Lossless Arithmetic Coding of Discrete Vehicular Trajectories

Masterarbeit

von

Matthias Radig

aus

Düsseldorf

vorgelegt am

Lehrstuhl für Rechnernetze und Kommunikationssysteme
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

März 2013

Betreuer:
Dr. Markus Koegel

Zusammenfassung

Die Übertragung und Speicherung von Fahrzeugbewegungsdaten via Positionsmessungen in regulären Intervallen führt zu einer großen Redundanz, da Fahrzeuge physikalischen Barrieren unterliegen, die die möglichen Bewegungen stark einschränken. In einer vorhergehenden Arbeit wurde ein verlustbehaftetes Kompressionsverfahren vorgestellt, das die Redundanz dieser diskreten Trajektorien erkannte und eine effiziente arithmetische Kodierung der Daten ermöglichte. In dieser Arbeit werden die Prinzipien des verlustbehafteten Kodierers auf den verlustfreien Fall übertragen. Dies beinhaltet unter anderem die Analyse und Lösung von Problemen, die durch die Verwendung eines geodätischen Koordinatensystems entstehen, das verwendet werden muss, um Informationsverlust zu vermeiden. Zudem werden verschiedene Wahrscheinlichkeitsmodelle für den arithmetischen Kodierer vorgestellt, die unterschiedliche Anwendungsfälle abdecken und hohe Flexibilität bieten, um Kompromisse zwischen Kompressionsrate und Nutzung von Laufzeitmitteln sowie Anzahl benötigter Stichproben zu finden. Die Modelle werden mit Hilfe von Realdaten aus dem OpenStreetMap-Projekt getestet. Die Ergebnisse zeigen, dass die mit unserer Methode kodierten Trajektorien um etwa 20% kleiner sind, als mit den derzeit besten anderen verlustbehafteten und -freien Kompressionsverfahren kodierte.

Abstract

Tracking the movement of vehicles by transmitting or storing their position in regular intervals results in a great amount of redundancy, since vehicles are bound by physical barriers that greatly limit the possible movements. In a previous work, a lossy compression scheme was designed that exposed the redundancy in such discrete trajectories and provided an efficient arithmetic code for them. In this work, the principles of the lossy coder are translated to the lossless domain. This includes the analysis and solution of challenges introduced by the usage of a geodetic coordinate system, which is required to avoid information loss. We furthermore introduce different probability models for the arithmetic coder that cover different use cases and offer great flexibility in terms of resource consumption and required sample data vs. compression. These models are evaluated using real-world data from the OpenStreetMap project. The results show that the discrete trajectories encoded with our scheme are about 20% smaller than the encoding of the best existing lossy and lossless schemes.

Acknowledgments

I wish to thank the people who supported me during the creation of this thesis: Dr. Markus Koegel for his advice and guidance; Kristin Westermann for her encouragement during stressful times and her financial support that allowed me to focus completely on the thesis; and René Bartelmus for providing additional proof-reading and helpful suggestions. I also thank Dr. Koegel, and all the other people who contributed, for the research that lead to the lossy compression scheme which this work is based on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There are numerous systems that measure, transmit, and store positional data of vehicles and other moving objects. Examples include On-Board-Units that track the movement of vehicles for billing purposes or databases that store large amounts of movement traces for the creation of maps. The required financial and technical upkeep of such systems depends, among other things, upon their consumption of storage space and network traffic. It is therefore beneficial to compress this positional data as much as possible.

A *trajectory* is the path of a moving object. It describes the geometry of the motion and does not depend on the speed of the object. A *discrete trajectory*, or *trace*, is a series of sequential, related position measurements. The real, continuous trajectory of an object is converted into a discrete one by sampling. This yields a sequence of points that are a subset of the continuous trajectory. For the scope of this work, a discrete trajectory is expected to have a constant sampling rate, i. e., the time between individual measurements is always the same and no measurements are omitted. This allows to estimate the speed and acceleration of the object in addition to its position and direction. We focus on vehicular movements on the surface of the earth. Such movements are usually measured with GPS or similar methods, using a geodetic coordinate system with latitude and longitude.

The primary goal is to specify a lossless compression scheme based on arithmetic coding for series of geodetic coordinates, and provide an accompanying proof-of-concept implementation to verify the viability of the scheme on real-world data. This requires us, before anything else, to consider what the input of such a coder should be; specifically what values are possible and how they can be represented in a computer program. Additionally, an arithmetic coder needs a probability assigned to each value. The efficiency of the code is determined by how accurately those probabilities reflect reality. For this reason, we propose different probability models, discuss their use cases, and compare their compression performance. There are two secondary goals:

First, the developed coding scheme should be scalable to small, embedded devices. These devices may have limited storage space, memory, and computational power. To be used on a variety of different device types, the algorithm should provide the possibility to sacrifice compression performance for lower resource consumption.

Second, the proof-of-concept implementation that is part of this work should not use any floating-point arithmetic[1]. Obviously, any rounding operation performed on the input data will in general result in information loss. Apart from the input data, we will work with probabilities. Because these probabilities only determine how coordinates are mapped to the encoded stream and vice versa, rounding operations are safe as long as they are identical during encoding and decoding. Unfortunately, that is not true for floating-point arithmetic either, as the encoding and decoding processes might happen on a number of different machines with varying rounding strategies. In essence, we need precise control over all arithmetic operations that are part of the coding process and are therefore restricted to integer arithmetic.

For the scope of this work, we define the term "compression ratio" as

$$\textit{compression ratio} = \frac{\textit{size reduction}}{\textit{uncompressed size}} = 1 - \frac{\textit{compressed size}}{\textit{uncompressed size}} \ .$$

By this definition, the compression ratio is the reduction in size relative to the uncompressed size. Therefore, a higher compression ratio is better. This is a more intuitive metric, especially when plotted against some parameter.

The rest of this thesis is structured as follows: in the first part of Chapter 2, other compression schemes for discrete trajectories are discussed. The second part is a detailed look at the workings of the existing lossy arithmetic coder that our own extends upon. We describe the basic idea behind the existing coding scheme as well as the weaknesses that this work intends to improve upon. In Chapter 3 we discuss the input that the coder gets. After a brief explanation of geodetic coordinate systems, we introduce parameters that define the format of the input data. From those parameters, we formally derive the set of possible values that the individual elements of a discrete trajectory can assume. We call this set the *domain* of the coder. In Chapter 4, necessary information theoretical concepts for the following chapters are explained. Furthermore, examples of their application to the domain of discrete trajectories and a high-level overview of their usage in this work are given. After the domain of the coder is defined and the basics regarding arithmetic coding are covered, Chapter 5 focuses on the core part of any arithmetic coding scheme: the probability distribution of the symbols to be coded. We aim to design flexible probability models that can be trained for a specific use-case with

[1] Floating point arithmetic can of course be used for the parts of the implementation that are not related to the coding scheme, i. e., the calculation of compression ratios etc.

sample data. We propose and evaluate three different models intended to cover a variety of use-cases. Chapter 6 compares the proposed models using real-world vehicular traces collected for the creation of street maps. It features a large section where the parameters of each model are optimized for our sample data. This is not only required for the following comparison of the compression ratios, but allows us to quantify the effects that the individual parameters have. After parameters are chosen for each model, we compare the compression ratios of those different configurations to each other as well as the lossy arithmetic coder and another lossless coder for discrete trajectories. Chapter 7 concludes this thesis. We discuss ideas for future works that extend and improve upon our probability models, and argue that the principles of our coder might be transferable to create coders for purposes outside the vehicular domain. We then end this work by summarizing the most important results and accomplishments.

Chapter 2

Related Work

A number of compression schemes for discrete trajectories have been proposed in earlier works. They can be distinguished by a few key properties: Most of them are true to the original data only up to a certain error threshold. They are therefore lossy compression algorithms. Others are lossless—they do not alter the original data in any way. There are also compression schemes that require the entirety of the data before any single part of a trace can be coded, while others require only a certain subset. Naturally, the algorithms also differ in their runtime complexity as well as difficulty of implementation.

One approach are line simplification algorithms: a discrete trajectory can be imagined as a number of lines, represented by their endpoints. Line simplification seeks to eliminate points that hold only little information. Usually, such algorithms guarantee that the deviation from the original line is below a configurable maximum error threshold ε, with a larger value allowing for greater compression. A prominent example is the algorithm by Douglas and Peucker from [7]: between a start point s and end point e, the maximum distance between the connecting line and any intermediate point p is determined. If it is below ε, the intermediate points are omitted. If there are no intermediate points, no operation is necessary. Otherwise, the process recursively proceeds, trying to omit points between s and p as well as p and e, and so on, until no more points can be omitted. Since the algorithm recursively processes intermediate points, beginning with the first and last one, the complete set of points has to be available before the algorithm can be executed.

More complex geometrical constructs can be used, e. g., cubic splines [9] or clothoids [4, 12], to model the underlying movements with greater flexibility, resulting in increased compression performance.

The algorithms referenced above aim to find a geometric pattern in a complete data set and simplify it using geometrical models. Contrarily, the authors of [10] use a kinematic

model to estimate the result of the next position measurement, given its predecessors. The difference between estimate and actual position, called the *innovation* by Koegel et al., is then arithmetically coded using an appropriate probability model. This lossy compression scheme successively encodes individual measurements, i. e., it can operate on data streams, and therefore, on traces of arbitrary length. It offers the currently best compression ratios.

The lossless compression of discrete trajectories has received less attention. In [6], Diesterhöft proposes an alternative representation of discrete trajectories using byte-aligned difference vectors. This reduces the size of the trajectory and exposes the remaining redundancy such that general-purpose compression algorithms like deflate or Prediction by Partial Matching (PPM) can be used to compress even more. This method yields compression ratios comparable to the results in [10]. In contrast to [10], the coder from [6] uses neither a kinematic model nor a compression algorithm specific to its domain.

2.1 Innovation-based compression schemes

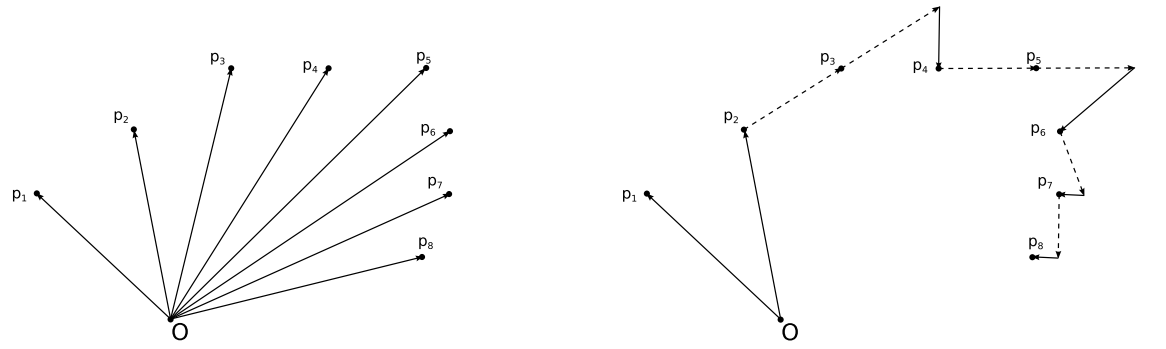
This work extends on the ideas and findings of [10]. The following section provides an overview of the compression scheme developed by Koegel et al. We describe the algorithm as it behaves with optimized parameter values. Koegel et al. evaluated a number of different diverse configurations; for instance, different kinematic models were tried, but the description below considers only the model of constant motion, as it resulted in the best compression ratios.

The coder operates on the individual points of discrete trajectories. It needs only the previous measurements to encode a point, not the complete trajectory. Therefore, it acts as if the discrete trajectory were a data stream. This allows to actually implement a stream-based coder that works efficiently on long traces whose total size would exceed a system's memory. Another use case would be the real-time encoding of a trajectory while the vehicle that is being tracked is still in transit.

In the following, we describe how the algorithm encodes the individual positions of a trace. The first two positions p_1 and p_2 are written uncompressed, as the used kinetic model—that of constant motion—requires the last two positions to calculate a prediction. These positions are referred to as the *initialization vector*. The predicted next position $\mathbf{p'_n}$ is then calculated as

$$\mathbf{p'_n} = \mathbf{p_{n-1}} + \frac{\mathbf{p_{n-1}} - \mathbf{p_{n-2}}}{\Delta t} \cdot \Delta t = 2 \cdot \mathbf{p_{n-1}} - \mathbf{p_{n-2}} \ .$$

The velocity $\frac{\mathbf{p_{n-1}} - \mathbf{p_{n-2}}}{\Delta t}$ is estimated from the previous two measurements, multiplied by the time Δt between measurements (the inverse of the sampling rate), and then added to the



(a) Positions are encoded as vectors relative to the origin

(b) Positions are encoded as sum of previous position (points), next expected offset (dashed arrows) and innovation (solid arrows)

Figure 2.1: Different representations of the same discrete trajectory

latest known position $\mathbf{p_{n-1}}$. The actual position $\mathbf{p_n}$ can be represented by the *innovation*, i. e., the difference $\mathbf{I} = \mathbf{p_n} - \mathbf{p'_n}$. Since the estimate $\mathbf{p'_n}$ is extrapolated from previous data points, it is known by both the encoding and decoding side. Thus, $\mathbf{I}$ contains all necessary information to obtain $\mathbf{p_n}$.

The advantage of using the innovation over the original vector is best demonstrated by an example. Figure 2.1 shows the two different representations of the same discrete trajectory. In Figure 2.1a the vectors are comparatively large and pairwise distinct. Figure 2.1b shows the alternative representation of the same trajectory. The dashed arrows denote the previously described estimated next position, while the solid arrows denote the innovation. Points p_1 and p_2 are the initialization vector, coded as absolute positions, since the kinetic model requires the previous two positions to give an estimate for the next one. All following points can then be represented solely by the innovation vectors, i. e., the solid arrows, which are notably smaller in size than their counterparts from Figure 2.1a. Note that when the measured object moves with constant velocity, $\mathbf{p'_n} = \mathbf{p_n}$ and therefore $\mathbf{I} = \mathbf{p_n} - \mathbf{p'_n} = 0$. Zero vectors are omitted in the image. The most important property of innovation vectors in comparison to absolute positions is their tendency to repeat. Figure 2.1b demonstrates this nicely with points p_3 and p_5, for whom the innovation is zero, and p_7 followed by p_8 that have the same non-zero innovation. This tendency to repeat is easily explained: vehicles have a set of frequent behaviors—accelerating, moving straight-forward, decelerating, taking turns—that result in similar innovations. For example, moving with a constant velocity means that the corresponding innovation is zero, as is the case with point p_3 and p_5. Contrarily, we could expect the same kind of repetition from absolute positions only if the vehicle was driving repeatedly through a circular pattern.

Exposing the redundancy in the data set is only the first step. It still remains to define an efficient code for the innovation vectors. Koegel et al. used an arithmetic code for their efforts. The compression performance of such a code can be almost optimal, given that the underlying probability model is a close approximation of reality. The best of the evaluated probability models is an empirically determined one, i. e., probabilities that were assigned to possible innovation vectors by statistical evaluation of a set of sample traces.

In this work, we will build on the efforts of Koegel et al. to create a lossless compression scheme. This poses unique challenges that are not encountered in the lossy counterpart:

1. The lossy coder works with Cartesian coordinates represented as floating point numbers. The geodetic input data are converted prior to the compression procedure. This conversion involves real-number operations and therefore—on a finite machine—rounding. That cannot be allowed in a lossless compression algorithm. The nature of geodetic coordinate systems and an appropriate binary representation are discussed in Chapter 3. The exclusion of floating point numbers is not discussed in detail—it is simply a technical detail of our implementation.

2. Because rounding operations that affect the data points need to be avoided, the translation of metrical distances in Geodetic coordinate systems is challenging. However, it seems natural to assign the same probability to the same metrical deviation in the domain of vehicles, independent of the current position. Possible solutions are proposed in Chapter 5.

Chapter 3

The Domain of the Coder

3.1 The Coordinate System

To design a compression scheme for discrete trajectories it is imperative to consider the properties of the coordinate system used for position measurements. Vehicular movements are most commonly measured using a geodetic coordinate system. Such systems usually use two angles, latitude φ and longitude λ, to indicate the horizontal position in parallel to the surface of the earth. A third component indicates the elevation, i. e., the vertical position. As vehicles usually move along the surface of the earth, our coding scheme focuses on latitude and longitude.

Given a point P, ISO 19111 [3] defines the latitude φ to be the angle from the equatorial plane to the perpendicular through P and the longitude λ as the angle from the prime meridian plane to meridian plane of P (Figure 3.1). For coordinates to be meaningful, a *geodetic datum* is required that positions the coordinate system in relation to the earth and usually includes a *reference ellipsoid* to model the earth's surface. For instance, GPS uses the World Geodetic System 1984 (WGS 84) [2].

One aspect that will become important in Chapter 5 is the metrical irregularity of the system: the metrical distance travelled on the surface of the earth when deviating by one unit of longitude depends on the latitude. As demonstrated in Figure 3.2, when the latitude increases, the length of units of longitude on the surface of the earth decreases.

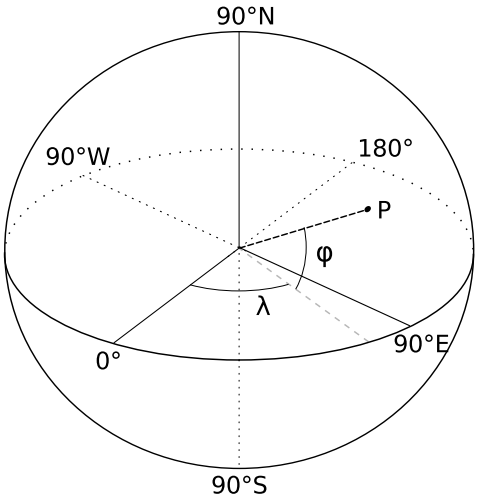
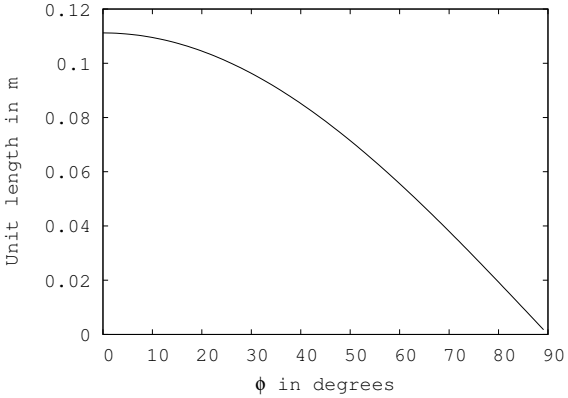
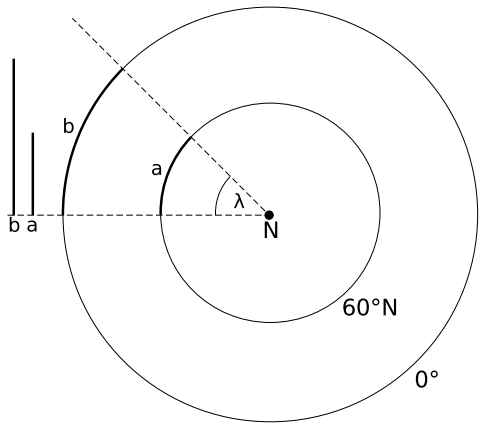


Figure 3.1: Latitude φ and longitude λ of a position P on the surface of the earth.



(a) The length of a line projected on the surface of the earth by one unit of longitude, on different circles of latitude. In this case, one unit of longitude refers to an angle of 10^{-6} degrees.



(b) Arc lengths on different circles of latitude for the same longitude λ, as seen from a top-down view.

Figure 3.2: The covered distance on the surface of the earth when deviating by one unit of longitude depends on the latitude.

3.2 Formal Definition of the Domain

A GPS measurement module provides latitude and longitude in degrees, using a certain radix r and fractional precision p. The radix would usually be either two or ten but the proposed algorithm allows any natural number except one. The fractional precision is typically between 5 (meter-perfect) and 7 (centimeter-perfect) for radix 10. These two parameters dictate the domain of our coder:

$$
\begin{aligned}
D_{lat} &= \left\{ \frac{n}{d} \in \mathbb{Q} \mid d = r^p, n \in \{-90 \cdot d, \ldots, 90 \cdot d\} \right\} \\
D_{lon} &= \left\{ \frac{n}{d} \in \mathbb{Q} \mid d = r^p, n \in \{-180 \cdot d, \ldots, 180 \cdot d - 1\} \right\} \\
D &= D_{lat} \times D_{lon}
\end{aligned}
$$

Notably, D_{lon} does not include 180. In a geodetic coordinate system, 180 and -180 are equivalent, since the longitude is circular. Which one of the values is included is simply a matter of convention. Contrarily, D_{lat} includes 90 as well as -90 because they represent different positions—the north and the south pole.

The coder maps the absolute positions to an innovation vector, which is the difference between a reference position and the actual position that is coded. Both the absolute positions and the innovations are elements of D. However, it is important to keep in mind that the probability models discussed in Chapter 5 are designed to code innovations and not absolute positions.

3.3 Binary Representation of the Domain

Each position consists of two rational numbers that share a common denominator with each other as well as with other positions. Since the base and precision are the same for all positions, the denominator can be discarded and a position can be digitally stored using a pair of integers, holding the value of the respective numerators. This is also known as fixed-point notation with a scaling factor of r^p.

Chapter 4

Arithmetic Coding

4.1 Information Content and Entropy

While we use a model of constant velocity to predict the next position of a vehicle, there are additional influences that cannot be predicted: the input of the driver and the noise introduced by the GPS measurements themselves. The innovation is therefore the subject of a random process, that we model as follows:

"An *ensemble* X is a triple $(x, \mathcal{A}_X, \mathcal{P}_X)$, where the *outcome* x is the value of a random variable, which takes on one of a set of possible values, $\mathcal{A}_X = \{a_1, a_2, \ldots, a_i, \ldots, a_I\}$, having probabilities $\mathcal{P}_X = \{p_1, p_2, \ldots, p_I\}$, with $P(x = a_i) = p_i, p_i \geq 0$ and $\sum_{a_i \in \mathcal{A}_X} P(x = a_i) = 1$." [11]

In the context of coding, $s \in \mathcal{A}_X$ is called a *symbol*, while $\mathcal{A}_X$ is called the *alphabet*.

In our case, the outcome x denotes the next position in a discrete trajectory, given as the innovation, i. e., the difference between expected and actual position. The alphabet $\mathcal{A}_X$ is the set of all possible innovations, which corresponds to the set D defined in Section 3.2, plus some control symbols, like a symbol that marks the end of a trajectory. The frequency of occurrence of each innovation vector is then given by $\mathcal{P}_X$. This is, in essence, the same model that the authors of [10] used, with the difference that we use the geodetic coordinate system of the input data whereas they converted the input data to Cartesian coordinates. This results in different alphabets.

For a given ensemble X, information theory offers a concept to quantify the amount of information that a certain outcome contains, and the average information content over all possible symbols: The *Shannon information content* $h(x)$ of an outcome x and the *average information*

content or *entropy* $H(X)$ are defined as

$$h(x) = \log_2 \frac{1}{P(x)}$$
$$H(X) = \sum_{x \in \mathcal{A}_X} P(x)h(x) = \sum_{x \in \mathcal{A}_X} P(x)\log_2 \frac{1}{P(x)}$$

For the special case $P(x) = 0$, $0 \cdot log(\frac{1}{0})$ is defined to be 0. [11]

The entropy serves as a lower bound for the average symbol size of any code of $\mathcal{A}_X$. These concepts were introduced by Claude E. Shannon in [13], but we use the notation and definitions from [11], which differ in some places.

4.2 Joint Ensembles

Given two ensembles X and Y, XY is called the *joint ensemble* which has outcomes xy with $x \in \mathcal{A}_X$ and $y \in \mathcal{A}_Y$. It has the *joint probability distribution* $P(x, y)$ with

$$\begin{aligned} P(x = a, y = b) &= P(x = a) \cdot P(y = b | x = a) \\ &= P(y = b) \cdot P(x = a | y = b) \end{aligned}$$

and if x and y are independent

$$P(x = a, y = b) = P(x = a) \cdot P(y = b) \text{ . [11]}$$

In this work, joint ensembles have two applications: First, a discrete trajectory consists of multiple consecutive points that need to be encoded individually. It seems natural then to think of each point as the outcome of a different ensemble, while the whole trajectory is the joint ensemble of those.

Second, an ensemble X that represents a single measurement can be factorized into ensembles Y and Z such that the alphabet size is reduced and the individual probabilities are increased. Consider the example

$$X = (x, \{000, \ldots, 999\}, \mathcal{P}_X) \qquad \text{with } \mathcal{P}_X(x) = 10^{-3}.$$

We can define two other ensembles

$$Y = (y, \{0, \ldots, 9\}, \mathcal{P}_Y) \qquad \text{with } \mathcal{P}_Y(y) = 10^{-1}$$
$$Z = (z, \{00, \ldots, 99\}, \mathcal{P}_Z) \qquad \text{with } \mathcal{P}_Z(z) = 10^{-2}$$

resulting in the joint ensemble

$$YZ = (yz, \{000, \ldots, 999\}, \mathcal{P}_{YZ}) \qquad \text{with } \mathcal{P}_{YZ}(yz) = \mathcal{P}_Y(y) \cdot \mathcal{P}_Z(z) = \mathcal{P}_X(yz).$$

YZ is thus equivalent to X. However, the individual alphabets $\mathcal{A}_Y$ and $\mathcal{A}_Z$ are smaller and the probabilities are higher. Therefore, by cleverly subdividing ensembles, fixed-size data types can be used to represent symbols and probabilities even when alphabet sizes become very large and probabilities become very small.

4.3 Arithmetic Coding

In this section, an arithmetic coder is described as a black box. It is explained what functionality it provides but not how it works. Understanding the internals of arithmetic coding may be helpful but is not required to understand the following chapters. Detailed explanations of arithmetic coding and the prerequisite information theoretical concepts can be found in [11].

Given an ensemble $X = (x, \mathcal{A}_X, \mathcal{P}_X)$, an arithmetic coder takes a string of symbols $s = s_1 s_2 \ldots s_m \in \mathcal{A}_X^*$ as input and outputs a bit string b such that

$$\text{length}(b) \leq \sum_{i=1}^{m} h(s_i) + 2 \ .$$

This means that an arithmetic coder generates at most a constant overhead of 2 bits compared to an optimal encoding, where the length of the coded string equals the information content of the source string. [11]

In practice however, it is impossible to determine $\mathcal{P}_X$ exactly. The efficiency of an arithmetic code depends on how well the probability model reflects the underlying real-world process. In Chapters 5 and 6 we focus on the proposition and evaluation of different probability models for the domain of vehicular trajectories.

In the above explanation, the different symbols of the source string are independent and identically distributed. This is not a requirement of an arithmetic coder. A source string s

can be seen as the outcome of a joint ensemble S, i. e.,

$$S = X_1 \dots X_n,\ s = x_1 \dots x_n$$

where the individual probability distributions are given by

$$\mathcal{P}_{X_i}(x_i) = P(x_i|x_1, \dots, x_{i-1}) \ . \ [11]$$

In simpler terms, a probability model for arithmetic coding may use the previously coded symbols to compute the probability distribution for the next symbol. In the case of vehicular trajectories, this is an absolute requirement as most of the redundancy is caused by the relative proximity of consecutive measurement points.

We use Bob Carpenter's Java implementation [5] of the arithmetic coding algorithm. It is generic, so we can employ our own probability models implemented as Java objects. The different probability models are described in Chapter 5.

In this specific implementation, the `int` primitive type is used to represent probabilities and symbols. Since probabilities are not discrete, they are represented by intervals defined via three positive integers: the lower and upper bound of the cumulative probability interval and the total cumulative probability. For example, the triple $(5, 10, 20)$ represents the cumulative probability interval $[\frac{5}{20}; \frac{10}{20}] = [\frac{1}{4}; \frac{1}{2}]$. Each symbol is assigned a unique probability interval. The usage of a fixed-size data type places hard limits on the alphabet size (2^{32}) and minimum usable probability ($\frac{1}{2^{31}-1}$). These limits can be circumvented using joint ensembles, as demonstrated in Section 4.2. In Section 5.1.1, we provide specifics on how to apply joint ensembles to the problem of coding innovation vectors.

Chapter 5

Probability Models

In the following, we propose probability models for the innovation vectors. These models are based on some assumptions made about the probability distribution of innovation vectors. We assume that the distribution peaks at the origin, and decreases exponentially in all directions. This is supported by the findings in [10]. Furthermore, we assume that the distribution is, to a certain extent, radially symmetric. Our models do not take the orientation of the vehicle into account, and therefore deviations in all directions are equally possible.

5.1 Grids

All of the models described in this chapter, with the exception of the uniform model from Section 5.2 use one or two discrete grids. Grids serve two different functions: first, a grid is able to map a 2-dimensional innovation vector to a symbol. A grid that is used this way in a model is called the *symbol grid* of that model. Second, a grid is needed to provide geometrical context to a probability distribution. Such a grid is then referred to as the *probability grid* of a distribution or model. For most models, the same grid serves as symbol and probability grid. Both types of grids can also be used to determine a corresponding rectangular area for a symbol or probability.

Some properties are common to all grids: they are rectangular shaped and aligned such that the horizontal axis corresponds to the longitude and the vertical axis to the latitude. The number of grid nodes along each axis is odd, so that there is always one grid node that can be identified as the center of the grid. Having a clear center makes sense, since we assume the probability distribution of the innovations to peak at the center. The way a grid is sized depends on the respective probability model. For a detailed description of the models, it is necessary to formally name the defining parameters of a grid.

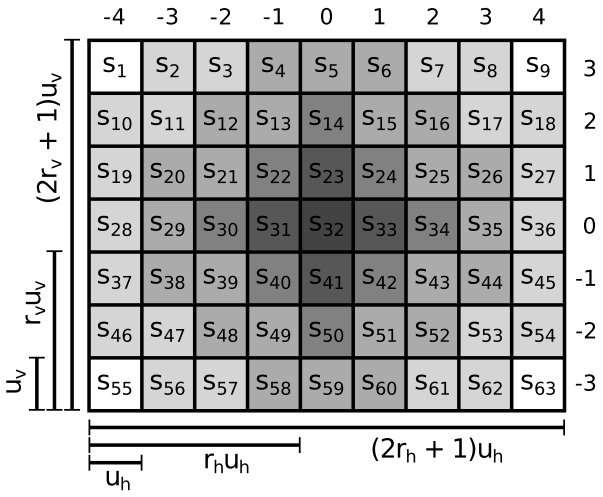


Figure 5.1: A grid, as defined by its parameters. For each grid node index (i,j), the mappings $R(i,j)$ (rectangles), $f(i,j)$ (color codes), and $\sigma(i,j)$ $(s_1, \cdots, s_{63})$ are shown.

The vertical and horizontal length of grid nodes u_h and u_v are called the *horizontal and vertical unit*. The number of grid nodes along each axis is given in dependence of the grid radii r_h and r_v as $2r_h + 1$ and $2r_v + 1$. In consequence, the geometrical length of the axes are $(2r_h + 1)u_h$ and $(2r_v + 1)u_v$.

Grid nodes are identified via indices $(i,j) \in \{-r_h, \cdots, r_h\} \times \{-r_v, \cdots, r_v\} =: I$ where $(0,0)$ always denotes the center node. With this index set, three utility functions for the grid can be defined:

The *frequency mapping* $f : I \to \mathbb{N}_0$ maps a grid node to the frequency with which it is encountered. For convenience, we name the sum over all grid nodes $f_{total} = \sum_{i \in I} f(i)$. The probability of a grid node can then be calculated by normalization of the frequency values $\frac{f(i,j)}{f_{total}}$.

The *rectangle mapping* $R : I \to \mathcal{P}(\mathbb{R}^2)$ maps a grid node to the geometrical area that it covers. This is necessary to resample the frequency mapping f from one grid to another with different dimensions, as done by the metrical model that is subject of Section 5.3.2.

The *symbol mapping* $\sigma : I \to \mathcal{A}_X$ maps a grid node to a symbol. Its inverse σ^{-1} maps each symbol that represents a grid node to the corresponding index. The symbol mapping does not necessarily have to be defined. The grid's purpose may be limited to provide frequencies for its nodes, and the associated areas. σ is defined for a grid if and only if it is a symbol grid. In this case, the frequency for a symbol is given by $f \circ \sigma^{-1} : \mathcal{A}_X \to \mathbb{N}_0$.

To be used in an arithmetic coder, usually at least one additional control symbol, called the

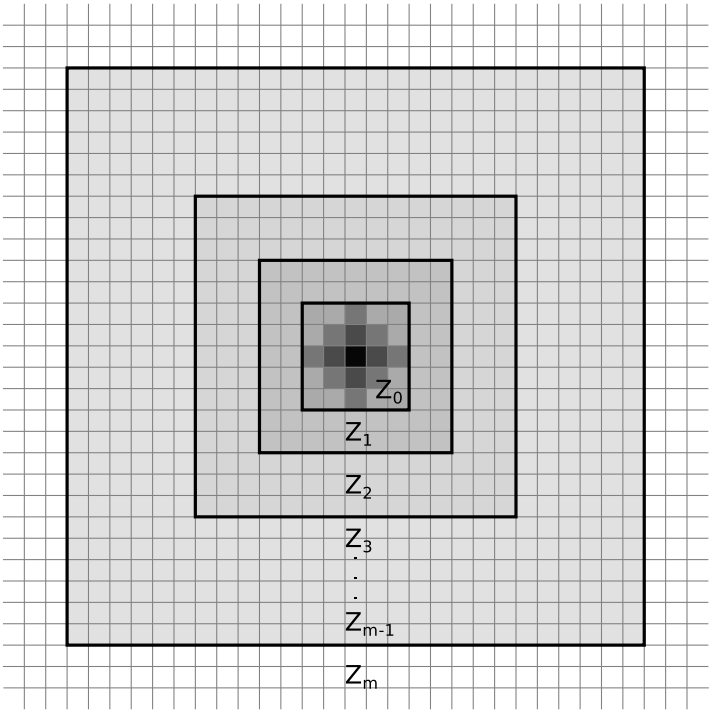


Figure 5.2: Schematic depiction of the different zones of a trained probability distribution. Z_0 assigns different probabilities to its different nodes. The other zones each redistribute a single probability uniformly over their nodes.

end-of-file (EOF) symbol, is required, because the size of the trace is generally not known. This is omitted from the formal description of the grid and the models, because it introduces unnecessary clutter. For example, the symbol mapping σ would not be invertible without limitation, as the EOF symbol does not have a corresponding grid node. In an implementation, the addition of an EOF symbol is easy: first, another counter c_{EOF} is defined, which holds the frequency of the EOF symbol. All that remains is to replace f_{total} with $f_{total} + c_{EOF}$ when calculating the probabilities. The remaining probability interval of $\frac{c_{EOF}}{f_{total}+c_{EOF}}$ is then assigned to the EOF symbol. Other control symbols may be added this way, if required by a specific implementation.

5.1.1 Grid Zones

Due to constraints given by the number of training data as well as the available memory, it is not feasible to empirically determine the probability of every possible node in a large grid. A grid node that is very improbable would require enormous amounts of sample data. Consider a grid node with an actual probability of 10^{-10}: we would need sample traces with a total length of about 7 billion position measurements to have a probability of $\frac{1}{2}$ to encounter the

grid node at least once[1]. In the actual implementation, the frequency mapping f will only be partially calculated from training data.

We propose to divide grids into multiple disjunctive zones centered around the origin, as illustrated by Figure 5.2. The size of the *center grid* Z_0 is chosen small enough that the available training set provides a good approximation of the distribution. Ideally, every node from the zone is encountered at least once during training. Those probabilities are used "as is"; each grid node is assigned an individual probability. Around the center zone, multiple concentric *grid rings* $Z_1, \ldots, Z_{m-1}$ are laid out. Grid nodes that are part of a grid ring are not assigned an individual frequency. Instead, they share the accumulated frequency of themselves and their peers. The remaining part of the grid is called the *outer grid* Z_m. Like the grid rings, it uses a shared frequency count for its nodes.

Formally, the zone Z_j is defined as the set of indices of the grid nodes that are part of the zone, such that

$$Z_j \subset I$$

$$\bigcup_{j=0}^{m} Z_j = I$$

$$\forall j, k = 0, \ldots, m : j \neq k \Leftrightarrow Z_j \cap Z_k = \emptyset \ .$$

Using these different zones, we can define a probability distribution that requires knowledge about only a certain subset of f. For an ensemble $N = (n, I, \mathcal{P}_N)$, where n resembles a grid node, the probability that the outcome lies in Z_j is

$$P(n \in Z_j) = \frac{1}{f_{total}} \sum_{i \in Z_j} f(i) \ .$$

We can then approximate the probability of a grid node $i \in I$ with

$$p_i = \begin{cases} \dfrac{f(i)}{f_{total}} & \text{if } i \in Z_0 \\ \dfrac{P(n \in Z_j)}{|Z_j|} & \text{if } i \in Z_j, j = 1, \cdots, m \end{cases} \ .$$

To learn this distribution from test data, $|Z_0| + m$ counters are required instead of |I|; for all zones except Z_0, only the accumulated frequency of the nodes in the zone is required to calculate $P(n \in Z_j)$. Since the number and size of the zones can be varied, zone based probability models offer great flexibility for finding a sufficient trade-off between compression

[1] $1 - 10^{-10}$ to the power of 7 billion is approximately $\frac{1}{2}$.

performance, size of the training set, and required runtime resources.

Zones also circumvent the problem of very large alphabets in combination with fixed-size data types. Consider the ensemble $X = (x, \mathcal{A}_X, \mathcal{P}_X)$. In this case, the grid is required to have a symbol mapping. Otherwise there would be no relation between the alphabet and the grid. For each zone Z_j, we define an ensemble $X_j = (x_j, \mathcal{A}_{X_j}, \mathcal{P}_{X_j})$:

$$\begin{aligned} \mathcal{A}_{X_j} &= \{s \in \mathcal{A}_X | \sigma^{-1}(s) \in Z_j\} \\ \forall s_i \in \mathcal{A}_{X_j} : \quad p_i &= P(x_j = s_i) = P(x = s_i | \sigma^{-1}(x) \in Z_j) \ , \end{aligned}$$

where σ^{-1} is the inverse symbol mapping described in Section 5.1. The ensemble X_j then is a model for the case that it is certain that the next position lies in zone Z_j. We can define an additional *multiplexer ensemble* $M = (k, \{0, \ldots, m\}, \mathcal{P}_M)$ where k is the outcome of a random variable dependent on x, whose value denotes the zone that x lies in:

$$k = j \Leftrightarrow \sigma^{-1}(x) \in Z_j \ .$$

The probability distribution of k is simply the sum of the probabilities of each member of a zone:

$$P(k = j) = P(\sigma^{-1}(x) \in Z_j) = \frac{1}{f_{total}} \sum_{i \in Z_j} f(i) \ .$$

The joint ensemble MX_k can now be used to factorize the original ensemble X. For a symbol s with $\sigma^{-1}(s) \in Z_j$, we first code the symbol j with $P(k = j)$ and then s with $P(x_j = s)$:

$$\begin{aligned} P(k = j) \cdot P(x_j = s) &= P(\sigma^{-1}(x) \in Z_j) \cdot P(x = s | \sigma^{-1}(x) \in Z_j) \\ &= P(x = s) \cdot P(\sigma^{-1}(x) \in Z_j | x = s) \\ &= P(x = s) \ . \end{aligned}$$

Every symbol $s \in \mathcal{A}_X$ is split into two code symbols, using the ensembles M and the appropriate X_j. When decoding, the correct ensemble X_j is chosen after the outcome of M has been decoded. Since $P(k = j) \cdot P(x_j = s) = P(x = s)$, the codes are equivalent. This factorization allows small probabilities and large alphabets to be represented by products of larger probabilities and conjunctions of smaller alphabets, and the limitations of fixed-size data types can be circumvented.

All of our models that use empirically determined probability distributions split their grid into zones to avoid the need for an unrealistic amount of sample data. A separation into center grid and outer grid is usually enough for a model to work properly. The usage of grid rings is optional, but may improve compression ratio and reduce resource consumption. Using

several grid rings instead of a larger center grid adds two kinds of bias to the model: First, the distribution is expected to exhibit some radial symmetry, and second, symbols that are close to each other geometrically are assumed to be close in probability as well. From another perspective, models using grid rings assume a relation between grid nodes in proximity to each other as well as nodes that have a similar distance from the center. This relation is exploited to gather information about multiple grid nodes from an individual sample datum.

5.2 Uniform Model

The uniform model assigns the same probability to each code symbol. An arithmetic coder utilizing a uniform distribution is very similar to an uncompressed representation where all symbols are represented as different bit strings of minimal equal length.

Let $\mathcal{A}_X$ be an alphabet with $|\mathcal{A}_X| = n$. The bit length of an uncompressed symbol is $\lceil \log_2(n) \rceil$. The size of an encoded symbol is $\log_2(1/p)$. In this case, the probability $p = \frac{1}{n}$ resulting in a symbol size of $\log_2(n)$. It clearly follows that

$$\lceil \log_2(n) \rceil - 1 < \log_2(n) \leq \lceil \log_2(n) \rceil \ .$$

With a uniform distribution, the average length of a code symbol can be reduced by at most 1 bit. Since the alphabet that encodes positions in a geo coordinate system with sufficient precision is quite large, the compression ratio achieved by a uniform probability model can be expected to be low. For example, using base 10 and a precision of 6 decimal places, $|D_{lat}| = 180 \cdot 10^6 + 1$ and $|D_{lon}| = 360 \cdot 10^6$, resulting in an uncompressed bit size of $\lceil \log_2(D_{lat}) \rceil + \lceil \log_2(D_{lon}) \rceil = 57$ for a single coordinate. Using arithmetic coding with a uniform distribution, the average bit length per symbol is given by $\log_2(D_{lat}) + \log_2(D_{lon}) \approx 55.8548$ resulting in an estimated compression ratio of $1 - \frac{55.8548}{57} \approx 2\%$. Nonetheless, a uniform model is still useful for transmitting the starting position of a trajectory.

The uniform model is the only one in this chapter that does not use a grid. A uniform distribution over the complete alphabet is a special case that allows a different method of factorizing an ensemble. The geo coordinates are represented as integers with a certain amount of digits. We can split these into smaller numbers, e.g. groups of 3 digits, and then encode them individually with a uniform distribution. On the decoding side, the groups are decoded and then concatenated, yielding the original number. This is possible only because for a uniform distribution, the probabilities of the individual digits are independent of each other. This type of factorization was already demonstrated by the example in Section 4.2.

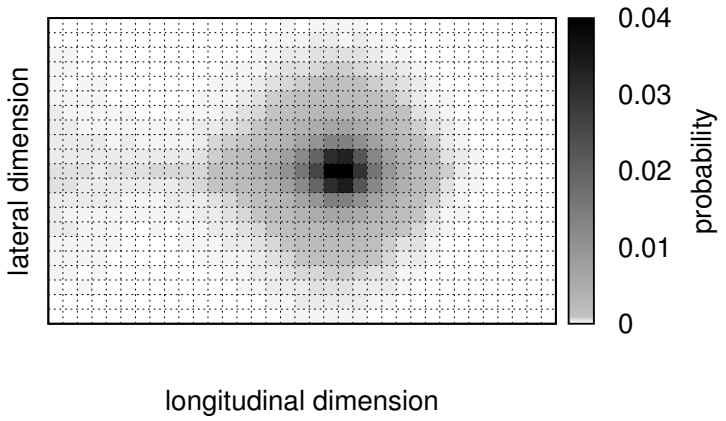


Figure 5.3: Heat map of a distribution from [10] obtained from sample data.

5.3 Empirical Models

In [10], Koegel et al. evaluate multiple innovation-based probability models and achieve the best results with distributions that are calculated empirically, i. e., from sample data. Using a training set of traces, the innovation of each measurement is calculated and counted, resulting in a frequency mapping for symbols $f' : \mathcal{A}_X \to \mathbb{N}_0$ assigning a probability of

$$p_i = \frac{f'(s_i)}{f'_{total}} \quad \text{with} \quad f'_{total} = \sum_{s \in \mathcal{A}_X} f(s)$$

for each symbol $s_i \in \mathcal{A}_X$. A resulting distribution is shown in Figure 5.3.

Our formal definition of grids distinguishes between grid nodes and symbols. Specifically, the frequency mappings used by our models are of the form $f : I \to \mathbb{N}_0$; their domain differs from that of f'. This allows us to describe probability distributions without direct relation to the alphabet. In our notation, the equivalent to the frequency mapping f' would be $f \circ \sigma^{-1}$, i. e., using the inverse symbol mapping σ^{-1} to get the grid node of a symbol, and then the frequency mapping f to obtain the assigned probability.

5.3.1 Basic

The basic model uses only a single grid. It maps probabilities directly to its symbols, i. e., the possible innovation vectors:

$$p_i = \frac{f(\sigma^{-1}(s_i))}{f_{total}}$$

where f is the frequency mapping, σ^{-1} is the inverse of the symbol mapping of the grid, and f_{total} denotes the number of training data. When training the probability distribution for the basic model, no complicated calculations are required.

However, the simplicity of the basic model has a price: The deviation from the expected position is calculated in units of latitude and longitude, i. e., as angles. Since the metrical distance that corresponds to a unit of longitude depends on the current latitude (as illustrated in Figure 3.2 in Section 3.1), two geometrically different deviations might yield the same outcome. From another perspective, the probability to deviate 10 m to the east from the expected position would depend on the current latitude.

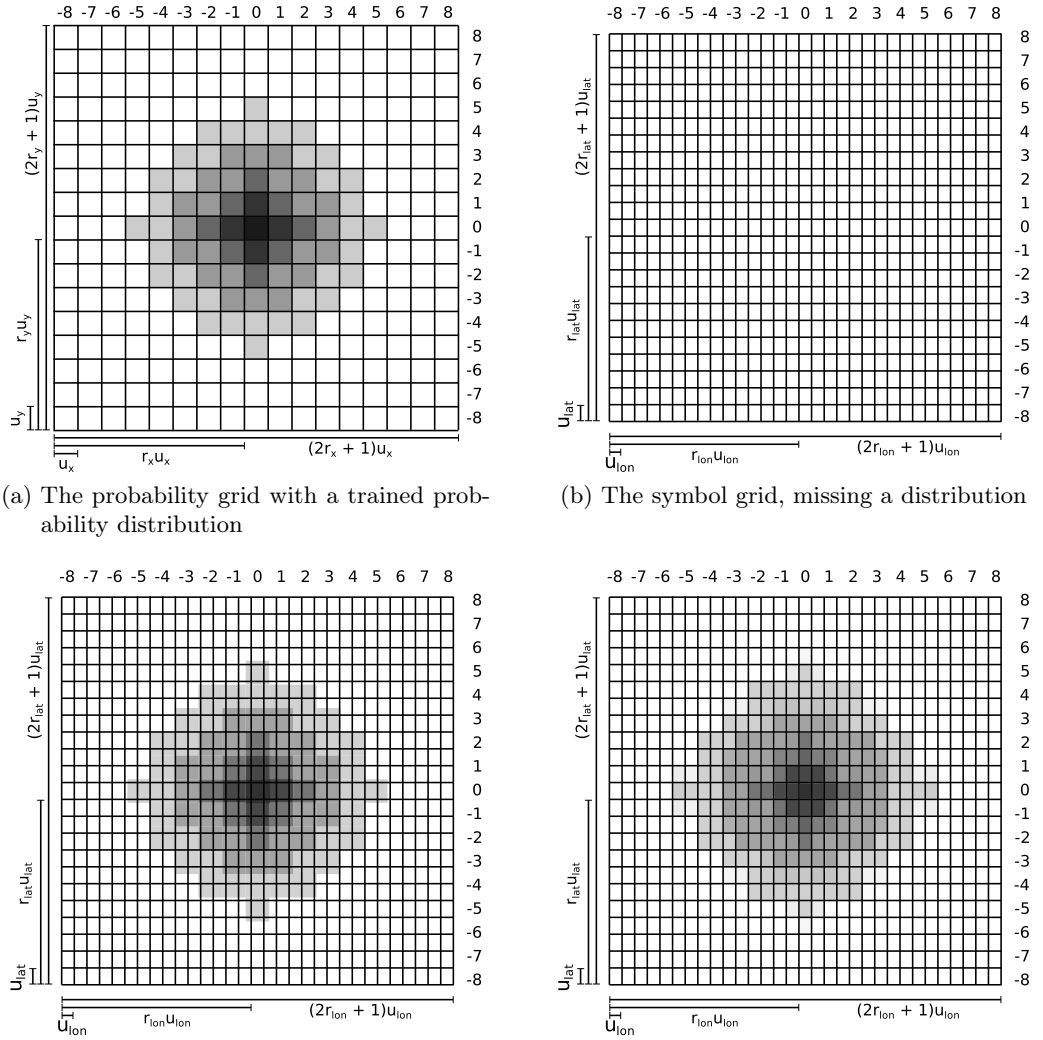
It is intuitive to expect that the movement and steering behavior of vehicles is not, in fact, dependent on the latitude. Therefore, a model that does measure deviations in metrical units is discussed in the next section.

5.3.2 Metrical

A solution to the problem of latitude-dependent geometry is to calculate the innovation in metrical distance instead of angular distance. Metrically similar innovations are mapped to the same outcome. This results in a probability distribution that is more in check with how a vehicle is expected to behave. However, the coder may not use metrical deviations; converting geo coordinates to Cartesian coordinates would generally result in loss of information. Instead of converting the coordinates, we devise an algorithm that converts the universally applicable metrical model into a locally applicable geodetic model.

In contrast to the basic model, the probability grid is defined in metrical units. The area that each grid node covers is therefore constant; previously, the units u_{lon} and u_{lat} were used, whose actual size measured in meters is dependent on the current latitude. This means that the probability grid is independent from the position that is to be encoded. However, it does not assign probabilities to the code symbols; instead, it provides the probability that a measurement falls into a certain area.

The probability for a symbol has to be calculated individually for each reference position. At a specific reference position, the abstract units u_{lon} and u_{lat} have a real, metrical equivalent. Using those values in combination with the dimensions of the probability grid, a symbol grid



(a) The probability grid with a trained probability distribution

(b) The symbol grid, missing a distribution

(c) The symbol grid, with the distribution of the probability grid in the background

(d) The symbol grid, with the distribution rasterized to the grid nodes

Figure 5.4: A visual explanation of the parameters of the two grids and the resampling of the probability distribution. The numbered coordinates are those from the probability grid. The indices of the symbol grid are different.

with dimensions similar to the probability grid's can be created. The last step is a resampling of the probabilities from the probability grid to the raster of the symbol grid. The result is a distribution over the elements of the symbol grid, i. e., the code symbols. This process is illustrated by Figure 5.4.

Before the algorithm is explained in more detail, we name the necessary values for both grids, using the same scheme as in Section 5.1, but with different indices:

For the probability grid we use the index p as well as x for the horizontal axis and y for the vertical axis. u_x and u_y denote the length of grid nodes; r_x and r_y are the radii such that the area that the grid covers is given by $(2r_x + 1)u_x \times (2r_y + 1)u_y$. The indices of the grid nodes form the set I_p, which is the domain of the rectangle mapping $R_p : I_p \to \mathcal{P}(\mathbb{R}^2)$ and frequency mapping $f_p : I_p \to \mathbb{N}_0$. As with the basic model, f_p is trained from sample data.

The symbol grid has the same parameters with the indices s for the grid, and lon, and lat for its horizontal and vertical axis. However, none of these parameters are chosen. They are calculated depending on the reference position and the parameters of the probability grid. We start with the units:

$$\begin{aligned} u_{lat} &= \frac{\pi \cdot R_e}{|D_{lat}| - 1} &&= \frac{\pi \cdot R_e}{180 \cdot r^p} \\ u_{lon} &= \frac{2\pi \cdot R_e \cos\varphi_e}{|D_{lon}|} &&= \frac{\pi \cdot R_e \cos\varphi}{180 \cdot r^p} \end{aligned}$$

where R_e is the radius of the earth and r^p is the radix to the power of the precision, i. e., the number of fractions that each degree is split into. For demonstration purposes, a spherical model of the earth is used. Since the implementation is aimed towards GPS measurements, the formulas used there are for the reference ellipsoid from WGS 84 [2]. u_{lat} is the approximate metrical distance projected on the surface of the earth while traversing one unit of latitude. u_{lon} measures the same for the longitude, at the current latitude φ. The corresponding grid sizes $(2r_{lat}+1)u_{lat}$ and $(2r_{lon}+1)u_{lon}$ are chosen to get the largest possible symbol grid whose area is completely contained in the probability grid. We start with a formula that gives a real-valued radius, such that the dimensions would match exactly:

$$(2r_a + 1)u_a = (2r_b + 1)u_b \Leftrightarrow r_a = \frac{(2r_b + 1)u_b}{2u_a} - \frac{1}{2} \; .$$

With the help of that formula, we determine the appropriate integer values for the radii

$$r_{lat} = \left\lfloor \frac{(2r_y + 1)u_y}{2u_{lat}} - \frac{1}{2} \right\rfloor \; , \qquad r_{lon} = \left\lfloor \frac{(2r_x + 1)u_x}{2u_{lon}} - \frac{1}{2} \right\rfloor \; .$$

This completes the symbol grid for the reference position. Each index has a corresponding

symbol via the symbol mapping $\sigma_s : I_s \to \mathcal{A}_X$ and rectangle via the rectangle mapping $R_s : I_s \to \mathcal{P}(\mathbb{R}^2)$. The only thing missing to assign probabilities to symbols is the frequency mapping $f_s : I_s \to \mathbb{N}_0$ of the symbol grid.

For each index $i_s \in I_s$, we have a corresponding area $R_s(i_s)$ that intersects one or more elements from the range of R_p. That enables us to redistribute the frequency count f_p from the probability grid to the nodes of the symbol grid, using the area of the intersecting rectangles as weight:

$$f_s(i_s) = \left\lfloor \sum_{i_p \in I_p} f_p(i_p) \cdot \frac{area(R_s(i_s) \cap R_p(i_p))}{area(R_p(i_p))} \right\rfloor$$

With the frequency count f_s, each symbol s_i is assigned a probability p_i as before:

$$p_i = \frac{f_s(\sigma_s^{-1}(s_i))}{f_{total}}$$

In reality, the grid of latitudes and longitudes is, of course, not rectangular. However, in the vehicular domain, the innovation vectors are mostly in the order of magnitude of up to 15 meters long. The circumference of the circles of latitude and longitude on the surface of the earth is usually—depending on the latitude—much higher, such that the bend of the surface is negligible. In very close proximity to the poles, the circles of latitude become very small, and the metrical model will be unable to estimate usable probabilities as well.

It might be counter-intuitive that this model, designed for lossless coding, features an explicit rounding operation when rescaling the frequency count. However, this rounding is performed only on the frequency count and therefore affects the probabilities assigned to symbols, not the symbols themselves. In essence, the values of the probabilities are irrelevant to the correctness of the coding process, as long as the resulting probabilities are exactly the same for the encoding and decoding step. They are, however, very relevant to the compression performance of the coding process.

5.4 Adaptive Model

Instead of using a static, pre-determined distribution, a probability model could approximate the distribution “on-the-fly”. The adaptive model we propose uses the same grid layout as the basic empirical model from Section 5.3.1. However, the counters are all initialized to one at the start of the coding process and are appropriately increased each time a data point is

coded. In consequence, the approximation of the real distribution gets better the longer the coding process runs. In general, we expect the performance of the model to increase with the size of the trace that is encoded.

The fact that the model has none of its data in the beginning and only gradually learns how to efficiently code its alphabet complicates the relationship between parameter values and compression ratios. For the empirical models, a suitable size for the innermost zone Z_0 can be determined by evaluating the sample data. For a large enough data set, like the one we use for the evaluation in Chapter 6, even large grids with dimensions of 40m × 40m can be improved by an even larger grid. The same is not true for the adaptive model. The larger Z_0 is, the more expensive are the first few symbols.

Since the frequency mapping is mutable, we write f_j for the frequency mapping after $j-1$ symbols have been coded. Similarly, we write $(p_i)_j$ for the probability of grid node i and corresponding symbol $\sigma^{-1}(i)$. Symbols in Z_0 are initially distributed uniformly, i. e.,

$$\forall i \in Z_0 : f_0(i) = 1 \text{ and } (p_i)_0 = \frac{1}{|Z_0|} \ .$$

More generally, for the j-th symbol that is to be coded, the distribution is given by

$$\forall i \in Z_0, j = 0, 1, 2, \ldots : (p_i)_j = \frac{f_j(i)}{|Z_0| + j} \ .$$

For each step j that codes the symbol $s_j = \sigma(i)$, the frequency count for s_j is increased:

$$f_{j+1}(i) = f_j(i) + 1 \ \Rightarrow (p_i)_{j+1} = \frac{f_j(i) + 1}{|Z_0| + j + 1} \ .$$

The impact that a single data point has on the adaptive distribution therefore depends on $|Z_0|$. This dependency weakens with increasing j and $f_j(i)$. Choosing the size of Z_0 means choosing a trade-off in compression performance between shorter and longer traces.

While the previously examined properties of the adaptive model lead us to expect generally lower compression ratios than the empirical models on a set of diverse traces, it also has some advantages. There is no predetermined probability distribution that has to be learned and stored. This leads to a smaller and slightly simpler implementation. Unlike the basic empirical model, distances can be measured in angles without significant risk of skewing the probability distribution as long as all data points of a trace are comparatively close in latitude. We assume that this is generally true for the vehicular domain, with occasional fringe cases caused by measurement or processing errors. There are some other variables that could have an impact on the distribution: the driver of the vehicle, the vehicle itself, or the quality of the

GPS module used. The empirical models cannot differentiate between these variables, but the adaptive model can.

We expect the adaptive model to have the least problems in proximity of the poles. It avoids the problems of the basic model, since its distribution is —for the vehicular domain—trained by data from nearby latitudes only. It also sidesteps the problems of the metrical model, since it does not use any geometrical approximations, but samples the distributions for its symbols directly.

Chapter 6

Evaluation

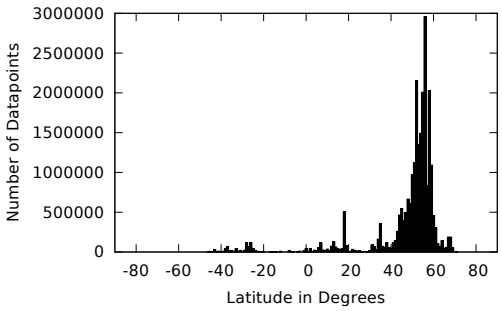
6.1 Sample Data

We used a set of 30540 vehicular traces, obtained from the OpenStreetMap project (OSM) [1]. All traces share the sampling rate of 1Hz. They vary in length from 100 to 125435 data points, i. e., individual position measurements.

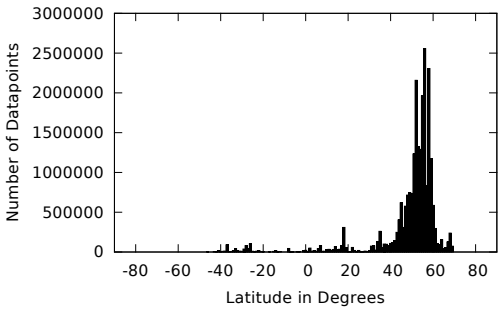
For the scope of this evaluation, we randomly divided the sample traces into three disjunctive sets of 10180 traces each: the *training set*, the *cross-validation set* (CV), and the *test set*. The training set is used to determine the distributions for empirical models. Other models do not use it. The cross-validation set is used to find good values for the different parameters of the models. It is important to keep this separate from the test set, which is used to compare the models to each other as well as the coders from [10] and [6]. Using the same set of traces for parameter optimization and comparison could give an unrealistic advantage to our models as the parameters would be optimized for the specific set of sample data used for comparison.

There are two key properties that have a measurable impact on the compression ratio: the length and the latitudinal position. The length is especially important when comparing the adaptive model to the empirical ones. A broader coverage of latitudes is expected to cause problems for the basic model because the area that it covers varies with the latitude.

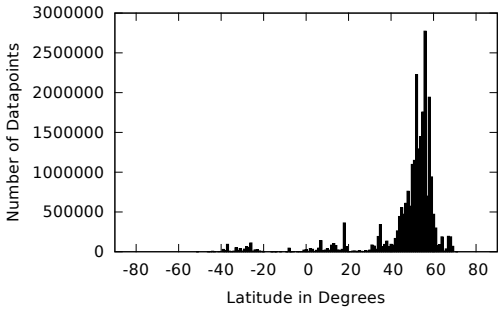
Figures 6.1a-6.1c show an analysis of the latitudes that occur in our three data sets. Most of the traces lie between 40°N and 60°N. We decided that additional, less biased sets of sample data were needed, to avoid training models specific to certain latitudes. These reduced sets trade size—and therefore, statistical reliability of the results—for representativity. We divided the range of the latitude into 12 classes and then chose the reduced sets such that the number of data points belonging to a single class would not exceed 250000, as shown in Table 6.1.



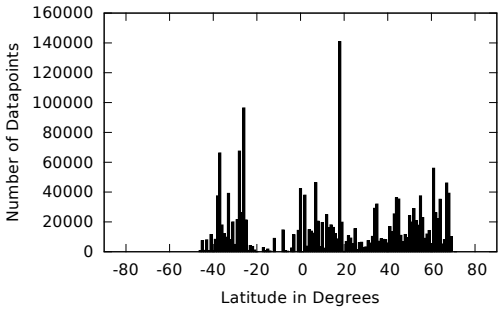
(a) Latitude coverage of the training set



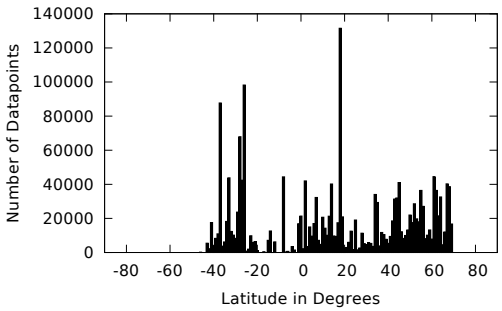
(b) Latitude coverage of the CV set



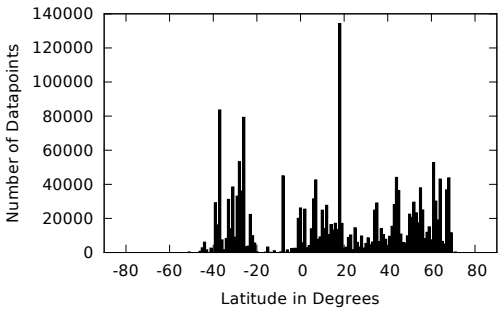
(c) Latitude coverage of the test set



(d) Latitude coverage of the reduced training set



(e) Latitude coverage of the reduced CV set



(f) Latitude coverage of the reduced test set

Figure 6.1: Analysis of the latitude coverage in the different sets of sample data

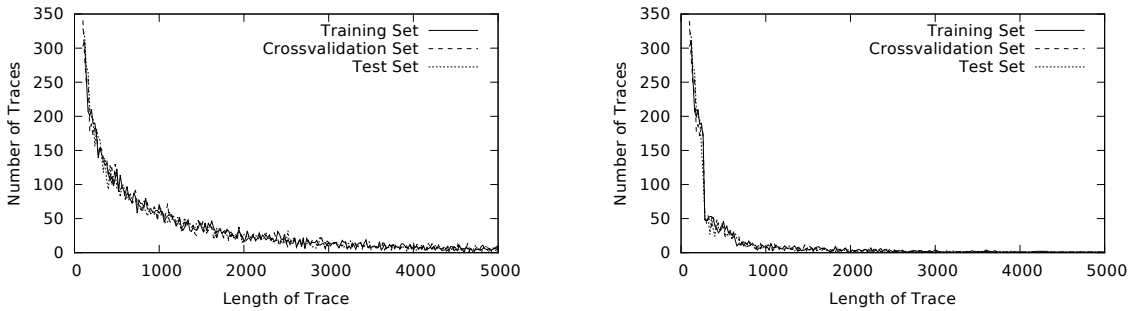
Class	$-90°$ to $-75°$	$-75°$ to $-60°$	$-60°$ to $-45°$	$-45°$ to $-30°$	$-30°$ to $-15°$	$-15°$ to $0°$
Training	0	0	8198	244457	249656	95816
CV	0	0	214	249998	249982	113165
Test	0	0	3810	249921	249965	101777

Class	$0°$ to $15°$	$15°$ to $30°$	$30°$ to $45°$	$45°$ to $60°$	$60°$ to $75°$	$75°$ to $90°$
Training	249968	249987	249999	250000	249999	0
CV	249962	249991	250000	249983	249988	0
Test	249947	249992	250000	249954	249995	0

Table 6.1: The distribution of data points in the reduced sets among twelve classes of latitude

		Training	CV	Test
Full	absolute	24547377	23923690	24014409
	relative	33.87%	33.00%	33.13%
Reduced	absolute	1851490	1866644	1858756
	relative	33.20%	33.47%	33.33%

Table 6.2: The number of total data points in each set of sample data



(a) Length distributions of the three complete data sets

(b) Length distributions of the three reduced data sets

Figure 6.2: The number of traces in each data set vs. the trace length

While the reduced sets visibly diverge from a uniform distribution, they are an improvement on the original sets and offer reasonable variety in latitudes. Unfortunately, the data sampled from OSM did not include any traces in proximity of the poles. This left us unable to evaluate the effects of latitudes less than 15 degrees from the poles on the individual models.

The absolute and relative overall size of the data sets is shown in Table 6.2. For both the full and reduced sets, the total data points are distributed evenly among training, CV, and test set.

For the evaluation of the adaptive model to be reliable, it is required that the cross-validation and test sets include a significant part of shorter traces. Since the model trains its distribution on-the-fly, short traces are its weakest point. Figure 6.2 confirms that there are a lot of shorter traces in all of the sample sets. It also shows that traces of similar length are distributed evenly among the training, CV, and test sets. In the following, we will see that all models performed significantly worse on the reduced sets. The reason for this is that the full sets include many very long traces, whose entropy is lower since they often include longer motionless periods.

6.2 Parameter Optimization

Before comparing our different models to each other and to other compression schemes, we needed to choose parameters like the center grid size, grid granularity, etc. For some parameters a suitable value was chosen through reasoning. Others were chosen by optimizing the compression ratio over their possible values, using the cross-validation set as reference data.

When discussing the grid size, we use the terminology from Section 5.1. The radii in this context determine the size of the center grid, not the complete grid.

6.2.1 Basic Empirical Model

For the basic model, we needed to consider only the size and layout of the different zones. We chose to evaluate two variations of the basic model: A simpler one with only the center grid and the outer grid, as well as one with multiple additional grid rings between the inner and outer grid.

For the simple version, only the size of the center grid had to be chosen, or more specifically, a latitudinal radius r_{lat} and a longitudinal radius r_{lon}. We chose to always set $r_{lon} = \frac{r_{lat}}{2}$. The sample data uses a base of 10 and precision of 6. On average, the metrical equivalent to one longitudinal unit is about 5–6 cm on the surface of the earth, while one latitudinal unit corresponds to about 11 cm (Figure 3.2a). More formally, we assumed that on average

$$u_{lon} \approx \frac{u_{lat}}{2}$$

and chose

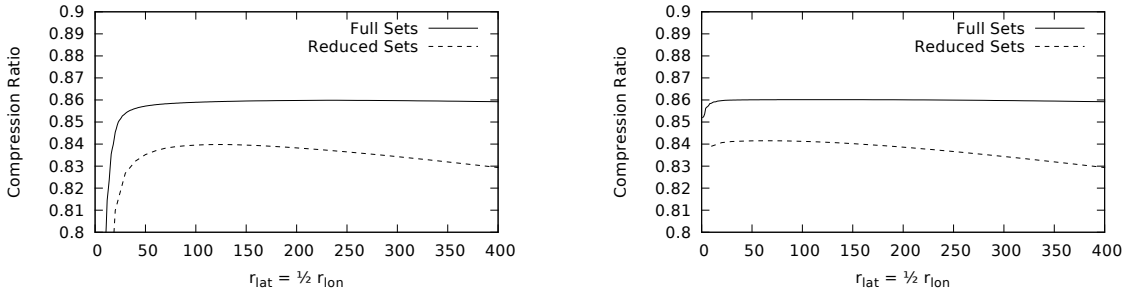
$$r_{lon} = 2 \cdot r_{lat}$$

resulting in a roughly quadratic shape of the grid:

$$(2r_{lon} + 1)u_{lon} \approx \frac{(4r_{lat} + 1)u_{lat}}{2} = (2r_{lat} + \frac{1}{2})u_{lat} \approx (2r_{lat} + 1)u_{lat} \ .$$

This left us with a single parameter r_{lat} to optimize.

The extended version of the basic model is similar, except that we added multiple grid rings around the center zone. We chose a fixed size for the cumulative area covered by the center grid and grid rings. Like before, r_{lat} parameterizes the size of the center grid. The grid rings are laid out successively around the center grid, each with a latitudinal distance $d_{lat} = 5$ and a longitudinal distance $d_{lon} = 2d_{lat} = 10$ from the preceding grid ring. The last grid ring is



(a) Basic model, simple variant

(b) Basic model, extended variant

Figure 6.3: Compression vs. grid size for the basic empirical model

a possible exception, as its dimensions are chosen to exactly fit to the targeted total area of $1601u_{lon} \times 801u_{lat}$.

Figure 6.3 shows the relation between the latitudinal radius r_{lat}, that determines the size of the center grid, and the compression ratio for the two variations of the basic model. For both models, the compression ratio monotonically increases with the size of the center grid up to a certain r_{lat}, from which on it monotonically decreases. This shows a conflict in choosing an appropriate grid size: A larger center grid means more accurate probability estimates for the symbols it covers—but as the symbols farther from the center are less frequent, they require a large amount of sample data for their estimates to be reliable. From a certain point on—about $r_{lat} = 125$ for the simple model and $r_{lat} = 75$ for the extended one—even the very large training set containing over 24 million data points is not sufficient. As a side note, the fact that the extended model's falloff point is lower does not mean that it requires more training data; on the contrary, the extended model already gives a fairly good approximation for most symbols that are covered by the grid rings. As a consequence, the optimum can be reached with a smaller center grid, and the falloff starts earlier. In both cases, it is visible that the compression ratio almost plateaus before it reaches the falloff point. We chose the values 125 and 50 for r_{lat} for the simple and extended model respectively.

6.2.2 Metrical Empirical Model

The metrical model needs additional parameters compared to the basic model, as described in Section 5.3.2. Those parameters determine the resolution of the probability grid, i. e., values for u_x and u_y. The latitudinal unit u_{lat} varies only slightly with the position and is—because of the base and precision of the input data—always close to 11 cm, so we set u_y to the same. The longitudinal unit is a different matter—it's metrical equivalent can be small, so the sampling

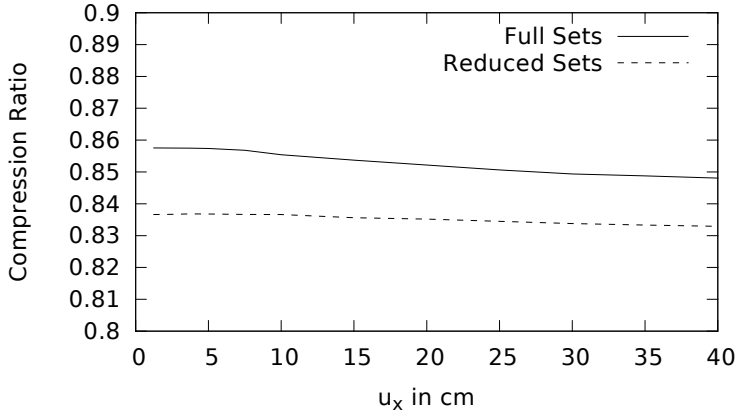
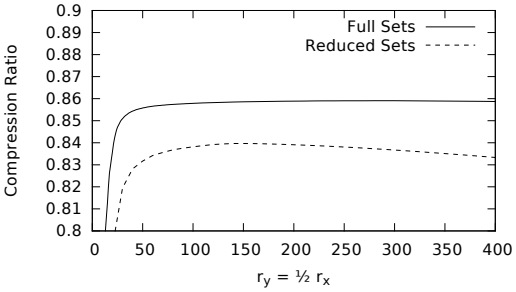
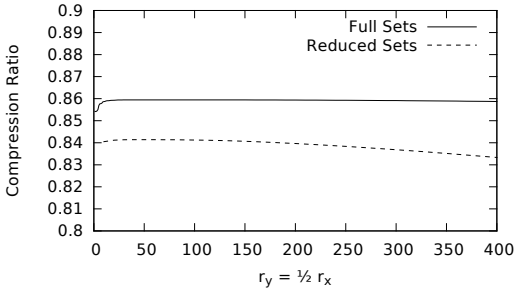


Figure 6.4: Compression vs. sampling resolution for the metrical empirical model



(a) Metrical model, simple variant



(b) Metrical model, extended variant

Figure 6.5: Compression vs. grid size for the metrical empirical model

resolution in the longitudinal direction should most certainly be higher. Figure 6.4 shows the compression ratio vs. u_x. The curve plateaus around 5 cm. Incidentally, this is the same as what we expected the average value of u_{lat} to be, which made comparing the basic and metrical model a bit easier.

Like before, we used two variations of the model; a simple one consisting only of a center grid and an outer grid as well as an extended one with multiple grid rings, arranged to cover a total area of $1601u_x \times 801u_y = 1601 \cdot 5\text{cm} \times 801 \cdot 11\text{cm} \approx 80\text{m} \times 88\text{m}$.

After choosing our grid resolution, we had to choose an appropriate size for the grid, i. e., values for r_x and r_y. Since $u_x \approx \frac{u_y}{2}$, we set $r_x = 2r_y$. As in the previous section, we end up with a single parameter that determines the grid size and optimize it via iteration. As the results in Figure 6.5 show, the relation between grid size and compression ratio is similar to the basic model's. We decided to choose the equivalent values, namely $r_y = 125$ for the simple variant and $r_y = 50$ for the extended one, to simplify comparison of the models' performance.

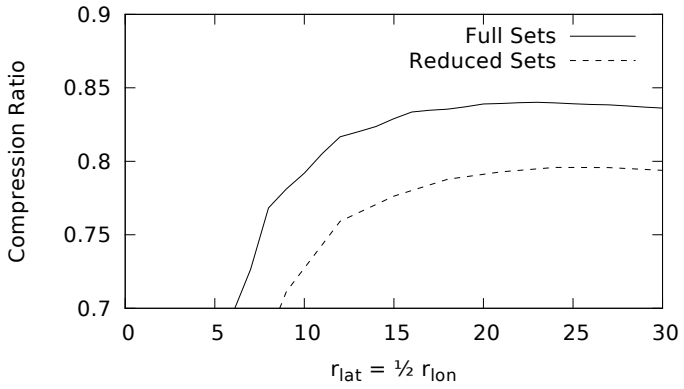


Figure 6.6: Compression ratio vs. grid size for the adaptive model without grid rings

Center grid		Counters in D_0		Counters in $D_{1,\ldots,m-1}$	
r_{lat}	r_{lon}	absolute	relative	absolute	relative
1	2	16	0.1667	80	0.8333
10	20	862	0.9170	78	0.0830
20	40	3322	0.9776	76	0.0224
50	100	20302	0.9966	70	0.0034
100	200	80602	0.9993	60	0.0007
200	400	321202	0.9999	40	0.0001

Table 6.3: The number of parameters to be trained for the center grid and the accumulated grid rings for different values of r_{lat}.

6.2.3 Adaptive Model

Unlike the trained models, the adaptive model has a lot less data to train its distribution. Using a simple model with only a center grid and an outer grid is unfeasible. As seen in Figure 6.6, the compression ratio reaches its maximum already when r_{lat} is only around 23, as opposed to the basic model, where the falloff point was at $r_{lat} = 125$. Therefore, the adaptive model relies heavily on the usage of grid rings.

In contrast to the trained models, the number of grid rings used is a concern. When using large center grids, the relative number of parameters to train for the grid rings is negligible (Table 6.3); however, if the adaptive model cannot reliably train a larger grid, it might also struggle with too many zones.

In contrast to the previous models using grid rings, we introduce a new parameter: a rate of growth $\alpha > 0$ for the width of the grid rings. The size of the first ring is defined as before, by fixed values $d_{lon}^{(1)} = 10$ and $d_{lat}^{(1)} = 5$, while the size of the i-th ring is defined recursively by

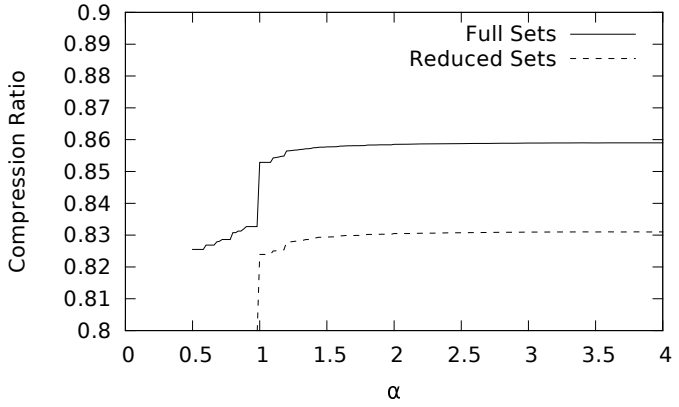
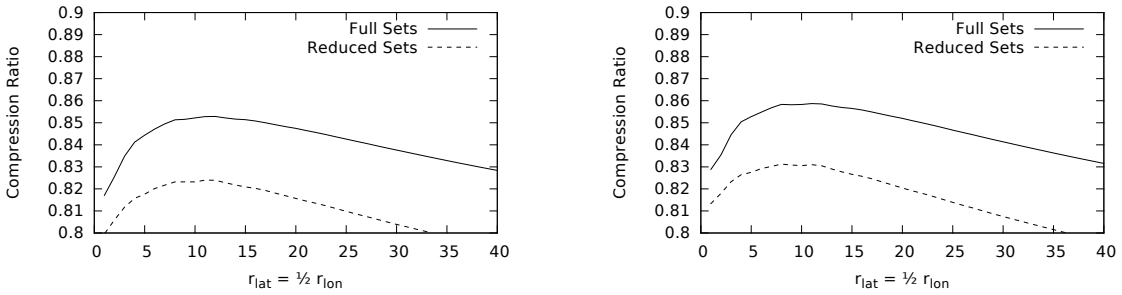


Figure 6.7: Compression ratio vs. growth rate α for the adaptive model with grid rings



(a) Adaptive model with linearly spaced zones

(b) Adaptive model with exponentially spaced zones

Figure 6.8: Compression vs. grid size for the adaptive model

$d_{dim}^{(i)} = \alpha \cdot d_{dim}^{(i-1)}$. The special case $\alpha = 1$ means that a fixed width is used for all grid rings. In all other cases, the width grows (or shrinks, if $\alpha < 1$) exponentially with increasing i.

As before, we set a total area of $1601u_{lon} \times 801u_{lat}$ to be covered by the center grid and grid rings.

Figure 6.7 shows how an exponential growth of the grid zones alters the compression ratio. Unsurprisingly, a growth rate below one performs worse, as the shrinking size leads to an increased number of zones far from the center. Growing zones lead to a notable improvement in compression ratio. We therefore chose $\alpha = 2$ to evaluate the grid size in combination with exponentially growing zones. Although higher values perform slightly better, simply doubling the zone width with each zone makes it easier to picture the grid.

As with the empirical models before, the optimal size for the center grid was determined by iteration. Figure 6.8 shows the effect of the grid size on the compression ratio. We examined

two variants with $\alpha = 1$, where zones grow linearly, as well as $\alpha = 2$, meaning doubling dimensions with each successive zone. The exponential variant performs strictly better than its linear counterpart. With a larger center grid, the difference decreases. At $r_{lat} = 40$, the difference is negligible, with both variants performing significantly worse. The optimum for both variants is around 10–13. We chose the exponential variant with $r_{lat} = 12$ for the final evaluation.

6.3 Compression Ratios

We compared the compression ratios of the different models and variations using the optimized configurations that were found during the parameter optimization. We evaluated a total of five different configurations: The basic and metrical empirical models, each with and without grid rings, and the adaptive model with exponentially growing zones. Additionally, we encoded the test set with the lossy innovation-based coder from [10] and the lossless byte-based coder from [6].

For the lossy coder, we set the discretization parameter $\varepsilon = 5\text{cm}$. ε determines the allowed error and therefore the distance between grid nodes. The dimensions of the grid were left at the default values. We used the trained model, which is similar to our basic model without grid zones, except for the usage of Cartesian coordinates.

The byte-based coder was configured to use parameters optimized for discrete vehicular trajectories at the precision of the test data. Specifically, we chose number 4 from its available profiles. Profile 4 is an improvement on the original work [6]. It was introduced by Koegel in [8]. The functionality consists of two parts: First, the discrete trajectory is converted from a series of absolute positions to a series of byte-aligned difference vectors. Next, a general-purpose compression algorithm is used for further compression. We chose the *Prediction by Partial Matching* (PPM(n)) method, which uses a distribution calculated dynamically for each byte from n preceding bytes for an entropy coding. More specifically, we used a PPM(16)-based arithmetic coder. This method yielded the best compression results in the original evaluation in [6].

Table 6.4 shows that our coding scheme performed generally better than the other two. This is true for both the full and the reduced test set. The basic model with zones achieved the best results. The relative compression ratio between coders a and b is calculated as

$$1 - \frac{1 - ratio_a}{1 - ratio_b} = 1 - \frac{compressed\ size_a}{compressed\ size_b} ,$$

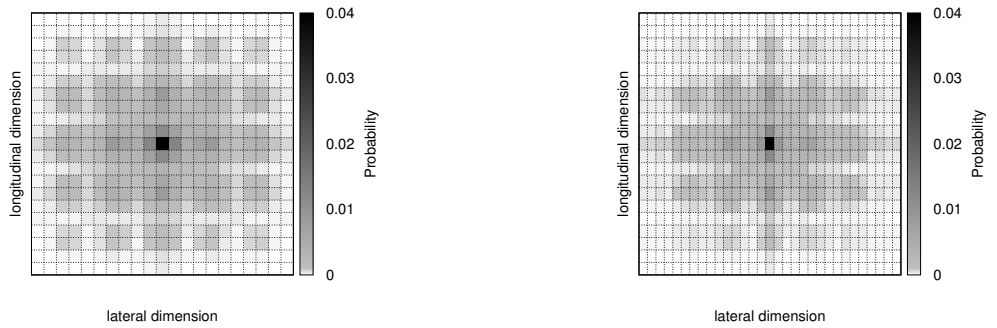
Model	Parameters	Compression Ratio Full Sets	Reduced Sets
Basic	r_{lat}=125, r_{lon}= 250	85.87 %	83.82 %
Basic-Zones	r_{lat}= 50, r_{lon}= 100	85.99 %	84.01 %
Metrical	r_y =125, r_x = 250	85.75 %	83.80 %
Metrical-Zones	r_y = 50, r_x = 100	85.95 %	84.00 %
Adaptive	r_{lat}= 12, r_{lon}=24, α=2	85.62 %	82.84 %
Lossy Coder	ε = 5cm	82.34 %	79.44 %
Byte Coder	Profile 4	82.85 %	77.06 %

Table 6.4: The compression ratios on the test sets using the optimized parameters chosen in Section 6.2

where a is our coder. The compression ratio relative to the output of the lossy coder is 20%. For the byte coder it is 18%. For the reduced sets, containing a higher variation of latitudes and shorter individual traces, the relative reduction in size was 22% and 30%, respectively.

The different probability models are very close to each other in terms of compression ratios. As was to be expected, the models with additional zones come out on top. The combination of center grid and grid rings provide fine approximations, even for larger innovations, and the usage of a smaller center grid means that the training set does not need to be as large as for the models without grid rings. There is another major advantage to the zone-based models: they use significantly less memory than their zoneless counterparts. With $r_{lat} = 50$, $r_{lon} = 100$, 70 grid rings and an EOF symbol, the model needs to hold 20302 probabilities in memory. For the simpler model without grid rings we used $r_{lat} = 125$ and $r_{lon} = 250$, resulting in 125752 counters, including the EOF symbol; the memory consumption is increased by factor six, and still the compression ratio is slightly lower.

A surprising result is that the basic models perform better than their metrical counterparts. One explanation is that none of the traces in our test data are close enough to the poles to expose the weakness of the basic model. Additionally, the traces from the OpenStreetMap project stem from a number of different sources. There might be other effects that introduce entropy which does not depend on the kinetic behavior of cars. For example, some rounding operations might have occurred at any point between GPS measurement and us downloading the data, e. g., in the GPS device itself or during the conversion to a text file. The coordinates might also have been converted from binary to decimal. These are possible sources of entropy that the metrical model is not designed for. Figure 6.9 shows the trained distributions of the basic and metrical model. The distribution from the basic model shows a distinct pattern of discretization. This pattern is most likely caused by a discrepancy between the precision of the input data and the precision of the actual measurements; specifically, the measurement precision seems lower than the precision of the numbers that store the result. The metrical



(a) Basic model

(b) Metrical model

Figure 6.9: The trained distributions of the basic and metrical model

distribution hints at the same pattern, but less distinct. In the metrical model, the same innovation can be mapped to different outcomes depending on the latitude. Therefore, the pattern of discretization is softened in the metrical model. Since the OSM project accumulates traces from heterogeneous sources, it is reasonable to use a high precision to avoid losing information from high-precision sources. In other use cases, where the precision of the input data can be tailored to the measurements, the metrical model would likely outperform the basic one. Using the coder on a larger scale, for example with planes, the innovations might be generally larger, such that the error introduced by assigning probabilities to angular deviations instead of areas becomes greater. In any case, it is safe to say that the basic model would perform poor in the closer proximity of the poles.

For the specific use case of compressing arbitrary traces from numerous heterogeneous sources at a given base of 10 and precision of 6, the zone-based basic model or the adaptive model are the best choices. While the empirical model offers a higher compression ratio, its implementation requires more work—for the trained distribution, one requires a data format and accompanying parser as well as a utility program to train a distribution and an appropriate amount of sample data. Selecting the sample data is another source for possible problems. For good results, the training set must be representative of the data that is encountered in the specific use-case. The adaptive model, in turn, needs only a few additional lines of code to increase its counters as symbols are encountered. Additionally, the adaptive model is less prone to suffer from varying latitudes, and might even achieve better compression ratios for traces in proximity of the poles.

Chapter 7

Conclusion

In this work, we successfully translated the concept of innovation-based coding from [10] to the lossless domain. We created a proof-of-concept coder that operates on geodetic coordinates, given as fixed-point numbers, and can encode and decode discrete trajectories without loss of information.

To this end, we introduced the concept of grid zones. These offer solutions to a number of different issues. The lossy coder has a limited grid size and needs to restart the coding process when a position outside the grid is encountered. It also lacks flexibility when it comes to the required size of the training set: short of reducing the overall grid size, there is no way to reduce the amount of sample data required to reliably train its models. Using grid zones, we designed models that cover the complete domain of input data, without exceptions that cause a restart of the coding process. By varying the layout and size of the grid zones, a model can be trained with a smaller set of sample traces—or even not trained at all, as the adaptive model—and still yield good compression ratios.

There was an additional challenge caused by the geodetic coordinate system that had to be used due to the lossless nature of the coding scheme: Varying length of longitudinal units, depending on the current latitude. The metrical model was introduced to solve this problem. For the use case of compressing OSM traces, we saw that the metrical model does not improve compression over the basic model. We showed that the input data contains certain patterns for which the basic model is more appropriate. We still believe that the metrical model would improve compression ratios if the input data were of higher quality, or if it included traces in proximity of the poles. Nonetheless, it is a positive result that the basic model performs so well. We expect the basic model to be sufficient for many use cases of coding vehicular trajectories.

Our evaluation showed that our lossless coder performs better than the two state-of-the-art coders from the lossy and the lossless domain. Using the real-world data from the OpenStreetMap project, we achieved relative improvements in compression of 22% and 18%, respectively. For the reduced sets, containing a higher variation of latitudes and shorter individual traces, the relative improvement was 20% and 30%, respectively.

Overall, we designed a coding scheme that surpasses the state-of-the-art coders for the vehicular domain. It features flexible models that allow trade-offs to be made between compression ratios and complexity in implementation, training data, and runtime resources. We gave a detailed explanation of how to optimize parameters for a real-world use case, such that the full potential of the models is used. Our implementation avoids floating-point arithmetic, so that it can be used to exchange data between machines of arbitrary type, instead of being constricted by the local floating-point implementation.

7.1 Future Work

The implementation that is part of this work is aimed towards scientific evaluation of the coding scheme and its probability models. It is meant as a proof-of-concept, focused on modularity, extensibility, and readability. It sacrifices usability, runtime, and memory to achieve these goals. For day-to-day usage, at least a proper interface would be required. Furthermore, a real-world implementation could focus on a single probability model and be highly optimized, in contrast to the open architecture of our implementation.

There are also possible improvements for the models themselves. For example, none of our models take the direction of the vehicle into account. The metrical model could be extended to rotate the probability grid prior to resampling the probabilities. Depending on the size and granularity of the grid, this might be an expensive operation. This could be delegated to the GPU, as resampling is one of the typical use cases that modern graphics hardware excels at. As a simpler but more inaccurate version of a direction-aware model, a context-based model with a fixed number of direction classes could be used. For each class the probability distribution would be rotated appropriately at the time of training.

The coding scheme may be generalized to cover different use cases. A fairly intuitive generalization would be the coding of discrete trajectories of smaller or greater scale than those from vehicles. For example, with the prevalence of handheld devices that include GPS receivers and of accompanying services to track and record movements of individuals, we believe the compression of such traces to become equally useful as for the vehicular domain. Tracking

handheld devices might warrant more flexible and dynamic models than cars, as a single trace could contain movements on foot, on a bicycle, in a train or car, or even on a plane, and any combination thereof.

An even greater generalization would be the design of a generic arithmetic coder for time series. While our coder is designed for the vehicular domain, is has few properties that specifically target vehicles or even the movement of objects. There are lots of physical properties—for example temperature, pressure, or light intensity—that are similar to the movement of objects in that they are to a certain degree predictable. The combination of a prediction model to calculate an innovation and an arithmetic coder with an empirically trained or adaptive probability model to code possible innovations is therefore transferable to other domains.

Bibliography

[1] The OpenStreetMap Project. Data is licensed under the Open Data Commons Open Database License (ODbL). Online resource: http://www.openstreetmap.org/.

[2] World Geodetic System 1984. Online resource: http://earth-info.nga.mil/GandG/wgs84/index.html.

[3] ISO/FDIS 19111:2002(E) Geographic information – Spatial referencing by coordinates. Technical report, International Organization for Standardization (TC 211), 2007.

[4] Ilya Baran, Jaakko Lehtinen, and Jovan Popovic. Sketching Clothoid Splines Using Shortest Paths. *Computer Graphics Forum*, pages 655–664, 2010.

[5] Bob Carpenter. Arithcode project: Compression via arithmetic coding in java. version 1.1, 2002. Online resource: http://www.colloquial.com/ArithmeticCoding/.

[6] Andreas Diesterhöft. Byte-Kodierung und verlustfreie Kompression von Fahrzeugbewegungsdaten. Bachelor's Thesis, Heinrich-Heine-Universität Düsseldorf, January 2012.

[7] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, dec 1973.

[8] Markus Koegel. *A Long Movement Story Cut Short - On the Compression of Trajectory Data*. PhD thesis, Heinrich Heine University, Düsseldorf, Germany, January 2013.

[9] Markus Koegel, Wolfgang Kiess, Markus Kerper, and Martin Mauve. Compact Vehicular Trajectory Encoding. In *VTC '11-Spring: Proceedings of the 73rd IEEE Vehicular Technology Conference*, May 2011.

[10] Markus Koegel, Matthias Radig, Erzen Hyko, and Martin Mauve. A Detailed View

on the Spatio-Temporal Information Content and the Arithmetic Coding of Discrete Trajectories. *Mobile Networks and Applications*, 2012.

[11] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. http://www.inference.phy.cam.ac.uk/mackay/itila/book.html.

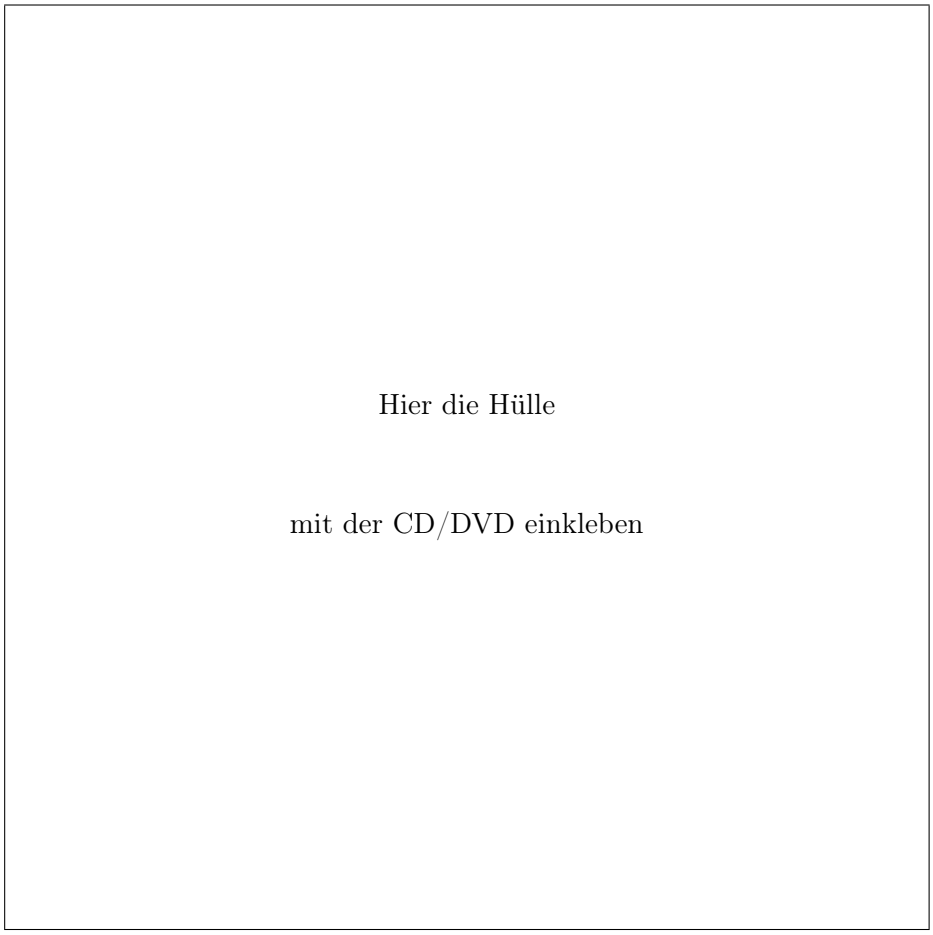
[12] James McCrae and Karan Singh. Sketching piecewise clothoid curves. *Computers & Graphics*, 33(4):452–461, 2009.

[13] Claude Elwood Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, jul 1948.

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 26. Februar 2013 Matthias Radig



Diese DVD enthält:

- eine *pdf*-Version der vorliegenden Masterarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Masterarbeit samt aller verwendeten Skripte
- die Quelldateien der im Rahmen der Masterarbeit erstellten Software (Module geocoder und tools)
- die Quelldateien des benutzten arithmetischen Kodierers von Bob Carpenter (Modul arithcoder)
- die zur Auswertung geschriebenen Skripte und deren Ergebnisse
- die Websites der verwendeten Internetquellen