Intelligent Systems and Robotics

Diploma Thesis

A Three Dimensional Map Format for Autonomous Vehicles

Paul Czerwionka

Matr. 3773050

Supervisor: Prof. Dr. Raùl Rojas
Assisting Supervisor: Dipl.-Inf. Fritz Ulbrich

Institute of Computer Science, Freie Universität Berlin, Germany

September 29, 2014

I hereby declare to have written this thesis on my own. I have used no other literature and resources than the ones referenced. All text passages that are literal or logical copies from other publications have been marked accordingly. All figures and pictures have been created by me or their sources are referenced accordingly. This thesis has not been submitted in the same or a similar version to any other examination board.

Berlin, September 29, 2014

(Paul Czerwionka)

Abstract

Autonomous vehicles of the future need maps with much more information than provided by the existing street maps. State of the art navigation systems already have digital maps, but they lack in detail, accuracy and assume a flat world. Based on the existing two dimensional map format *RNDFGraph* our autonomous car "Made In Germany" has been using, this thesis shows the development process and application of a more detailed format covering the third dimension, the Advanced Road Network Definition Graph (*ARNDGraph*). Over 300 km of Berlin's road network were mapped and autonomously travelled. The experiments include runtime and file size comparisons between the *RNDFGraph* and the *ARNDGraph*.

Contents

List of Figures

Acronyms

ABS Antiblock Braking System. 1

ACC Adaptive Cruise Control. 1

AVUS Automobil-, Verkehrs- und Übungs-Straße. 18

BGL BOOST Graph Library. 9, 10, 24

BMBF Federal Ministry of Education and Research. 7

DARPA Defense Advanced Research Agency. 7

DGPS Differential Global Position System. 2, 8

DMU Distance Measurement Unit / Odometry. 8

EBD Electronic Brakeforce Distribution. 1

ESC Electronic Stability Control. 1

GPS Global Positions System. 2, 3, 8, 13

ICC International Congress Centrum Berlin. 18, 20, 27

IMU Inertia Measurement Unit. 3, 8

INS Inertia Navigation Systems. 3

MiG Made in Germany. 5

OROCOS Open Robot Control System. 8

OSG Open Scene Graph. 8, 14

OSM Open Street Map. 9, 15

RDDF Route Definition Data File. 9

RNDF Route Network Definition File. 9–11, 14

RTT Real-time Toolkit. 8

SLAM Simultaneous Localization and Mapping. 9

SpoB Spirit of Berlin. 5

WGS84 World Geodic System 1984. 2, 13

CHAPTER 1

Introduction

The first paved roads were built in ancient Egypt approximately 4600 years ago [1]. They were needed to transport the massive stone slabs from the quarries to the building sites of the pyramids. Even though technology has advanced and our roads now are now built with much more sophistication, they still serve the same purpose of making transportation more efficient. It is no surprise, the first known road map was also drawn in ancient Egypt: The Turin Papyrus Map dates back to 1180 BC. It depicts the location of stone quarries, a gold mine and a settlement. Annotations about these places and the distances between them are also included. [2]

Egyptian hand drawn maps linked art, science and craftsmanship into the practice of cartography. Whoever had the most accurate maps outclassed their competition by being able to pinpoint their location and find the quickest and/or safest route between two locations.

Today, the most commonly used maps are road maps. But roads are not used to push stone slabs to the tombs of a ruling class. Roads are used for everyone's benefit. Individual and public transportation was jump started by the invention of the first mass produced car in 1908.

Cars have not changed much since their invention. Yes, they became faster, bigger and heavier, but they still need to be operated by a human being. The Vienna Convention on Road Traffic in 1968 states:

> 1. Every moving vehicle or combination of vehicles shall have a driver.
> ...
> 5. Every driver shall at all times be able to control his vehicle ...
> [3]

Cars will not change from being driven by a human to being driven by a computer from one day to the other. These changes will be subtle, however, they have already begun. In the past, pressing the brake pedal in a car meant transferring the physical force to the brake shoes. In today's cars, a myriad of computer systems like Antiblock Braking System (ABS), Electronic Brakeforce Distribution (EBD) or Electronic Stability Control (ESC) manage the physical force. The interface (brake pedal) stayed the same, but the actual movement of the brake shoes changed to computer control. More advanced drive assistance systems like Adaptive Cruise Control (ACC) regulate the car's speed by regulating gas and brake.

Automatic car parking systems control the steering wheel, while the driver is only operating gas and brake. All the car's actuators (steering wheel, throttle and brake) are already controlled by computers. Most driving decisions are made by the driver.

It will take years before autonomous vehicles are allowed to operate on a public road without human supervision; this is recently a topic of much discussion. A proposed amendment to the Vienna Convention on Road Traffic is on its way, stating that driver assistance systems may control the vehicle as long as the driver is able to intervene at any time.

Transportation dictates the design of the modern city and autonomous vehicles will change this design. These new cars will still need roads, but will not need parking spaces close to where we want to go. An autonomous vehicle can drop us off at our destination and then find a parking space by itself, or use the time when we don't need it to refuel, clean or transport somebody else. Self driving cars will have a massive impact on transportation in the future and therefore will also change the design of the modern city. But even autonomous vehicles will need maps.

1.1 Motivation

Maps are still used for two main purposes, localization and navigation. Localization helps to situate ourselves by comparing the landmarks seen around us with the landmarks given on the map. Navigation guides us from one location to another in an unfamiliar environment.

The map format I am presenting will cover both aspects. Because this map is primarily read by autonomous robots it needs to be digital. A graphical representation is not needed for the robot, however, to find errors quickly and to edit the map by humans a graphical representation together with a graphical interface for editing will be introduced.

1.1.1 Localization

Localization is the process of comparing the landmarks or features you see with the landmarks or features drawn on the map. The more precisely you can detect these landmarks the more precisely you can tell where your position on the map is.

Usually landmarks used in maps are immovable objects like mountains, towers, etc.. A compass orients us towards particular landmarks. This procedure is tedious and prone to errors.

The Global Positions System (GPS) revolutionized the process of localization. Instead of comparing the surrounding with the map, the GPS gives absolute coordinates in the World Geodic System 1984 (WGS84), which can be translated into a location on the map. However the mean error of the GPS position is more than ten meters. The position is calculated by measuring the travelling time of the signals from satellites to the GPS receiver. The travelling time is influenced mainly by atmospheric effects including weather. This influence can be measured and minimized by using the Differential Global Position System (DGPS) which works by having fixed locations and known positions. By calculating the delta of the GPS signal and the known position, atmospheric errors can be measured and transmitted to any other GPS receiver in the same area. Which in return use this measured error for a more precise position estimate.

Another challenge, especially in urban environments, is called the multipath problem. Sometimes there is no direct line of sight to a satellite and the signal is reflected by high-rise buildings, resulting in a signal with longer travelling time. GPS is a very easy way to get a rough estimate of your position, but the position can vary greatly in accuracy. Sometimes, for example in tunnels, no GPS signal can be received at all and other means of localization are necessary.

Odometry is another way of tracking the car's position, but because of wheel slip and wheel spin it is not very precise. Inertia Navigation Systems (INS) or Inertia Measurement Unit (IMU) also work very well for a short time, but they suffer the same problem as odometry. Since the current position is derived from previous positions, and the error is cumulative, accuracy worsens significantly over time. Because road markings are immovable and can be tracked very precisely with cameras[4] or lidar sensors[5], they are almost perfect to be used as landmarks in a map. The only issue is they are not very distinguishable from each other and a global localization would give ambiguous results. By combining the best GPS positions with odometry, inertia systems and the tracking of lane markings a localization in the map within centimetres at any time can be done.

1.1.2 Navigation

Navigating a vehicle through today's traffic is a very complicated task and starts before the car even begins to move. It starts by planning a path from point A to point B through a network of streets. This must also be done without breaking traffic laws such as, travel on one way streets, making turns where allowed and not crossing unbroken lane markings.

To plan an optimal path, whether it be the shortest or the quickest path, a map of the street network is needed.

Usually the task of navigation starts when we decide which route to take to get from our starting point to our destination. A route consists of a sequence of roads, including left, right and maybe even u-turns. Planning a route more precisely, it includes which lane to take, and whether it is unusual to make a right turn from the left most lane on a multi lane road. Human drivers plan ahead. If we know a right turn needs to be made we make sure we are already in the right most lane way before we reach the intersection. For bigger and more complicated intersections we might have multiple lanes for left or right turns. An anticipating driver will choose the lane depending on which lane he has to be in after the turn. The task of navigation includes not only the choice of which road to take, but also which lane to drive in.

Another reason a map is needed a priori is that we normally drive faster than our breaking distance permits. The way streets are built we are already driving so fast where we have to assume the road is going to continue behind the corner because we will not be able to stop otherwise. Also in dense urban traffic other vehicles may block your line of sight to the lane markings. This part of the map is not needed before we start to drive, but it is needed once we start moving and following the road.

The information about how the street is shaped is used for multiple purposes. It is used to improve the localization of a vehicle and is also needed to drive high speed. As the car increases speed it needs to look ahead further and needs to know the curvature of the road

before it is in the corner. For comfortable driving the preferred speed in a curve is calculated by estimating the centrifugal acceleration. For safety the car should enter the curve with the maximum speed the car should have inside the curve. So the curvature of the road should be known to the car before it enters the curve. Sensors often can not detect the road ahead far enough. Therefore an accurate map is needed.

1.1.3 Project Requirements

For testing purposes, especially the behaviour module or the controller, we needed to be able to drive in a controlled environment. Luckily Berlin has an abandoned airport in the city center, the Tempelhof Airport. Pre World War II it was one of Europe's iconic airports. During the Berlin Blockade in 1948/1949 it was part of the Berlin Airlift which fed two million Berliners for almost a year. In 2008 the airport was closed and most of the area is now a public park. It hosts trade shows, music festivals or in our case proved to be a perfect testing ground for our autonomous vehicles. Virtual maps of artificial road networks were created for Tempelhof airport test manoeuvres or scenarios in early stages of development. Though these maps were very small, they still allowed us to test multiple crossings, right of way behaviour, traffic lights and blocked roads. In preparation to drive on the German autobahn these maps included long straights, long curves and several adjacent lanes for driving up to 120 km/h, doing lane changes and overtaking slower vehicles.

Some requirements resulted not only from the technical needs but also from the perspective of software development. With more than 20 active developers and only one car, not everyone could test his part of the software in the car. Also testing with the actual car was very time consuming. Having a simulator sped up software development immensely. One of the general requirements of the project was to have a simulator and a simulator would not have been of much use without a map.

As any other map the digital map should represent the real world only as detailed as needed. If it is too detailed, the creation of these maps would consume more time than they should. Also the application programming interface should not be overly complicated which will not only waste time in developing itself but also everyone's time who will have to use the module. Especially if the module is used by many other developers. With these decisions in mind, we decided on a two dimensional map.

Our car detected obstacles using fusioned lidar and radar sensors[6] and a stereo camera system. To avoid collisions with obstacles on the road the robot either matched the speed accordingly or even stopped in front the obstacle.

While driving autonomously in Berlin on a two dimensional map situations were encountered which could not be handled by the flat road representation. The first one we encountered on the autobahn. The highway has a small slope upwards a hill. Behind the hill was a sign mounted above the autobahn and spanning across its whole width. As the car drove up the slope at a certain point it aligned itself with the sign. From the point of view of the car the sign was recognized as static obstacle in the middle of the autobahn and the car started to slow down. Continuing on the slope a couple of meters later, as the car pitched down the sign above the autobahn and was no longer classified as an obstacle.

This was only one part of the problem. The other one was the car could not "see" what was

behind the hill. Having a three dimensional map will not help the car to see through hills, but at least the car would know it cannot see that part of the road and drive appropriately, for example by not doing overtaking manoeuvres.

With the help of three dimensional maps these problems could be solved.

Looking at 2.6 closely an autobahn underpass was modelled. Because the map has no height information it is impossible for the car at the crossing to tell if it positioned on top of the autobahn or if it is inside the underpass. With the *RNDFGraph* this was handled by also taking the cars orientation into account. This solution is far from elegant and only works if roads don't have the same orientation. In other cases like a multilevel highway or a parking garage it is still ambiguous on which level the car actually is.

1.2 Contribution

Based on the existing two dimensional map format for the autonomous vehicles Spirit of Berlin (SpoB), Made in Germany (MiG) and e-Instein I will present a new three dimensional map format. It can be used to model urban street networks and highways specifically for autonomous vehicles. By using splines to interpolate the road, the map will have a small file size while still being more accurate than commercial navigation systems for cars.

The three dimensional map will aid:

- the robot in deciding which sensor can "see" which parts of the road more precisely
- estimating the cars stopping distance, this can be done more accurately since going uphill or downhill is a very big factor
- the vehicles driving on multistory highways or in multistory parking structures
- in a more energy efficient driving

Additionally already existing two dimensional maps will be converted to the new format and the graphical editor will be changed so not only the lateral and longitudinal positioning can be modified but also vertical positioning.

CHAPTER 2

Preliminaries and Related Work

The following chapter will explain the context that the three dimensional map format was developed and tested in.

2.1 The Autonomos Project

At the Freie Universität Berlin Raul Rojas has been building autonmous robots since 1998. After many successful participations in the RoboCup, winning the World Championship twice and the European Championship five times, it was decided to expand. In 2006 the decision was made to participate in the Grand Urban Challenge. A retrofitted drive-by-wire dodge caravan was bought, already modified so a disabled person could operate it using only a joystick and a touch pad. With only a small budget and 6 months of development time, the car named "Spirit of Berlin" (or short SpoB) made it into the semi finals of a competition held by the Defense Advanced Research Agency (DARPA).

After returning to Germany, additional funding was granted by the Federal Ministry of Education and Research (BMBF) and the Autonomos Labs were born.

So far the project has worked on three different cars:

Spirit Of Berlin (SpoB)

The first car in which the autonomous team participated at Grand Urban Challenge in 2007. Even though the car still exists it has been stripped of most of its sensors.

Made in Germany (MiG)

The most expensive car was called "Made in Germany" or MiG. It was a Volkswagen Passat Variant 1,8l TSI 2010 Model , a 6th Generation Passat, modified by the Volkswagen AG to our specifications. The sensor setup was very extensive and included 7 radars on various frequencies (26-77 GHz) and pointed every direction. 6 Ibeo Lux laser scanners were incor-

Figure 2.1: The Autonomos Labs Car Family; from left to right: Spirit of Berlin, e-Instein and Made in Germany

porated into the body of the car. A Velodyne 64 HDL-E was mounted on the roof. Two guppy color cameras were used for traffic light detection and a black and white high dynamic range stereo camera system was used for lane and obstacle detection. A fifth camera was used for a secondary lane detection system. An Applanix POS LV consisting of an IMU, Distance Measurement Unit / Odometry (DMU) and a DGPS was also used.

e-Instein

The latest member of the autonomous car family was not only fully electric, but it also featured fewer, less expensive sensors to reduce costs. The e-Instein was a Mitsubishi iMiev and due to its zero emissions and small size it can be driven indoors. The sensor setup was much slimmer compared to MiG. It only had one Ibeo Lux lidar, one radar facing forward, a stereo camera system facing forward and a Velodyne HDL 32 on top. An Applanix POS LV is used for GPS localization and as an inertial measurement unit. The main focus of this vehicle was to reduce the number of sensors needed for safe autonomous driving in different situations including indoor.

2.2 Software Framework

Most of the software was written in C and C++. The open source software frame work Open Robot Control System (OROCOS) was used throughout the project. Components were developed in conjunction with the Real-time Toolkit (RTT). Multiple components to receive data, process data and manipulate data including data fusion between different sensors, were chained together. In this chain of data handling, display modules were also added to show the raw or processed data and to debug the process at different stages. These display modules used the Open Scene Graph (OSG) to show various data in the same three dimensional environment. For the map almost no data needed to be processed; a simple singleton gave other modules access to the map and no complex processing chain was needed. The map only used OSG to display the road and additional map data in the

same environment the sensor data was displayed. The map modules itself used the BOOST Library, especially the BOOST Graph Library (BGL) and the Boost Serialization.

2.3 Other Map Formats

Simple occupancy grids and landmark maps as used in Simultaneous Localization and Mapping (SLAM) approaches[7] only contain geometric information. While geometric information was needed, topological information was also needed. Information like which lanes are connected to each other, special purpose lanes, if lane changes are allowed, relations between traffic lights and lanes, speed limits, traffic signs and so on are required to obey traffic rules.

Open Street Map (OSM), on the other hand, provides mostly topological information with too limited geometric information and accuracy.

Proprietary formats used by TomTom, Navigon and other car navigation devices are not openly documented and therefore could not be used.

The only open and freely available format was OpenDrive [8]. OpenDrive is developed by the VIRES Simulationstechnologie GmbH in cooperation with Daimler Driving Simulator. Their scope is to describe road-network for the use in a simulator. Which was one of our requirements, but even more important was the ability to model real road networks. Lanes can be marked as being in a tunnel or on a bridge. The format also supports tilting and even cross fall of roads, but no real up and down. Thus over- and underpasses cannot be built.

Recently Bender et al. presented *lanelets*[9] as a map format for autonomous vehicles, which looks very promising. Their approach shows some similarities to our two dimensional map format, as in separating geometrical and topological information.

2.4 RNDF

The Route Network Definition File (RNDF) format was developed for the Grand Urban Challenge held in 2007. RNDF is a successor to the Route Definition Data File (RDDF) which was used in both of the earlier Grand Challenges 2005 and 2006. Both formats share the same concept: use latitude and longitude to describe way points. A sequence of those way points, with the addition of a width, marks a travel corridor in RDDF or a road in RNDF. The exact shape of the corridor or the road is not defined. The RNDF Documentation explicitly states:

> "For some road segments, the RNDF will specify sparsely placed way points. Within these segments, the implied vehicle behavior requirement is to use road-following techniques to stay in the appropriate travel lane, finding the drivable path to the next checkpoint." [10]

It is notable that every participating team in the Grand Urban Challenge 2007 was given the RNDF file and a satellite picture before challenge, and every team had the opportunity the make changes to the file.

> "The editing required 3 h of a person's time. In an initial phase, way points

were shifted manually, and roughly 400 new way points were added manually to the 629 lane way points in the RNDF." [11]

Therefore every team was able to modify the given RNDF to their interpretation of the RNDF format.

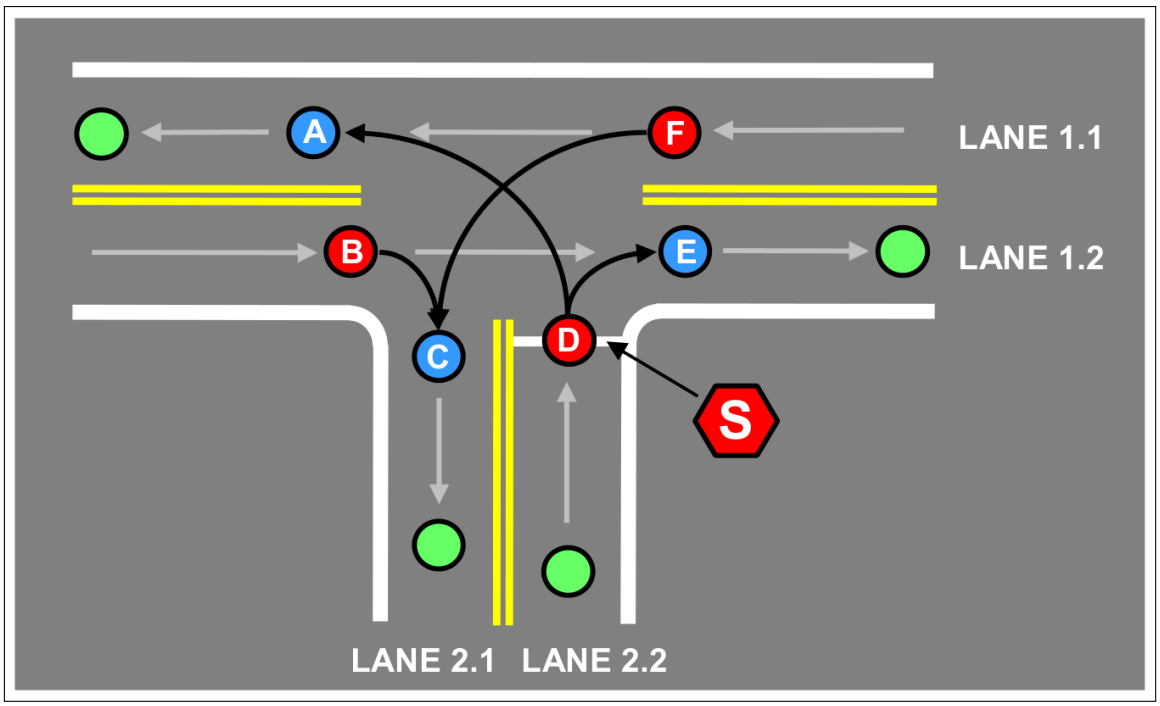

Figure 2.2: Interpretation of an intersection in the RNDF format. Letters mark vertices, gray arrows mark edges in a sequence of way points and black arrows mark connecting edges. The vertex labelled D has a stop sign. Picture taken from the RNDF documentation.

2.5 RNDFGraph

The RNDFGraph is an object graph model based on the RNDF format. It has been implemented using the BGL. While it's origins are the RNDF format minor parts have been changed and additional information was added.

Like with RNDF, a sequence of way points mark a road lane. We start off with a simple, directed, weighted graph. Each vertex of that graph corresponds to a way point in the sequence. Each vertex is connected to its successor by a directed edge. The edge weight is set to the distance between the way points.

Intersections are modelled with exit and entry way points. Any way point can be marked as an exit way point with another way point being an entry way point. These two way points are also connected with an edge in the graph. The edge is called a connecting edge and will be discussed later.

2.5.1 Improvements to RNDF

RNDF was created specifically for the Grand Urban Challenge as a route format which represents an artificial urban route network. In this artificial urban route network, the robots had to navigate from one checkpoint to the next. While the RNDF was never intended to be used to model real world road networks it was decided to use the already implemented format and to extend the format where needed. This was primarily done in my already published work in 2011 [12]. The major improvements were:

1. Only stop signs were defined, missing any other signs and traffic lights.
2. Roads were modelled as a sequence of way points with no interpolation of points in between.
3. Segments had no real use except for holding lanes together and defining speed limits.
4. Lanes could only be connected by connecting edges which did not have any properties.
5. Lanes always had a constant width.
6. Lanes were not further categorized, e.g. turning lane, parking lane, highway entry and exit.
7. Lanes had no association with their neighbouring lanes; every lane had its own sequence of way points.
8. Left and right lane boundaries could only be double yellow, solid yellow, solid white or broken white.
9. It was very difficult to model traffic islands or roundabouts.
10. The RNDF is two-dimensional and did not support bridges, tunnels or overpasses.

[12]

All of these points have been covered in "Optimized Route Network Graph as Map Reference for Autonomous Cars Operating on German Autobahn" except 9. and 10. This thesis will especially cover 10.

2.5.2 Splines

The most simple solution would be to sample the road every 10 centimetres and save these points. This would lead to accurate maps, but also a lot of data would have to be stored. In *RNDFGraph* the course of the road is mimicked with spline interpolation. By using splines, only the support points for the spline need to be stored, and every point between the support points can be interpolated. The challenge, however, is to place the support points in such a way, so the resulting spline interpolation matches the course of the road. Support points are also the link to the graph. Every vertex in the graph is also a support point for the spline. For further reference a spline in the *RNDFGraph* context is called the *laneSpline*.

Splines were originally thin wooden strips held in place by lead weights at support points. That way the wooden strips provided an interpolation for the points in between. They where mostly used in construction of boat hulls and air planes. With the invention of computers, splines became known as piecewise polynomial functions. While linear spline interpolations change their gradient at support points, quadratic and cubic spline interpolations offer

smoother changes in curvature. Bezier splines need a high degree polynomial to fit a complex curve. To find a fitting high degree polynomial linear equation systems need to be solved which leads to a higher computational effort.

Akima Spline Interpolation

Though a number of spline interpolations were available, for *RNDFGraph*, the Akima Spline Interpolation[13] was chosen due to some unique features.

The Akima Spline Interpolation produces curves which are close to manually drawn, continuously derivable curves. The first derivative can be interpreted as the direction of the road.

The Akima Spline Interpolation defines a cubic polynomial between every support point.

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, x \in [x_i, x_{i+1}] \tag{2.1}$$

Further the slope t is defined for a support point with the slope between the previous two and next two support points.

$$t_i = \frac{|m_{i+1} - m_i|m_{i-1} + |m_{i-1} - m_{i-2}|m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|} \tag{2.2}$$

Therefore the slope at any support point can be calculated. To interpolate the polynomial for every $x \in [x_i, x_{i+1}]$ the coefficients a_i, b_i, c_i, d_i need to be determined.

a_i is straight forward by evaluating S_i for x_i:

$$S_i(x_i) = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3 = a_i = y_i \tag{2.3}$$

b_i can be found by evaluating the derivative of S_i for x_i and 2.2 .

$$S_i'(x_i) = b_i + 2c_i(x_i - x_i) + 3d_i(x_i - x_i)^2 = b_i = t_i \tag{2.4}$$

c_i and d_i are a bit more complicated. Two more equations are needed. S_i for x_{i+1}

$$S_i(x_{i+1}) = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 = y_{i+1} \tag{2.5}$$

and the definition of a slope

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \tag{2.6}$$

combining them results in

$$m_i = \frac{a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 - y_i}{x_{i+1} - x_i} \tag{2.7}$$

since $a_i = y_i$ and $b_i = t_i$

$$m_i = \frac{t_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3}{x_{i+1} - x_i} \tag{2.8}$$

$$m_i = t_i + c_i(x_{i+1} - x_i) + d_i(x_{i+1} - x_i)^2 \tag{2.9}$$

so

$$c_i(x_{i+1} - x_i) = m_i - t_i - d_i(x_{i+1} - x_i)^2 \tag{2.10}$$

and

$$d_i(x_{i+1} - x_i)^2 = m_i - t_i - c_i(x_{i+1} - x_i) \tag{2.11}$$

Taking a closer look at the derivative of $S_i(x_{i+1})$:

$$S_i'(x_{i+1}) = b_i + 2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 = t_{i+1} \tag{2.12}$$

again $b_i = t_i$

$$2c_i(x_{i+1} - x_i) + 3d_i(x_{i+1} - x_i)^2 = t_{i+1} - t_i \tag{2.13}$$

in combination with 2.11 gives:

$$2c_i(x_{i+1} - x_i) + 3(m_i - t_i - c_i(x_{i+1} - x_i)) = t_{i+1} - t_i \tag{2.14}$$

$$\Leftrightarrow c_i = \frac{3m_i - 2t_i - t_{i+1}}{(x_{i+1} - x_i)} \tag{2.15}$$

combining 2.13 and 2.10

$$2(m_i - t_i - d_i(x_{i+1} - x_i)^2) + 3d_i(x_{i+1} - x_i)^2 = t_{i+1} - t_i \tag{2.16}$$

$$\Leftrightarrow d_i = \frac{-2m_i + t_i + t_{i+1}}{(x_{i+1} - x_i)^2} \tag{2.17}$$

In conclusion a_i, b_i, c_i and d_i can be calculated by knowing the gradients t_i, t_{i+1} and x_i, x_{i+1},m_i. Replacing t_i and t_{i+1} with 2.2 and m_i with 2.6 all the coefficients of S_i can be calculated with the values of $x_{i-2}, .., x_{i+3}$ and $y_{i-2}, .., y_{i+3}$.

The Param

Usually a spline $S(x) = y$ is defined with $x \in X$ being along the x-axis and $y \in Y$ being along the y-axis in a coordinate system. With the *RNDFGraph*, this is not to be confused with the coordinate system the map is using.

To avoid confusion the x component it called *param*. The *param* is used to address any point on the spline. It is defined as the sum of all euclidean distances along the sequence of support points. The *param* at the first support point x_0 is 0. The *param* at the second support point x_1 is the euclidean distance to the first. The *param* of the third support point x_2 is the euclidean distance from the first to the second plus the distance from the second to the third.

The *param* can be described recursively:

$$\begin{aligned} &param(x_0) = 0 \\ &param(x_n) = param(x_n - 1) + ||S(x_n), S(x_{n-1})|| \end{aligned}$$

Y is the maps coordinate system and every $y \in Y$ can be transformed to a WGS84 coordinate used by the GPS and vice versa. So every y is actually a tupel (a, b) marking a point in the graphs coordinate system.

2.5.3 Meta Information

To follow the traffic rules additional information is needed. This meta information is added to the *VertexData* and *EdgeData* objects. Each vertex and each edge has a corresponding data object.

Information added to edges is street type, such as highway, acceleration lane, deceleration lane, round about or street. Speed limits were added to edges. Type of the lane boundary is also noted (e.g. double yellow, broken yellow, broken white, solid white, solid yellow).

A vertex can be marked as a traffic light, stop sign or right of way sign. The lane width can also be set for every vertex. With the RNDF format it was only possible to set the road width for the whole lane, by moving this information from the lane context into the vertex context it is now possible to change the lane width at every vertex.

The most common use case for the autonomous car is to find the closest lane to the cars current position. Just searching for the closest support point does not give accurate results. The problem is two fold; first the closest spline needs to be found and then the closest point on the spline needs to be determined. To find the closest spline efficiently, a bounding box for each polynomial is calculated and stored in a spatial k-d tree. This data structure allows us to find the closest polynomial in logarithmic time depending on the number of support points in our map. Using Brent's Method the closest point on the spline is found. Brent's Method combines inverse quadratic interpolation, the secant method and the bisection method to find zero of a function [14]. In our case it is used to find the closest point on a spline with a given minimum accuracy and since the autonomous car subsequently searches for a point close to the last point the earlier found index is stored. The next search on the same polynomial will start from the stored index.

In the *RNDFGraph* a voting system is implemented. Each edge votes their likelihood of being the neighbour edge of every other edge.

So far there is no connection between neighbouring lanes in the graph. If two edges are adjacent to each other and lane boundaries stored in the *EdgeData* indicate lane changes are allowed, so called lane changing edges are inserted at the beginning and at the end of these edges, connecting each other. The weight of lane changing edges is considerably higher than the weight of normal edges to avoid unnecessary lane changes.

2.5.4 Creation of RNDFGraph Maps

The *RNDFGraphEditor* is a graphical interface to create *RNDFGraphs* from scratch or to edit existing maps.

Like other display modules in the autonomous project the, *RNDFGraphEditor* is implemented using the OSG Library. All the sensor data can be displayed while editing the map. Also the same display module used to display the *RNDFGraph* while driving is used in the *RNDFGraphEditor*.

The process to create a new map from scratch is as follows:

1. Drive the to be mapped road manually while logging with at least one camera, the applanix, the velodyne, and at least one lane detection system.

2. With the *RNDFGraphEditor* it is possible to play and seek in the log file and visualize all acquired sensor data.
3. It is also possible to add Google Maps or OSM as an underlying texture.
4. By combining Google Maps, the lane detection system and the velodyne data, it is very simple to draw lanes.
5. The images from the cameras make is possible to watch for traffic signs and set speed limits.

2.6 Existing RNDFGraph Maps

In the past over 60 maps have been created using the *RNDFGraph*. In the following text a few examples are given. The maps can be divided into ones used for testing certain modules or features on private areas or in the simulator and therefore are artificial and those ones which model real road networks.

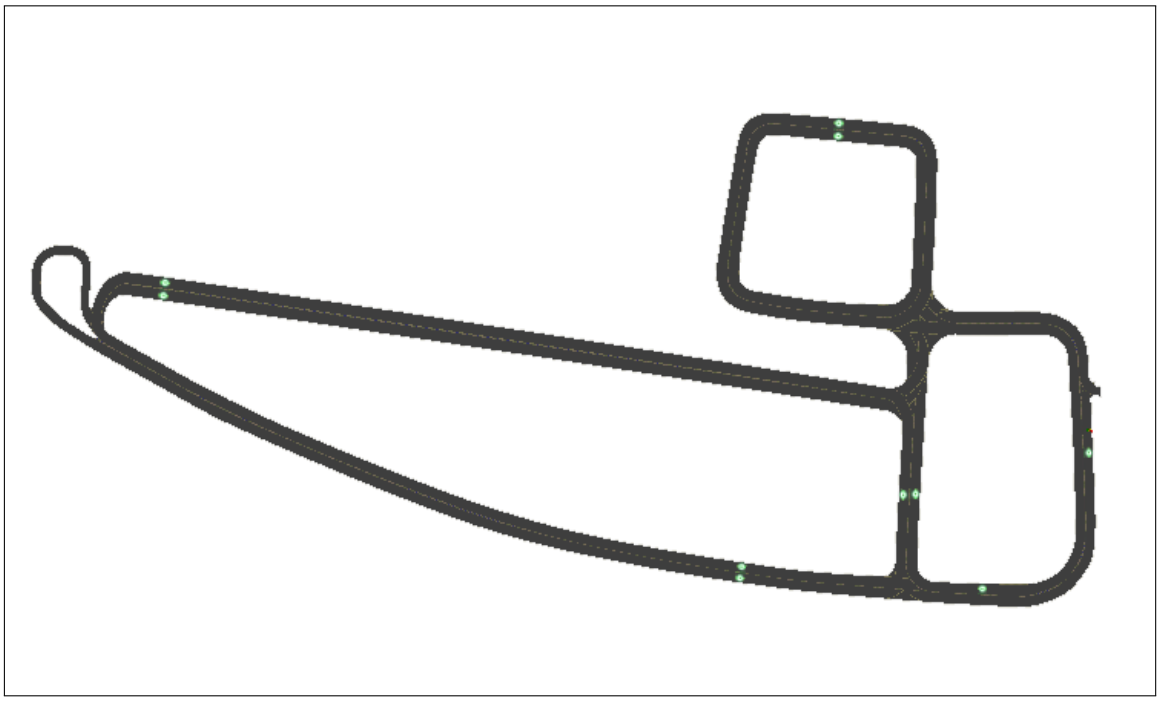

Figure 2.3: THF Presentation ; this map is used to test the car in different urban traffic scenarios. Traffic lights, other cars and cyclists, pedestrians, handling of construction sites and "left before right" was tested with this map before taking the car into real traffic.

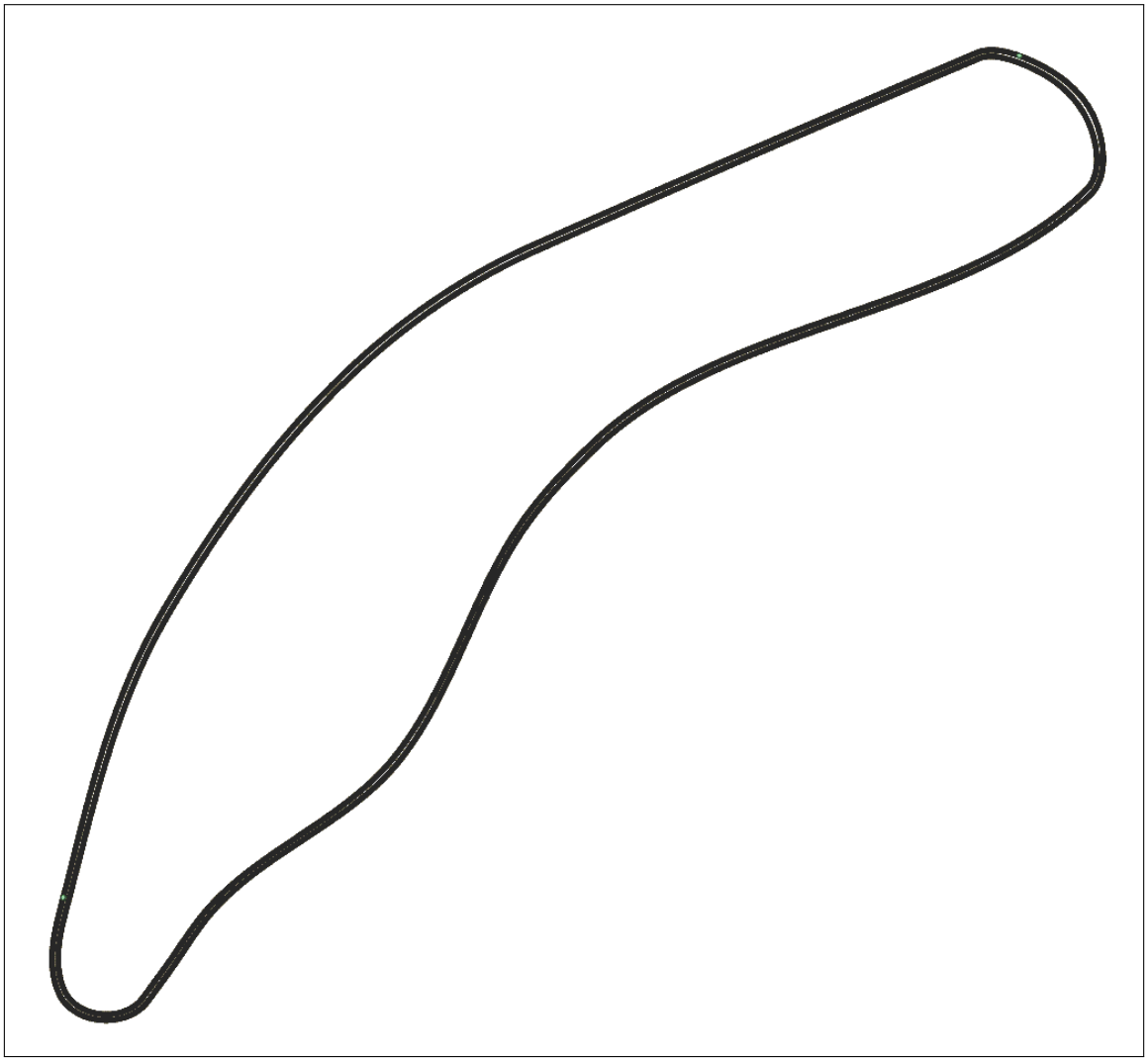

Figure 2.4: THF Slalom ; this was specifically designed to test the lateral and longitudinal controller at high speed (70 - 120 km/h). While we were not allowed to drive above 100 km/h in public traffic we wanted to make sure the controllers have enough leeway. Its main features were a long left curve, a long right curve and a tight bend at the end where the vehicle had to slow down to make the corner.

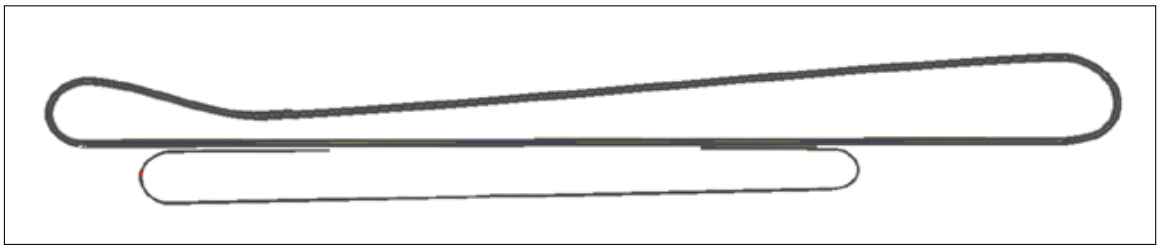

Figure 2.5: THF Speedway ; is used to test the behavior module in more complex highway situations. It features two parallel lanes, an acceleration lane and a deceleration lane. The two parallel lanes are used to test lane changes at high speeds (70 km/h - 120 km/h). In the acceleration lane, the car has to match the speed of the moving traffic on the autobahn before merging.

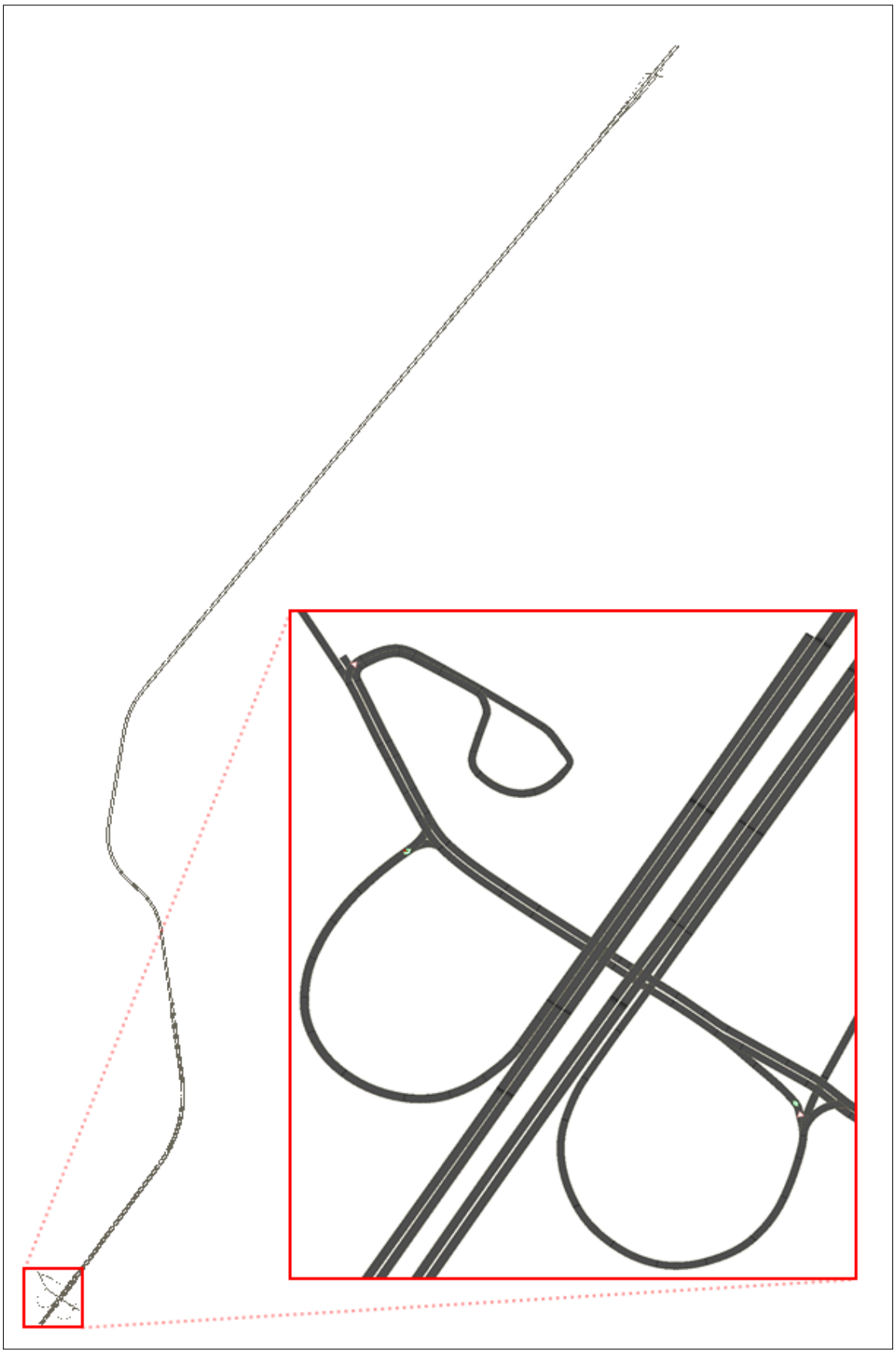

Figure 2.6: AVUS ; The AVUS in the south western part of Berlin is the oldest highway in Europe. While being a public highway, until 1998 it was also used as a racetrack. It connects the ICC and the Messe Berlin fairgrounds to the German autobahn network. Needless to say it is one of the busiest highways in Germany. It's close proximity the the Freie Universität Berlin makes it the perfect testing grounds for our autonomous cars.

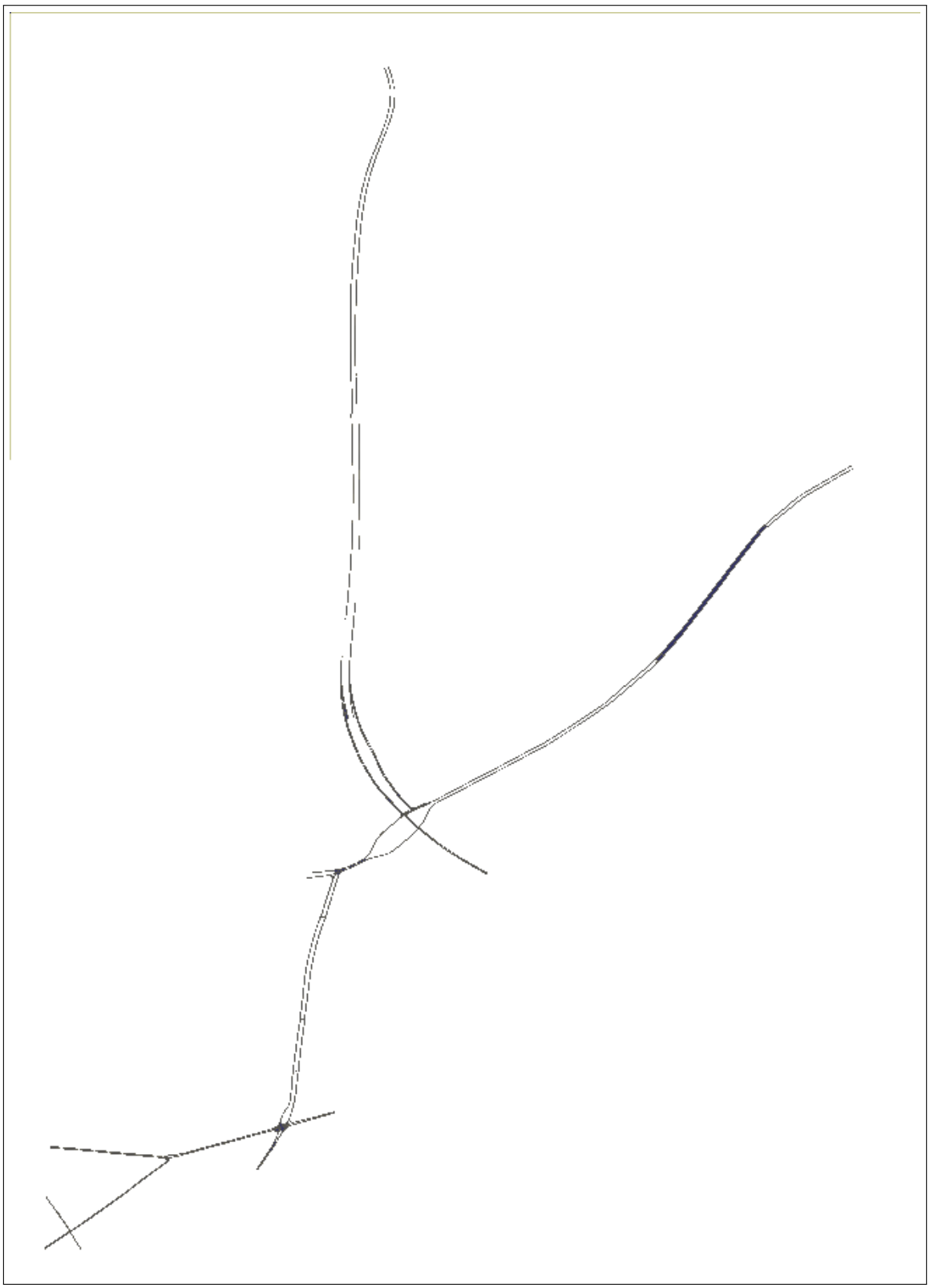

Figure 2.7: Dahlem ; The Dahlem map represents the area close to the Computing Studies Faculty featuring mostly small streets with and without traffic lights and sometimes cobble stone roads with no lane markings.

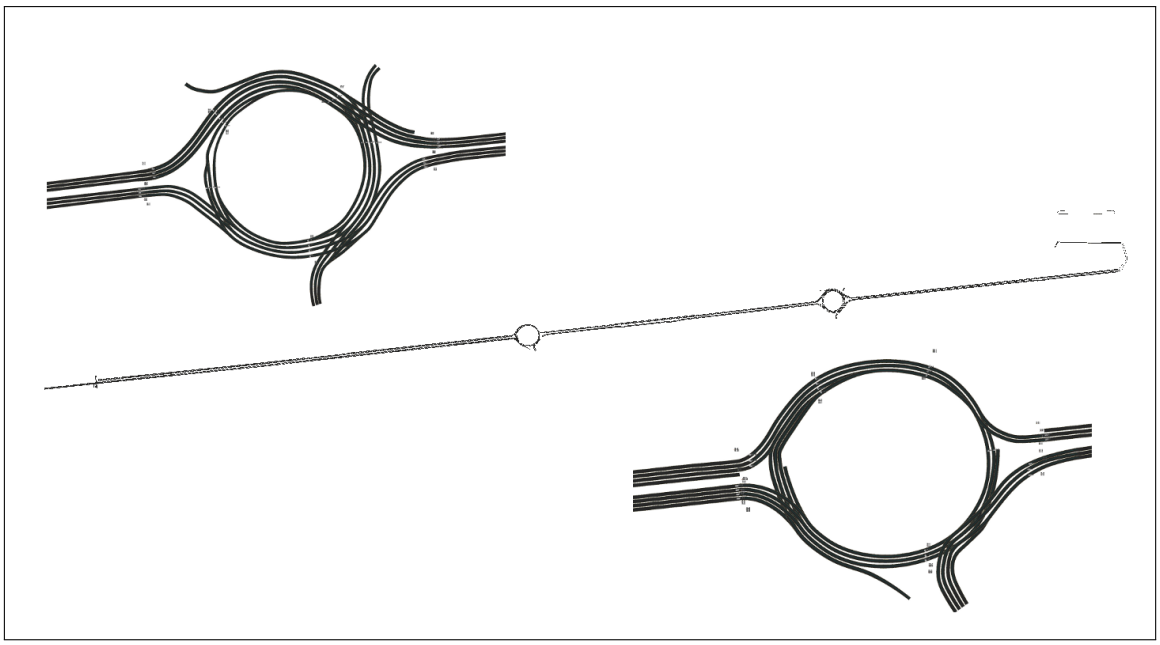

Figure 2.8: Kaiserdamm ; This map shows the road network from the ICC to the Brandenburg Gate and back. It includes two big roundabouts "Ernst Reuter Platz" and "Siegesäule" and more than 50 traffic lights. The challenge on this road is detecting all traffic lights, even the ones inside the roundabouts which are located at unusual angles and making two lane changes in very busy inner city traffic. This challenge was handled and successfully demonstrated at the Autonomous Press Conference in 2011[15].

CHAPTER 3

Advanced Road Network Definition

While a lot of problems were solved with the *RNDFGraph*, some still remain. Not all of the earlier described problems can be solved by simply making a three dimensional map.

The scope of this work is the three dimensional map itself, which is the starting point to the solution of the posed problems. The map format developed in this thesis is called Advanced Road Network Definition, or short *ARNDGraph*.

Many other modules need to be changed in order to benefit from the new map format. This work will focus on developing the map format itself and creating a graphical editor. Also the other modules will be modified to work with the new format. However utilizing the new features in the various modules will be part of future work.

The challenge here lies with not rewriting every module, but keeping the adjustments to other modules reasonable.

3.1 RNDFGraph3D

As a first proof of concept step, the *RNDFGraph* was converted to a *RNDFGraph3D*, by simply adding a third component to the earlier described way points. The way points of the *RNDFGraph* way points are defined by x and y coordinates and in the *RNDFGraph3D* way points are described in x,y and z coordinates. As shown in 2.5.2, the same equations can be used regardless of the number of dimensions the coordinate system uses. Through use of templating, only minor changes in the spline interpolation library were implemented.

While this easy approach solves the "sign on top of the hill problem" and the "multilayer road problem", it creates other problems. For example curves on a autobahn: In a curve, the highway also banks in the same direction the curve is turning to, e.g. on a left curve the left most lane is lower then the right most lane. If this was modelled using *RNDFGraph3D* it would look like a pair of steps because the lanes do not tilt sideways. This could be solved by tilting the lane, somewhat like OpenDrive[8] does, but it would complicate the process of modifying lanes. Changing one lane results in changing width and tilting of adjacent lanes.

For my approach I have chosen a different way.

3.2 ARNDGraph

With the *RNDFGraph*, the middle of the road was modelled. The road surface was described by going perpendicular to the road direction to the left and to the right. While this is very easy when constructing a road map using a recorded trajectory of the vehicle, on downside the resulting map does not really describe the road surface but the path the vehicle took.

To describe the road surface more exactly, the format I am introducing does not model the middle of the road. Instead, the left and right road boundaries are described. Usually these boundaries are lane markings, but they can also be a curb or no boundary which simply marks the end of the road surface.

This change will also make refining the map easier; for example, if the left lane marking needs to be moved, while the right lane marking stays in the same position. With the *RNDFGraph* the only way to do this was to move the *laneSpline*, and thus also moving the right lane marking. The only way to counter that movement was to change the lane width. By describing the lane through left and right lane boundaries refining the map is much easier.

As mentioned earlier, when the road ascends and descends, tilting the road becomes necessary. While other approaches describe lane surface tilting with an angle, if the lane surface is described as the area between two lane boundaries, the lane surface implicitly tilts when one boundary is higher then the other one.

An optimized RNDFGraph could not be edited. In the RNDFGraphEditor it was only possible to move vertices. But during the optimization process vertices are removed which only purpose was to keep a support point.

3.2.1 Improvements to RNDFGraph

Changing the two dimensional spline to three dimensional turned out to be quite simple. Since the Akima Spline Interpolation only takes into account a weighted gradient of the two points before and the two points after, only the calculation of the gradient had to be changed from two dimensional to three dimensional. Fortunately the BOOST Lbrary already has optimized functions to calculate the gradient in three dimensions.

With the *RNDFGraph*, graph and splines are tied closely together. Every support point for a spline is also a vertex in the graph and vice versa. But they serve different purposes. For instance, if the speed limit changes, a new edge must be added to the graph. To add a new edge, a new vertex and support point must also be added. However, by adding a support point the curvature of the spline is changed. So in a worst case scenario other support points need to be adjusted to refit the spline to the curvature of the road. So we end up changing support points because the speed limit changed.

Also the other way around is "not needed". For example, if the road curvature changes, we need to add additional support points to make the spline fit the curvature. However, by adding another support point we add another vertex and enother edge, even though the *EdgeData* does not need to be changed. So we end up adding support points and edges where they are not required.

The *RNDFGraph* optimization technique minimizes the problem of unneeded edges. By separating support points and vertices the optimization becomes obsolete.

But if we would not tie these two different purposes together we would not need to optimize the graph at a later stage. By being able to choose more carefully where edges start and end we could end up with a graph that has even less edges and vertices than an optimized *RNDFGraph*.

With the *RNDFGraph* a vertex is described by the x and y coordinate of its corresponding support point, and spline. We change the definition of a vertex to a parameter on the spline. The parameter describes the distance from the start of the spline, where the parameter is 0, to a given point on the spline. A spline given a parameter returns the x,y and z coordinate of that point on the spline.

Having the car follow a given spline is done by a "low level" controller. This controller handles the delay between giving commands to the car's steering motor and the car actually turning. Since we have a *driveSpline* for every lane, following the road is done by the low-level controller. The decision on which *driveSpline* to drive is done by a higher-level "authority", in part of the software we call the behaviour. It uses the graph to navigate from the car's current position to a given destination. To calculate the quickest path, well known path finding algorithms can be used. In our case we use the Johnson's All Pairs Shortest Path Algorithm[16] once the map is loaded, saving the results in a VxV matrix. The weights of the edges are estimations on how quickly the car can travel each edge. This step is called the macro plan generation. Once that is done we have a list of edges the car needs to travel on to get to the destination. As mentioned earlier each edge has a *driveSpline* which is given to a the low level controller which then makes sure the car travels on these splines.

However, while driving along the planned path, the plan can change: for example in a highway scenario another vehicle is driving slower than our car wants to travel it will initiate a lane change. Before a lane change is done the car checks if its destination is also reachable from the lane it wants to change to and that the time-to-destination is not significantly greater than before.

Also while driving along a spline, a micro plan is generated. The micro plan first consists only of the cars trajectory generated with the *driveSpline* of the edge. If there are obstacles close to the *driveSpline*, the trajectory can be smoothly altered while still keeping within the lane boundaries.

The driveSpline

The drive Spline is the suggested trajectory a vehicle should take while driving on a lane. While technically the spline has no constraints, the *ARNDGraph* makes sure the *driveSpline*'s support points are always between the left and the right boundary of an edge. Also the support point's z-axis component has to be on the plane between the two boundaries.

Using the intercept theorem, the drive spline support points are moved onto the plane between the left and the right boundary spline.

Since the *driveSpline* can be different for different vehicles, i.e. a motorbike will have different constraints than a 40 ton truck, it's inclusion into the map is questionable. In our case it is

included because the behaviour relies on it. A rework of the behaviour module could remove that dependency.

The Graph

With the *RNDFGraph* splines and graph are closely connected. Every spline support point is also a vertex in the graph and vice versa.

The Graph is implemented using the BGL[17]. This way all the graph algorithms in the Boost Library can be used. The BGL itself uses a weighted directed adjacency list to store the graph.

Like any graph, the *ARNDGraph* consists of edges and vertices. In our case the edges represent a lane or a stretch of a lane. While the splines are used to interpolate the exact curvature of lane markings, the edge represents a connection between two locations.

Edges are used to assign logical information about the lane. Every edge has an *EdgeData* object stored in a `boost::property_map`. The *EdgeData* object holds type of road (street, highways, roundabout, e.t.c.), speed limits, pointers to the left and right boundary spline and the *driveSpline*. The edge weight must be be stored in a separate map, however, so the BOOST graph algorithms such as Johnson's All Pairs Shortest Path Algorithm[16] can be used to calculate the shortest path between all vertices.

In the *ARNDGraph* two new boundary types were added: curb and no boundary.

A vertex represents a specific point on a lane. Since every edge needs to be connected to two vertices, they are also start and endpoints of edges.

As for edges, for every vertex a *VertexData* object is stored in a `boost::property_map`. The *VertexData* stores meta information about that location. Such items are road signs, traffic lights, checkpoints, give way signs or decision points for the braindriver project[18].

Neighbour Lanes

With the *ARNDGraph*, a lane is neighbour of another lane if they share a boundary spline. It is possible to define neighboured edges in the *ARNDGraphEditor*.

For the car to change lanes, it needs to know if a lane has a neighbour lane. For that purpose we define: A lane is neighbour of another lane if they share a boundary. In our terms that means a lane is the neighbour of another lane if the left boundary spline of one lane is the right boundary spline of another.

With that information we can navigate from one lane to another, but standard graph algorithms like Djikstra[19], Bellman Ford[20], etc. will fail because they can not "relate" to the above neighbourhood relationship. For that purpose, lane changing edges are added to the graph. These edges connect the vertices at the beginning and the end of neighbouring edges. If a lane change is allowed in both directions, two lane changing edges are added. If it is only possible to change from one edge to the other, but not back, only one edge is added in the direction where the lane change is allowed. A penalty is added to the weight of these edges to ensure a lane change is only done when needed.

3.2.2 Converting RNDFGraph to ARNDGraph

Over 60 maps using the *RNDFGraph* have been created, reusing these maps is a requirement. Therefore, a conversion from *RNDFGraph* to *ARNDGraph* was implemented. The conversion was done as follows:

1. the *laneSpline* was copied to the *driveSpline*
2. left lane boundary spline was calculated by going half the lane width orthogonal to the laneSpline to the left, the same was done for the right lane boundary spline
3. *VertexData* was copied
4. *EdgeData* was copied

3.2.3 ARNDGraphEditor

To change the *ARNDGraph* simply and quickly, a graphical interface called the *ARNDGraphEditor* was developed. It was implemented using the Open Scene Graph Library and therefore should be easily portable to Windows or Mac OS X.

In continuation of separating geometrical and topological data, the *ARNDGraphEditor* features two different edit modes: Spline Edit to change geometrical data, and Edge Edit to change topological data.

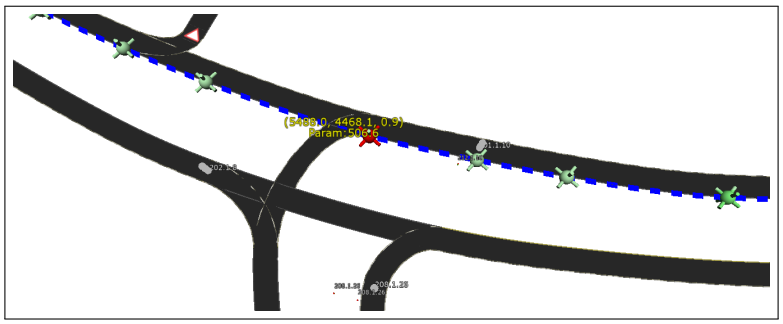

Figure 3.1: ARNDGraphEditor, Spline Edit Mode ; In Spline Edit mode each support point is highlighted with *Cursor3D*. They can be moved by dragging the *Cursor3D* to a new position.

In the Spline Edit mode the support points of every spline, can be moved. These support points can not only be moved on the x and y axis but also on the z axis. To be able to conveniently do this, a *Cursor3D* is added to the Open Scene Graph. It consists of a sphere with three cylinders, one cylinder for every axis x, y and z aligned in the corresponding direction and a sphere in the middle. If the mouse cursor is pointed at the sphere or the x, y cylinder, the *Cursor3D* can only be moved in the xy plane. If the mouse cursor is pointed at the cylinder representing the z axis the cursor can only be moved along the z axis. That way support points can easily be moved in three dimensional space.

Edge Edit mode is used to change the topological information. In this mode edges on the same spline can be merged into one edge or one edge can be split into two edges. Vertices

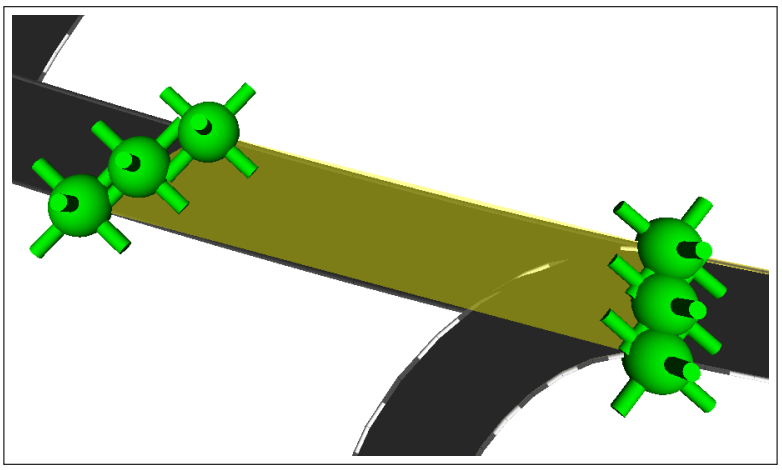

Figure 3.2: ARNDGraphEditor, Edge Edit Mode ; In Edit Edge Mode the start and end param value for left, right boundary and *driveSpline* can be modified by dragging the green *Cursor3D*.

can be marked as being a traffic light, a stop sign or a right of way sign. Connecting edges can be added between vertices. The left, right and drive spline *param* can be moved along the corresponding spline.

While this edit mode should only change topological information this is not entirely the case. Usually when moving a vertex along a spline, the spline itself and the geometrical information does not change. However, if the vertex has incoming or outgoing connecting edges, the splines of the connecting edge have to be changed. For this reason some geometrical information has to be changed in Edge Edit Mode.

CHAPTER 4

Experimental Results

4.1 Experiments and Evaluation

For the experiments the *RNDFGraph* of Berlin is converted to an *ARNDGraph* and manually optimized by merging unneeded edges.

This map covers parts of the city's highways and urban streets in close proximity to the Free University of Berlin, as well as the heavy traffic street Kaiserdamm from the ICC to the Brandenburg Gate. This was done by combining the *RNDFGraphs* shown in Figure 2.6, 2.7 and 2.8 into a single graph. The total length of all edges is about 331 km. Roughly two thirds of the map (229 km of all edges) are highways. The remaining third is classified as streets and roundabouts. The graph is weakly connected. Interesting is the number of edges after the automatic optimization techniques described in [12] compared to the number of edges in the *ARNDGraph*. The overall size of the file was also noted, however, since the *ARNDGraph* will need about two to three times as many splines than the *RNDFGraph*. An increased file size in that magnitude should be expected. The file size containing the *RNDFGraph* was 1 301 011 bytes, while the file size with the *ARNDGraph* was 3 623 891 bytes. The difference is within the expected magnitude.

To compare the two graphs, the length of all edges was divided by the number of edges. This metric estimates how many meters of road were covered per edge on average. Because edges used as urban streets are much shorter than edges for highways, the same metric is used again but only for edges used as highways and only for edges used as streets.

The average length of any edge using *ARNDGraph* was higher than the average length of any edge in an optimized *RNDFGraph*. As a result the ARNDGraph needed less edges to cover the same road network.

For localization the most important functionality is to find the closest point on a spline for a given point. Since this function is used very frequently it is imperative to be very fast. To measure the time, this function was called 1 million times with random points. The fastest, the slowest and the average time were recorded and compared between *RNDFGraph* and *ARNDGraph*. As execution time always depends on the processor and memory used these experiments were conducted on a MacBook Pro 6,2 with a Intel(R) Core(TM) i7 CPU M

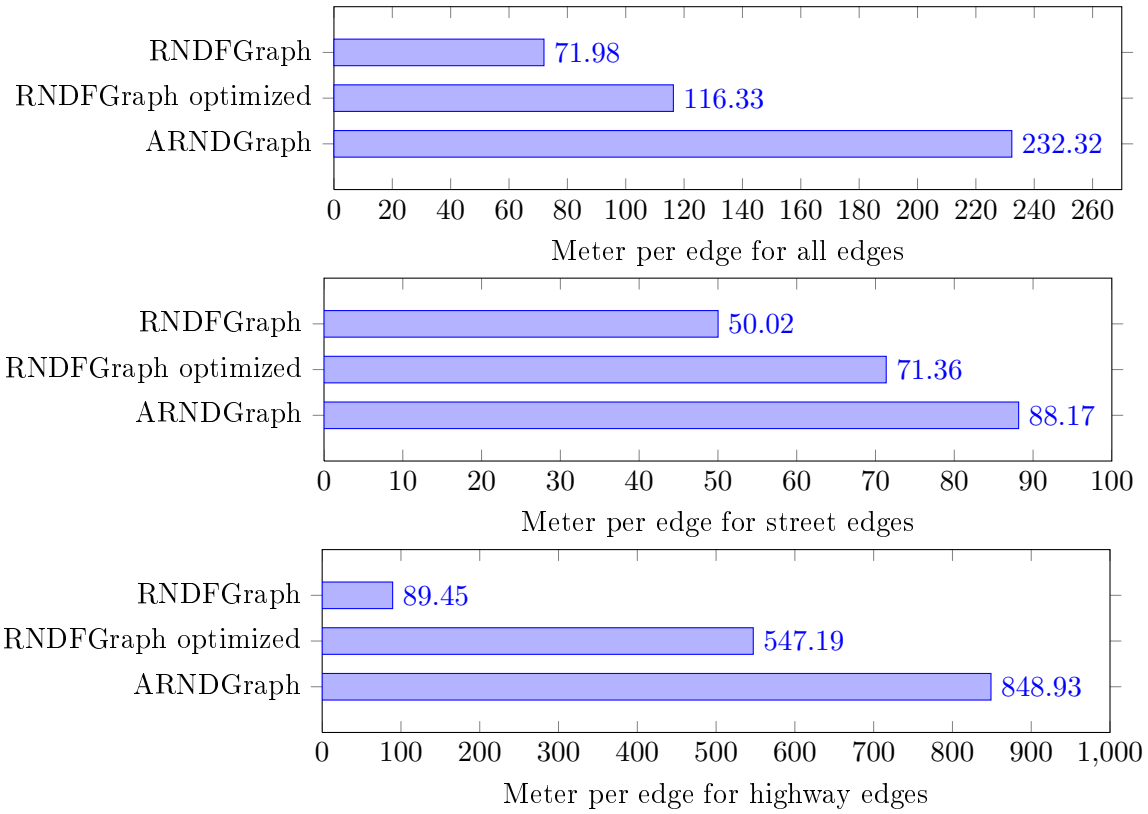

Figure 4.1: Meters of Road Covered per Edge; Overall average edge length, average length for edges classified as street and average length of edges classified as a highway were compared.

620 running at 2.67GHz with synchronous DDR3 memory at 1067 Mhz. A native installed Linux was used with a 3.16.0 kernel.

Figure 4.2 shows the runtime to find the closest point on a spline for a given random point. While the *ARNDGraph* is a bit slower it fully supports three dimensional coordinates.

Of course the autonomous car "Made in Germany" also successfully navigated and continuously localized itself through Berlin using the *ARNDGraph*. The car planned its route from the outskirts of Berlin to the Brandenburg Gate in the center of the city, recognizing traffic lights, planning lane changes and switching from the highway to inner city streets.

All the artificial maps explained earlier were also successfully converted to *ARNDGraphs* and can be used.

4.2 Conclusion

Every *RNDFGraph* map can be converted to an *ARNDGraph* map. Using the *ARNDGraphEditor*, the map can be further refined and three dimensional information can be added.

Localization can now be accurately done on multilevel highways or parking garages.

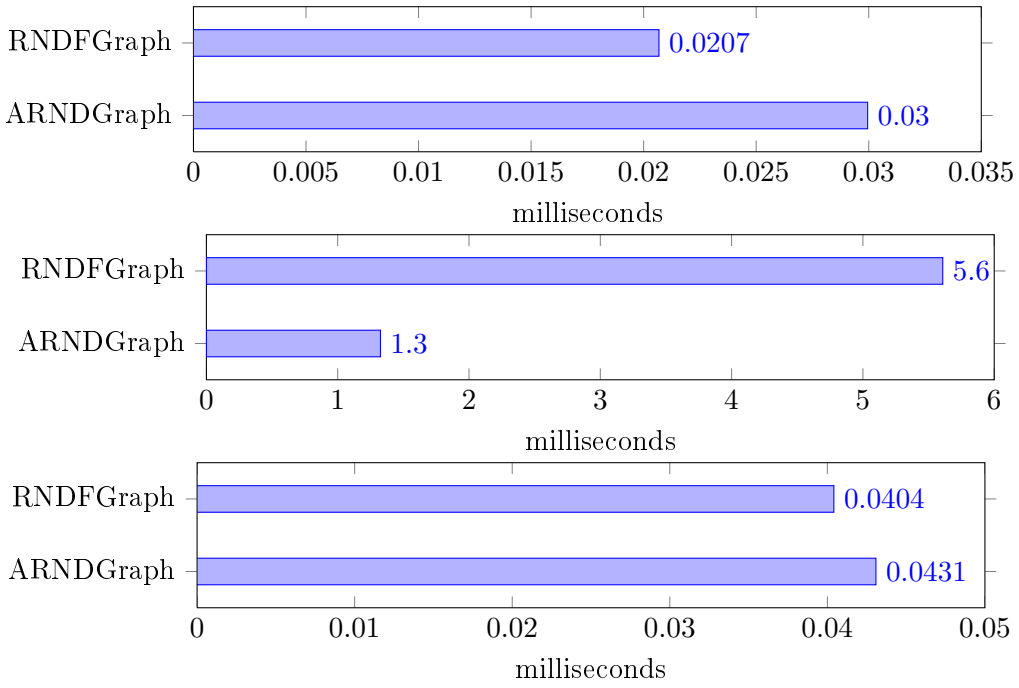

Figure 4.2: Runtime to find closest point on spline; minimum, maximum and average time.

Many other methods of map building use three dimensional point clouds for mapping. This however generates massive amounts of data since every point needs to be stored. With our method only a few points need to be saved. We only need to save the support points, the rest of the points can be interpolated to a given accuracy or maximum error.

With the *ARNDGraphEditor* it is now possible to manually optimize the graph. I have shown the manual graph optimization is superior to the automatic optimization used with the *RNDFGraph*.

4.3 Future Work

The now three dimensional map format offers a multitude of future work.

While the map is now three dimensional and the road can ascend and descend the currently implemented obstacle detection and tracking still only features two dimensional obstacles with two dimensional positions. To successfully perceive an obstacle as being above the road, the obstacle detection and tracking also needs to work in three dimensional space. An iteration on obstacle perception can now be done.

To plan a route, more energy efficient road ascending and descending slopes can now be taken into account.

Optimal and comfortable driveSpline calculation can now consider slope, tilting, left and right lane boundaries and curvature. The driveSpline can also depend on the vehicles features. A truck might have a different optimal driveSpline than a car, which in turn would also be different from the one of a motorcycle.

Automatic refinement of an existing map can also be researched. How many support points are needed for a maximum error? It depends on what kind of a road is to be mapped. After mapping the AVUS it became apparent that highways need much less support points than streets. To work out a good balance between the number of support points and the necessary accuracy would be another research topic.

Because the ARNDGraph has a very small memory footprint the map can be transmitted very quickly. Synchronizing the same map across multiple cars, maybe a cloud based approach, would also be a very interesting future work.

For now the map only covers parts of Berlin. If the maps should cover the whole city or even long distance drives to other cities the graph needs to be optimized even further.

Bibliography

[1] J. N. Wilford. World's oldest paved road found in egypt. http://www.nytimes.com/1994/05/08/world/world-s-oldest-paved-road-found-in-egypt.html, 1994.

[2] J. A. Harrell and V. M. Brown. The oldest surviving topographical map from ancient egypt: (turin papyri 1879, 1899, and 1969). *Journal of the American Research Center in Egypt*, 29:pp. 81–105, 1992.

[3] United Nations. Vienna convention on road traffic. http://www.unece.org/fileadmin/DAM/trans/conventn/crt1968e.pdf, 1968.

[4] M. Aly. Real time detection of lane markers in urban streets. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 7–12. IEEE, 2008.

[5] P. Lindner, E. Richter, G. Wanielik, K. Takagi, and A. Isogai. Multi-channel lidar processing for lane detection and estimation. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, Oct 2009.

[6] D. Göhring, M. Wang, Michael Schnurmacher, and T. Ganjineh. Radar/lidar sensor fusion for car-following on highways. In *ICARA'11*, pages 407–412, 2011.

[7] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[8] Opendrive format specification, rev. 1.3. http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.3D.pdf, 2010.

[9] P. Bender, J. Ziegler, and C. Stiller. Lanelets: Efficient map representation for autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, June 8-11, 2014*, pages 420–425. IEEE, 2014.

[10] Defense Advanced Research Projects Agency. Route network definition file (rndf) and mission data file (mdf) formats. http://archive.darpa.mil/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf, 2007.

[11] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 2008.

[12] P. Czerwionka, M. Wang, and F. Wiesel. Optimized route network graph as map reference for autonomous cars operating on german autobahn. In *ICARA*, pages 78–83. IEEE, 2011.

[13] H. Akima. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM*, 17(4):589–602, October 1970.

[14] R. P. Brent. *Algorithms for Minimization Without Derivatives*, volume "Chapter 4: An Algorithm with Guaranteed Convergence for Finding a Zero of a Function". Prentice-Hall, Inc., Englewood Cliffs, New Jersery, 1973.

[15] Autonomos press conference 2011. `http://www.idw-online.de/pages/de/news440476`, 2011.

[16] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, pp. 636–640, 2nd edition, 2001.

[17] The Boost Graph Library (BGL). `http://www.boost.org/doc/libs/1_56_0/libs/graph/doc/`.

[18] M. Waibel. Braindriver: A mind controlled car. `http://spectrum.ieee.org/automaton/robotics/robotics-software/braindriver-a-mind-controlled-car`, 2011.

[19] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[20] R. Bellman. On a routing problem. *Quart. Appl. Math.*, 16:87–90, 1958.