"On the Cluster Analysis of Pedestrian Movements in Traffic"

Project Work

Christian Feist

November 16, 2010

The following company has supported this work:

Supervisors:
Dipl.-Ing. Martin Roehder
Prof. Dr. Ralf Möller

Hamburg University of Technology
Software Technology Systems
`http://www.sts.tu-harburg.de/`
Schwarzenbergstraße 95
21073 Hamburg
Germany

Dedicated to the paradise from which we cannot be driven.

Declaration

I, Christian Feist, solemnly declare that I have written this project work independently, and that I have not made use of any aid other than those acknowledged in this project work. Neither this project work, nor any other similar work, has been previously submitted to any examination board.

Hamburg, November 16, 2010

Christian Feist

Contents

1 Introduction

1.1 Motivation

Since the establishment of cars as a main means of transportation and the corresponding increase of traffic, road safety has become a major development area for manufacturers and politicians to cope with multiplying numbers of fatal traffic accidents. While most of the improvements made so far mainly benefit the driver and his passengers inside the car, the urban accident figures of the world's growing metropolises have shifted attention towards the weaker participants of modern traffic. As urbanization is increasing, the development focus of today's automotive safety systems has moved to the complex scenarios of built-up areas and protecting the driver and the non-occupants from the physical and psychological aftermath of pedestrian accidents has not only become a profitable competitive advantage but also the subject of legislative discussion.

1.1.1 Accident Statistics

To be able to relate to this trend we will state some statistical observations from different sources ([WHO09] from Europe and [NHT08] as well as [IIH09] from the U.S.) that are among those illustrating its foundations. They show the relevance of pedestrian accidents in modern traffic as well as some important situational data that is connected.

"Pedestrians constitute the most important non-occupant victim group." All investigated studies show that accidents with pedestrians make for a significant amount of total traffic accidents. Our sources claim figures of 12% for the U.S., 18% for the EU and 14% in Germany. All of these numbers are above those for cyclists or other non-occupant movement methods such as rollerblading or skateboarding.

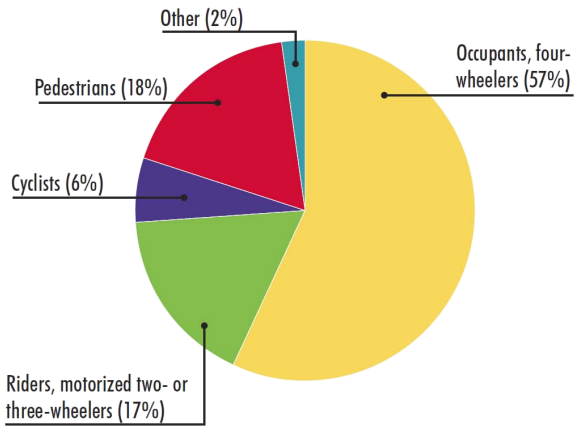

Figure 1.1: Deaths by road user category for the European Union

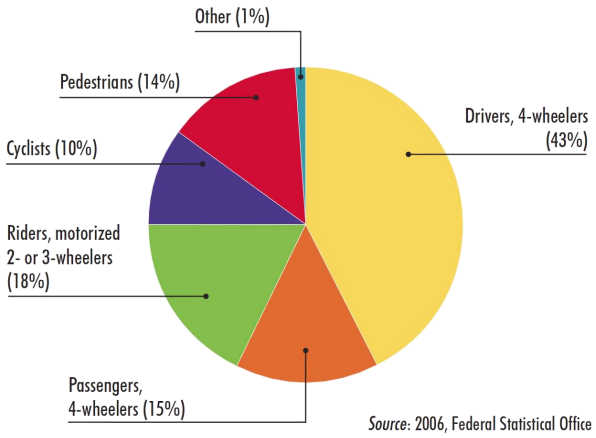

Figure 1.2: Deaths by road user category for Germany

"Most pedestrian casualties occur in urban scenarios." According to the latest U.S. study from 2009, 71% of pedestrian deaths occured in urban areas with 58% on major roads. 53% of all victims were hit on roads with an allowed speed of 40mph or less. This corresponds to the general traffic accidents figures. Germany, for example, reports about two thirds of all accidents inside built-up areas and even though the amount of deaths is higher outside the cities, the number of injuries inside urban areas is roughly 62% higher than anywhere else.

"Each traffic accident costs every citizen a significant amount of money." The European study shows that every death in traffic on its soil costs the state of Germany around 1.162.000 €. Figure 1.3 shows the cost for the annual injuries and deaths per person for different states.

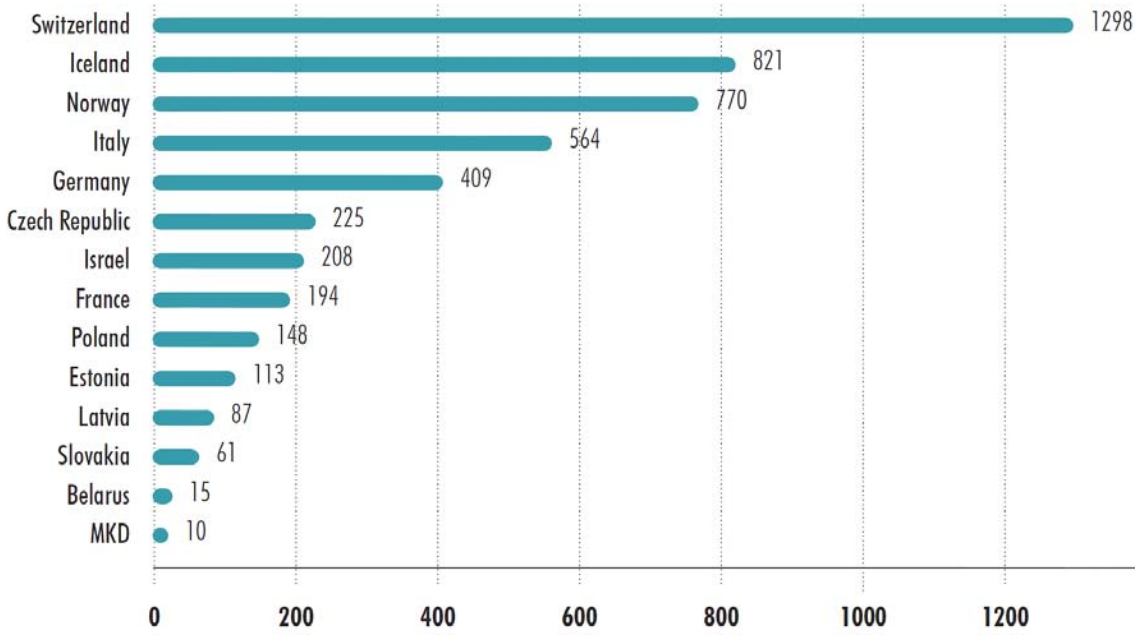

Figure 1.3: Costs (in €) of road traffic deaths and injuries per person in selected countries in the WHO European Region

"Pedestrian accidents happen because of perception failures." Even though the weather conditions were clear in 89% of the accidents, the U.S. study shows that visibility was hindered by night-time in 70%. However, all investigated resources support the claim that not only the drivers lack perception as the victims are often at the age over 70 (62% higher victim rate than lower ages in the U.S.) or under the influence of alcohol (53% of killed pedestrians over 16 years in the U.S. had blood alcohol concentrations of at least 0.08%).

1.2 The Objective

With the sensor technologies available inside a modern car and its development towards a connected multimedial information system, new ideas to fill the gaps of human perception and assist the driver of the future inside the ever-growing complexity of urban traffic situations are in technological reach. Vehicles are on their way to master the identification of traffic participants, road signs, lane borders and their location on the map. Modern communication technologies enable them to spread their findings and receive information perceived by other cars and local structures in the environment.

This enables current active safety systems to mitigate some collisions when they are unavoidable. Due to the fixed and nearly exhausted physical limits regarding maneuvers like braking or avoiding, turning increasingly available information into situational prediction is the key to gain the extra time needed to further lower the rates of injured and died pedestrians.

This thesis is part of a project that moves the focus of study from the car to the pedestrians and takes the situational context of the scene into account to achieve a secure assessment of the appropriate reaction earlier than current models. To be able to predict pedestrian behavior within critical traffic situations it has to be examined and analyzed for existing patterns first. Our project presents proof that such patterns exist and introduces an effective framework to exploit this knowledge for a better prediction of an identified pedestrian's future movement. One of the foundations to implement such a model based on recorded information across different scenarios is to answer the question tackled by this thesis:

"Given a vast amount of movement data gathered for a traffic scene, how do you find and efficiently represent the dominant routes taken by pedestrians?"

2 Background

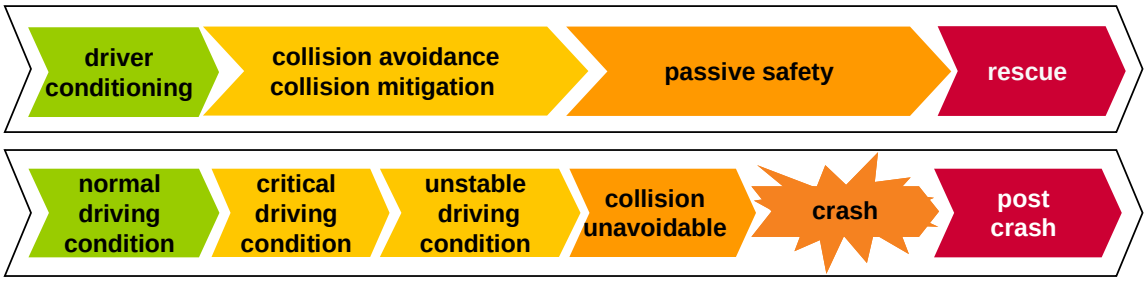

Figure 2.1: Process of accident development and strategies of the integral safety approach [Gol]

2.1 Integral Automotive Safety

To understand the context of our work, let us first introduce the basic design of today's approach to automotive safety. The participating safety features can be classified by their aims and the situtation in which they become active. The integral design considers the system as a whole which can be aligned to the development process of an accident as shown in figure 2.1. Therefore, we will present the characteristics and typical systems for each part of an integral safety system while following the example of a road accident with a wild animal.

2.1.1 Driver Conditioning

Driver conditioning systems include the comfort and driver assistant funtions active during usual driving conditions. They gather, filter and present information for the driver and issue warnings if the measured data shows uncommon behavior. Examples include:

- **Low temperature warning** - issued below 4 °C because of possible slippery conditions
- **Lane departure warning** - monitors the car's lateral position within its lane and warns if a line is crossed
- **Light sensor** - is coupled with headlights to automatically react to dusk or tunnels
- **Adaptive Cruise Control** - laser- or radar-based system to track objects ahead and adjust vehicle speed

Within our rural road scenario, let us assume we have an upper class car driving through foggy surroundings. The car has automatically switched the lights on and its navigation system has informed the driver of the currently allowed top speed.

2.1.2 Collision Avoidance and Mitigation

If the situation becomes uncommon and dangerously unstable, a class of safety systems that is often referred to as "active safety" seeks to avoid a possible crash or to limit its aftermath. These systems are listening to their sensors and have actuators that have a direct effect on the car's driving behavior. The following are among those found in modern cars:

- **Antilock Braking System** - ensures maneuverability during full brake application by releasing brake pressure in a defined interval
- **Electronic Stability Control** - prevents oversteer and understeer by braking individual wheels according to the driver's desired steering angle
- **Emergency Brake Assist** - interprets the velocity of the brake pedal to issue maximum brake pressure earlier

Our driver hears the warning from the distance radar and sees the animal in front of his car. As he quickly lifts his foot from the gas to the brake pedal, a modern ESC system identifies a "panic brake" pattern and prepares the brake by already attaching the blocks the brake disc. As the brake pedal is pushed down rapidly, the brake assist overrides with maximum pressure after only a few millimeters down to ensure faster deceleration. However, once the wheels begin to block, the brake is limited by ABS so our driver can steer the car to avoid the animal.

Figure 2.2: System design for integral safety with **sensors**, **algorithm**, **actuators** and **passive features** [RSK10]

2.1.3 Passive Safety

Should a crash be unavoidable, the passive safety systems are the last line of defense not only for the driver and his passengers but also for potential collision opponents. These systems range from simple static or mechanical constructions to sensor-activated countermeasures. Some examples populate the following list:

- **Airbags** - crash sensors around the car trigger explosion-inflated nylon fabric bags to prevent occupants from hitting interior parts like the steering wheel or windows
- **Seatbelt fastener** - automatically strengthens and locks the application of the seatbelt for all passengers

- **Car body systems** - specially manufactured body parts have a wide range of applications including side impact protection or soft hoods to mitigate pedestrian impacts

In our common road scenario, the vehicle leaves the road because of the driver's dodging maneuver and diagonally hits a large rock. The vehicle determines which seats within the car are occupied and fastens their seatbelts and enables airbag functionality. In the moment of collision, the front and side airbags inflate within $0.04s$ to protect the driver's head. The car body absorbs impact energy because of its specific manufacturing process and preserves the room for passengers inside by channeling the impact energy to outer parts of the frame.

2.1.4 Rescue

After the crash, modern cars employ rescue systems to enhance the chances of finding the vehicle and evacuating the driver from the inside. This stage of integral safety can consist of the following mechanisms and more:

- **Unlocking all doors** - to ensure easy entry for rescuers the central locking of the car will release all door locks
- **Enabling the warning lights** - activating the blinking orange lights on the car's front and back increases awareness of the wrecked car
- **Automatic emergency call** - systems like "eCall"[Inf10] call the closest emergency central and transmit GPS position and number of passengers

Our crashed vehicle dials the unified emergency number via its GSM module and transmits the values from its GPS antenna and the seats' occupancy data. While the warning lights are turned on and the doors are unlocked, an audio connection to the rescue service is established that enables the driver to speak to the emergency central and give information about the nature of the accident and his injuries. The central can issue warnings to local radio stations and over the traffic information frequencies that warn other drivers about the crash site.

2.2 Clustering

Our goal of grouping pedestrian movements by their recorded data moves the focus of our thesis to the field of data mining. Therefore, this chapter will cover the theoretical foundation of the automatic partitioning of data into groups, known as clustering.

2.2.1 Definition

Clustering (or "Cluster Analysis") is a form of unsupervised learning and is largely used as a method for statistical data analysis. It describes ways to split a set of measured data into subsets that contain observations which are similar by some criterion. [TSK05] divides the aim of clustering into two situational goals: to find groups of data within the set that are either meaningful ("Clustering for Understanding") or useful ("Clustering for Utility").

Clustering for Understanding

Clustering for Understanding tries to find the natural underlying structure of the given data. It resembles the human talent to group observations into meaningful classes and to assign new observations to these "labels". In this context a cluster is seen as a potential class and cluster analysis can be defined as "the study of techniques for automatically finding classes" [TSK05]. Early examples can be found in Biology, mainly in the classification of living things into the biological taxonomy. Today's market research, information retrieval or medical virus analysis also makes heavy use of said methods.

Clustering for Utility

Clustering for Utility creates classes to be able to represent individual observations of data by a more abstract (and often coarser) label. These methods will often try to find a single data object that is then used to represent each measurement of its own cluster. "Therefore, in the context of utility, cluster analysis is the study of techniques for finding the most representative cluster prototypes" [TSK05]. As it finds a simpler representation for the data set, typical applications are compression (such as vector quantization) or performance optimization of algorithms for large data sets by performing calculations only on the representatives without losing much accuracy.

Cluster Analysis can be broken down into further categories based on the shape of its outcome (a "clustering"). We will follow the categorization from [TSK05] that specifies the following distinct feature characteristics:

- Hierarchical or Partitional:

  When clustering the data, the method can either produce a nested or unnested structure. Partinional clustering is the simple division of the data into categories. If we allow a category to have subcategories, we are building a tree structure in which every node of the tree is a cluster and each node (except for leaf nodes) holds the union of its children ("subclusters"). This is called Hierarchical Clustering. Both results can be transformed to the other form by trivial set theory.

- Exclusive or Overlapping or Fuzzy:

  Considering a data point's form of membership in a clustering, we have different possibilities. Exclusive means that each observation is a member of exactly one cluster as opposed to allowing multiple memberships ("overlapping"). A reason to favor a non-exclusive method might be to avoid the creation of new classes that are characterized just by a combination of already known features (leading to "overshooting" the data). A distinct method from these two is to treat clusters as fuzzy sets. This means every data point is a member of every cluster with a weight between 0 and 1. If we apply the constraint that these weights must sum up to 1, we define a probabilistic way of clustering the data with each weight being an observation's probability to belong to a certain cluster. De facto, this is often transformed back to an exclusive system again by assigning the data point to its most likely cluster only.

- Complete or Partial:

  Regarding the amount of points of the underlying base set of data that has to be in a membership, we can choose to allow ("partial") or disallow ("complete") data points not being assigned to any cluster at all. A reason to do so is the avoidance of very small or even singleton clusters that could be classified as not being part of any well-defined structure within the data but as representing noise or outliers. However, note that if we were to use cluster analysis to retrieve data points by their class, this technique would have the potentially undesirable effect of making some of the data inaccessible.

Furthermore, the form of a single cluster itself can be used to separate different methods of cluster analysis. A detailed definition of the categorization into "well-separated", "prototype-based", "graph-based", "density-based" or "shared-property" clusters is beyond our scope. Feel free to refer to [TSK05] which also gives some examples. We will proceed with the introduction of the actual algorithmic base that is later used to achieve our goals by cluster analysis.

2.2.2 K-Means

By the definitions made earlier, the k-Means clustering algorithm can be classified as a complete prototype-based exclusive partitioning technique. It is typically applied on data in a continuous n-dimensional feature space, because the resulting prototypes are found as centroids (usually the mean of a cluster) and are not part of the data set per se. For feature spaces that do not define a method to average a cluster to a prototype one would have to use a different algorithm like k-Medoid which chooses an actual participant as representative. Following [TSK05] the algorithm can be described as in 2.1. Let us have a look at each step in detail.

Algorithm 2.1 k-Means algorithm

0: Select k points as initial centroids.
repeat
  Form K clusters by assigning each point to its closest centroid.
  Recompute the centroid of each cluster.
until Centroids do not change.

Selecting the initial centroids Distributing the representatives before the first run is a non-trivial but crucial step for the end result of clustering with k-Means. Even though the method produces deterministic results for the same distribution of initial centroids, results can significantly vary for different placements. The classical way to begin k-Means is by distributing the centroids randomly. As mentioned, this may yield poor clusterings as random distribution may as well be unfortunate. A way to achieve a better partitioning is to repeat the procedure with random initial centroids each time and then chose the k-Means run that resulted in the least sum of the squared error (SSE, also "scatter"). This is the global squared sum of all distances of the data points to their assigned centroid which can

be formally written as

$$\text{SSE} = \sum_{i=1}^{K} \sum_{x \in C_i} dist(c_i, x)^2 \tag{2.1}$$

Assigning data points to clusters The measurement of similarity used to assign data points to centroids is some form of distance norm within the feature space. Apart from the obvious Euclidean distance or the Manhattan norm, [TSK05] also mentions cosine similarity or the Jaccard measure as possibilities for non-Euclidian feature spaces. Depending on the problem at hand, other distance measures like the Mahalanobis or Hamming [Ham50] distance have been suggested. An important factor for practical use is the computational complexity of the distance measure as it needs to be calculated repeatedly for each data point and centroid.

Recomputing the centroids For continuous spaces it can be shown that choosing the mean of a cluster as its centroid minimizes the SSE and therefore generates the best result at this step of the algorithm. It is defined as

$$c_i = \frac{1}{n_i} \sum_{x \in C_i} x \tag{2.2}$$

This property also holds for other distances in the Euclidian space.

Abort criterion of the algorithm What is formulated quite generally in our algorithm description as "centroids do not change" actually translates to a lack of change in membership of any data point. This strict criterion is often softened by already stopping if less than a certain percentage of the data have been reassigned. This is practical because most of the algorithm's convergence happens during the first steps and some combinations of initial centroid distribution and chosen distance measure can cause small oscillations in memberships in the final stage of the algorithm.

2.2.3 Finding k

A significant drawback of the standard k-Means algorithm is that you have to specify k in advance. Choosing a "wrong" value prevents the method from finding a meaningful partitioning and in most scenarios the correct number of clusters is unknown beforehand. As we have seen in 2.2.2 running k-Means multiple times and trying to maximize a performance measure can help to choose the best distribution of initial centroids and the same can be done to decide on k. Based on this technique, other algorithms have been suggested that make use of different performance measures and some changes in structure to tackle the problem of automatically finding the natural fit of k for the underlying data set.

X-Means

X-Means is such an algorithm. It uses k-Means as a building block and increases k by splitting certain clusters into two and reevaluating the achieved clustering using the Bayesian information criterion (BIC).

BIC scoring This performance measure maps the chosen model and its underlying data to a scalar score value by considering its posterior probability. Based on [PM00] it uses the following approximation for the j-th model M_j also known as the Schwarz-criterion:

$$\mathrm{BIC}(M_j) = \hat{l}_j(D) - \frac{p_j}{2} \cdot \log R \tag{2.3}$$

where $\hat{l}_j$ is the log-likelihood of the data D according to model M_j from the maximum-likelihood point, p_j the amount of free parameters and R the size of the data set. We can express the maximum-likelihood estimation (MLE) for the variance under the identical spherical Gaussian assumption as

$$\sigma^2 = \frac{1}{R-K} \sum_i (x_i - \mu_{(i)})^2 \tag{2.4}$$

and the point probabilities as

$$\widehat{P(x_i)} = \frac{R_{(i)}}{R} \cdot \frac{1}{\sqrt{2\pi}\hat{\sigma}^M} \exp(-\frac{1}{2\hat{\sigma}^2} \|x_i - \mu_{(i)}\|^2) \tag{2.5}$$

Then the log-likelihood of the data can be written as

$$l(D) = \log \prod_i P(x_i) = \sum_i (\log \frac{1}{\sqrt{2\pi}\sigma^M} - \frac{1}{2\sigma^2} \|x_i - \mu_{(i)}\|^2 + \log \frac{R_{(i)}}{R}) \tag{2.6}$$

Fixing $1 \leq n \leq K$ and only focusing the data points D_n belonging to cluster n yields

$$\hat{l}(D_n) = -\frac{R_n}{2} \log(2\pi) - \frac{R_n \cdot M}{2} \log \hat{\sigma}^2) - \frac{R_n - K}{2} + R_n \log R_n - R_n \log R \tag{2.7}$$

The authors of [PM00] define the free parameters p_j as the sum of $K-1$ class probabilities, $M \cdot K$ centroid coordinates and one variance estimate. To calculate the global log-likelihood for all clusters we can simply sum up the local log-likelihoods for each cluster and use the number of data points of the complete set as the value for R.

As described in [PM00] the X-Means algorithm goes through the steps shown in algorithm description 2.2. As before, we will inspect each part of the algorithm in detail. However, the suggested

Algorithm 2.2 X-Means algorithm

repeat
  Improve Parameters
  Improve Structure
until $K > K_{max}$
return the best-scoring model found during the search

performance optimizations are not in our focus and will not be discussed. The interested reader is pointed to [PM00] for some recommendable insights.

Improving parameters The "Improve-Param" phase consists of just running usual k-Means to its convergence. Therefore, it takes care of recalculating data point assignments and centroid positions as described in 2.2.2.

Improving structure Changing the structure of the clustering basically means deciding on the birth and location of new clusters. X-Means achieves this by splitting existing partitions in two and comparing the resulting BIC score against the one determined for the undivided one. In detail, this is done by splitting each cluster of the previous run and spawning two new centroids within its borders. Their location is calculated by moving them "a distance proportional to the size of the region"[PM00] in opposite directions on a random axis. Then for each previous cluster 2-Means is run locally, in a sense that only the data points of the parent region are reassigned to the new children. Now the model test mentioned earlier is performed and (depending on the score) either the parent or its children are dismissed. This way, regions of the data set that are not yet well-fit by the model receive finer clustering while others remain unchanged. After making these local decisions, the model is evaluated by a global BIC test, which means scoring the overall fit of all centroids to the complete set of underlying data.

Abort criterion of the algorithm X-Means demands the specification of a maximum value for k beforehand to stop iterating through the improvement of parameters and structure. If K_{max} is reached, it returns the model (and therefore the value of k) that has scored the highest global BIC score.

G-Means

The G-Means algorithm has a similar structure compared to X-Means. As its competitor, it uses recurring runs of k-Means on that data that is step-by-step partitioned into more and more clusters depending on the local outcome of its performance measure, the Anderson-Darling test for Gaussian fit. We will describe this criterion as well as some other peculiarities of this algorithm suggested by its authors in [HE03].

The Anderson-Darling test This statistical test will determine between the following two hypotheses for a given set of input data and a confidence level α:

- H_0: The data around the center are sampled from a Gaussian.
- H_1: The data around the center are not sampled from a Gaussian.

We will see later that choosing H_1 and rejecting H_0 for the data of a single cluster will cause the algorithm to split this cluster into two instead of keeping the larger one. The test is based on the one-dimensional Anderson-Darling statistic and tests for normality based on the empirical cumulative distribution function (ECDF). Given a sorted list of input data x_i with mean 0 and variance 1, defining $z_{(i)} = F(x_{(i)})$ with F being the $N(0,1)$ cumulative distribution function the statistic is defined as

$$A^2(Z) = -\frac{1}{n}\sum_{i=}^{n}(2i-1)(\log(z_i)+\log(1-z_{n+1-i})) - n \tag{2.8}$$

Considering the results of [Ste74] we need to correct the statistic because in our cluster analysis the values for μ and σ are estimated:

$$A^2_*(Z) = A^2(Z)(1 + \frac{4}{n} - \frac{25}{n^2}) \tag{2.9}$$

Put in the same format as in [PM00] we can specify the outer structure of G-Means as in algorithm 2.3.

Algorithm 2.3 G-Means algorithm

repeat
  Improve Parameters
  Improve Structure
until $K = K_{previous}$ (No more clusters were added.)

As the improvement of parameters again only means running k-Means to convergence (just as with X-Means) and the abort criterion is trivial, we will just focus on the details of the structural improvement step.

Improving structure To decide on the possible birth of new centroids, we have to make a local comparison of the data's representations by a single parent cluster or its two children c_1 and c_2. Therefore we need to spawn two child centroids within the data of their parent and run 2-Means to make new assignments. The spawn process of G-Means is far more sophisticated than just using a random axis for their displacement. The authors of [HE03] suggest a method based on the main principal component s of the data with Eigenvalue κ, which chooses the children's locations around the center c of the data as $c \pm s \cdot \sqrt{\frac{2\kappa}{\pi}}$. This way the centroids are spawned in their expected location under the Anderson-Darling test's hypothesis H_0. See [Cor97] for a powerful way to calculate these values. Once we have run 2-Means to convergence, we can reduce the data to a single dimension by projecting it onto the connection $v = c_1 - c_2$ of the final child centroid locations via $x'_i = \langle x_i, v \rangle / \|v\|^2$. Afterwards, the list of projected values acquired is transformed to have mean 0 and variance 1. We then form the list of cumulative values Z by calculating $z_i = F(x'_{(i)})$ and then test it using Anderson-Darling. If $A^2_*(Z)$ is among the non-critical values at confidence level α we discard the children and keep the parent cluster, because we accept H_0. Otherwise, we replace the original cluster with its children c_1 and c_2.

2.3 Edit Distances

An edit distance in information theory is a distance metric to measure the amount of difference between two sequences. Usually a set of elementary operations that transform one sequence into another is defined for such a metric and then the number of operations needed for the given case is determined. They can be used within a variety of domains where no geometric distance can be defined. As any other kind of distance metric, edit distances need to have some specific properties to make them practical (see [Keo]):

- $D(A,B) = D(B,A)$ **(Symmetry)**
  Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

- $D(A,A) = 0$ **(Constancy of Self-Similarity)**
  Otherwise you could claim "Alex looks more like Bob, than Bob does."

- $D(A,B) = 0$ iff $A = B$ **(Positivity / Separation)**
  Otherwise there are objects in your world that are different, but you cannot tell apart.

- $D(A,B) \leq D(A,C) + D(B,C)$ **(Triangular Inequality)**
  Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

2.3.1 Levenshtein Distance

The prime example of an edit distance is the one suggested for two strings by Vladimir Levenshtein in [Lev66]. It is defined as the minimum number of edit operations to transform one string into the other. The operations available are insertion, deletion and substitution of a character. For two strings u and v we can specify a distance matrix D by the following rules using $m = |u|$ and $n = |v|$:

$$D_{0,0} = 0 \tag{2.10}$$

$$D_{i,0} = i, 1 \leq i \leq m \tag{2.11}$$

$$D_{0,j} = j, 1 \leq j \leq n \tag{2.12}$$

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} & +0 \text{ if } u_i = v_j \\ D_{i-1,j-1} & +1 \text{ (substitution)} \\ D_{i,j-1} & +1 \text{ (insertion)} \\ D_{i-1,j} & +1 \text{ (deletion)} \end{cases}, 1 \leq i \leq m, 1 \leq j \leq n \tag{2.13}$$

The bottom right value of this matrix is the final value for the Levenshtein distance. The matrix at hand also contains all information needed to determine the minimal sequence of operations by a backtracking algorithm. In some applications, it is helpful to consider the following upper and lower bounds for the Levenshtein distance:

- It is always at least the difference of the sizes of the two strings.

- It is at most the length of the longer string.

- It is zero if and only if the strings are identical, see 2.3.

- If the strings are the same size, the Hamming distance [Ham50] is an upper bound for the Levenshtein distance.

2.3.2 Damerau-Levenshtein Distance

Working in the field of spell-checking, Frederick J. Damerau defined a distance metric for strings that considered not only insertion, deletion and substitution, but also the transposition of two adjacent characters. In [Dam64] he also claimed that these operations correspond to 80 percent of all human misspellings. As the distance measure used is almost identical to Levenshtein's suggestion, the addition of transposition to the algorithm is called the Damerau-Levenshtein distance. Instead of the definitions made in formula 2.13 we need to write

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} & +0 \text{ if } u_i = v_j \\ D_{i-1,j-1} & +1 \text{ (substitution)} \\ D_{i,j-1} & +1 \text{ (insertion)} \\ D_{i-1,j} & +1 \text{ (deletion)} \end{cases}, (1 \leq i \leq 2, 1 \leq j \leq n) \vee (1 \leq i \leq m, 1 \leq j \leq 2) \quad (2.14)$$

and

$$D_{i,j} = min \begin{cases} D_{i-1,j-1} & +0 \text{ if } u_i = v_j \\ D_{i-1,j-1} & +1 \text{ (substitution)} \\ D_{i,j-1} & +1 \text{ (insertion)} \\ D_{i-1,j} & +1 \text{ (deletion)} \\ D_{i-2,j-2} & +c \text{ if } u_i = v_{j-1} \wedge u_{i-1} = v_j \text{ (transposition)} \end{cases}, 3 \leq i \leq m, 3 \leq j \leq n \quad (2.15)$$

to include transposition.

3 Implementation

3.1 Measured Data

To determine the right methods to process it, we should first look at the data that has been measured in the course of this thesis and how it has been acquired. To gain meaningful insights into pedestrian movement in traffic, it is of utmost importance to study it in relevant situations and record movements in a way that simplifies further analysis.

3.1.1 Locations

Following the arguments made in 1.1.1 our work is concerned with pedestrians in crossing situations. To be able to make more general statements, the team chose three locations that offered a good perspective on zebra crossings. The most important beneficial features these areas have in common are:

- A frequented zebra crossing over a wide street (at least two lanes)
- Access to an indoor location with electricity and a window on higher ground with a nearly orthogonal alignment to the crossing path
- Allowance to record anonymous pedestrian movement data, see 3.3.1

We ended up favoring the following three candidates:

Private property, Ingolstadt With an explicit authorization to record pedestrians, it was possible to use the private property of a local production plant for our purposes. It featured a highly-frequented zebra crossing and high camera perspective. Being a workplace, traffic movements offered on weekends were expectedly few.

University campus, Kaiserslautern A narrow street intersecting two main areas of the TU Kaiserslautern's campus guaranteed a high amount of pedestrians crossing. A taller building nearby helped to create a good view over the area that produced interesting movement patterns heavily influenced by some natural attraction points like a bus stop or staircases.

Public street corner, Aachen This large crossing of two roads within the city area of Aachen provided us with some special scene properties including a refuge island in the middle of the pedestrian crossing as well as a pedestrian light that organized traffic flow. A rather flat perspective made this a tricky measuring station to set up and evaluate.

Figure 3.1: Camera view on our Kaiserslautern scenario

3.1.2 Setup

In the rooms available we installed a camera and a workstation for each scenario. The camera is built into a frame that protected it from direct illumination and then fixed on the window pane by using vacuum cups. The connection to the measuring workstation is established over a gigabit LAN connection. The cameras had to be intrinsically and extrinsically calibrated using a large calibration object in the form of a board with black dots on white ground. Proper calibration is crucial for the correct projection from pixel to 3D real-world coordinates. The machines used as workstations are powered by two quadcore Intel Xeon processors and have 8GBs of RAM at their disposal. They have been equipped with the Automotive Data- and Time-triggered Framework (ADTF [Ele10]) and are running a 64bit installation of Windows 7. Detailed descriptions of the calculations done on the data within ADTF are given later in 3.3.1.

3.1.3 Volume

	Ingolstadt	Kaiserslautern	Aachen
Days	106	100	28
Hours	1272	1200	336
Image data (GB)	1330	368	342
Serialized data (GB)	400	61	95
Volume reduction	0.70	0.83	0.72
Pedestrians total	148598	14757	7584
Pedestrians per day	1401.87	147.57	270.86
Pedestrians per hour	116.82	12.30	22.57

Table 3.1: Volume statistics of measured data

3.2 Tool Chain

To understand the context of this thesis and its significance to our pedestrian safety system, let us first give an overview of the overall system design and the involved components. As described earlier, our approach to improve current systems consists of a ground-based part that analyzes pedestrian behavior and derives a model for their movement and a mobile one that uses this model to draw environmental-based conclusions about the current safety situation within the car. The exact communication methods for the model are not discussed here as well as the technical issues regarding the car's reaction on possible threats. We will sketch the flow of data from a large amount of tracked pedestrian paths to a network representation of characteristic trajectory segments that can be used for a probability-based movement prediction of a person tracked by the car's sensors. Mind that this thesis' work is concentrated on the first of the following steps and we will therefore not go into the depths of each block. Details about the structure of data consumed and produced by the component in focus are given in 3.3.1 followed by specific algorithmic solutions.

3.2.1 Bundle Extraction

As this part is this thesis' addition to the system, we will leave detailing it to upcoming chapters. It deals with replacing human specification of filters to extract similar structures from the data with a fully autonomous data mining system that automatically determines meaningful groups of movements through the scene. We will use an approach based on evolved clustering techniques coupled with a meta-logic to generate each of these bundles and a corresponding rough graph that mimics a group's main direction.

3.2.2 Isolation

The isolation step uses the information on bundles and their outlines to generate a true representative trajectory (called a master trajectory) from each of them. This allows concentrating the data of a main stream of movement through the scene into a very small and handy format for further analysis. It is calculated by building slices that are orthogonal to the outline direction through the bundle's trajectories and then averaging each measurement. This requires resampling the given outline and the underlying tracks to generate approximately equidistant measurements as the raw trajectories have completely arbitrary amounts of corner points. The sampling rate is usually increased which requires interpolation to generate data between two measurements. For this study, linear interpolation is used, but nonlinear methods are likely to improve the result and should be the object of further research.

3.2.3 Segmentation

Because each measurement does not only hold position coordinates of a pedestrian, there might still be heavy divergences within a recognized bundle. The segmentation step tries to find features that allow splitting a given master trajectory into segments of different characteristics. A good example would be the velocity distribution among the underlying trajectories of a bundle. There may be different profiles of movements that should be considered distinct, like pedestrians stopping before crossing and those proceeding without lowering their speed. Generally, the system is able to calculate even

more features than the object tracker records. Some of them might be scene-related, such as velocity in direction of a certain object or distance to a line specified by the user. Segmentation can identify different behavior within a master trajectory's scope by clustering such features and can then construct a tree of master trajectory segments that models separation in more than just the local dimensions. Links between the segments hold the assigned transition probabilities that mirror the distribution of trajectories participating in a segment.

3.2.4 Prediction

To use this tree of master trajectory segments for a given scene for prediction, we need a method to decide which initial segment a pedestrian is to be assumed on. In our system this decision is made by classifiers which are defined for each segment. Using the measured data from the static camera system, they are trained for the characteristic feature dimensions determined in the segmentation step. Early studies suggest that correlation of features over a sliding window can significantly improve separability and thereby classifying performance. The resulting parameters from training are verified against a test data set, to see whether decisions by the learned feature combinations hold to be sufficiently correct. This step of the system completes the ground-based and long-term situation analysis. The tree is then made available to the car in some form and can there be compared to incoming sensor data of current pedestrian movement. As a human is tracked in the car's range, the measurements are matched to data points on possible master trajectories via interpolation. Then the corresponding classifiers calculate their decision on the likelihood of the sequence of measurements matching their segment and the pedestrian is assumed to be on the segment with most certainty. From there, the transition probabilities are evaluated to construct the likelihood for the person to be on any following segment. Should a graph through the tree that is intersecting with the road receive a sufficient conditional probability, the pedestrian's time to reach each segment along the way is calculated to construct his likely movement path in the three relevant dimensions to determine the vehicle's optimal reaction: latitude, longitude and time.

3.2.5 Action Concept

The collision detection algorithm determines whether or not the determined probable pedestrian movement trajectory intersects with the car's extrapolated path and what time to collision (TTC) is left. It also calculates the cross-section of car and human to assess the relative collision spot on the vehicle's front. These parameters are input to a decision logic that chooses between three possible reactions to the witnessed sensor data:

1. Warning the driver about the potential threat caused by the presence of a pedestrian determined to cross the street. This action should be undertaken if the probability is the highest of all possibilities but its absolute value is still rather low or if the TTC is high enough to let the driver decide freely.

2. Giving a steering recommendation with an impulse on the steering wheel. This reaction is designed for medium TTCs and cases where the cross-section shows that the impact location of

the pedestrian will be on the side of the front bumper rather than central. Collision avoidance by steering has the chance of leaving the target completely unharmed even if the distance is rather close, but it also has unresolved technical and legal issues regarding the direction of steering without creating a different danger situation.

3. Apply emergency braking without any action by the driver. This is reserved for cases in which the TTC is below the time needed to stall the car and the classification significantly certain. Ignoring prior steering recommendations will also lead to this reaction.

3.3 Applying Clustering

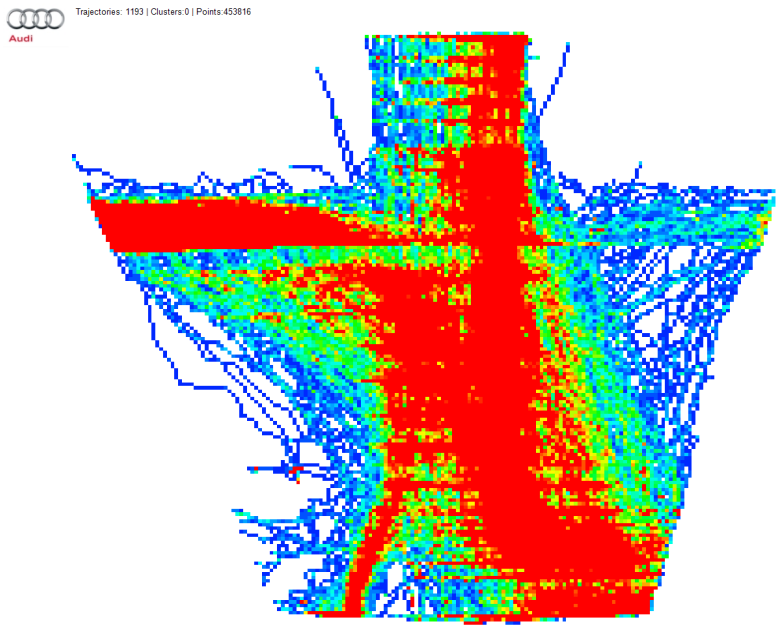

Figure 3.2: Projected top-down view with trajectory grid (Kaiserslautern)

3.3.1 Inputs and Outputs

The task of this thesis' algorithmic implementation is the identification, filtering and preparation of the main groups of pedestrian movement in a given scene. Before detailing the inner structure of the algorithm, we will define its position in the data flow through the system.

Input

As mentioned earlier, the initial data source is a static camera system aiming at a pedestrian crossing from a higher angle. The connected workstation receives a video stream in a resolution of 1280 by 1024 pixels. These images are not recorded directly for two reasons:

1. Privacy:

   To be granted permission to study human behavior in public situations you must ensure anonymity to be able to use the data without the explicit consent of each individual that has been recorded.

Furthermore, other scene components (especially cars) have recognizable features that are linked to individuals or companies and are not be monitored or tracked either. For one of the scenes the system had to obey particularly strong secrecy constraints as the camera was placed inside Audi's production plant and was likely to produce footage of publicly unknown products or activities.

2. Storage Volume:

   Recording a video stream this large requires a significant amount of memory per second. This poses a challenge on the workstation's hard drive writing bandwidth as well as the actual amount of disk space required per day. As the measurement systems were designed to operate without network connectivity in buildings that offered mainly a good view over the scene and access to electricity, the data had to be transferred by hand using external drives. The goal of maintaining organizational effectivity made a longer interval between data acquisitions desirable.

Performance improvements of an earlier work have allowed the real-time conversion into a foreground-background-segmented black and white stream of the same size at 30Hz. This solution leaves little room for identification of individuals and is directly compressed using a run-length coding scheme and therefore satisfies both requirements.

The recorded data is afterwards fed into a time-consuming object tracking algorithm that produces binary serialized trajectory files. Object tracking is Kalman-filtered and ignores objects over a maximum size. The objects created from the tracking algorithm are identifiable by a unique ID and mainly consist of an array of measurements holding float values for position, velocity, acceleration, size of the object and more. It should be noted that this datatype is included from the Automotive Data and Time triggered Framework (ADTF), which has the benefit of direct compatibility to objects tracked by car sensors. The tracking and binary serialization process reduce the volume of data by a factor of about four compared to the segmented video stream.

Within the analysis framework these C++ structures for the traces of moving objects are held in a pool that automates their transitions between persistent and volatile memory as well as wrapping them into managed .NET objects for easier handling and automatic garbage collection. Trajectories can be grouped and retrieved using string tags to simplify access within the large amounts of data recorded.

They are then filtered to keep only pedestrian movements in the pool for further processing. This is achieved by an extensible XML-defined filtering framework allowing for tree-structured logical compositions of different Boolean filter components such as "has crossed a line in the scene" or "had a maximum velocity of". Integrated into an efficient user interface, this solution can be used to avoid noise like trees shaking in the wind and to eliminate car trajectories from the scene. The remaining trajectories form the input set for clustering and bundle extraction.

Output

As described in the system overview, the clustering step is succeeded by master trajectory isolation which reduces a bundle of similar trajectories to one representative trajectory by resampling and averaging their measurements. These averages are to be calculated over slices through the bundle roughly

orthogonal to the bundle's tangent direction. This defines the required output of the clusterer as two main structures:

1. Groups of trajectories that are classified as similar and as significant for the overall characterization of pedestrian movement in the scene. Their number should be minimal and strongly correlated to the mainly used combinations of entry and exit points of the scenario. Generating too many bundles results in master trajectories that carry less information and makes the decision of the classifier 3.2.4 both more expensive and more error-prone. Obviously divergent trajectories should still not be within the same bundle as should be outliers diverging from the overall scene characterization. Both cases are likely to spoil or even prevent the calculations of the isolation phase.

2. A single sequence of points in the scene (called an outline) for each group of trajectories that was distinguished by step 1. This outline is used by the isolation algorithm as the tangent approximation for the bundle needed to generate the locations and directions of said slices for averaging. For a meaningful master trajectory the outline should lie within the bundle's dimensions and should smoothly follow the common direction of the underlying trajectories. Hard corners, opposite direction or zig-zag-patterns cause isolation to use single measurements within the group more than once or to stop prematurely because a slice did not cut through all participating trajectories.

Technically, the trajectory bundles are realized by tagging all trajectories belonging together with a common name that can be linked to their outline. The outline itself is given as a sequence of cluster centroids with their scene coordinates.

3.3.2 Clusterer Implementation

Preceding the solution to bundle extraction and outline generation downstream we will examine the design and optimization of the algorithm's main building block - the clusterer itself. The theoretic foundations studied earlier in 2.2 introduce a variety of approaches to cluster data in a space of features. This chapter will describe the peculiarities of the given environment and the decisions made to recombine and adjust the underlying methods to find a meaningful partitioning for the given inputs.

Comparison of different street scenes has led to the assumption that the reasonable number of clusters should be treated as unknown beforehand and should be determined by the algorithm itself without human supervision. This constraint rules out the basic k-means algorithm and instead demands a higher-order logic on top of it to find a proper value of k for the set of input data. Examples of such structures have been examined in 2.2.3 and a derivate of their core idea will be used to tackle the given problem.

Structure

As analyzed earlier, the common approach behind k-Means successors like X-Means or G-Means is their approach to find the right amount of clusters by clustering by k-means, changing the value for k

and then deciding whether the new model represents the data better than before. This is repeated as long as a more suitable fit has been found compared to the run before.

This method is partly adopted for this project. However, we change the structure from iteration to hierarchical recursion. Instead of running a global clustering step and evaluating the new partitioning on all the data, the algorithm presented here uses 2-Means within a cluster and decides by a so-called split criterion whether or not to force a local split of this cluster into two new ones. This is inspired by X-Means, but we do not use a global refinement step after a split has occurred. The described procedure is then started on the complete base set and repeated until no further splits have occurred. Then the assignment step is recalculated again once to refine the partitions. The algorithm is outlined in description 3.1.

The split itself is suggested by randomly choosing a normalized direction vector in the dimension of the feature space and spawning two new centroids in positive and negative distance of the base set's standard deviation from the base centroid. This method has also been suggested by [PM00].

Algorithm 3.1 Outer structure of our algorithm

```
0: C_0 ← D
   C ← C ∪ C_0
   repeat
      |C|_previous ← |C|
      for all C_i ∈ C do
         Spawn c_1 and c_2
         Run 2-Means
         if Criterion(c_1, c_2, C_i) = true then
            C ← C \ C_i
            C ← C ∪ c_1 ∪ c_2
         end if
      end for
   until |C| = |C|_previous
   Reassign points to centroids
```

The presented technique has been chosen for its good parallelizability in comparison to its minor accuracy drawbacks. As only a local subset of the data is considered by the inner clustering steps, the algorithm is prone to slightly overshooting the data with respect to the final count of clusters. This is because of the theoretical possibility of splitting through a part of the base set that would be well represented as one cluster. The subsequent steps are likely to create smaller partitions on both sides of the border without any means to connect them. However, in the course of this thesis we will see that these effects do not harm the results as the data's structure as well as the algorithm's goal tolerates the error. How parallelism in structure is exploited for better performance is discussed in 3.3.3.

As within the group of algorithms mentioned earlier, differences in their performance on various shapes of measured data are mainly the result of their evaluation criterion. For a number of reasons, our clusterer will be determined by its own approach to make this decision. The development stages and final characteristics of this split criterion will be discussed in 3.3.2.

Algorithm 3.2 Inner 2-Means structure

0: Select c_1, c_2 as initial centroids.
repeat
$previous_1 \leftarrow c_1$
$previous_2 \leftarrow c_2$
Form 2 clusters by assigning each point to its closest centroid.
Recompute the centroid of each cluster.
until $(d(c_1, previous_1) < d_t) \wedge (d(c_2, previous_2) < d_t)$

The structure of the inner k-means block remains unchanged with a single exception: The abort criterion has been relaxed for faster convergence without a significant loss in accuracy. As you can see, the condition for stopping has been shifted from "no data point has changed its cluster membership" to "no cluster centroid has traveled further than a certain threshold" in comparison to the last pass through the loop. Since the concept of outlines is a simplification by itself and merely a base for further exact calculations, the small error in centroid positions is acceptable at this stage. Early tests have shown that the speed of convergence increases by magnitudes especially since small oscillations of membership in the final stage of the algorithm are avoided. Furthermore, having a threshold parameter allows for manual adjustment in the trade-off between speed and accuracy in real-world units.

Split Criteria

The previous chapter has already pointed out the significance of the split criterion. Its task is to review the larger base cluster and the newly suggested child clusters and make the binary decision which model represents the underlying data in a better way. The recursive structure limits the available factors that can be considered in the process to those that are directly related to those three clusters (see Parallelism 3.3.3). The decision made should therefore be characterized as local rather than global. The following list gives an overview of the more meaningful quantities at hand:

- Members count
- Standard deviation from centroid
- Centroid position
- ID of source trajectory for each member

As the performance of this criterion is vital to a meaningful partitioning of the feature space, we can compile a wish list of desirable behaviors. These need to be transformed into solid mathematical terms that translate to a single Boolean.

- Convergence: The criterion should lead the algorithm to a deterministic stop in a reasonable amount of time.
- Cluster size: The data should neither be split into too small groups that carry little information and cause centroid counts to rise over a representational number, nor should clusters with a high member count assimilate smaller groups that are clearly separated geometrically.

- Distance: For meaningful outlines and a more correct partitioning of end points later, centroid distance should not decrease below a critical measure as this has proven to be limiting undesirable effects on bundle-shaped data.

G-Means suggests that the criterion should take a cluster's member count as a normalizing factor to avoid "bad decisions about clusters with few data points" (see [HE03]). This thesis' implementation uses this advice, but not within a distribution-dependent statistical test like Anderson-Darling or BIC. Evaluations of these techniques have shown them to be too strict for use with connected data like ours. Instead, the standard deviation of the members with respect to their centroid has been chosen as the measure of fit within a cluster. This leads to the first version of our own split criterion:

$$\text{Split iff: } \frac{\frac{\phi_1}{n_1} + \frac{\phi_2}{n_2}}{2} \cdot b < \frac{\phi_P}{n_P} \tag{3.1}$$

with standard deviation ϕ_i within cluster i and member count n_i for the children and 2 and their parent P.

At first glance, this equation leads to desired behavior in terms of the first two points of our wish list. As the data is divided into smaller and smaller clusters, the standard deviation of each of them is decreasing as desired. However, at the same time, they lose members with every split, which ultimately leads to convergence because the standard deviation of the parent is normalized over a member count of twice the size of a child's fellowship on average.

The factor b introduced between parent and child allows shifting the balance by penalizing the children. It became necessary as a one to one ratio has led to far too many splits during initial tests. This shows the main downside of this formula as a universal split criterion for our task: There is an artificial weight factor that can only be humanly determined for a given set of data and has no real-world meaning across different scenes. This sort of trial-and-error approach to splitting is inacceptable for an algorithm designed to automatically decide on the number of clusters needed for any pedestrian movements recorded.

Further exploration of mathematical possibilities has led to a solution that incorporated the third wish from our list into the formula. Since weighing the two sides of the equation has been proven necessary, linking the factor to real-world distance seemed to have potential. Of course, the factor is hard to normalize to fit all possible scenarios. That's why it has been composed as the ratio between a given nominal distance and the actual centroid distance of the two children. The improved criterion has the shape of

$$\text{Split iff: } \frac{\frac{\phi_1}{n_1} + \frac{\phi_2}{n_2}}{2} \cdot \frac{d_{nominal}}{d_{actual}} < \frac{\phi_P}{n_P} \tag{3.2}$$

and makes use of the ratio between a given nominal distance $d_{nominal}$ and the current euclidian distance d_{actual} between children 1 and 2.

This change transfers the abstract and rather arbitrary weighting factor from before into a value of physical meaning. Setting a nominal value in meters will cause the criterion to penalize the children's

side whenever the suggested split results in a centroid distance below this value. On the other hand, children of a longer distance stand a chance of being spawned even if their member count is small compared to their parent. We will see in the results (see 4.2) that adjusting this factor is easy and even universal across different scenarios and leads to anticipated behavior of the clusterer.

ID-awareness

As described before, the data we are dealing with in this project is of specific shape. Clustering the local coordinates in two dimensions to find the "hot spots" of pedestrian movement within a scene leaves valuable information unconsidered: the links. As every trajectory can be tracked by a unique identifier within the application, the clusterer can be made aware of a measurement's source using a simple dictionary. Maintaining said dictionaries leads to both memory and performance costs, so one should ask whether this will actually solve a problem.

Looking at our data, we faced the following scenario. Two pedestrians cross the street, but at different speeds. The camera system tracks both persons at the same defined frequency and generates measurements for their track. This obviously leads to the slower person being measured more often per distance unit than the quicker. Generally this behavior is tolerated and unavoidable. However, it causes undesirable outcomes if you then apply clustering on local coordinates. The reason is the algorithm's way of updating the representative's position to the center of gravity within a cluster. This treats every measurement equally, causing a trajectory with more measurements per distance to gain more influence than others and to divert the meaningful position of the centroid.

Using the dictionary approach, we can solve this problem and make our algorithm more resistant against outliers. Instead of shifting centroid positions into the mean coordinates of all members, we introduce an intermediate step and first average the position of the participating measurements that belong to the same trajectory. Afterwards, the representative is moved into the center of these averages. The effect on the resulting outlines can be seen in 4.2 and is discussed in the evaluation (see 5.2.2).

3.3.3 Performance Optimizations

Making the clusterer aware of the source of measurements can be seen as a means of result optimization. This chapter deals with the raw computational performance of the algorithm and its memory demands. We will describe how both can be balanced out and how some geometrical and structural properties can be exploited to achieve the best throughput for great amounts of data.

Partial Distances

The nature of k-Means requires a lot of distance calculations. Each iteration consists of a large number of nearest neighbor queries that grow with the number of measurements and even exponentially with the number of clustering dimensions. Therefore, avoiding unnecessary distance comparisons can save a good amount of CPU time.

Different approaches have been suggested to accelerate the search for the nearest neighbor to significantly improve clustering performance. Examples include using the triangle inequality as described in [Elk03], tree structures like Kd-trees [KMN+02] or R-trees [PF00] that are impractical in higher di-

mensions and finally partial distances [CGRS84]. The latter is a smart method to end a single distance calculation prematurely. It changes the inner core of k-means to use the steps described in algorithm 3.3. It is evident that this technique can save unnecessary calculations whenever the exit condition

Algorithm 3.3 Partial distances algorithm

```
0: d_min = 0
   for i = 1 to N do
      d = 0
      for j = 1 to K do
         d = d + (x_j − c_ij)^2
         if d > d_min then
            Next i
         end if
         Next j
      end for
      d_min = d
      min = i
      Next i
   end for
```

$d > d_{min}$ is fulfilled before the distance calculation has finished. This is possible because the complete Euclidian norm is not needed for a mere comparison calculation since monotony holds for the sum of squares below.

The drawback of this simple method is that time savings are heavily dependant on the currently found d_{min}. If it is large, most of the calculations will still be completed. Again, some approaches have introduced that can precondition the problem by finding a better d_{min} beforehand to lead the algorithm into the profit zone earlier. We will use an elegant method by [AZHHH08] that has evolved from other solutions as described in their studies. It trades memory for speed by saving the previously smallest distance d_{prev} to its nearest neighbor for each data point. Under the assumption that the nearest centroid has moved a small amount since the last iteration, d_{prev} is a good value for d_{min}. The structure of the PD algorithm is therefore changed to mimic the description in 3.4.

As described, we have to assign each measurement to its nearest centroid once initially before the improved method can be used. But this step can still employ usual partial distances, which saves time at the beginning. Generally, it should be noted that this shortcut is most successful for higher dimensions of data, but we will see that it effectively improves performance even in our two-dimensional case.

Parallelism

Exploiting parallelism within algorithmic structures is a powerful tool to multiply the throughput of a calculation. We will examine our possibilities in two stages.

Algorithm 3.4 Optimized partial distances algorithm

0: run PD (initial step using PD as before)
  $d_{min} = d_{prev}$
  for $i = 1$ to N **do**
    $d = 0$
    for $j = 1$ to K **do**
      $d = d + (x_j - c_{ij})^2$
      if $d > d_{min}$ **then**
        Next i
      end if
      Next j
    end for
    $d_{min} = d$
    $min = i$
    Next i
  end for

Outer Structure The more obvious parallelizing task concerns the X-Means-like structure on top of the k-Means blocks within the algorithm. We have shown in 3.3.2 that the method re-runs itself on parts of the data again and again without any global information needed for each step down the tree. Therefore, it can be written recursively as:

Algorithm 3.5 ClusterRecursive(D)

0: $C_0 \leftarrow D$
  Spawn C_1 and C_2
  Run 2-Means
  if Criterion(C_1, C_2, C_0) = true **then**
    return $C \leftarrow$ ClusterRecursive(MembersOf(C_1)) $\cup$ ClusterRecursive(MembersOf(C_1))
  else
    return $C \leftarrow C_0$
  end if

The transition from recursive to multithreaded is then trivially achievable by spawning new threads for each call of the recursive function. A barrier needs to be implemented afterwards that puts the thread on hold until its children have returned their results. The parallel version of the outer structure consequently has the form of algorithm 3.6.

This will cause the number of running threads to grow exponentially with the number of splits and enables multicore systems to utilize their potential. However, the effect only kicks in after the first few splits have occurred, which is a major drawback because the first stages of the algorithm operate on the largest numbers of data points and therefore offer the best gain when parallelized. Luckily, we do not have to treat the inner k-Means block as atomic with respect to parallelization.

Inner Structure Any k-Means-based algorithm is parallelizable at its very core. This property rises from the fact that the most time-consuming operations within the algorithm are completely independent from each other. Finding the smallest distance to any centroid for a given measurement has no influ-

Algorithm 3.6 ClusterParallel(D)

0: $C_0 \leftarrow D$
Spawn C_1 and C_2
Run 2-Means
if Criterion$(C_1, C_2, C_0) =$ true **then**
  $Thread_1 \leftarrow$ ClusterParallel(MembersOf(C_1))
  $Thread_2 \leftarrow$ ClusterParallel(MembersOf(C_2))
  return $C \leftarrow$ ResultOf($Thread_1$) $\cup$ ResultOf($Thread_2$)
else
  return $C \leftarrow C_0$
end if

ence at all on the answer to the same question for another data point. Shared memory access during these operations is read-only or can be organized in a way that cross-writing can be avoided. Simply distributing the for-loop calculating these distances among multiple threads almost divides calculation time by their number[1]. The changed structure is made aware of the count of participating threads and the number of the current thread running and automatically determines the indices of data points falling into its part of the work.

Employing inner parallelism greatly enhances multicore performance of the whole algorithm, especially during the first splits. The method is able to provide the architecture with a sufficient number of threads to utilize a modern system's full capacity and achieve a respectable throughput on large data sets. Measurements of the algorithm's raw performance in different configurations are presented in 4.1.

3.3.4 Bundle Extraction

Now that we have introduced our implementation of the clusterer, we will show which strategies can be applied to use it as a powerful building block to find the bundle structures within our pedestrian movement data. Let us first define the characteristics of a bundle.

It should be

- grouping all pedestrian paths that would be humanly classified as "going the same way" or could be seen as leading to a trail on softer ground.
- consisting of those tracks that lead from similar beginnings to similar ends.
- of limited thickness to be able to represent its trail as a single trajectory with limited standard deviation.
- one-directional only.

The following pages will present two different clustering-based strategies to find these properties in our measurements and a third one that synthesizes both into a hybrid approach.

[1] Notice that APIs like [Mic10] let the developer define a loop as parallel and handle threading themselves.

Centroid Sequence Tracking

The idea behind our first strategy is that typical movements within the scene will lead to areas of higher measurement density, no matter which trajectories were involved. The whole system is based on the assumption that these characteristic paths through the scene exist and therefore there must be a resulting density distribution that can be modeled using clustering.

As our clusterer autonomously determines a meaningful number of partitions and positions its centroids in a meaningful center of gravity within each cluster (as described in 3.3.2), it is able to put out the points that characterize the whole scene by clustering two-dimensional coordinates of the data. We can then treat them as nodes to build a graph of paths taken, simply by finding the ordered sequence of clusters that each trajectory has contributed measurements to. Trajectories of equal cluster sequence can then be grouped and rightfully be called a bundle by the standards defined earlier.

The correct sequence is found by considering the trajectory's first measurement's coordinates and from there iterating through the list of clusters that have reported to contain its traces, always choosing the nearest neighbor. This ensures proper comparability between different sequences in the later process.

A major benefit of using cluster sequences is the trivial detection and distinction of movement directions. Usually clustering would have to consider the direction angle as a third dimension to separate trajectories moving along similar paths but with opposing orientation. Using our approach this is reduced to an efficient comparison of ordered sequences that is far less prone to noise than the direction of the velocity vector saved for each measurement.

Early tests have shown that this exhaustive search approach is fast enough in relation to the clustering process. Note that it is also independent from the starting point of its search as opposed to possible back-tracking algorithms that could be found. The examination of performance optimization potentials or alternatives to this technique is therefore not in the focus of this project.

Simplification Methods Still there is a drawback of this simple method that spoils the results' usefulness for the overall system. Perfect sequence equality is a very strict condition that leads to far more bundles being separated than necessary. Examine the following scenario: In a corner situation, three clusters A, B and C have been found that are arranged in an L-shape. Now there are trajectories that have traces in all three clusters, going from A to B to C and then there are others who go directly from A to C without actually diverting much from the course of the others - they rather take a small shortcut.

It is obvious to the human eye that these trajectory sets should be called a single bundle. Yet the algorithm finds two sequences of clusters that have been taken. This behavior of the clusterer in forking points is hard to avoid and therefore we need to solve this problem in the meta-logic. This is why we introduce a similarity measure for cluster sequences to soften the perfect equality criterion for grouping them. Different methods have been examined out of which an approach from the field of determining string differences has been chosen as a promising base: the formerly introduced Damerau-Levenshtein distance.

However, this does not hold as a good bundling criterion without changes, because it just counts the edit operations needed to turn a given sequence into another one without taking their length into account. We have fixed this by dividing the outcome by the two sequences' mean length. Using this to decide whether or not to join two bundles by checking for a fixed threshold results in some similar bundles being joined but still not all of them. Performance has been improved by awarding bonus if the first or last cluster passed has been equal. This forms the following criterion for sequences a and b:

$$\text{join sequences iff: } \frac{d(a,b) - \beta(a_0,b_0) - \beta(a_{|a|},b_{|b|})}{\frac{|a|+|b|}{2}} < t \tag{3.3}$$

with

$$\beta(x_i,y_j) = \begin{cases} 1 & \text{if } x_i = y_j \\ 0 & \text{if } x_i \neq y_j \end{cases} \tag{3.4}$$

The threshold t can be fixed for all scenes and gives a measure of the allowed relative divergence of the two sequences tested.

End Point Clustering

The positive effects of specifically considering the end points on the simplification process of cluster sequences found (and therefore on bundle extraction itself) have motivated a different strategy. Their equality has proven to be a strong statement about similarity in pedestrian movement. Usually, a scene could be defined by a number of entry and exit opportunities. The amount of persons entering and leaving the area can be considered equal (with very limited room for exceptions), so the different binary links between entry and exit points could very well define all characteristic groups of paths taken through the scene. The appeal of this technique is increased by the observation that most scenes show high densities in these areas as entering or exiting the scenario often involves some form of bottleneck (like a staircase or sidewalk not as wide as the crossing area).

Assuming this we have changed the input to our clusterer to hold only start and end points of all trajectories recorded. It should then determine the correct number of entry and exit points (without considering the difference) and provide us with a partitioning of the end points. We can then easily find the binary cluster sequences belonging to the trajectories and group them by exact equality.
This method has proven very powerful for our input data as it can be shown that pedestrians going from the same start to the same end direction stick to very similar movement paths during the way. It is resistant against minor deviations along the way and tends to rather undershoot than overshoot the number of bundles. Problems arise when trajectories end in unexpected places due to the tracking breaking off or if the density assumption fails to hold and end points are scattered across larger areas. However, the most damaging case is the one where there are two or more distinct bundles leading from the same start to the same end point. Such pedestrian behavior can be caused by environmental objects like refuge islands in the middle of the street and should be covered by our algorithm.

Hybrid Approach

A strategy to eliminate this drawback has been found in the combination of the prior two. As end point clustering operates on a very small fraction of the measured data, it needs very little time to calculate. Therefore, we use it as a first step to find all groups characterized only by similar entry and exit points. To avoid disregarding distinct bundles within such a link we then cluster again using the centroid sequence tracking method which will find splits along the way that fail to obey the similarity criterion defined in formula 3.3.

3.3.5 Trajectory Outlines

As discussed earlier bundle extraction is only one part of the expected work of our algorithm. Having found the groups of paths within our scene we want to construct a polygon graph that roughly represents the common form of each bundle and complies to the constraints of the following isolation stage as described in 3.3.1.

One could argue that our centroid sequence tracking approach actually produces such outlines and it is able to in many cases. However, experience has shown that clustering yields much better results if the input data is already more sharply defined rather than scattered. The amount of clusters generated and the quality of adaptation of the sequence to a bundle's shape increases significantly when clustering only one prefiltered group.

As with end point clustering, the size of the input set for bundle-internal clustering is reduced and the computational effort is small compared to the base set. Therefore, it is reasonable to add a further clustering step to our algorithm and enhance the resulting outlines significantly. Clustering within a bundle should in nearly all cases result in clusters that look like the base set was orthogonally cut into slices. The distance split criterion is designed to enforce that no two clusters ever lie besides one another, which avoids zigzag schemes of the outline later.

Like in centroid sequence tracking, the correct sequence for the outline is found using a nearest neighbor scheme beginning from a trajectory's first measurement. The assumptions made and the constraints held by the previous steps guarantee the construction of a valid sequence. Examples of outlines generated with different values for the distance criterion can be found in figures 4.1 and 4.2.

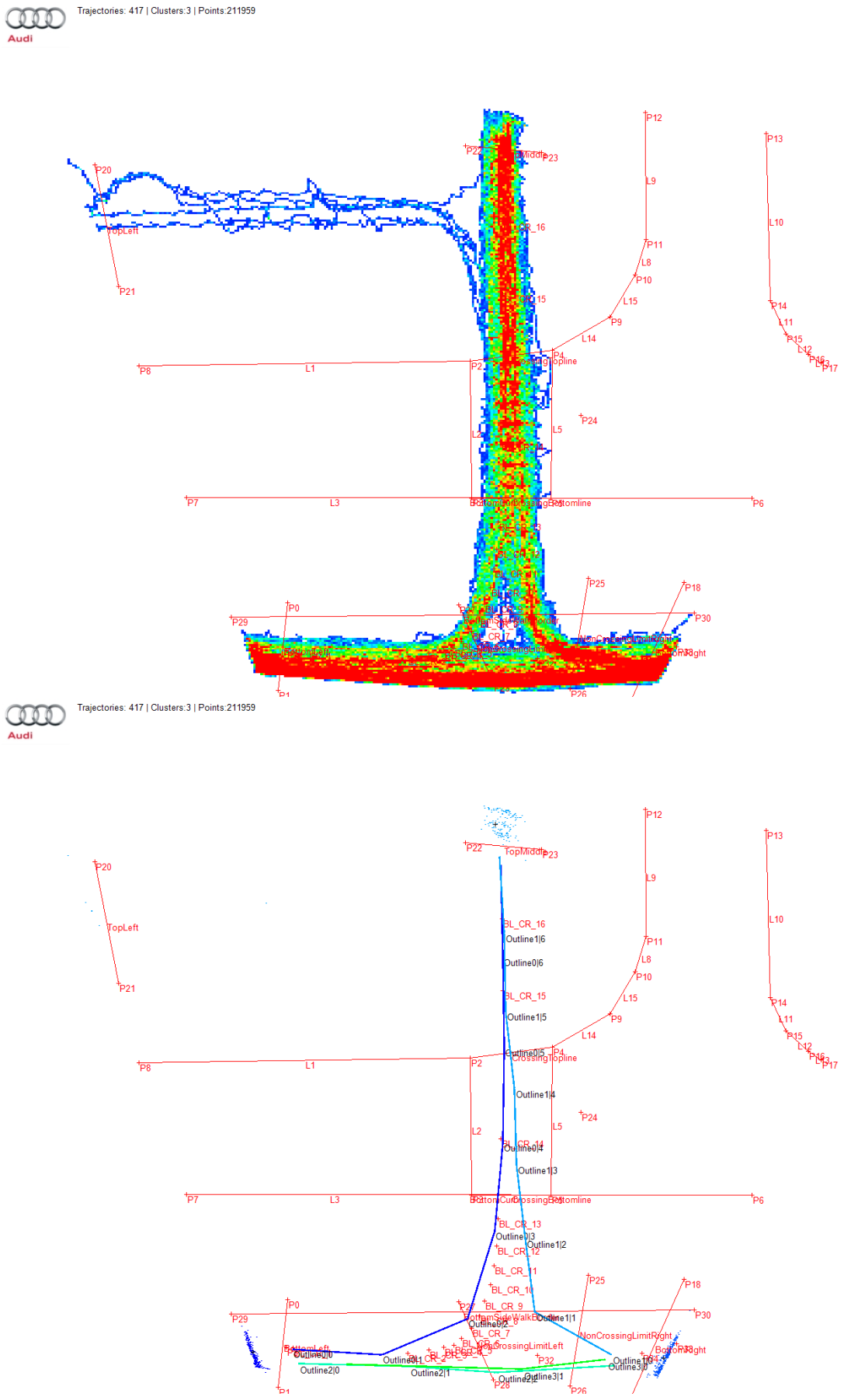

Figure 3.3: Hybrid bundle extraction performed on the Ingolstadt scenario

4 Performance Results

To evaluate the algorithms developed in the course of this thesis, they have been tested for computational speed and performnce regarding their goal. The data presented in the following sections has been measured on a PC equipped with two Intel Xeon quadcore CPUs and 8GB of system memory. The underlying data is taken from the Kaiserslautern and Ingolstadt scenarios.

4.1 Calculation Time Comparison

The performance data in 4.1 has been measured by clustering two data sets of different sizes with different combinations of switches for the clusterer. Outer threading and the use of partial distances have been enabled or disabled and the number of threads used to distribute the inner distance calculations has been changed. Each combination has been repeatedly calculated 20 times to average out the non-deterministic outcomes caused by random centroid spawning. The average cluster count determined is given as a median, but has been calculated arithmetically into the average time needed per created cluster. This value is presented to honor the fact that more split decisions generally result in a higher total calculation time. All times are given in seconds.

Data Points	Outer threading	Inner threads	Avg. time	Median clusters	Avg. time per cluster
166409	enabled	16	38615.3	4	9267.29
166409	enabled	8	37393.05	4	9606.01
166409	enabled	1	46829.7	3	14493.83
166409	enabled	1, **PD disabled**	51783	3	16452.76
166409	**disabled**	1	141122.85	3	46559.98
290659	enabled	16	147669.1	6	24309.41
290659	enabled	8	193923.2	6	31785.71
290659	enabled	1	145149.1	3	49905.84
290659	enabled	1, **PD disabled**	201804.85	3	57677.15
290659	**disabled**	1	768795.95	4	199857.37

Table 4.1: Computational performance measurements on two data sets

4.2 Split Behavior

Nominal distance $d_{nominal}$	Median cluster count	Average distance from master
3.0	16	0.0940
4.0	11	0.1131
5.0	9	0.2101
6.0	8	0.1431
7.0	7	0.2396

Table 4.2: Split behavior measurements, first set (40 trajectories)

For tables 4.2 and 4.3 two bundles of crossing pedestrians have been clustered with different values (in meters) for our distance split criterion. Each setting has been applied 9 times as base for the given averages. The distance value presented is the arithmetic average of the mean distance between corresponding samples of the master trajectory for the bundle and the calculated outline resampled to match the master trajectory's sample count.

Nominal distance $d_{nominal}$	Median cluster count	Average distance from master
3.0	24	0.1286
4.0	21	0.2284
5.0	13	0.3248
6.0	11	0.2437
7.0	8	0.3779

Table 4.3: Split behavior measurements, second set (80 trajectories)

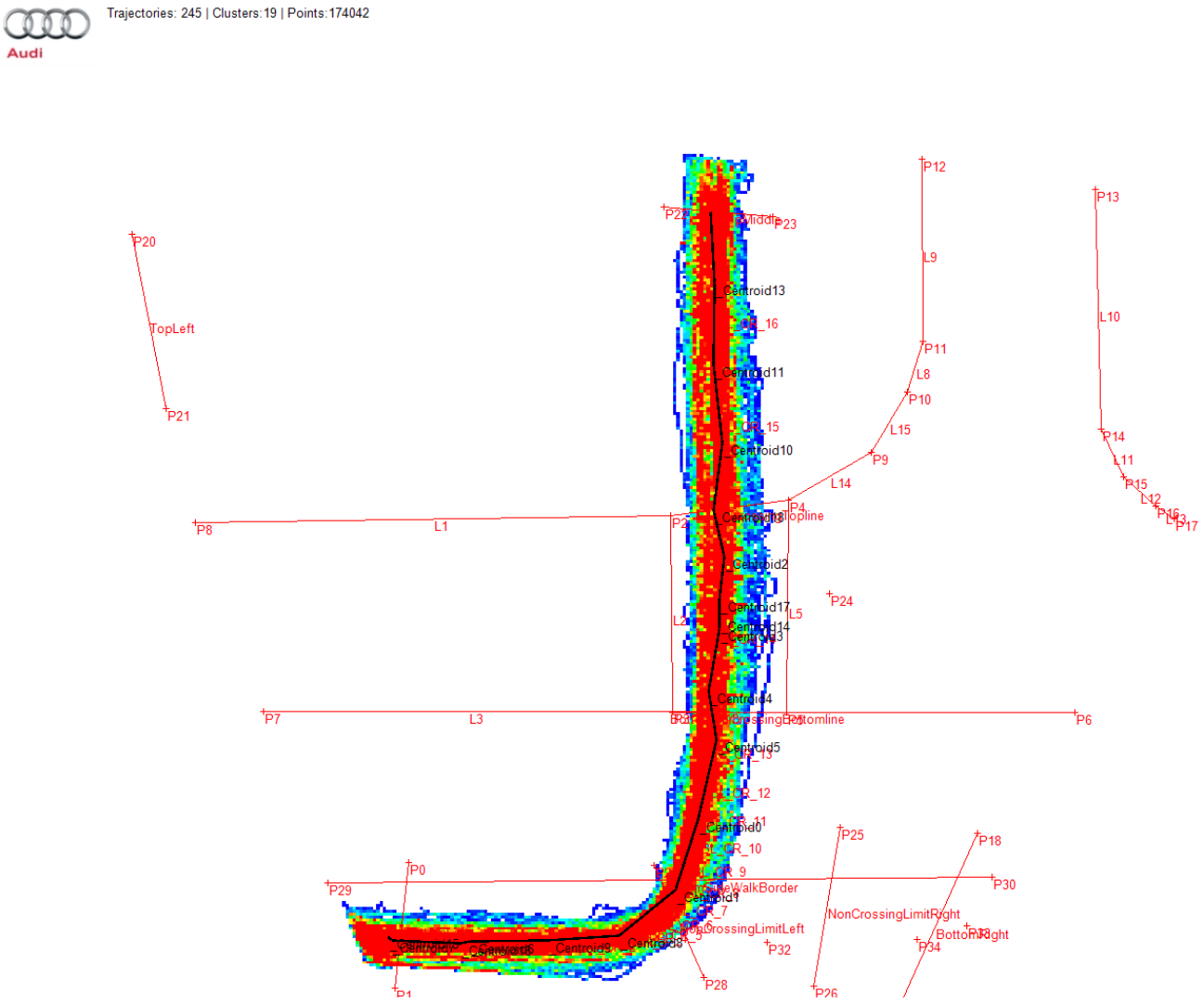

Figure 4.1: Outline (black) for a bundle generated with $d_{nominal} = 3.0$ (Ingolstadt)

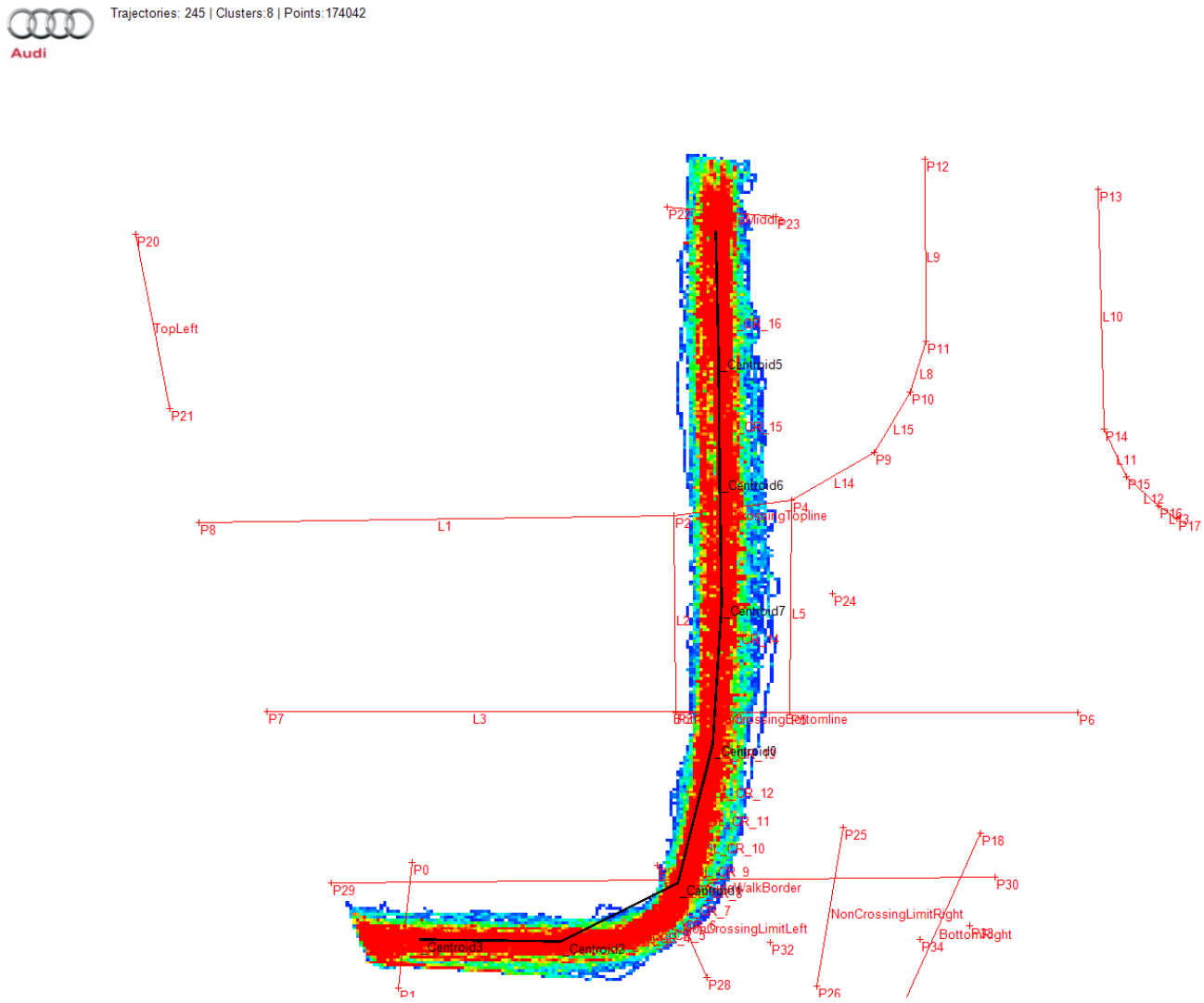

Figure 4.2: Outline (black) for a bundle generated with $d_{nominal} = 5.0$ (Ingolstadt)

5 Conclusion

In the context of an advanced driver assistant system we have tackled a fundamental problem in the field of data mining: the cluster analysis of traces of connected data points by a similarity measure. In this chapter, we will discuss the qualitative achievements of this thesis and evaluate the quantitative analysis of our method.

5.1 Practical Achievements

We have shown a concept for the use of classical clustering algorithms as a building block to cluster a data set that contains link information. As planned, we have found ways to isolate the main directional bundles within our measured trajectories and generate a representative outline to be used in following stages of the overall system. This has required an in-depth look into the discipline of data mining and the analysis and comparison of different algorithms to inspire our own approach. The paragraphs below cover some of the lessons learned during this project and give examples of their application.

Domain Knowledge

Even though it is desirable for a method to be general and universally valid, never forget that solving the problem at hand with the information you are given is the main goal of the project. From the example of split criteria we can see that knowing the shape of the data in advance can motivate the rejection of established research and lead to simpler yet in this domain more powerful logic. Statistical tests like Anderson-Darling may have proven useful in astronomic applications and other areas with Gaussian data distribution, but they seem to ask the wrong questions regarding bundled trajectories. However, we have been able to define our resulting criterion in a way that renders it valid outside the borders of our scenario as the balance factor required carries a domain-specific meaning and can remain constant across different instances of the same problem.

Dimension Reduction

As the studies of G-Means [HE03] have taught us, the key to apply complicated calculations on large sets of data can be the reduction of the problem to a lower dimension. At first glance, clustering our data for similar movements is a three-dimensional problem of latitude, longitude and direction. But cluster analysis on those three features does not only lead to the problems of normalization and noisy direction vectors, it also comes with a major performance drop. Therefore, our suggestion of using end point clustering and centroid sequence tracking simplifies the procedure by avoiding calculations and sensor errors without losing precision in bundle recognition.

Parallelism

Performance and memory optimization are often considered a refinement and placed in the "tidy up" section of projects in science and business environments. Our experience during this study tells us that this should be considered bad practice in data mining or applications of similar input complexity. Optimizing algorithms for speed and space consumption early leads to a far better overview of their usefulness on larger data sets and the ability to test their performance without simplifying reality too much. Especially the identification of parallelization opportunities within the structure of the chosen methods is always worth a closer look and the effort of implementation.

5.2 Evaluation of Performance Results

5.2.1 Computational Performance

The data presented in table 4.1 shows the benefits of performance optimization by exploiting parallelism. We see that an increasing number of threads used results in significantly lower calculation time up to the point of saturation because of the increasing scheduling costs for the operating system. We can also see the better impact of increasing thread counts on larger data sets because of a better ratio between the amounts of work and scheduling costs (23.5 percent speed-up from 8 to 16 inner threads compared to 3.5 percent on the smaller data set). The speed-up by using partial distances averaged at 12.7 percent in the given tests for a single distance calculation thread.

5.2.2 Split Behavior

The values from tables 4.2 and 4.3 show an inversely proportional relationship between the nominal distance and the number of clusters suggested by the algorithm. We can see that from the tested values a nominal distance of 3m results in the lowest divergence to the master trajectory. Higher values generally lead to higher mean distances, even though the algorithm might also be lucky with a lower cluster count if the centroids are well-positioned. Future studies could include a measure of the angles inside the outline to find a good ratio between mean distance and zig-zag patterns.

5.3 Future Potential

During this thesis we have touched on many subjects from the fields of traffic safety, human behavior, movement tracking and data mining. However, the techniques implemented and presented here are by no means exhaustive as time constraints have prevented many chances of further research directions that have occurred to us. Some promising additions are introduced in this section to inspire future studies.

5.3.1 Clustering in a Space of Trajectories

While clustering in two dimensions only and finding bundles by centroid tracking or end point similarity yields benefits and simplifies subtasks, it still remains a two-staged approach that ignores link information once and applies it later on a much coarser level. If an appropriate technique was found to cluster complete trajectories, it could have a great impact on computational effort and especially on

the quality of the results. An approach based on k-Medoid seems reasonable, since there is no trivial solution to build a mean trajectory inside a bundle. The success of cluster analysis in a trajectory space comes down to the choice of an effective similarity measure.

Possible Distance Measures

To be meaningful, the distance measure chosen should fulfill the basic requirements for distances as specified in 2.3. We will examine two of the more promising candidates.

Surface between Trajectories A possible indicator for similarity among different trajectories could be based on the surface spanned between them. As with integral methods for functions, the distance becomes zero when both images are of equal shape. The big advantage of using the surface is the lack of necessity to resample the trajectories. Instead, simple triangle-based techniques from the field of 2D/3D visualization can be applied directly on the data points. To avoid equal distance scores for the two cases of both trajectories running in parallel at a given distance and both of them running in opposite directions half of the time and exactly on top of each other afterwards, we could incorporate the variance over the individual triangle sizes. Low variance would be a strong indicator for similar path shape. It might also solve the problem of determining the walking direction among similar shaped paths.

Gesture Recognition Methods The problem at hand could also be seen from the perspective of gesture recognition, as established methods from this field should be applicable to trajectories and be able to evaluate shape similarity. A simple technique could split the trajectory into equidistant pieces and assign each piece to a direction class, e.g. by rounding it into 45 °C steps. The trajectories can then be compared by the editing distance of their sequences of direction classes. This way similar shape results in low distances and the movement direction is clearly distinguishable, but both paths could be displaced by a large distance within the scene. Therefore, we would need to incorporate a translation distance factor to determined significant shift differences on both axes. For simplicity, considering endpoints only is sufficient to solve the problem as shape similarity dictates similar distances along the way.

5.3.2 Genetic Algorithms for Sequence Distances

If we stick to our approach of clustering data points without link information, the following step of centroid sequence tracking could be improved by replacing the Damerau-Levenshtein distance with algorithms from the field of genetic sequence alignment. Distances calculated by techniques such as Needleman-Wunsch [NW70] or Smith-Waterman [SW81] are based on a penalty matrix that awards different scores for edit operations depending on the elements involved. In our case, this matrix could be constructed to mirror the physical distances between centroids to avoid joins of severely different paths that look similar to the current algorithm because they only differ in one edit.

List of Figures

Acknowledgement

This project work has been achieved while participating in a skilled and motivating environment within the offices (and cars) of Audi Electronics Venture GmbH, a sibling and department of AUDI AG's headquarters in Ingolstadt, Germany. The modeling and implementation of this task is my respectful contribution to the efforts of my restless team in AEV's driver assistant systems cluster.
Therefore, my primary expression of gratitude is devoted to my tireless team leader *(may I say Dr.?)* Martin Roehder for his trust, input and motivation along the way. You have been a true mentor from day one and it has been an honor to work on this project alongside you. This summer has been so much more than just an internship, thank you.
Special thanks to my dear teammate Dipl. Ing. Sean Humphrey for countless discussions, razor-sharp questions and his inspiring work attitude. Please, get some rest, buddy...it is well-deserved.
Many thanks to the rest of the people at AEV-3, especially Dr. Ing. Björn Elias and Dr. Ing. Björn Giesler for their interest in my work and a lot of bright moments on and off the job, Christina Gackstatter and Jessica Reinhold for many shared laughs in our office and, of course, to Andreas Reich for letting me share his playground. A shout out to my student colleagues doing their theses within AEV, including Florian Schmidt, Michael Keller, Jörg Mohrig and Andreas Lehmann. I hope I will see each one of you again and wish you all the best until that happens.
I need to thank Professor Dr. Ralf Möller for providing me with the freedom of his kind remote supervision and with much of the basic logic applied during my work. Organization of projects like this becomes so much easier when there are people involved who share the enthusiasm to explore different working environments.
Last but not least, I want to express my gratitude to my family, especially to my little sister Micky for relentlessly making sure that this thesis meets its deadline.

Bibliography

[AZHHH08] AL-ZOUBI, Moh'd B. ; HUDAIB, Amjad ; HUNEITI, Ammar ; HAMMO, Bassam: New Efficient Strategy to Accelerate k-Means Clustering Algorithm. In: *American Journal of Applied Sciences* 5 (2008), S. 1247–1250

[CGRS84] CHENG, D. ; GERSHO, B. ; RAMAMURTHI, Y. ; SHOHAM, Y.: Fast Search Algorithms for Vector Quantization and Pattern Recognition. In: *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing* 1 (1984), S. 1–9

[Cor97] CORSO, Gianna M. D.: Estimating an eigenvector by the power method with a random start. In: *SIAM Journal on Matrix Analysis and Applications* 18 (1997), S. 913Ű937

[Dam64] DAMERAU, Frederick J.: A technique for computer detection and correction of spelling errors. In: *Communications of the ACM* (1964)

[Ele10] ELEKTROBIT AUTOMOTIVE GMBH (Hrsg.): *Automotive Data and Time triggered Framework*. Elektrobit Automotive GmbH, 2010. `http://www.elektrobit.com/download/1613/datasheet_eb_assist_adtf_dt_screen`

[Elk03] ELKAN, C.: Using the Triangle Inequality to Accelerate k-Means. In: *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)* (2003), S. 609–616

[Gol] GOLLEWSKI, T.: *Integrale Sicherheit - Herausforderungen, aktuelle Funktions- und Auslegungskonzepte aus Sicht eines OEMs*. Hanau-Steinheim, Integrated Safety

[Ham50] HAMMING, Richard W.: "Error detecting and error correcting codes". In: *Bell System Technical Journal* 29 (1950), S. 147–160

[HE03] HAMERLY, Greg ; ELKAN, Charles: Learning the K in K-Means. In: *"Learning the K in k-means", NIPS*, 2003

[IIH09] IIHS ; U.S. DEPARTMENT OF TRANSPORTATION: FATALITY ANALYSIS REPORTING SYSTEM (Hrsg.): *Fatality Facts*. U.S. Department of Transportation: Fatality Analysis Reporting System, 2009. `http://www.iihs.org/research/fatality_facts_2009/pedestrians.html`

[Inf10] INFORMATION SOCIETY AND MEDIA (Hrsg.): *eCall - saving lives through in-vehicle communication technology*. European Commission: Information Society and Me-

dia, July 2010. http://ec.europa.eu/information_society/doc/factsheets/049-ecall_july10_en.pdf

[Keo] KEOGH, Eammon: *Clustering*. TUHH, (Lecture Slides for Introduction to Machine Learning and Data Mining)

[KMN+02] KANUNGO, T. ; MOUNT, D.M. ; NETANYAHU, N. ; PIATKO, C. ; SILVERMAN, R. ; WU, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 24 (2002), S. 881–892

[Lev66] LEVENSHTEIN, V. I.: Binary codes capable of correcting deletions, insertions, and reversals. In: *Soviet Physics Doklady* 10 (1966), S. 707–710

[Mic10] MICROSOFT (Hrsg.): *.NET Framework 4*. Microsoft, 2010. http://msdn.microsoft.com/de-de/netframework/default.aspx

[NHT08] NHTSA ; NHTSA NATIONAL CENTER FOR STATISTICS AND ANALYSIS (Hrsg.): *Traffic Safety Facts*. NHTSA National Center for Statistics and Analysis, 2008. http://www.walkinginfo.org/facts/docs/PedTSF_2008.pdf

[NW70] NEEDLEMAN, S. ; WUNSCH, C.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. In: *Journal of Molecular Biology* 48 (1970), March, Nr. 3, 443–453. http://dx.doi.org/10.1016/0022-2836(70)90057-4. – DOI 10.1016/0022–2836(70)90057–4. – ISSN 00222836

[PF00] PROIETTI, G. ; FALOUTSOS, C.: Analysis of Range Queries and Self-spatial Join Queries on Real Region Datasets Stored using an R-tree. In: *IEEE Transactions on Knowledge and Data Engineering* 5 (2000), S. 751–762

[PM00] PELLEG, Dau ; MOORE, Andrew: X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: *In Proceedings of the 17th International Conf. on Machine Learning*, Morgan Kaufmann, 2000, S. 727–734

[RSK10] RESSLE, A. ; SCHRAMM, Stefan ; KÖLZOW, Thorsten: Generierung von Verletzungsrisikofunktionen für Fussgängerkollisionen. In: *crash.tech* (2010)

[Ste74] STEPHENS, M. A.: EDF Statistics for Goodness of Fit and Some Comparisons. In: *Journal of the American Statistical Association* 69 (1974), Nr. 347, 730–737. http://dx.doi.org/10.2307/2286009. – DOI 10.2307/2286009. – ISSN 01621459

[SW81] SMITH, T. F. ; WATERMAN, M. S.: Identification of common molecular subsequences. In: *Journal of Molecular Biology* 147 (1981), Nr. 1, 195 - 197. http://dx.doi.org/DOI:10.1016/0022-2836(81)90087-5. – DOI DOI: 10.1016/0022–2836(81)90087–5. – ISSN 0022–2836

[TSK05] *Kapitel* 8. Cluster Analysis: Basic Concepts and Algorithms. In: TAN, Pang-Ning ; STEINBACH, Michael ; KUMAR, Vipin: *Introduction to Datamining*. Addison-Wesley, 2005

[WHO09] WHO ; WORLD HEALTH ORGANIZATION EUROPE (Hrsg.): *European status report on road safety - Towards safer roads and healthier transport*. World Health Organization Europe, 2009. `http://www.euro.who.int/__data/assets/pdf_file/0015/43314/E92789.pdf`