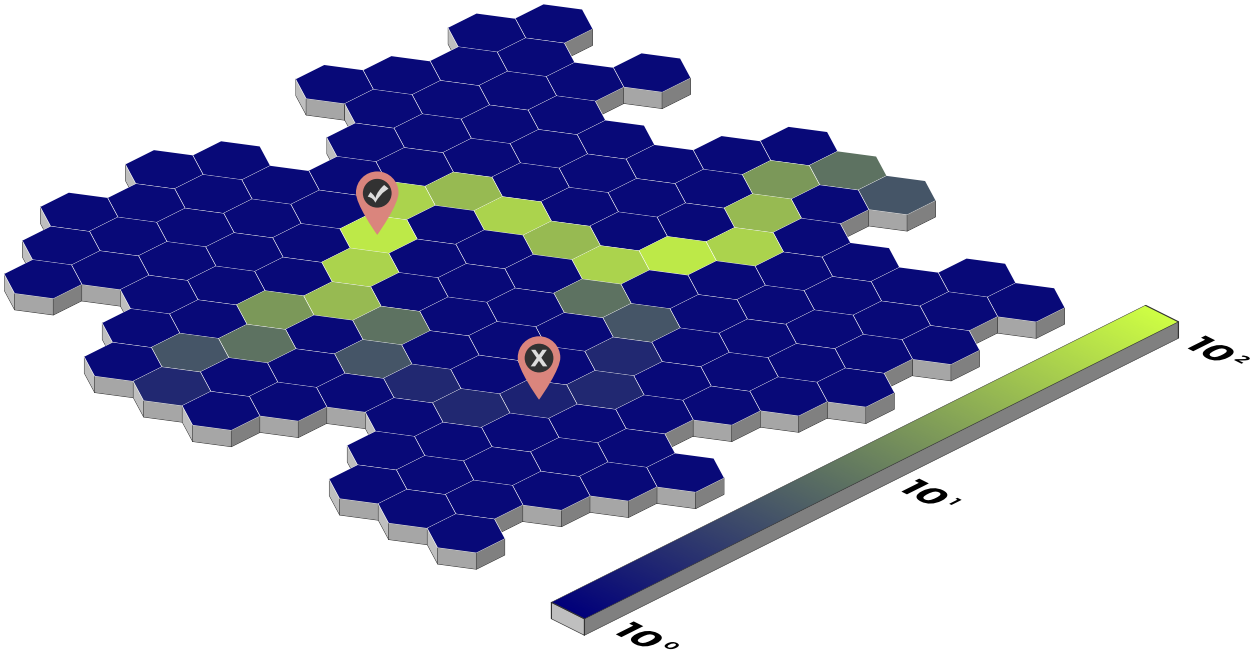
CHALMERS
UNIVERSITY OF TECHNOLOGY



A Visual Aid for Cartographic Road Adjustment

Bachelor's Thesis in Computer Engineering

William Voong
Victor Wallsten

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Bachelor's Thesis 2021

A Visual Aid for Cartographic Road Adjustment

William Voong
Victor Wallsten

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

A Visual Aid for Cartographic Road Adjustment
William Voong
Victor Wallsten

© William Voong, Victor Wallsten, 2021.

Supervisors:
Ulf Norell, Department of Computer Science and Engineering
Mathias Andreasson, CPAC Systems

Examiner:
Arne Linde, Department of Computer Science and Engineering

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

The Authors grant to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Authors warrant that they are the authors of the Work, and warrant that the Work does not contain text, pictures or other material that violates copyright law.

The Authors shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Authors have signed a copyright agreement with a third party regarding the Work, the Authors warrant hereby that they have obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Cover: Visualisation of an isometric heat map in a honeycomb style. From [1]. Used with permission.

Department of Computer Science and Engineering
Gothenburg, Sweden 2021

Abstract

The objective of this project, carried out at CPAC Systems, was to develop an application that supplies a visual aid on which manual assessment for cartographic road additions and adjustments can be based. It is meant to be used at worksites where the landscape often changes. When the landscape changes, so do its roads, and the cartographic information in the system needs to be updated. A model was created based on data extracted from CPAC's database, containing cartographic information about a worksite and a half hour's worth of vehicular movement by five vehicles. An application that achieves the objective was developed by using Matplotlib to plot a combination of a decaying density heat map and line segments filtered via orthogonal vector projection and ray casting. Additionally, the application provides the user with reasonable suggestions for road additions. A feature that provides suggestions for road adjustments was not implemented due to time constraints; this could be the starting point for future work.

Keywords: cartography, density heat map, orthogonal vector projection, ray casting, haskell

Sammandrag

Målet för detta projekt, utfört på CPAC Systems, var att utveckla en applikation som förser användaren med ett visuellt hjälpmedel som underlättar manuell bedömning av kartografiska vägtillägg och -justeringar. Det är tänkt att användas på platser där landskapet ofta förändras och behov finns av att kartan representerar verkligheten väl. Data extraherad från CPAC:s databas, som innehåller kartografisk information om ett område och fem fordons rörelsemönster under en halvtimme, lästes in och användes som grund för modellen. En applikation som uppfyller målet utvecklades genom att använda Matplotlib för att rita upp en kombination av ett avtynande färgdiagram och linjesegment filtrerade via ortogonal vektorprojektion och ray casting. Dessutom förses användaren med rimliga förslag på vägtillägg. En funktion som förser användaren med förslag på vägjusteringar implementerades ej på grund av tidsbrist; detta kan vara utgångspunkten för framtida arbete.

Nyckelord: kartografi, färgdiagram, ortogonal vektorprojektion, ray casting, haskell

Acknowledgements

We want to thank CPAC Systems for letting us take part in the development of one of their systems. Special thanks to our in-company supervisor, Mathias Andreasson, for bouncing development ideas with us.

A big thank you to those who helped us shape this report into its final form: Sara Petersson who assisted with the proofreading, and our supervisor at Chalmers, Ulf Norell, who was always there to help—providing excellent guidance and feedback—throughout the writing process.

Finally, a heartfelt thank you to our good friend Fawzi Aiboud Nygren for lending his artistic talent to us, creating the amazing cover image and many of the figures found in the report.

William Voong, Victor Wallsten, Gothenburg, June 2021

Contents

1
Introduction

The following chapter includes a brief presentation of the company at which the project was carried out, a description of the system that was further developed, and the purpose, objectives and scope of the project.

1.1 Background

The project was carried out at CPAC Systems: a company that develops value-creating products for clients such as Volvo Group and Yamaha.

The system to be further developed during the project is used in vehicles to help the driver get a clean cartographic overview of the worksite at which the vehicle is located. The overview includes roads, various types of zones, traffic signs, and real-time positions of the vehicles connected to the system.

Information about each vehicle is sent periodically with a frequency of 1 Hz and stored in a database that also contains information about the worksite. Vehicular data relevant to the project includes a unique identifier, a timestamp, and their position and speed.

Relevant worksite data includes what class it belongs to—a road, zone, or traffic sign—and its position in the case of a traffic sign, or shape in the case of a road or a zone. A shape comes in the form of a list of positions. For roads, the list represents a polyline—a sequence of lines between each sequential pair of positions in the list. For zones, the list represents a polygon drawn in the same way as the polyline for roads, with an additional line between the first and last of the positions in the list.

Vehicular positions are stored as coordinates in latitude/longitude/altitude format, whereas worksite positions are stored as x/y/z coordinates based on Universal Transverse Mercator projection. Both coordinate systems are based on the same model of the Earth: The World Geodetic System 1984 Ellipsoid.

1.2 Purpose

The system described in Section 1.1 is used in places where the landscape often changes, such as at an open-pit mine. When the landscape changes, so do its roads, and the cartographic information in the system needs to be updated to better reflect

reality. Currently this is done by a two-step process: a driver activates a function that records the route taken by the vehicle, and this information is passed on for manual assessment before being put into effect.

CPAC considers the route-recording step to be inefficient and wants it to be automated. Hence, the purpose of the project is to automatically supply a basis for manual assessment of cartographic road updates, removing the need for the aforementioned route-recording step.

1.3 Objectives

The main objective is to develop an application that supplies a visual aid on which manual assessment for cartographic road additions and adjustments can be based. To achieve this goal, two subgoals are set:

Subgoal 1. Read the data into a unified format
Subgoal 2. Turn the unified data into a visual aid

For the application to be deemed adequate, the following solution requirements need to be met:

Requirement 1. The visualisation is an accurate representation of the given data. In other words, the data is parsed and visualised without unreasonable distortion.
Requirement 2. The visual aid can reasonably be used by a human for assessing whether a cartographic road should be added or adjusted.

Additionally, to be deemed successful, at least one of the following features needs to be implemented:

Feature 1. The visual aid provides reasonable suggestions for road additions.
Feature 2. The visual aid provides reasonable suggestions for road adjustments.

1.4 Scope

The output does not need to be machine-readable. Since the visual aid is meant for humans, to ease parsing the output the positional data is restricted to two dimensions—a top-down view—not taking altitude into account.

No direct access to CPAC's database is given. Data is supplied via plain text files containing data extracted from the database. At first, the supplied data did not include vehicle identifiers; this was amended late into the project.

To avoid unnecessary complications during development, the application does not need to be integrated with CPAC's interfaces, nor does a graphical user interface need to be implemented.

For confidentiality reasons, neither the aforementioned text files nor any source code will be shown in this report.

2

Theory

The following chapter introduces systems, models, libraries, formulas, and algorithms used in the project.

2.1 Coordinate Systems for Geopositioning

The following section gives an introduction to the geopositional coordinate systems used during development: World Geodetic System 1984, latitude and longitude, and Universal Transverse Mercator.

2.1.1 World Geodetic System 1984

The World Geodetic System 1984 (WGS84) is a coordinate system and reference frame for the Earth for practical applications of mapping, charting, geopositioning, and navigation. WGS84 includes, among other things, a widely used reference for an ellipsoid that represents an approximation of the Earth (WGS84 Ellipsoid).

The WGS84 coordinate system is a right-handed, Earth-fixed orthogonal coordinate system [2]. Its definition follows the criteria outlined in the International Earth Rotation and Reference Systems Service (IERS) Technical Note No. 36 [3], repeated below:

- It is geocentric, the center of mass being defined for the whole Earth including oceans and atmosphere
- Its scale is that of the local Earth frame, in the meaning of a relativistic theory of gravitation
- Its orientation was initially given by the Bureau International de l'Heure orientation of 1984.0
- Its time evolution in orientation will create no residual global rotation with regard to the crust

The coordinate system is graphically depicted in Figure 2.1.

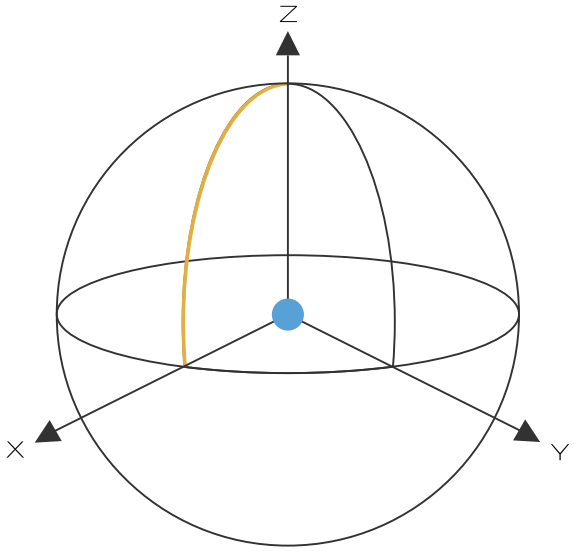


Figure 2.1: The WGS84 coordinate system. The blue dot marks the Earth's center of mass. The orange line represents the IERS Reference Meridian. From [1]. Used with permission.

In Figure 2.1, the origin and axes are defined as follows:

- The origin is the Earth's center of mass
- The X-axis is the intersection of the IERS Reference Meridian and the plane passing through the origin and normal to the Z-axis
- The Y-axis completes a right-handed, Earth-centered and -fixed orthogonal coordinate system
- The Z-axis is the direction of the IERS Reference Pole

The origin and Z-axis serve as the geometric center and rotational axis of the WGS84 Ellipsoid: an approximation of the Earth with the intent of being a mathematically manageable reference useful for cartographic and gravimetric applications.

Two defining parameters are needed to determine the ellipsoid's geometric properties: the semi-major axis and the flattening factor of the Earth. By determining the geometric properties of the ellipsoid, it is possible to convert coordinate values between different coordinate systems based on that ellipsoid [2].

2.1.2 Latitude and Longitude

Latitude and longitude are positions on the surface of the Earth that can be located via angular measurements relative to certain reference points. The values are expressed in degrees, with the fractal part either in decimals or in minutes and seconds. Latitude represents a location north or south of the Equator, and longitude represents a location east or west of the prime meridian at Greenwich, London. Hence, the values are usually written with a suffix denoting direction: N for north, S for south, E for east, and W for west. Alternatively, the values can be written with a positive or negative signum, where positive values represent north or east, and negative values represent south or west. Latitudinal values are in the range $[-90°, 90°]$, and longitudinal values are in the range $[-180°, 180°]$.

Latitude, in geodetic applications such as the one used in this project, is the angle between the normal lines of the equatorial plane and the plane of the given position on the surface of the Earth. Given an elliptical, non-spheric model of the Earth, like the WGS84 Ellipsoid, the length of a latitudinal degree varies slightly between the Equator and the North Pole or South Pole due to the non-uniformity of the curvature of the model. To aid in locating latitudinal positions on maps or globes, equidistant lines known as parallels are plotted parallel to the Equator as illustrated in Figure 2.2.

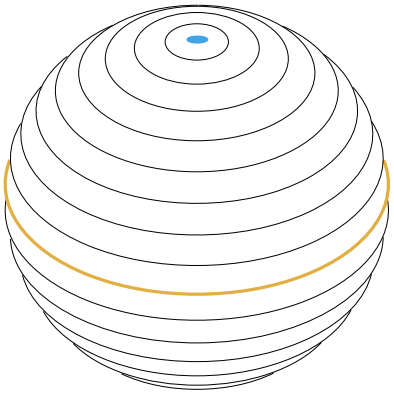


Figure 2.2: Globe with equidistant parallels plotted. The blue dot marks the North Pole. The orange line represents the Equator. From [1]. Used with permission.

Longitude is the angle between two lines, both starting at the centre of the Earth, one being drawn to the intersection of the Equator and the prime meridian, and the other being drawn to any other point on the Equator. To aid in locating longitudinal positions on maps or globes, equidistant lines known as meridians are plotted from pole to pole as illustrated in Figure 2.3.

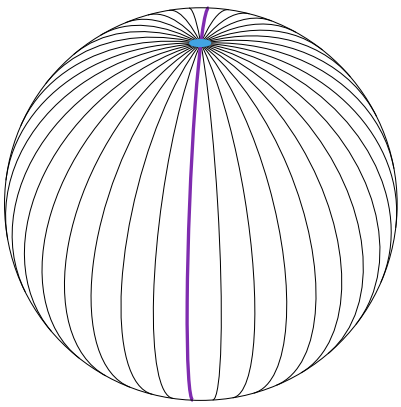


Figure 2.3: Globe with equidistant meridians plotted. The blue dot marks the North Pole. The purple line represents the prime meridian. From [1]. Used with permission.

Combining equidistant latitudinal parallels and longitudinal meridians yields a grid framework that is useful for determining positions on the Earth—see Figure 2.4 for an illustration.

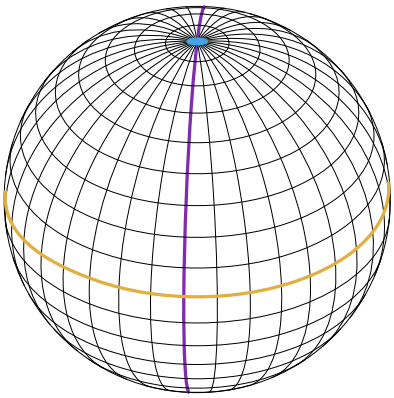


Figure 2.4: Globe with equidistant parallels and meridians plotted. The blue dot marks the North Pole. The orange line represents the Equator. The purple line represents the prime meridian. From [1]. Used with permission.

In Figure 2.4 adjacent parallels are 10° apart, as are meridians. Hence, for example, the position at the intersection of the parallel two lines south of the Equator and the meridian three lines east of the prime meridian can be written as {20°0'0"S, 30°0'0"E} using minutes, seconds and the directional suffixes, or $\{-20.0°, 30.0°\}$ using decimals and signums [4].

2.1.3 Universal Transverse Mercator

Universal Transverse Mercator (UTM) is a plane coordinate grid system used for locating positions on the surface of the Earth, comprised of 120 transverse Mercator projections. The Earth is divided into 60 longitudinal zones, each one being further split in two by the Equator. The central meridians of adjacent longitudinal zones are 6° apart, starting from Zone 1 at 177° W covering 180°–174° W, Zone 2 at 171° W covering 174°–168° W and so on. There are exceptions in a few areas—see Figure 2.5 for an illustration.

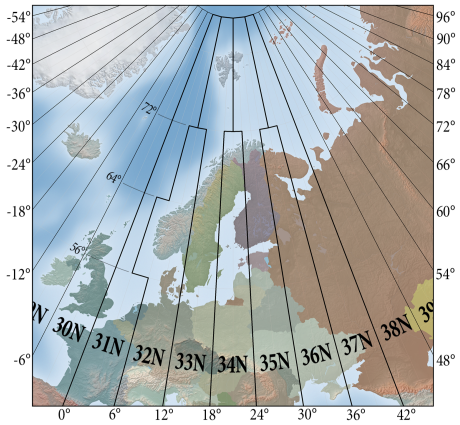


Figure 2.5: Parts of UTM zones 21–47, with exceptions in zones 31–37. The suffix N is used to denote being located in the northern hemisphere. From [5]. CC BY-SA 4.0.

For exact values and exceptions for all zones, see Table 1, 2, 3, 7 and 8 in [6].

A UTM-projected position is specified by the UTM zone number and the easting and northing coordinates in that zone, with the origin being the intersection of the Equator and the central meridian of the zone. To avoid negative numbers, the origin is defined as being 5×10^5 m East and the northing value differs depending on hemisphere. Origins in the northern hemisphere are defined as being 0 m North, increasing northwards, whereas origins in the southern hemisphere are defined as being 10^7 m North, decreasing southwards [7].

2.2 Density Heat Map

A density heat map is a two-dimensional generalisation of a histogram, constructed by grouping a set of points by their x and y coordinates and applying an aggregation function to compute the colour of the tile representing the bin [8]. See Figure 2.6 for an illustration.

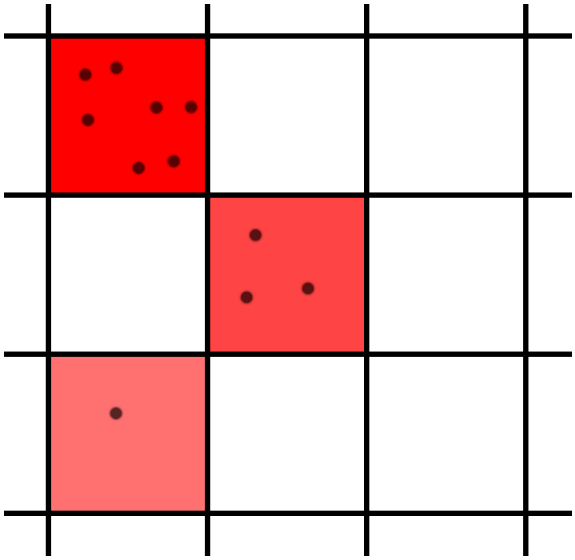


Figure 2.6: A density heat map. The tiles represent bins and the black dots represent sampled data. Notice how the colour of the populated tiles has a different intensity depending on the number of samples within each bin.

2.3 Orthogonal Vector Projection

Given a non-zero vector u in $\mathbb{R}^n$ and a vector y, the orthogonal projection of y onto u is given by $\hat{y}$ according to Equation 2.1. See Figure 2.7 for an illustration.

$$\hat{y} = \frac{y \cdot u}{u \cdot u} u \tag{2.1}$$

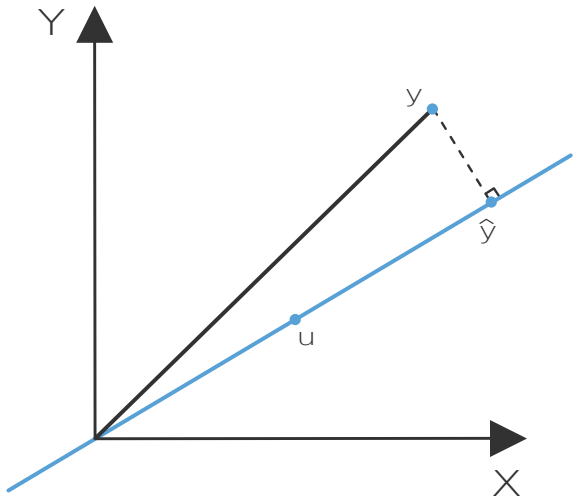


Figure 2.7: The orthogonal projection of y onto u, denoted by $\hat{y}$. From [1]. Used with permission.

For the line between y and $\hat{y}$, the shortest vector from y to the span of u is $\hat{y} - y$. The span of u is the set of vectors given by $c \cdot u$ where c is an arbitrary real-valued scalar [9].

2.4 Ray Casting

Ray casting—also called the *surrounding test* or the *odd-even algorithm*—is a binary classification method for determining whether a point is inside a polygon.

The algorithm can be described as follows: if a horizontal line is drawn from a point in one direction, an odd number of intersections between the drawn line and the polygon indicates that the point is inside the polygon—otherwise the point is outside the polygon. See Figure 2.8 for an illustration.

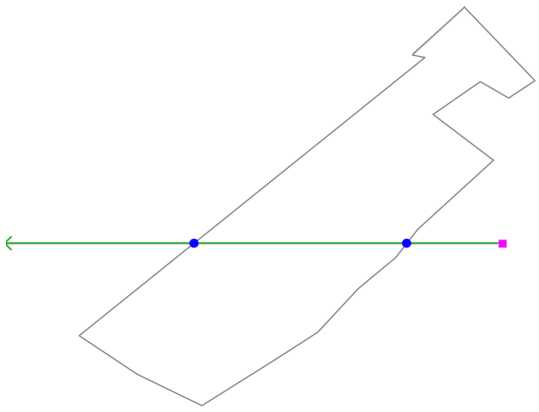


Figure 2.8: Ray casting: The pink square represents the point classified by the algorithm. The blue circles mark the intersections between the polygon and the green line drawn horizontally from the point through the polygon. The number of intersections is even, indicating that the pink square is outside the polygon.

It can be observed that each time the drawn line crosses a border of the polygon, the binary classification switches its state, starting from *outside* [10].

One way to implement this algorithm is to check the point with each segment of the polyline. However, one must account for the edge case when the drawn line intersects a vertex of a segment, as each vertex is a connection between two segments of the polyline and would be counted twice if intersected.

This edge case can be handled by only counting an intersected vertex if the segment's other vertex lies above the drawn line [11], as illustrated in Figure 2.9.

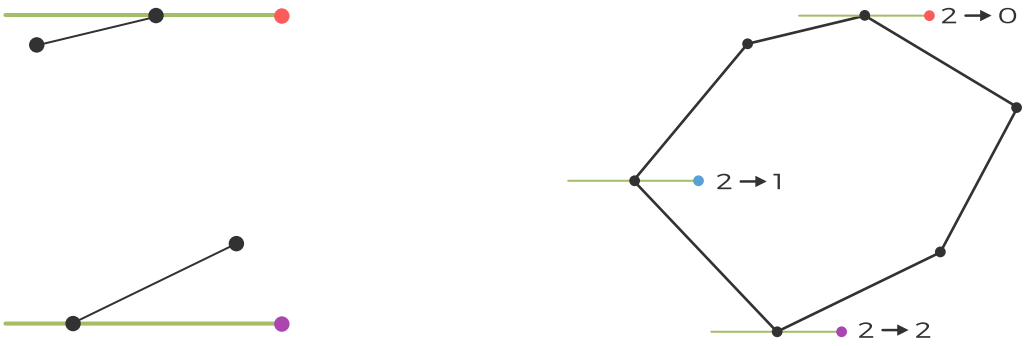


Figure 2.9: On the left: Two basic edge cases when horizontally drawn lines from the red and purple points intersect a vertex of a black segment. The intersected vertex in the topmost case is not counted because the other vertex lies below the line. This, in the context of the polygon to the right, changes the number of counted intersections as indicated by the arrows. Handling the edge cases in this way fixes the classification of the blue point from *outside* to *inside*. From [1]. Used with permission.

2.5 Principal Curve Analysis

A principal curve is a smooth one-dimensional curve that passes through the middle of a p-dimensional data set. Principal curve analysis is the method of determining such a curve [12].

2.6 Libraries

The following section contains descriptions of the Haskell libraries used in the project. They are all available on Hackage, the Haskell community's central package archive of open-source software [13].

2.6.1 Containers

Containers is a library providing efficient general-purpose implementations of various immutable container types including sets, maps, sequences, trees and graphs [14].

2.6.2 DSP

DSP is a digital signal processing library that supports various functions for linear algebra, interpolation, and Fourier transform [15].

2.6.3 HCoord

HCoord is a translation of the Java library Jcoord—a library for converting values between Ordnance Survey of Great Britain grid references, Universal Transverse Mercator references, and latitude and longitude. The accuracy of the conversions in HCoord has not been explicitly stated, but Jcoord conversions are claimed to be accurate to within 5 m [16].

2.6.4 Matplotlib

Matplotlib includes bindings to the homonymous Python library—a plotting library for application development, interactive scripting, and image generation across multiple operating systems. It supports major 2D plot types such as scatter plots, line plots, density heat map plot, and can also handle real-time data and GUI events [17].

2.6.5 Parsec

Parsec is a monadic parser combinator library designed to be simple, safe, and fast, with good error messages. It provides simple parsing functions and means of tying the parsing functions together, through which more sophisticated parsers can be built [18].

2.6.6 QuickCheck

QuickCheck is a library for automatic testing of Haskell programs. A large number of randomly generated cases are tested against a specification of the program, provided by the programmer, in the form of properties functions should satisfy [19].

3
Methods

The following chapter presents the methods used, divided into four sections. The first one describes various types of visual aids considered for the application, followed by a section on choosing what to implement. The third section details the implementation, and the final section explains how the application was tested.

3.1 Visual Aids Considered

In the following section multiple types of visual aids that were considered are presented in terms of implementability, fulfillment of solution requirements and potential issues.

3.1.1 Line Segments as a Visual Aid

One way to provide the user with a visual aid that fulfills the solution requirements is to visualise the route taken by each vehicle as a line segment. Creating a line segment is trivially done by simply drawing lines between the data samples sorted by time. However, in the beginning of the project, the data samples did not include vehicle identifiers, giving rise to a problem demonstrated in Figure 3.1 when multiple vehicles are included in the data samples.

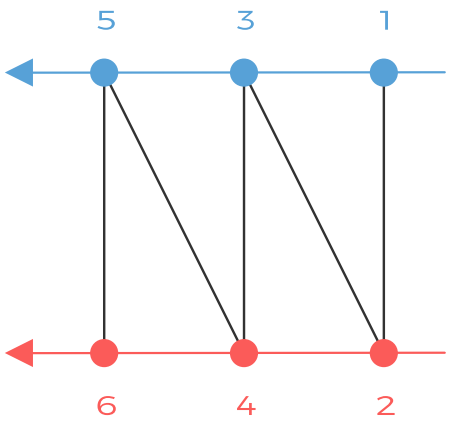


Figure 3.1: The red and blue dots represent samples from two vehicles driving in the same direction. The samples are numbered by time. The black line segment represents the resulting misleading line segment—the ideal solution would result in the separate blue and red lines. From [1]. Used with permission.

This problem breaks the requirement of accurate visual representation. The idea of line segments as a visual aid was rejected at first since not enough information was available to separate data by vehicle, but was reconsidered when circumstances changed and the samples included vehicle identifiers.

Displaying all line segments at once would end up looking like spaghetti given enough data samples, prompting a need for filtering algorithms, like grouping by vehicle identifier and dividing each line into appropriate chunks. The chunks could be constructed, for example, by cutting the line segment each time the speed of the vehicle turns to zero, or specifying the number of samples per chunk.

3.1.2 Principal Curve as a Visual Aid

Representing a road that goes through the middle of a cluster of points can be achieved by using principal curve analysis. One potential issue with this algorithm when points are far apart naturally—for instance, due to branching roads, as illustrated in Figure 3.2.

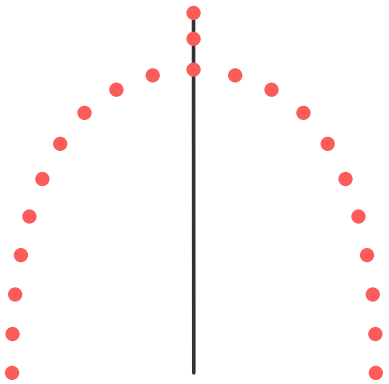


Figure 3.2: The red dots represent sampled data from branching roads. The black line shows the principal curve between the samples, thus showing that it might result in a bad suggestion. From [1]. Used with permission.

An idea considered for reducing the risk of running into this scenario was to take data from only one vehicle at a time in small chunks, and applying principal curve analysis on that subset of data. However, finding appropriate parameters for this solution might be excessively time-consuming. Either way, results might not meet the accurate representation requirement.

3.1.3 Density Heat Map as a Visual Aid

A solution that is not reliant on data samples separated by vehicle is using a density heat map as visual aid.

This solution does not represent a clear suggestion like that of a line segment representing a road. However, it is fairly easy for humans to parse the information. Without any additional functionality, a heat map superimposed over a cartographic map can be used to make decisions on road additions. By adding a periodic decay

to the heat map, it can also be used to make decisions on road adjustments. Also, various filtering algorithms like removing heat around roads or within zones can be added to provide the user with the option of visualising newly populated areas.

Implementation is simple due to the existence of supporting functions for plotting density heat maps in the Matplotlib library.

A disadvantage with this type of visual aid is that it is quite data-reliant—a heat map constructed from small amounts of data does not offer a clear visual aid.

3.2 Choice of Visual Aid

Relying solely on a density heat map has the problem of being an unclear visual aid when there is a low amount of data samples. Line segments give a baseless visual aid. Principal curve analysis can yield unreasonable results when there are branching roads. Since there are issues with each of these visual aids by themselves, a combination of visual aids was considered.

Principal curve analysis was ruled out completely due to estimating that the time needed to develop an adequate version of the algorithm was too long.

The visual aid chosen was a combination of the density heat map and the line segments because of how they compensate for each other's disadvantages. Line segments do not need to rely on a large amount of data samples to give a clear visual aid, thus making up for the disadvantage of the density heat map. Having an underlying heat map gives a basis for the line segments visualisation.

Furthermore, combining the two gave rise to a new idea for improving the visual aid of the line segments: filtering the segments by a specified *popularity threshold*, stipulating that the heat map intensity at each vertex in a given line segment must be equal to or greater than the threshold.

3.3 Implementation

The following section explains how the application was implemented—from reading the data to visualising it.

3.3.1 Reading the Data

Due to only being able to access the data through exported text files in a non-standard format, simply using a well-established library for reading the data was not possible. Hence, custom parsers needed to be written. This was one of the main reasons for the choice of programming language for this project—functional languages are well-suited for writing parsers and Haskell is no exception [20]. There are several alternatives, but the library of choice ended up being Parsec due to easily finding good tutorials [21][22]. Furthermore, the fact that the library has

been cloned in many other languages [23] increased the incentive to learn how to use it.

The grammar used in the text files containing the cartographic and vehicular information was easily deduced by simply looking at a few entries of each type. The *key: value* pairs (fields) always appeared in the same order and if a field had a value of *false* or *0 (zero)*, the field was omitted.

At first, simple parsers for numbers, singular fields and separators were written. These parsers were then combined to form more and more complex ones until a single parsing function per information type had been developed.

3.3.2 Unifying Data from Different Geopositional Systems

As mentioned in Section 1.1, vehicular positions and worksite positions are stored using different coordinate systems. Hence, in order to perform calculations based on the relationship between any given vehicle and the worksite, the positional data needs to be unified. The obvious solution is to simply convert all positions of the worksite data into the coordinate system of the vehicular data or vice versa. The choice ended up being converting vehicular positions into UTM coordinates due to conveniently resulting in the appropriate cell size mentioned in Section 3.3.3.

A library for making these kinds of conversions—HCoord—was used. Confirming the accuracy of the conversions was done by plotting both data sets and seeing whether they overlapped as expected.

3.3.3 Modeling a Heat Map with Decay Functionality

To model a basic heat map based on the available vehicular data, an appropriate method would be to use a data structure mapping coordinate pairs to an intensity value. Whenever a vehicle reports its position, the intensity associated with that position would increase. Using floating-point numbers for coordinates, however, would result in an unwieldy collection of mostly single-intensity entries. Rounding UTM-projected coordinates to integers would effectively turn the data structure into a grid system with intensity cells the size of 1 m^2. Different resolutions can be achieved by scaling the input coordinates, but the basic cell size was deemed appropriate.

Adding a decay functionality, to enable letting the intensity of a previously populated position fade away if no reports have been sent from that position in some predetermined amount of time, is a simple matter of adding three things:

1. A timer attached to each intensity value
2. Resetting the timer of a position when reported
3. A function that can be applied to each mapping periodically, reducing the intensity if the timer has run out

Given these stipulations, a suitable data structure is a Map. The one chosen for this

project can be found in the *containers* library: the Map in *Data.Map.Strict*. The type signature of a heat map with decay functionality as described would then be *Map (Int,Int) (Int,Int)*. In other words, a table mapping rounded (*x, y*) coordinate keys to (*intensity, timer*) values.

3.3.4 Filtering out Heat Map Data

To provide the user with the option of visualising newly populated areas, functions for filtering out data within a specified distance from roads or inside zones were added.

For filtering around roads, orthogonal vector projection as described in Section 2.3 was used if the point was located perpendicular to the segment. Otherwise, the distance from the segment's nearest vertex was used.

For filtering in zones, the ray casting algorithm from Section 2.4 was applied.

3.3.5 Visualising Data

The data was visualised by using the Matplotlib library. The library was picked because it includes many supporting functions for drawing various types of two-dimensional plots—most importantly density plots—as well as superimposing plots over other plots.

The worksite data was plotted by turning each entity—a single road, traffic sign, or zone—into a plot and folding the plots into one via superimposition. Scatter plots with various markers were used for traffic signs, and line plots were used for roads and zones. For visual clarity, the entity types were colour coded—black for roads, red for traffic signs and each type of zone was given its own colour.

The vehicular data was plotted as a density heat map with the density of a bin representing the amount of reports from a position within that bin.

3.4 Testing

The following section explains the methods used for verifying the correctness of the data reading and unification described in Sections 3.3.1 and 3.3.2.

3.4.1 Testing the Parsers

Testing the parsers was done via QuickCheck. For each parser, functions for generating strings according to the grammar and syntax of the provided files were written, as well as property-based testing functions to check that the output of the parsers had the expected properties—namely, that the values of the fields of the input string matched the corresponding values of the output data, with empty input fields resulting in a corresponding value of *false* or *0 (zero)*.

3.4.2 Testing the Coordinate Conversions

Verifying the accuracy of the coordinate conversions was done visually in two steps.

Firstly, separate density plots were made of the data before and after the conversion. The shapes and intensity patterns of the plots were compared to make sure the relative values of the data sets were not distorted.

Secondly, the post-conversion plot was superimposed over the plot of the worksite data.

If the shapes and intensity patterns match, and the post-conversion plot overlaps with the worksite data plot, it is reasonable to assume that the conversions are correct.

4

Results

The following chapter presents the results produced during the course of the project and how they relate to the objectives listed in Section 1.3.

4.1 Reading and Unifying the Data

The following section presents the results related to reading and unifying the data, and how they relate to the subgoals and requirements mentioned in Section 1.3.

4.1.1 Reading the Data

Text files for two data sets—worksite data and vehicular data—were parsed. See Figures 4.1 and 4.2 for the resulting plots of the respective data sets.

Figure 4.1: Framed plot of worksite data. The black lines represent roads, the red markers represent traffic signs, and other colours represent zones.

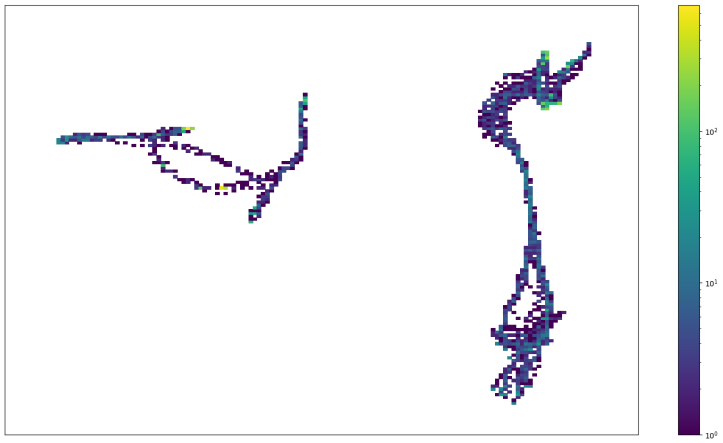


Figure 4.2: Framed plot of vehicular data. The colour bar on the right shows the intensity range.

The plots look plausible as representations of a worksite and vehicular movement. More importantly, this was confirmed by the in-company supervisor who supplied the data. Additionally, the parsing tests described in Section 3.4.1 all passed. Given all of this, the first half of Subgoal 1—*reading the data*—is accomplished and the first half of Requirement 1—*the data is parsed correctly*—is met.

4.1.2 Unifying the Data

After unifying the coordinate systems of the data sets as described in Section 3.3.2, the sets were plotted and superimposed. See Figure 4.3 for the resulting plot.

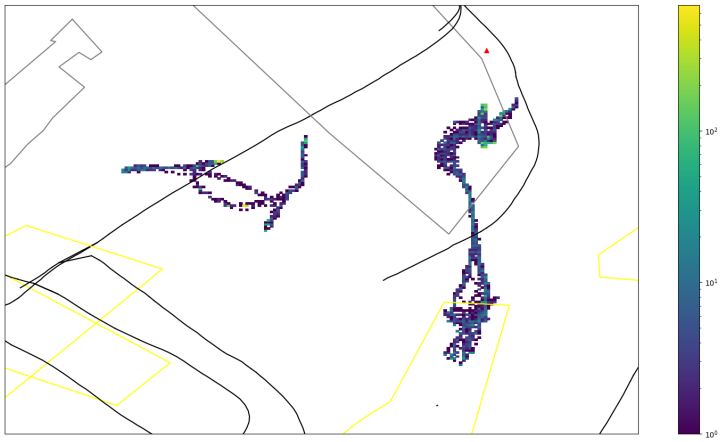


Figure 4.3: Framed plot of vehicular data plot superimposed over worksite data plot, zoomed in to better display the overlapping area. The colour bar on the right shows the intensity range of the vehicular data.

The conversion test described in Section 3.4.2 passed. Adding this onto the results of Section 4.1.1, the entirety of Subgoal 1—*reading the data into a unified format*—is accomplished and all of Requirement 1—*the visualisation is an accurate representation of the given data*—is met.

4.2 Turning the Unified Data into a Visual Aid

One of the subgoals of the project was to turn the unified data into a visual aid. As described in the Methods chapter, the choice of visual aid was a combination of a density heat map and line segments superimposed over the worksite plot.

The density heat map by itself partly fulfills Requirement 2 in that the visual aid can reasonably be used by a human for assessing whether a cartgraphic road should be added. Adding a decay functionality also enables assessing road adjustments, resulting in Requirement 2 being fully met.

The following section includes a demonstration of the aforementioned decay functionality, filtering the density heat map, and the combination of line segments and the density heat map as a visual aid.

4.2.1 Demonstration of Decay on a Density Heat Map

To completely fulfill Requirement 2—*The visual aid can reasonably be used by a human for assessing whether a cartographic road should be added or adjusted*—decay functionality was added to the density heat map.

Since data was limited, in order to provide a clearer demonstration of this feature, data was generated to demonstrate how decay can be used as a visual aid for road adjustments. See Figure 4.4 for an illustration.

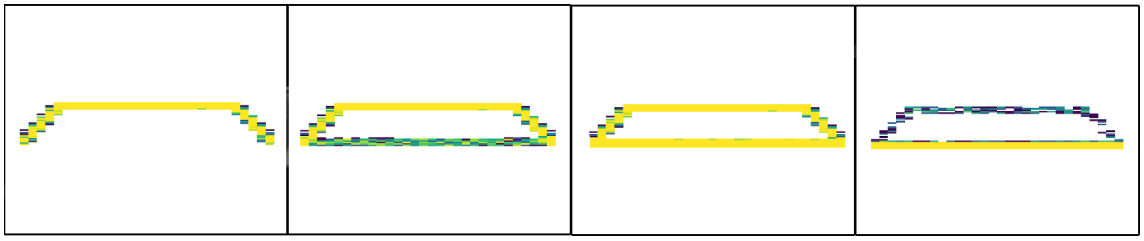


Figure 4.4: A sequential ordering of events from left to right plotted as density heat maps. The intensity range is the same as in previous figures, ascending from blue to yellow. The first subfigure shows a populated road along the top, with no population along the bottom. In the second one, the bottom road starts being populated. As the top road stops being populated and the bottom road keeps being populated, both reach equal intensity in the third subfigure. In the last one, decay has begun affecting the top road, reducing its intensity.

This simulation demonstrates how the road might change and become less populated on unused paths, slowly deteriorating. Simulating road activity by sampling randomly generated data that is aggregated on the density heat map, the visual aid shows how a new road appears between its end points. By applying decay on the

heat map, the old road deteriorates and eventually only the new road will remain in the visual aid, indicating that the old road should be adjusted.

4.2.2 Filtering the Density Heat Map

To give the user the option of only displaying newly populated areas, two filtering algorithms were implemented.

Filtering around roads was done primarily by using orthogonal vector projection. See Figure 4.5 for the resulting plot.

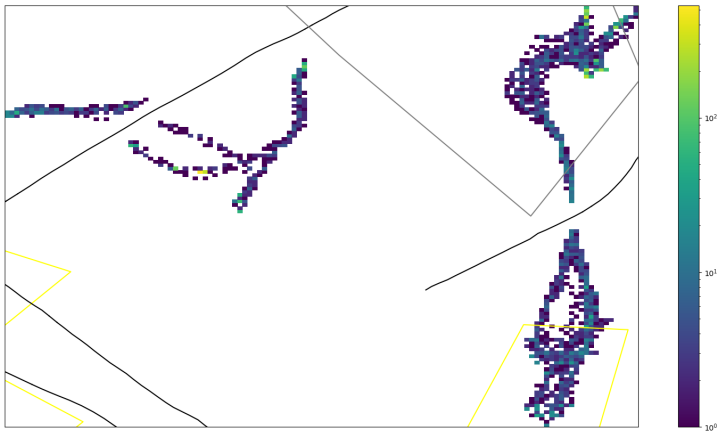


Figure 4.5: The visual aid after filtering out data within 3 meters of roads.

By applying the ray casting algorithm, data was removed from zones. See Figure 4.6 for the resulting plot.

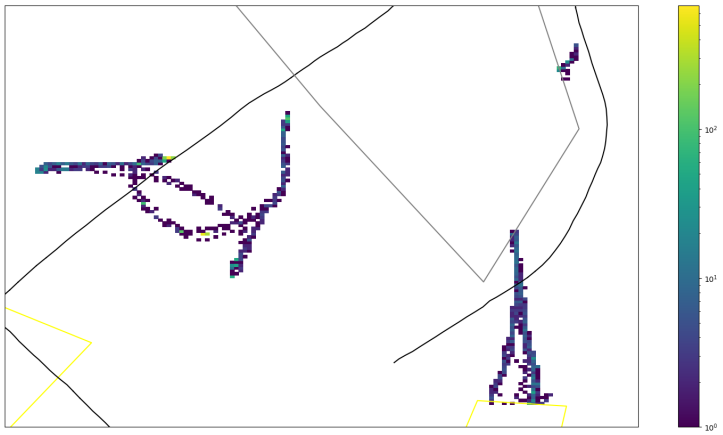


Figure 4.6: The visual aid after filtering out data inside zones.

In addition to supplying the user with visual options, these filtering algorithms can be used as means of implementing Feature 1—*the visual aid provides reasonable suggestions for road additions*—by narrowing down the list of suggested line segments.

4.2.3 Line Segments and Density Heat Map as a Visual Aid

The chosen visual aid is a combination of a density heat map and line segments, mainly for the reason that they make up for each other's disadvantages.

For instance, the line segments can provide a more reasonable visual aid when there is a low amount of data samples, and the density heat map can serve as a basis for the line segment suggestions.

To make the suggestions more reasonable as a visual aid, the line segments, representing routes taken by vehicles, can be divided in different ways. For example, one can cut a line segment each time the vehicle stops or specify the number of samples per chunk. Additionally, filters can be applied to narrow down the list of suggestions. This all relates to Feature 1: *the visual aid provides reasonable suggestions.*

An unfiltered plot can be seen in Figure 4.7.

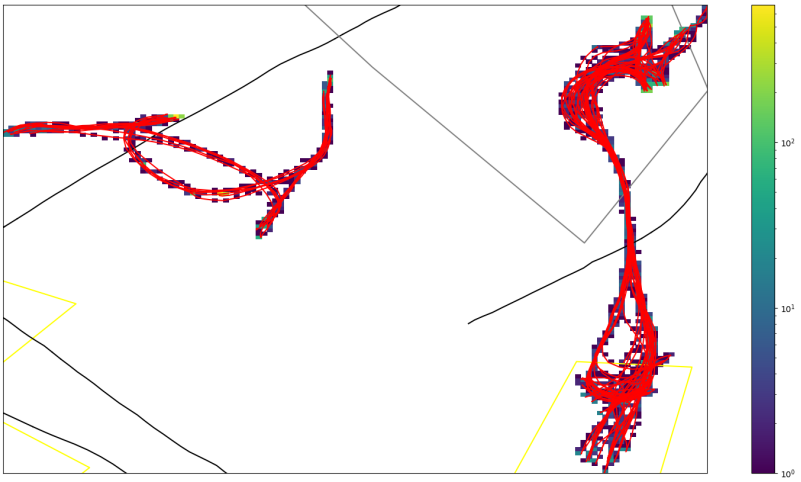


Figure 4.7: The combination of a density heat map and line segments as a visual aid, showing all line segments, intact and unfiltered.

Figure 4.7 shows how suggestions for road additions can be given, represented by the red line segments. Each line segment represents the route taken by one of the five vehicles during the 30-minute sampling session. Even though the sampling session was fairly short, each vehicle has driven back and forth several times. Hence, the line segments need to be trimmed somehow in order to be considered reasonable suggestions.

What follows are the results of trimming the initial line segments in various ways by combining methods previously described in Sections 3.1.1, 3.2 and 3.3.4.

In Figure 4.8, the line segments have been divided into chunks of 25 samples and filtered by a popularity threshold of 2: if any vertex in a suggested line segment has a lower intensity than the threshold, the suggestion will be excluded.

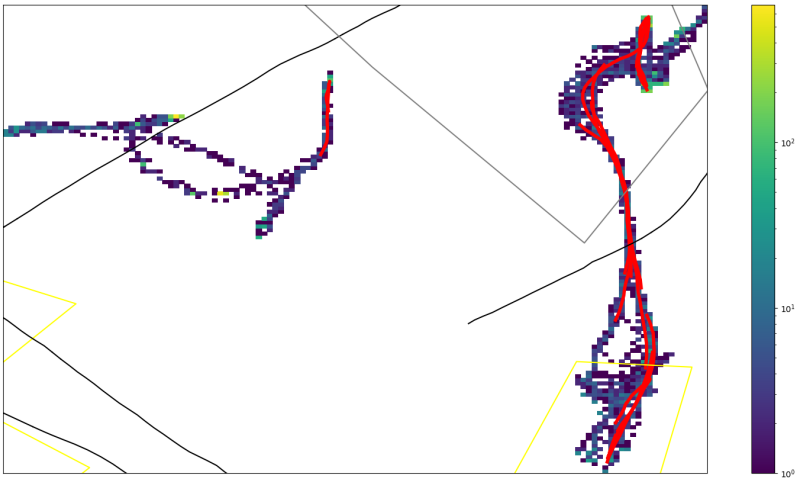


Figure 4.8: Visual aid with 25-sample line segments filtered with a popularity threshold of 2.

Figure 4.9 shows a single suggestion after cutting each line segment when the vehicle stops, filtering by a popularity threshold of 2 and choosing the longest one.

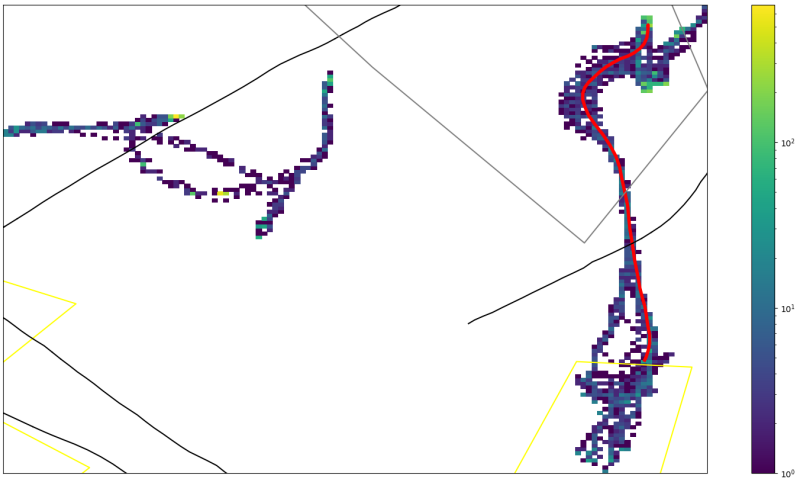


Figure 4.9: The longest suggestion after cutting each line segment when the vehicle stops and filtering by a popularity threshold of 2.

To avoid getting suggestions inside zones, filtering the density heat map via ray casting can be done, as shown in Figure 4.10.

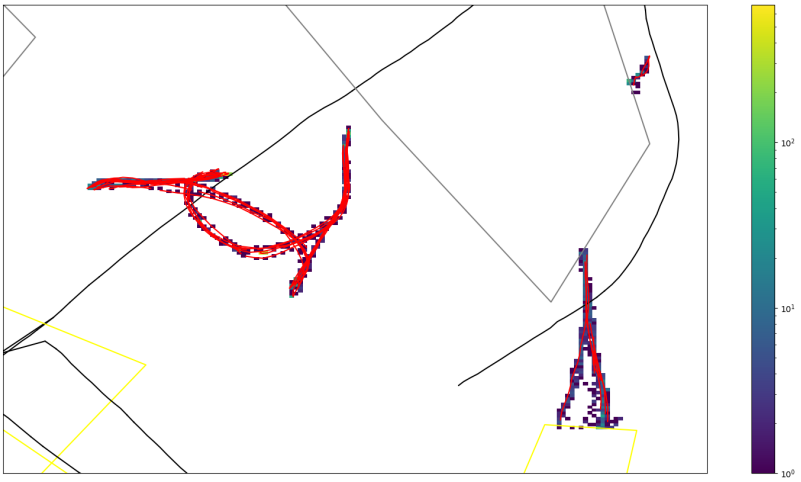


Figure 4.10: The combination of line segments as a visual aid with zone filtering via ray casting on the density heat map, demonstrating how suggestions can be given without overlapping with zones.

With the right combination of dividing and filtering the line segments, reasonable suggestions are given. Thus, Feature 1 is considered implemented.

5
Conclusion

The application developed during the project is able to accurately read data extracted from CPAC's database into a unified format. By plotting a decaying density heat map, the data was turned into a visual aid for assessing cartographic road additions or adjustments. Additionally, line segments were drawn and filtered in various ways to provide reasonable suggestions for road additions.

In conclusion, both of the subgoals were achieved, both of the requirements were met and one of the desired features was implemented.

5.1 Discussion

By filtering line segments with a popularity threshold of two, as shown in Figure 4.8, an unexpectedly large portion of the lines were filtered out—see Figure 4.7 for the plot with no threshold filtering. Reasonably this can be explained by the low amount of sample data that was put into the model. Also, since the filtering method requires every single vertex in a segment to match or surpass the threshold, the longer the segment, the less likely it is to pass the filtering mechanism.

To counteract the excessive filtering, one could divide the longer segments into shorter chunks. However, this would naturally lead to getting shorter suggestions, which might not be desirable. An idea to fix this is to tweak the filtering algorithm such that the requirement is less strict, for example by only needing a portion of the vertices to match or surpass the threshold, but there was no time to implement and evaluate this method within the frame of the project.

Due to not having access to vehicle identifiers until late into the project, it was difficult to come up with an adequate solution for the desired features of providing suggestions for road additions or adjustments—one of the problems is described in Section 3.1.1. When access was finally obtained, there was not much time left for development, leading to an unpolished Feature 1—*the visual aid provides reasonable suggestions for road additions*—and no time to start implementing Feature 2—*the visual aid provides reasonable suggestions for road adjustments.*

Verifying the accuracy of the coordinate conversions could have been done more rigorously, for example by comparing the results given by the library used—whose accuracy was not explicitly stated—with those of a well-established conversion method.

However, the accuracy was deemed sufficient for the scope of this project and if higher accuracy is needed, the conversion method can easily be replaced.

The choice of visual aid for achieving the subgoals and meeting the requirements—a density heat map—turned out to be good for several reasons: it is simple, easy to implement, provides a clear overview that is easy for humans to parse and the model could further be used for filtering suggestions when implementing Feature 1.

5.2 Future Work

The feature of providing suggestions for road adjustments was not implemented in this project. A solution that has been discussed is to implement a road deletion algorithm working similarly to the road addition algorithm, comparing existing roads to the opposite of the popularity threshold used for road additions when decay is applied. The combination of a road deletion and a road addition would form an adjustment suggestion.

The decaying demonstration was only done using generated data. It would be interesting to verify the usability of decay on a large amount of non-generated real-time data. Furthermore, the frequency with which decay should be applied needs to be tested—given the short time span in the provided data, no realistic tests could be conducted to determine a reasonable decay frequency.

The principal curve method described in Section 3.1.2 was not implemented. Evaluating how this compares to the current method for suggesting road additions could be useful.

Additonally, the current methods for filtering suggestions are unpolished. One issue mentioned is the strictness of the popularity threshold requirement—tweaking the requirement might yield better results.

Providing road adjustment suggestions was not implemented. Adjusting a road might lead to traffic signs being mispositioned, which should be considered when implementing this feature.

Additional features, such as providing zone additions or adjustments, could be considered.

References

[1] F. Aiboud Nygren, private communication, Apr. 2021.

[2] *DEPARTMENT OF DEFENSE WORLD GEODETIC SYSTEM 1984: Its Definition and Relationships with Local Geodetic Systems*, NGA.STND.0036_1.0.0_WGS84, National Geospatial-Intelligence Agency, United States Department of Defense, Arnold, Missouri, USA, 2014. Available: https://earth-info.nga.mil/php/download.php?file=coord-wgs84, Accessed on: 2021-03-23.

[3] *IERS Conventions (2010)*, IERS Technical Note 36, Federal Agency for Cartography and Geodesy, Frankfurt, Germany, 2010.

[4] A. Augustyn et al., "Latitude and longitude," in *Encyclopaedia Britannica*. Edinburgh, Scotland: Encyclopædia Britannica, Inc., 2021. [Online]. Available: https://www.britannica.com/science/latitude, Accessed on: 2021-04-01.

[5] Kbellis, "Modified UTM Zones," 2017. [Electronic image]. Available: https://commons.wikimedia.org/w/index.php?curid=63315557, Accessed on: 2021-05-06.

[6] *UNIVERSAL GRIDS AND GRID REFERENCE SYSTEMS*, NGA.STND.0037_2.0.0_GRIDS, National Geospatial-Intelligence Agency, United States Department of Defense, Arnold, Missouri, USA, 2014. Available: https://earth-info.nga.mil/php/download.php?file=coord-grids, Accessed on: 2021-03-31.

[7] *The Universal Grids and the Transverse Mercator and Polar Stereographic Map Projections*, NGA.SIG.0012_2.0.0_UTMUPS, National Geospatial-Intelligence Agency, United States Department of Defense, Arnold, Missouri, USA, 2014. Available: https://earth-info.nga.mil/php/download.php?file=coord-utmups, Accessed on: 2021-03-31.

[8] Plotly, "2D Histograms," 2021. [Online]. Available: https://plotly.com/python/2D-Histogram/, Accessed: 2021-05-21.

[9] D. Lay, *Linear Algebra and its Applications*, 3rd ed., Boston, MA, USA: Pearson Education, 2006.

[10] I. Sutherland, R. Sproull, and R. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys*, vol. 6, no. 1, pp. 12-14, Mar. 1974, doi:10.1145/356625.356626.

[11] "Point in polygon," in *Wikipedia: The Free Encyclopedia*, [Online]. Available: https://en.wikipedia.org/wiki/Point_in_polygon, Accessed: 2021-06-07.

[12] T. Hastie and W. Stuetzle, "Principal Curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, Dec. 1989, doi:10.1080/01621459.1989.10478797.

[13] haskell.org, "Hackage: The Haskell Package Repository," 2021. [Online]. Available: https://hackage.haskell.org/, Accessed on: 2021-04-30.
[14] Containers, Version 0.6.4.1, [Software library] : haskell.org, 2020. Available: https://hackage.haskell.org/package/containers, Accessed on: 2021-05-04.
[15] DSP: Haskell Digital Signal Processing, Version 0.2.5.1, [Software library] : M. Donadio and H. Thielemann, 2020. Available: https://hackage.haskell.org/package/dsp, Accessed on: 2021-04-30.
[16] HCoord, Version 1.0.0.0, [Software library] : D. Francesconi, 2017. Available: https://hackage.haskell.org/package/hcoord, Accessed on: 2021-03-22.
[17] J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007.
[18] Parsec, Version 3.1.14.0, [Software library] : D. Leijen, P. Martini, A. Latter, and H. V. Riedel, 2021. Available: https://hackage.haskell.org/package/parsec, Accessed on: 2021-03-22.
[19] QuickCheck, Version 2.14.2, [Software library] : K. Claessen, B. Bringert, N. Smallbone, 2020. Available: https://hackage.haskell.org/package/QuickCheck, Accessed on: 2021-05-11.
[20] P. Ljunglöf, "Pure Functional Parsing: an advanced tutorial," Lic. thesis, Department of Computing Science, Chalmers University of Technology and University of Gothenburg, Gothenburg, Sweden, 2002. [Online]. Available: https://gup.ub.gu.se/file/207637, Accessed on: 2021-05-02.
[21] B. O'Sullivan, D. Stewart, and John Goerzen, "Real World Haskell: Chapter 16. Using Parsec," 2008. [Online]. Available: http://book.realworldhaskell.org/read/using-parsec.html, Accessed on: 2021-04-30.
[22] Wikibooks, "Write Yourself a Scheme in 48 Hours: Parsing," 2020. [Online]. Available: https://en.wikibooks.org/wiki/Write_Yourself_a_Scheme_in_48_Hours/Parsing, Accessed on: 2021-04-30.
[23] haskell.org, "Parsec: Parsec clones in other languages," 2020. Available: https://wiki.haskell.org/Parsec#Parsec_clones_in_other_languages, Accessed on: 2021-04-30.

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

CHALMERS
UNIVERSITY OF TECHNOLOGY