Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

Generating Turn-by-turn Instructions for Reliable Cycling Navigation

Bachelor thesis

Author: Karel Mareš
Thesis supervisor: Doc. Ing. Michal Jakob, Ph.D.
Submission: May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Mareš Karel** Personal ID number: **499040**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Generating Turn-by-turn Instructions for Reliable Cycling Navigation

Bachelor’s thesis title in Czech:

Generování turn-by-turn navigačních instrukcí pro spolehlivou cyklistickou navigaci

Guidelines:

Algorithmic methods for generating turn-by-turn navigation instructions for navigation systems represent a long-standing yet still not fully resolved problem. The accuracy, unambiguity, and timing of instructions are crucial for the reliable navigation of a planned route. This problem is particularly challenging for bicycle navigation due to the density and complexity of the road, path, and trail networks on which cycling navigation occurs, and which prevents the direct application of methods designed for car navigation, which takes place on structurally simpler networks, and calls for tailored approaches. The exploration of such approaches is the goal of this thesis.

1. Familiarize yourself with the problem of generating instructions for turn-by-turn navigation systems and review existing approaches to solving the problem.
2. Based on a survey of real bicycle routes, identify a library of navigation instructions that is expressive enough to describe the complexities of bicycle navigation.
3. Design an algorithm for generating navigation instructions. As an input, the algorithm takes a cycling route specified as a path on the underlying bicycle transport network. As the input, the algorithm produces a sequence of navigation instructions that best correspond to the given route.
4. Implement the proposed algorithm and integrate it with real-world map data describing cycling transport networks.
5. Evaluate the implemented algorithm on a suitably selected sample of test cycling routes. Assess both the computational properties of the proposed method as well as the quality of the produced navigation itineraries.

Bibliography / sources:

[1] Golab, A., Kattenbeck, M., Sarlas, G., & Giannopoulos, I. (2022). It’s also about timing! When do pedestrians want to receive navigation instructions. Spatial Cognition & Computation, 22(1-2), 74-106.
[2] Hamburger, K., Röser, F., & Knauff, M. (2022). Landmark selection for route instructions: At which corner of an intersection is the preferred landmark located?. Frontiers in Computer Science, 4, 1044151.
[3] Mackaness, W., Bartie, P., & Espeso, C. S. R. (2014). Understanding Information requirements in “Text Only” pedestrian wayfinding systems. In Geographic Information Science: 8th International Conference, GIScience 2014, Vienna, Austria, September 24-26, 2014. Proceedings 8 (pp. 235-252). Springer International Publishing.
[4] Götze, J., & Boye, J. (2015). “Turn Left” Versus “Walk Towards the Café”: When Relative Directions Work Better Than Landmarks. AGILE 2015: Geographic Information Science as an Enabler of Smarter Cities and Communities, 253-267.

Name and workplace of bachelor's thesis supervisor:

doc. Ing. Michal Jakob, Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **02.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Ing. Michal Jakob, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Author statement

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24.5.2024 .....

Karel Mareš

Poděkování

Děkuji panu doc. Ing. Michalu Jakobovi, Ph.D. za vedení mé bakalářské práce a za podnětné návrhy, které ji obohatily. Děkuji rodičům a přátelům za podporu.

Karel Mareš

Název práce:

Generování turn-by-turn navigačních instrukcí pro spolehlivou cyklistickou navigaci

Autor: Karel Mareš

Studijní program: Otevřená informatika
Obor: Základy umělé inteligence a počítačových věd
Druh práce: Bakalářská práce

Vedoucí práce: Doc. Ing. Michal Jakob, Ph.D.
Department of Computer Science FEE

Abstrakt: Práce zkoumá generování instrukcí pro cyklisty v obohaceném směrovacím grafu bez použití externích orientačních bodů pro zadanou cestu. Navrhujeme agregaci hran pro lepší pochopení vazeb mezi hranami a definujeme nejednoznačnost cest. V závislosti na nejednoznačnosti okolních hran a předchozích manévrech se používají různé sady instrukcí. Metody agregace hran a generování instrukcí jsou důkladně popsány. Poté je navržená metoda generování instrukcí vyhodnocena a prezentována ve vyvinuté aplikaci.

Klíčová slova: obohacený směrový graf, generování instrukcí, agregace hran, nejednoznačnost cesty, rozhodovací bod, cyklistická navigace

Title:

Generating Turn-by-turn Instructions for Reliable Cycling Navigation

Author: Karel Mareš

Abstract: The thesis explores instruction generation for cyclists in an enriched routing graph without the use of landmarks for set path. We propose edge aggregation for better understanding of links between edges and define ambiguity of roads. Depending on ambiguity of surrounding edges and previous maneuvers, different sets of instructions are used. Methods of edge aggregation and instruction generation are thoroughly described. Afterwards the proposed method of instruction generation is evaluated and presented in developed application.

Key words: enriched routing graph, instruction generation, edge aggregation, road ambiguity, decision point, cycling navigation

Contents

List of Acronyms

PRIMARY I. class roads

SECONDARY II. class roads

TERTIARY III. class roads

SERVICE generally for access to a building, service station, industrial estate, etc., or unclassified roads connecting houses and buildings

RESIDENTIAL living streets (streets where is lowered allowed speed due to contact with pedestrians)

PATH narrow roads in the countryside, mostly unpaved

TRACK wide roads in the countryside, mostly unpaved

FOOTWAY footway

CROSSING crossing

STEPS stairs

CYCLEWAY dedicated road/street for cyclists.

UNKNOWN uknown

TRACK dedicated track for cyclists separated from car traffic

ZONE pedestrian zone with allowed access by bike

LANE mandatory cycle lane

SHARROW advisory cycle lane

NONE no cycling infrastructure is present

EXCELLENT brand new asphalt

GOOD old asphalt roads or concrete roads, or paving stones with very narrow gaps

BAD cobblestones with bigger gaps, not compacted unpaved roads

HORRIBLE roads hardly used for bicycle

IMPASSABLE roads that should be used only as a last resort

IE IE
Inside aggregated Edge

BE BE
Between aggregated Edges

Chapter 1

Introduction

1.1 Motivation

Opposed to road infrastructure for motorized vehicles, the infrastructure for cyclists is more dense and complicated. Cyclist infrastructure is combination of infrastructure for motorized vehicles, pedestrians and some road types primary for cyclists. This complex infrastructure presents opportunity for generation of more advanced instructions. Presentation of these instructions to the cyclist may be affected by outside elements. One of the elements are weather conditions. Reading from a display during sunny day can be as challenging as using the display during rainy day. Another element is ambient sound and listening to instructions, without headphones, during heavy traffic may be challenging. Other type of limitation is battery life of a navigating device. With the rise of headphones using surround sound and bone conducting technologies, providing auditory information may be beneficial opposed to displaying them. Auditory instructions carry higher risk of misinterpretation without the visual presentation and therefore should be as clear as possible.

Many nowadays research papers are based on data instruction generation using landmarks. Although in some areas are information about landmarks are updated quite frequently, in some areas with less population or less frequent visits from tourists this data about landmarks can be outdated. Roads do not change that frequently and in some situations, with no landmark nearby or really outdated dataset, navigation based purely on roads and paths could be used instead. Some road types are more significant and are more noticeable and could be used as a reference point.

1.2 Aim of this paper

This paper researches instruction generation in a routing graph composed of multiple infrastructures and the risk of misinterpreting them. Our goal is to generate instruction based only on enriched routing graph without using any landmarks.

Chapter 2

Related work

The development of cycling navigation systems combines understanding of geographic information, route planning algorithm and interface design. This section reviews current state of development in these areas and ideas from other research papers.

Cognitive approach to spatial communication is important, because understanding how humans communicate directions and other types of spatial information is fundamental to design effective system for instruction generation. The cognitive approach to the spatial information communication, discussed in article [1] by Michel Denis provides analysis of mental processes behind route description and user friendly instructions. This paper by Michel Denis emphasizes the role of mental imagery linked to following route descriptions and instruction enhancement by linking this visual imagery with 'mental maps', which, according by the article, cyclists naturally form. The study also highlights importance of linguistic structure of instruction and selection of reference points so the user to reduce redundancy and to refer to permanent landmarks.

Another paper [2] by Jakub Krukar takes similar approach to instruction generation based on landmarks, which are more intuitive for cyclists compared to distance-based instructions and as previous paper, from Michel Denis, this paper researches landmark identification and significance of these landmarks to the user. This landmark-based method used with the cognitive processing of spatial information helps the usability of navigational systems for cyclists. Article [3] then researches landmark selection more in depth and suggests which landmarks people find most helpful. Study's experiment reveals that participants prefer landmarks on the side of the road onto which they need to turn.

Study by Rousell and Zipf [4] claims that the arrival of location-aware smartphones has started the development of pedestrian navigation that prioritizes landmarks over distance and road names. Using these reference points aids users even when GPS accuracy is low. This statement is supported by article [5], where their experiments resulted in favor of a landmark-based approach, as it is easier for users to adapt to instruction structure.

The article [6] states that not only landmarks are important for good route description and in less familiar places landmarks are mentioned even in places other than decision locations.

Selection of appropriate landmarks is not trivial, the article [7] describes importance of a landmark and how to measure their salience. The salience can be calculated based on visibility, shape, importance and it's structure (like intersections).

However landmarks may not always be available. Research article [8] explores possibility of using street names instead of landmarks. Advantage of this approach is that the street plates are placed at convenient places. But these street plates might not always be visible or known.

Focus of this paper is on description of a road limited only to information about the structure of a routing graph and it's road types. We have not found many papers on this topic, so we will use ideas from landmark-based instruction generation, which will serve as a foundation for our work.

Chapter 3

Problem specification

We focus on generating instructions for in a enriched routing graph containing only information about road structure and road types for a given path. We want to ensure that the user does not deviate from the route and that the generated instructions have low ambiguity and therefore are less likely to be misinterpreted. In this section we define terms, which are necessary to provide insight into the algorithmic instruction generation.

3.1 Enriched routing graph

Data can be represented as a directed graph and we assign a point in the Euclidean space $\mathbb{R}^2$ to each node.

3.1.1 Space

Majority of maps represent data in two dimensions. We define our map objects space as Euclidean space $\mathbb{R}^2$. As distance function $d(x, y)$ for two points $x, y \in \mathbb{R}^2$, we will use haversine formula (denoted in G). The x represents longitude and y latitude.

3.1.2 Routing graph

In this space we use a routing graph to represent road network infrastructure.

Definition 3.1.1. Directed graph is a tuple (V, E, f, g), where:

- V is a set of nodes
- $E \subset \{e = (u, v) | (u, v) \in V\}$ is a set of oriented edges
  - u is is start node (denoted $SN(e) = u$)
  - v is is end node (denoted $EN(e) = v$)
- function $g : V \to \mathbb{R}^2$ assigns point $p \in \mathbb{R}^2$ to every node $v \in V$
- function h assigns a set of points in Euclidean space $\mathbb{R}^2$

$$h(e) = \{\alpha \cdot g(SN(e)) + (1 - \alpha) \cdot g(EN(E)) \mid \alpha \in [0, 1]\} \; \forall e \in E$$

This representation allows us to traverse between nodes and select nearby edges. Distance between two edges is defined as minimal distance between any two nodes of opposite edges.

Definition 3.1.2. Distance between two edges $e_1 = (u_1, v_1), e_2 = (u_2, e_2) \in E$:

$$d(e_1, e_2) = \min\{d(u_1, u_2), d(u_1, v_2), d(v_1, v_2), d(v_1, u_2)\}$$

Definition 3.1.3. Outgoing edges $O(v)$ for some node $v \in V$ is a set $O(v) = \{e \mid SN(e) = v \mid e \in E\}$

Definition 3.1.4. ingoing edges $I(v)$ for some node $v \in V$ is a set $O(v) = \{e \mid EN(e) = v \mid e \in E\}$

Definition 3.1.5. Neighboring nodes $N(u) = \{v \mid SN(e) = u \wedge EN(e) = v\}$ for $u \in V$, where $e \in E$, is a set of nodes which are connected by by oriented edge starting in node u.

3.1.3 Features

Each edge can hold additional information, which could be used for instruction generation.

Definition 3.1.6. The set F_e is a set of all possible features of all edges.

Definition 3.1.7. Feature function $f_e(e_i) : E \to F_e^n$ assigns set of features to each node $e \in E$.

3.1.4 Nearby significant edges

Some edges might be used as a reference point to other edges and therefore we define significance function.

Definition 3.1.8. Significance function $s(e) : E \to \mathbb{R}$ assigns a significance based on features $f_e(e_i)$ to the edge $e_i \in E$

And with significance function, we can define nearby edges, which significance is lower or higher.

Definition 3.1.9. Nearby more significant edges $S_D(e) = \{e_o \mid d(e, e_o) <= D \wedge s(e) > s(e_o)\}$ is a set of edges, where $e, e_o \in E$ and D is a maximal distance.

Definition 3.1.10. Nearby less significant edges $s_D(e) = \{e_o \mid d(e, e_o) <= D \wedge s(e) < s(e_o)\}$ is a set of edges, where $e, e_o \in E$ and D is a maximal distance.

3.1.5 Edge visibility

To use some edges as reference points, we might need to know if the edge $e_v \in E$ is visible from the edge $e \in E$. $h(e)$ is essentially line segment we can define orthogonal projection of a point onto the edge e and determine if the projection exists.

Definition 3.1.11. Orthogonal projection $proj_e(\vec{x})$ of $\vec{x} \in \mathbb{R}^2$ onto edge $e \in E$ is a function representing orthogonal projection onto a affine space prescribed by parametric function $h_{affine}(\alpha) = \alpha \cdot g(SN(e)) + (1 - \alpha) \cdot g(EN(E))$.

- If $\alpha \in [0, 1]$ then the projection exists.
- If $\alpha \notin [0, 1]$ the projection does not exist.

If the projection of some point $h(e_v)$ exist, then there might be some edges between which would obstruct the projection onto edge e. To determine, if there is a another edge between, we need to determine some orientation in the space $\mathbb{R}^2$. We can use direction of the line segment, representing edge e_j. The direction of edge is $dir(e) = g(EN(e)) - g(SN(e))$ and to get side of any point we can use determinant and the knowledge, that it represents oriented volume of parallepiped between vectors.

Definition 3.1.12. Edge $e_i \in E$ is on the left of edge $e_j \in E$ if:
$\forall x \in h(e_i) : \det(dir(e_j)(g(SN(e_j)) - x)) < 0$

Definition 3.1.13. Edge $e_i \in E$ is on the right of edge $e_j \in E$ if:
$\forall x \in h(e_i) : \det(dir(e_j)(g(SN(e_j)) - x)) > 0$

Definition 3.1.14. Side is an item from tuple $(left, middle, right)$.

Now that we have defined orientation in our space, we can determine if orthogonal projection $proj_e(\vec{x})$ for any point form e_v is visible.

Definition 3.1.15. Consider edge $e \in E$, $e_v \mid e \neq e_v$ and set of edges, $P = \{p_1, \ldots, p_n \mid p_i \in E \setminus \{e, e_v\}\}$ is a set of edges on the same side as e_v. Set $P_{points} = \bigcup_{p_i}\{x \mid x \in h(p_i) \wedge \exists proj_e(x)\}$ is a set of all points from each edge $e \in P$, for which exists projection onto edge e.
Visible edge e_v from edge e is an edge for which applies at leas one of these rules:

- Exists $proj_e(x)$ and $proj_e(y)$ does not
- $\exists x : proj_e(x) \neq proj_e(y)$
- $\exists x : d(x, proj_e(x)) < d(y, proj_e(y))$

where $x \in h(e_v)$ and $y \in P_{points}$

In other words we want to know if exists any point from e_v for which orthogonal projection is unique or if this point is closest, with regards to other edges.

3.2 Angles between edges and their orientation

To get angle between two edges $e_i, e_j \in E$ we need to get vectors $\vec{u} = g(EN(e_i)) - g(SN(e_i))$ and $\vec{v} = g(EN(e_j)) - g(SN(e_j))$.

Definition 3.2.1. Angle between two edges $e_i, e_j \in E$ is $angle(e_i, e_j) = \min\{\theta_1, \theta_2\}$, where

$$\theta_1 = \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|}$$

and

$$\theta_2 = \arccos \frac{-\vec{u} \cdot \vec{v}}{\|-\vec{u}\| \cdot \|\vec{v}\|}$$

Using law of cosines we can get oriented angle between ingoing edge and outgoing edge. Because with law of cosines, we can get angle only up to 180°, we need to determine on which side the end of the outgoing vector is, with regards to ingoing edge. If the end is on the right side, we calculate complementary angle.

Definition 3.2.2. Oriented angle between edges $e_i, e_j \in E$ is:

$$angle_o(e_i, e_j) = \begin{cases} \theta & \det(uv) \leq 0 \\ 360 - \theta & otherwise \end{cases}$$

where

$$\theta = \arccos\left(\frac{\|\vec{u}\|^2 + \|\vec{v}\|^2 - \|\vec{u} + \vec{v}\|^2}{2\|\vec{u}\|\|\vec{v}\|}\right)$$

When generating instructions and determining right maneuvers, we need orientation and length of edges.

Definition 3.2.3. Function $orient(e) = \frac{\vec{v}}{|\vec{v}|} \mid \vec{v} = g(EN(e)) - g(SN(e))$ assigns orientation to each edge $e \in E$.

Definition 3.2.4. Function $len(e) = d(g(EN(e)), g(SN(e)))$ assigns length of each edge $e \in E$.

3.3 Route

Result of our work is a list of instructions describing a path in routing graph.

3.3.1 Path

Definition 3.3.1. Path P in oriented graph $G = (V, E)$ is a sequence of nodes and edges $P = (v_1, e_1, \ldots, e_{n-1}, v_n)$, where:

- $v_i \in V$
- $e_i = (v_i, v_{i+1}) \mid e_i \in E$
- $EN(e_i) = SN(e_{i+1})\ \forall i \in [1, \ldots, n-1]$

For convenience we will might use only the sequence of edges: $P = (e_1, \ldots, e_{n-1})$

3.3.2 Instruction

Instruction is an object containing text information and is a way to present how to get to certain point. This instruction is presented as a text information and can be combination of maneuver and additional information. Sequence of such instructions then also describes how to get from one point to another.
Maneuver is the most important part of any instruction and represents reorientation in space.

Definition 3.3.2. Maneuver m between two edges e_i and e_j is an item from set of maneuvers

$$M = \{sharp\_left, left, slight\_left, straight, sharp\_right, right, slight\_right\}$$

and is presented as a part of text describing direction.

Definition 3.3.3. Fitness of a maneuver $fit_m(x) : \mathbb{R} \to \mathbb{R}$ is a function evaluating geometrical fitness of maneuver $m \in M$ based on angle x.

In some cases we can use different set of maneuvers, if the conditions for these maneuvers are satisfied. In case of advanced maneuvers, we can use them when crossing or joining road.

Definition 3.3.4. Advanced maneuver m_a between two edges e_i and e_j is an item from set of maneuvers

$$M_a = \{continue, join, opposite, get\_on\_other\_side\}$$

and is presented as a part of text describing direction.

Definition 3.3.5. Decision instruction at intersection v of edges $e_i, e_j \in E$ is a tuple

$$I^D_{(e_i,e_j)} = (f(e_i), f(e_j), S_D(e_i), S_D(e_j), m)$$

3.3.3 Decision point

User does not need to be notified at every intersection (node in a routing graph). If the user continues straight along same path or there is only one outgoing edge, the user is most likely to continue without any instruction.

Definition 3.3.6. Continuation is an instruction which does not need to be presented, because the user is most likely to to execute maneuver m, without any instruction, correctly. We will call this continuation rule.

Continuation rule is can be applied at intersection n if there is only one maneuver available or if we do not change orientation or road. Therefore we define term for consequent edges. This term will allow us to perceive larger segments of roads.

Definition 3.3.7. Consecutive edge for edge $e \in E$ in a graph G, denoted $c(e)$, is a sequence of edges and non-repeating nodes $P = (v_1, e_1, \ldots, e_{n-1}, v_n)$, where v_i is connected only to v_{i-1} and v_{i+1} $\forall i \in [2, n-1]$ and for some $i \in [1, n] : e_i = e$.

Definition 3.3.8. Decision point at intersection v is point, where the continuation rule would not be applied.

3.3.4 Edge aggregation

For better instruction generation and easier referencing of more significant edges we define term **Aggregated edge**. Finding appropriate edges to for edge aggregation is described in sections 4.3 and 5.1. Aggregated edge is theoretical term, introduced by us, creating links between less significant and more significant edges. Aggregated edge represents edges with similar orientation and are used for understanding more complex parts of a routing graph.

Definition 3.3.9. Aggregated edge $A(e)$ for $e \in E$, $e \in R$, is a tuple $(c(e), c_l, c_r)$, where $c(e)$ is a consecutive edge to e and c_l, c_r are consecutive edges of less significant edges on left and right side of $c(e)$.

Definition 3.3.10. Set of aggregated edges $A = \{A(e) \mid e \in R\}$.

Definition 3.3.11. Function $a(e) : E \to \{A\}$ assigns to each edge $e \in E$ set of aggregated edges.

Definition 3.3.12. Function $route_{ag}(P)$ assigns to each path P a sequence of aggregated edges $(A_1, \ldots, A_n)$.

Definition of aggregated edges will help us with further filtering of decision points. We define ambiguity of maneuver m between two aggregated edges as:

Definition 3.3.13. Risk function $amb_m(A_c, A_n, A_o^s, e_i, e_j)$ calculates ambiguity of misinterpretation of maneuver $m \in M$ between edges $e_i, e_j \in E$ and aggregated edges $A_c, A_n \in A$, where e_i is contained in A_c and e_j is contained in A_n. A_o^s represents set of nearby aggregated edges, which are considered as 'dangerous' and could be mistook for A_n.

3.3.5 Route description

Now that we have defined path $P = (e_1, \ldots, e_n)$ and set of maneuvers M we can define term route, which describes maneuvers between each pair of consequent edges (e_i, e_{i+1}).

Definition 3.3.14. Route is a sequence $R = (e_1, m_1, e_2, m_2 \ldots, e_n)$.

3.4 Objective

Main objective of this thesis is generation of reliable instructions in enriched routing graph for some set path P. Generation of instruction at decision point v is based on finding optimal instruction with great fitness of maneuver m and low ambiguity.
For sequence of edges

$$P = (e_1, \ldots, e_p)$$

we get sequence of aggregated edges

$$route_{ag}(P) = (A_1, \ldots, A_g)$$

and route

$$R = (e_1, m_1, \ldots, e_p)$$

We want to find sequence of maneuvers $(m, \ldots, m_{p-1})$, such that for each maneuver m fitness $fit_{m_i}(e_i, e_{i+1})$ is maximal and ambiguity of instruction

$$amb_{m_i}(A_i, A_{i+i}, A_i^s, e_i, e_{i+1})$$

is bellow some threshold.
We can assume sum of all $fit_{m_i}(e_i, e_{i+1})$ as combined fitness

$$fit_{combined}(e_1, m_1, \ldots, e_p) = \sum_{i=1}^{p-1}(fit_{m_i}(e_i, e_{i+1}))$$

and represent this problem as a optimization problem, where we want to find some $fit_{optimal} = \min(fit_{combined}(e_1, m_1, \ldots, e_p))$. Therefore our optimal sequence of maneuvers $(m_1, \ldots, m_{p-1})$ can be found as:

$$(m_{o_1}, \ldots, m_{o_{p-1}}) = \arg\min(fit_{combined}(e_1, m_1, \ldots, e_p))$$

subjected to

$$\forall i \in (1, \ldots, p-1) : amb_{m_i}(A_i, A_{i+i}, A_i^s, e_i, e_{i+1}) < \gamma_r$$

where A_i is referenced aggregated edge by e_i, A_{i+1} is referenced aggregated edge by e_{i+1}, A_i^s is a set of nearby aggregated edges and γ_r is a threshold for ambiguity.

Chapter 4

Approach

In this section we will describe our approach to the problem of instruction generation for cyclists.

4.1 Data representation

Data are represented as an enriched routing graph, where feature function $f(e)\ \forall e \in E$ assigns these values:

- Road type
- Cycle infrastructure
- Surface
- Number of Lanes
  - Edge on a map is represented as polygonal chain.
- Whether or not it is:
  - Oneway
  - Roundabout
  - Pavement

All of the enum types are listed in Appendix A. Edges are combination of 3 infrastructures. First being infrastructure for motorized vehicles, second being infrastructure for pedestrians and third being a infrastructure meant purely for cyclists. Having a graph composed of multiple infrastructures creates in some areas very dense sections of graph in which generation of sensible instructions without the knowledge of relations between edges is very difficult. Discovering these relations will help with generation of more advanced instructions and help with finding decision points.

Figure 4.1: Routing graph

4.2 Separating infrastructures

We can separate these infrastructures by their road type and create several routing graphs. Separation of infrastructures will help us evaluate significance function $s(e)$, because some intersection of nodes might include edge $e \in E$, which is of really low significance and probability of disregarding the road by the user is really high and therefore would have low impact on ambiguity of particular maneuver.

- Edge types which represent infrastructure for vehicles:

$$R_{types} = \{PRIMARY, SECONDARY, TERTIARY, SERVICE, RESIDENTIAL\}$$

- Edge types which represent infrastructure for pedestrians and cyclists near R_{types}:

$$P_{types} = \{CYCLEWAY, FOOTWAY, CROSSING\}$$

- Other edges:

$$O_{types} = \{o | o \notin R \wedge o \notin P\}$$

This separation based on road types creates 3 distinct sets:

- $R = \{e \mid e \in S \wedge roadtype(e) \in R_{types}\}$
- $P = \{e \mid e \in S \wedge roadtype(e) \in P_{types}\}$
- $O = \{e \mid e \in S \wedge roadtype(e) \in O_{types}\}$

Significance of edges is then set to:

$$s(o) \leq s(a) < s(r) \ \forall o, a, r \mid o \in O, a \in A, r \in R$$

Values of significance function are based on easier recognition of the road type. More frequently used road types closer to the infrastructure for vehicles have higher chance to be maintained and used more frequently, which increases chance, that the road is still visible or accessible.
Each set then represents routing sub-graph. By this separation, we might create more consecutive edges, which then represent larger segment of a road, this will reduce number of decision points and therefore filter instructions, which might be deemed unnecessary. As seen in Figure 4.2 crossing separates the road into more segments, by acknowledging their different infrastructure and by separation of infrastructures, we can create the consecutive edge and generating of instruction would be necessary only if the user would have to move form one infrastructure to another. In Figure 4.2, we can also notice somewhat parallel

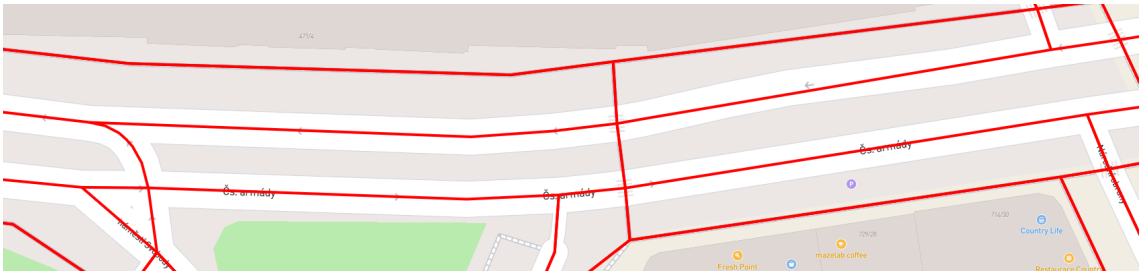

Figure 4.2: Consecutive edges

edges. Some of these edges could be referenced as a sidewalk to the road in the middle. This idea would help to decrease instruction ambiguity, although at the cost of more complex instruction. Because roads with higher significance are more likely to be recognized we want to find appropriate edges with higher significance and reference them instead of referencing a road type, which could be harder to find just as street plates referenced in article [8].
In top part of Figure 4.3, we can see branching of a footway. If we were to approach this intersection from the right of the figure without the knowledge of road bellow, we would be forced to generate instruction to take the footway more on the left. With the knowledge of road bellow, we could rely on previous reference of this road and we might not be forced to generate any instruction, if the previous instruction suggested following this road.

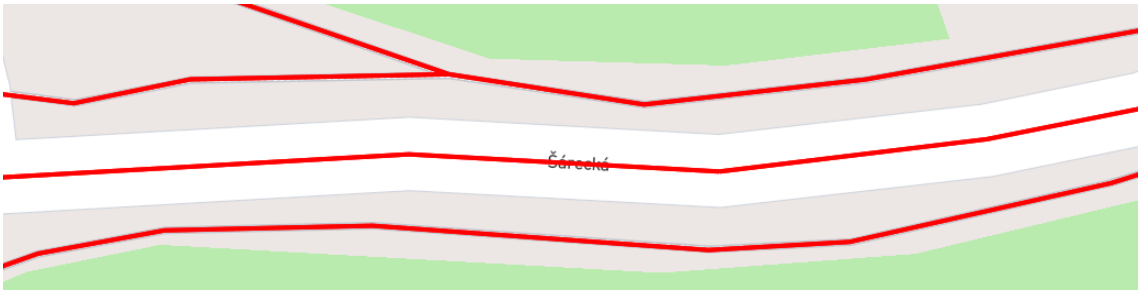

Figure 4.3: Filtering of decision point

4.3 Finding appropriate significant edges

Our goal is to find edges, which could be used as a reference point for a longer period of time and would copy trajectory of the road on which the user currently is, this would reduce amount of generated instructions (i.e. idea mentioned in Figure 4.3). Consider set of more significant edges $S_D(e)$ for some edge $e \in E$. To use significant edge as a reference point, similarly to landmark-based navigation, the significant edge needs to be visible from edge e. However without the knowledge of landmarks and obstacles between roads, the visibility (defined in 3.1.15) will have to suffice if we pick reasonable distance D for $S_D(e)$. By referencing more significant edges for longer periods of time, further decrease number of decision points and we will be able to generate minimal amount of instructions, as instruction generation would be required only if we deviate from the road significantly or change relative position to the road (meaning changing sides, joining the road or continuing in opposite direction). This is why we introduced term **Aggregated edge**. This term allows us separate instruction generation into two parts and use different instruction sets based on if the maneuver between two edges are referencing same aggregated edge or not. Figure 4.4 shows theoretical flowchart for our method of instruction generation.

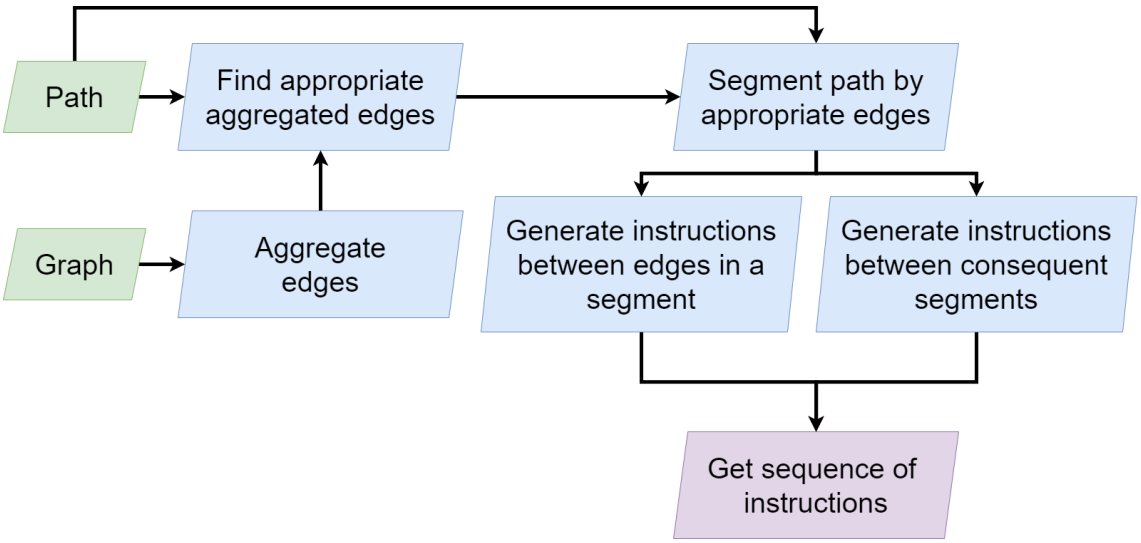

Figure 4.4: Algorithm

4.4 Road orientation

To generate maneuver m between two edges, we need to know their orientation. However the edge might not represent orientation of the road correctly. We also need to consider real orientation of the user as well as the instruction might be presented earlier and because of mentioned ‘mind maps‘ [1] user might assume the orientation based on his current orientation. On the right side of the Figure 4.5 we can see possible decision point on the left side of the road if we were to approaching this decision point from bellow. Imagine situation, where we approach this decision point from bellow and we want to turn left. If we were to calculate the oriented angle between the ingoing edge and outgoing edge facing slightly to the left, we would get angle approximately 135°. This could mean in some cases disorientation of the user, therefore we need find approach, which would find more approximate orientation of the road. We can calculate more approximate orientation by taking in account consecutive edge $c(e) = (e_1, \ldots, e_n)$ of an edge e, where we would assess the orientation of each edge $e_i \mid i \in [1, n]$ based on distance from the decision point.

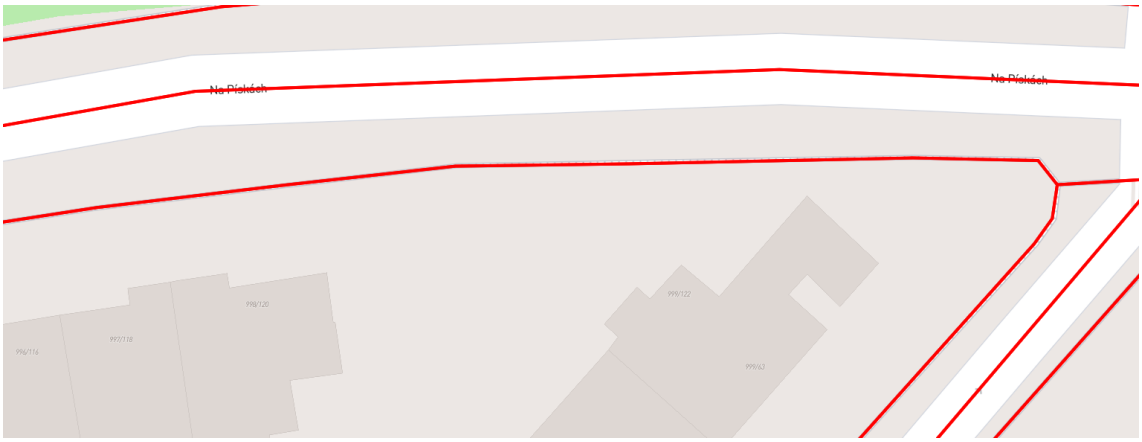

Figure 4.5: Edge orientation

4.5 Similarity of roads

Generation of maneuver based only on edge e_i and edge e_j might create misleading maneuver, because there could exist another edge, which wight be too similar if not better fitting. Therefore we need to find a metric on how two edges are similar to each other based not only on their geometry, but also on other attributes (such as road type). This might help us to generate more unambiguous instructions.

Chapter 5

Algorithm implementation

5.1 Edge aggregation

By constructing consecutive edges, for each infrastructure, and having a set of appropriate significant edges for each edge $e \in E$ we can create a structure **Aggregated edge** for better navigating and instruction generation. Consider consecutive edge $c(e) = (e_1, \ldots, e_n)$, where all considered edges have same road type. For each edge $e_i \in c(e)$ we can find set of less significant edges $s_D(e)$. We will consider only edges, which satisfy 4.3 (they are visible and have similar orientation). Because edge e is representation of a line segment, only condition needed is that the angle between these edges $\theta << 90°$. We will call these significant edges $S = \bigcup_i^n s_D(e_i) \mid e_i \in c(e)$. This set can be then divided into two subsets:

$$c_l = \{s \in S \mid \text{edge s is on the left of appropriate edge } e_i\}$$

$$c_r = \{s \in S \mid \text{edge s is on the right of appropriate edge } e_i\}$$

Edges from these two sets don't need to be connected but the consecutive edge $c(e)$ may be used as a reference point. Now we can specify meaning of items in the aggregated edge definition.

Definition 5.1.1. Aggregated edge $A(e)$, $e \in R$, is a tuple $(c(e), c_l, c_r)$.

In Figure 5.1 you can see links between edges represented by blue polygons. As you can see in the Figure 5.1 these aggregated edges create mostly links between edges, which are in a 'road-sidewalk' relation and the user would perceive them as one road, although the user is able to distinguish them. This helps us with selection of edges when calculating ambiguity for making maneuver $m \in M$. Edge aggregation can be time-consuming and therefore can be calculated beforehand.

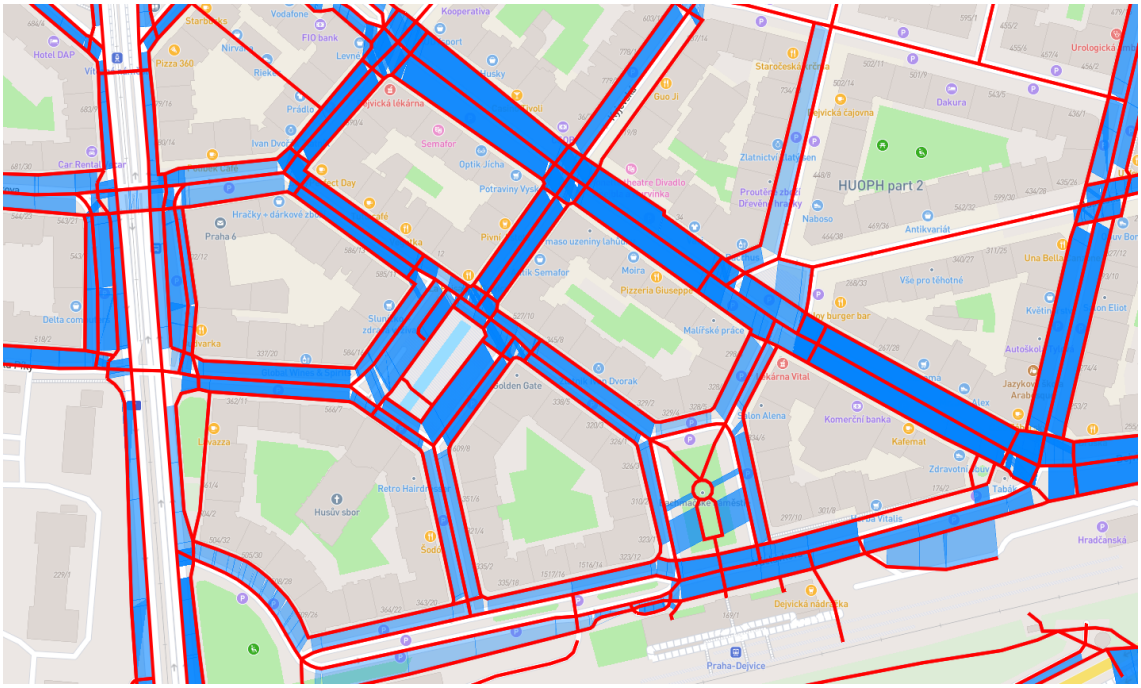

Figure 5.1: Edge aggregation

5.2 Instruction generation

To generate sensible instruction, we need to find appropriate maneuver based on geometry of surrounding edges. In some cases, multiple maneuvers are a viable option and therefore we need to find maneuver, which is the leas ambiguous, therefore we need to define functions, which would assess maneuver fitness based on geography and ambiguity of misinterpretation based on nearby edges.

5.2.1 Maneuver fitness

To get correct maneuver, we need to find the correct oriented angle of ingoing and outgoin edge

Definition 5.2.1. Function $pol(e)$ returns sequence of edges of $c(e) = (e_1, \ldots, e_n)$, where we will consider only edges $e_i \mid i \geq j$, where j is index of e in $c(e)$.

However getting orientation from whole $pol(e)$ without any weights might result in completely different orientation, because we do not know how the $pol(e)$ looks like, and average orientation would not be the solution. One approach is to use weighted average of orientation for each $e_i \in pol(e)$, where weights would depend on combined length of previous edges. Our interest in orientation based on distance can be represented by any non-rising function $i(x)$, where x is distance.

Definition 5.2.2. Function $orient_p(e) = \sum_{i=1}^{n} w_i \cdot orient(e_i) \mid w_i = \int_{l_p}^{l_p + len(e_i)} i(x)dx$, where l_p is sum of lengths of previous edges, for each $e_i \in pol(e)$.

With correct orientation we can define maneuver fitness. Fitness of a maneuver can be represented by function $fit_m(x)$, where $x = angle_o(e_i, e_j) \mid e_i, e_j \in E$ and $m \in M$.

Definition 5.2.3. Fitness of a maneuver $fit_m(x)$ for maneuver $m \in M$ and for oriented angle x is a function defined on $x \in [0, 360)$.

- $argmax(fit_{sharp\_left}) = 45°$
- $argmax(fit_{left}) = 90°$
- $argmax(fit_{slight\_left}) = 135°$
- $argmax(fit_{straight}) = 180°$
- $argmax(fit_{slight\_right}) = 235°$
- $argmax(fit_{right}) = 270°$
- $argmax(fit_{sharp\_right}) = 315°$

Additionally for convenience we define:

Definition 5.2.4. Fitness of maneuver m between two edges $e_i, e_j \in E$ is defined as:

$$fit_m(e_i, e_j) = fit_m(angle_o(orient_p(e_i), orient_p(e_o)))$$

where e_o is opposite edge to edge e_j.

If we use same function for all maneuvers $m \in M$ only shifted on a x axis, then we can compare how similar maneuvers are, by subtracting their fitness, and for better understanding we will normalize fitness of each maneuver to interval $[0, 1]$. In Figure 5.2 you can see fitness function for each maneuver scaled 100x.
These values are based on Figure 5.3.

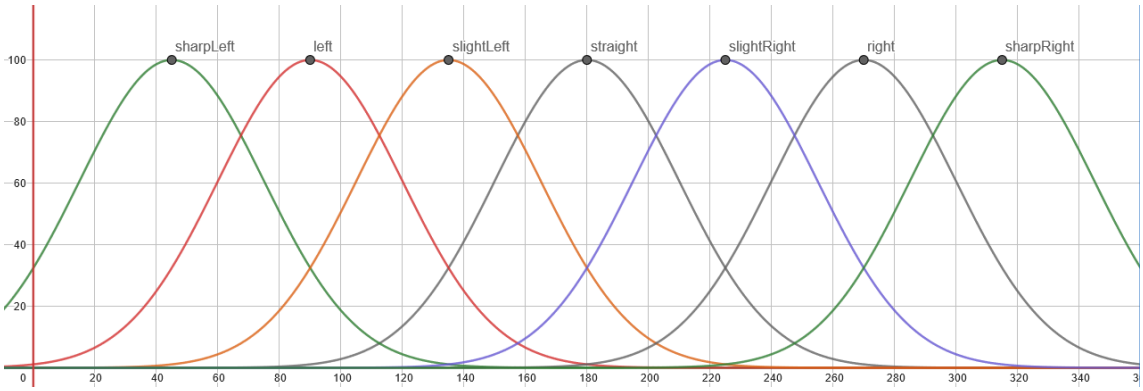

Figure 5.2: Maneuver fitness

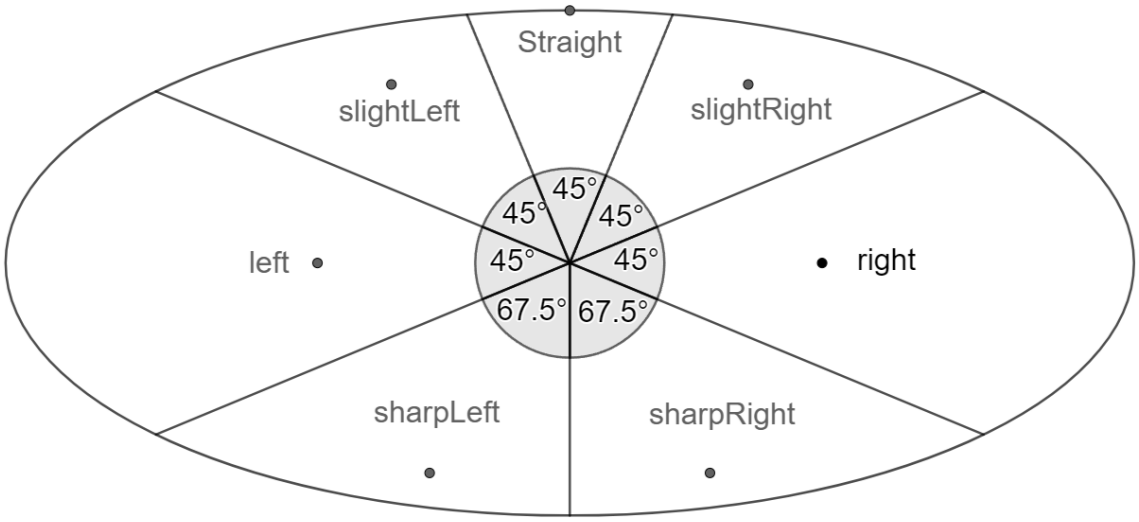

Figure 5.3: Orientation

5.2.2 Maneuver ambiguity

In section 4.5, we mentioned, that we need some way of distinguishing roads from each other. Because of that we define Maneuver ambiguity.

Definition 5.2.5. Ambiguity of maneuver $m \in M$ for edges e_i, e_j and set of other edges E_o, where $e_i, e_j \notin E_o$ is function

$$amb_m(e_i, e_j, E_o) = \begin{cases} 0 & E = \emptyset \\ 1 - |(fit_m(e_i, e_j) - \max\{fit_m(e_i, e_o) \mid e_o \in E_o\})| & otherwise \end{cases}$$

Ambiguity of maneuver m selects most fitting edge $e_o \in E_o$ and compares it to fitness of edge e_j, so this number represents how similar these two edges are in terms of orientation, the ambiguity is $|(fit_m(e_i, e_j) - \max\{fit_m(e_i, e_o) \mid e_o \in E_o\})|$.
Notice, that our fitness function is a normal distribution function. And by our ambiguity definition relies purely on ambiguity of oriented angles. This means that the optimal maneuver is maneuver with the lowest ambiguity.

5.2.3 Sectioning of path

Consider given path $P = (e_1, \ldots, e_n)$. We want to select an aggregated edge and use as longest as possible, because we do not want to flood the user with too much instructions and the user is most likely to follow road closest to referenced road.
We can then create sections composed of consecutive edge, which reference same aggregated edge, $S = ((e_1, \ldots, e_i), (e_{i+1}, \ldots, e_{i+1+j}), \ldots)$, otherwise denoted $S = (S_i, \ldots, S_m)$.

Definition 5.2.6. Sectioning S_i is defined as:

- S_1 is largest possible consecutive sequence of $e_1, \ldots, e_n$, which can reference same aggregated edge as e_1.
- S_i is largest possible consecutive sequence of $e_j, \ldots, e_n$, which can reference same aggregated edge as e_j, where e_j is last non referenced edge by all previous S_j, where $j < i$. For $i \in [2, \ldots, m]$.

This would allow us to separate instruction generation into two parts: instruction generation inside particular section (IE[1]) and instruction generation between these sections (BE[2]).

5.2.4 Finding appropriate aggregated edges

Some edges with lower significance can be aggregated to multiple consequent edges and we have to select which aggregated edges $a(e)$ that we want to address. If we divide edge e into sections $S_{visibility}$ by visibility (from each section is visible subset of $a(e)$), we can determine if these aggregated edges are consequent. In each visibility section can be multiple edges, and because of the visibility requirement they are on opposite sides and in this scenario we can select a more significant edge. If the significance equals then we select aggregated edge, which is on the same side as previous referenced aggregated edge with reference to edge e, because it is more likely, that it is continuation of previous edge. For longer edge, there can be multiple sections of visibility. After we choose appropriate aggregated edge for each section, we get sequence of aggregated edges $(A(e)_1, \ldots A(e)_n)$, between which reorientation instruction should be generated as well.
In figure 5.4 you can can see sidewalk represented as a consecutive edges. Consider that we approach this intersection from the bottom sidewalk to the left of the road. Even though graph does not indicate any path to continue straight, user would most likely continue straight onto unmarked sidewalk, therefore we need to indicate change of direction and reorient the user using more significant edges.

Definition 5.2.7. Function $route_ag(P)$ generates optimal sequence of aggregated edges using sectioning described in 5.2.3 resolves possible link to multiple aggregated edges using approach described in 5.2.4. For path $P = (e_1, \ldots, e_n)$.

[1] IE
[2] BE

Figure 5.4: Consequent edges problem

5.2.5 IE instruction generation

Because of our strict aggregation requirements (small angle and being closest edge to more significant edge), there is no possible way to get any closer to addressed significant road and we will be using only advanced maneuvers $m_a \in M_a$, because these maneuvers then have low probability of misinterpretation. Consider consecutive part of edges of path P, denoted $P_a = (e_1, \ldots, e_n)$, which can reference same aggregated edge $A(e) = (c(e), c_l, c_r)$, for some $e \in E$. Our approach is to notify user, that the orientation with respect to consecutive edge $c(e)$ has changed or that he needs to get across the road and get on the other side (join other set of tuple of $A(e)$). In Figure 5.5 we can see a main road with two adjacent sidewalk and a crossing. Consider that we arrive from the bottom left side sidewalk and we need to get to the sidewalk on the other side via the sidewalk and then we can continue in our general direction or in the opposite direction. We need to inform the user about getting on the other side and about general direction. Algorithm is presented in Figure 5.6.

5.2.6 BE instruction generation

Instruction generation between aggregated edges is more difficult, because we need to assess maneuver ambiguity of transfer from one aggregated edge to another and find appropriate aggregated edges to compare it to.

Ambiguity of aggregated edges

Consider current aggregated edge A_c, next aggregated edge A_n and set of other aggregated edges $A_o^s = \{A_{o1}, \ldots, A_{on}\}$ As A_o^s we will then consider set of aggregated edges which are connected to A_c by some oriented edge $e \in A_{oi}$ and $A_c, A_n \notin A_o^s$.

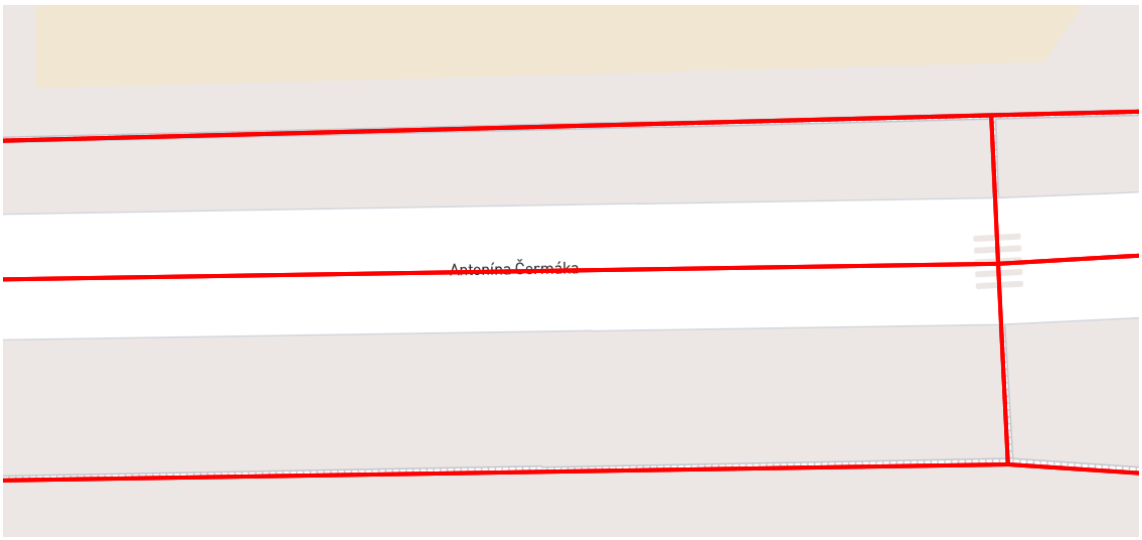

Figure 5.5: IE orientation

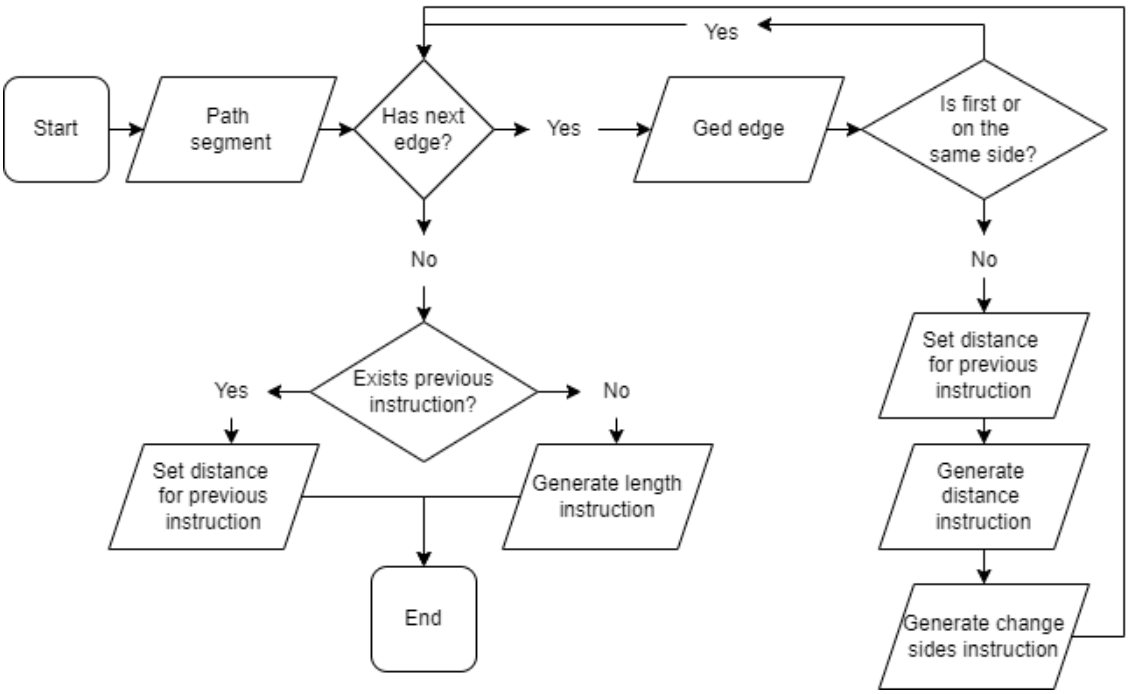

Figure 5.6: IE instruction generation

Definition 5.2.8. Ambiguity of maneuver m between aggregated edges A_c and A_n with other connected aggregated edges edges A_o^s is a function:

$$amb_m(A_c, A_n, A_o^s, e_i, e_j) = \begin{cases} 0 & A_o^s = \emptyset \\ fit_{max} & otherwise \end{cases}$$

where

$$fit_{max} = \max\{(1 - |fit_m(e_i, e_j) - fit_m(e_i, e_o)|)\} \cdot \alpha_r(e_j, e_o) \cdot \alpha_d(e_j, e_o)$$

for all $e_o \in A_o^s$ and

$$\alpha_r(e_j, e_o) = \begin{cases} 1 & roadtype(e_j) = roadtype(e_o) \\ \beta_r \in [0, 1] & otherwise \end{cases}$$

$$\alpha_d(e_j, e_o) = d_{degrading}(d(g(SN(e_j)), g(SN(e_o))))$$

where $d_{degrading}$ is a non-rising function representing our interest in nearby decision point or other edge from the current decision point and function $roadtype(e)$ that assigns a road type based on features $f(e)$ to an edge $e \in E$.

Part of formula $1 - |fit_m(e_i, e_j) - fit_m(e_i, e_o)|$ represents geometrical ambiguity of edges e_o and e_j. In other words, we take a look at aggregated edge $A_o \in A_o^s$ and assume worst possible outcome. In Figures 5.7 and 5.8 we see closer look at intersection of two roads. If we were to make maneuver 'turn left' coming from the bottom onto the left sidewalk, we would consider three outgoing sections with high significance, because all of them have sidewalks on the left side and are close to edge, where the maneuver is to be executed.

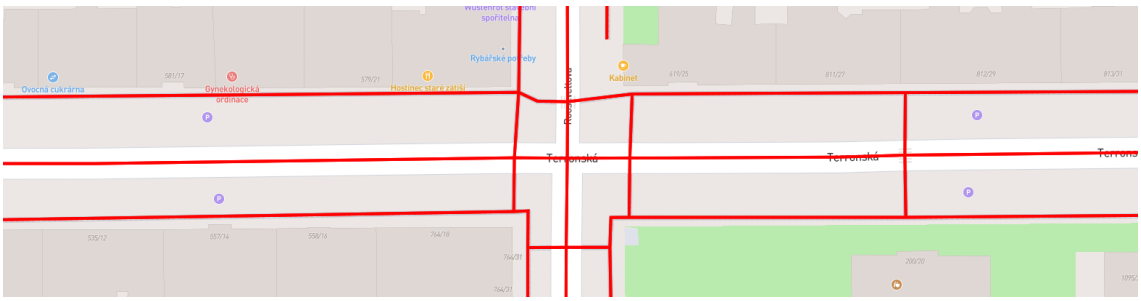

Figure 5.7: Intersection

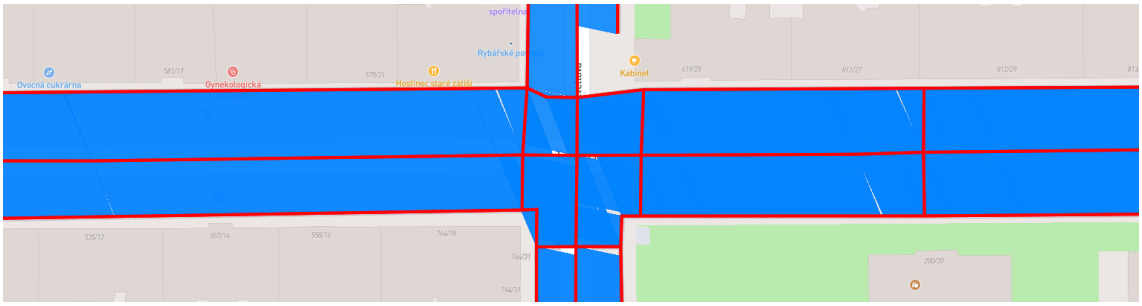

Figure 5.8: Aggregated intersection

Finding optimal maneuver

We want to find maneuver m with low ambiguity and high fitness for current aggregated edge A_c, next aggregated edge A_n and set of other aggregated edges $A_o^s = \{A_{o1}, \ldots, A_{on}\}$, with e_i being last edge from A_c and e_j being first edge from A_n. Consider subset of maneuvers $m_{great\_fitness} = \{m \mid fit_m(e_i, e_j) > \gamma_f\}$ for $m \in M$. This subset represents maneuvers, which would be considered by the user as a possible maneuver, based purely on geometry. This would of course not suffice and we need to find subset of these maneuvers, which are less ambiguous $m_{low\_amb} = \{m \mid amb_m(A_c, A_n, A_o^s, e_i, e_j) < \gamma_r\}$ for $m \in m_{great\_fitness}$. Now we can define term for the least ambiguous maneuver from m_{low_amb}.

Definition 5.2.9. Optimal maneuver $m \in m_{great\_fitness}$ is defined as:

$$optimal(A_c, A_n, A_o^s, e_i, e_j) \begin{cases} \emptyset & m_{low\_amb} = \emptyset \\ \arg\max_{m_{low\_amb}}(fit_m(e_i, e_j)) & otherwise \end{cases}$$

Optimal maneuver does not need to exist, because it would be too risky to execute. Therefore we need to find some way reference the orientation.

Using different set of maneuver

Our approach to non existing optimal maneuver is to select an edge, which causes this ambiguity and present this information to the cyclist. We can find this edge by removing the edge e_r from A_o^s and calculate recalculate the ambiguity without the edge e_r. It this does not resolve the problem, the intersection is deemed too complex and other sources of navigating should be used.

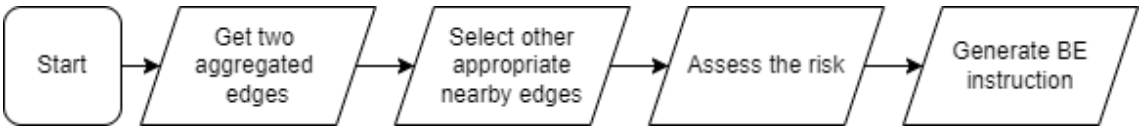

Figure 5.9: BE instruction generation

5.2.7 Instruction joining

Consider list of generated instruction $L = (d_1, m_1, \ldots, m_n)$, where $m \in M \cup M_a$ and d_i is a distance instruction. We can present each m_i and d_{i+1} as one instruction $distanced_i$. For each pair $distanced_i, distanced_{i+1}$, if the distance d of instruction $distanced_i$ is lower than some $dist_{min}$, we can generate new instruction by joining these instructions by 'and then' and replace $distanced_i$ for newly joined instruction. The algorithm is presented in Figure 5.10.

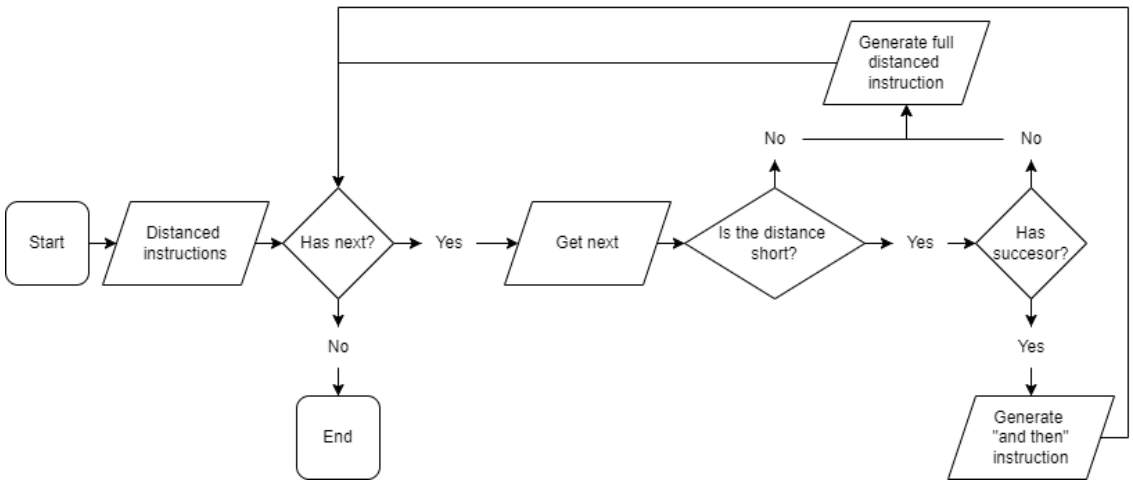

Figure 5.10: Instruction joining

5.2.8 Instruction presentation

Generated instructions are then presented as in a text form, they are tied to an intersection and by approaching to this intersection the generated instruction should be presented to the user. The structure of the text instruction can be described by Backus Naur Form:

$<$ *instruction* $>$:= 'maneuver is too risky' $|<$ *instruction_part* $>|$
$<$ *instruction_part* $>$ ' and then ' $<$ *instruction_part* $>$
$<$ *instruction_part* $>$:=$<$ *maneuver* $>$ ' onto ' $<$ *road_type* $><$ *distance* $>|$
$<$ *extended_maneuver* $><$ *distance* $>$
$<$ *maneuver* $>$:= 'turn sharp left' | 'turn left' | 'turn slight left' | 'continue straight' |
'turn slight right' | 'turn right' | 'turn sharp right'
$<$ *maneuver_extended* $>$:= 'continue' | 'continue in the opposite direction' |
'get on the other side' | 'join the ' $<$ *road_type* $>$
$<$ *distance* $>$:= ' ' | ' for ' $<$ *number* $>$
$<$ *digit_nonzero* $>$::= '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
$<$ *number* $>$:= '0' $|<$ *digit_nonzero* $>|<$ *number* $><$ *number* $>$
$<$ *road_type* $>$:= 'primary road' | 'secondary road' | 'tertiary road' | 'service road' |
'path' | 'track' | 'footway' | 'crossing' | 'steps' | 'cycleway'

$< side >:=$ ‘on the left side of the ’ $< road\_type >|$
‘on the right side of the ’ $< road\_type >$

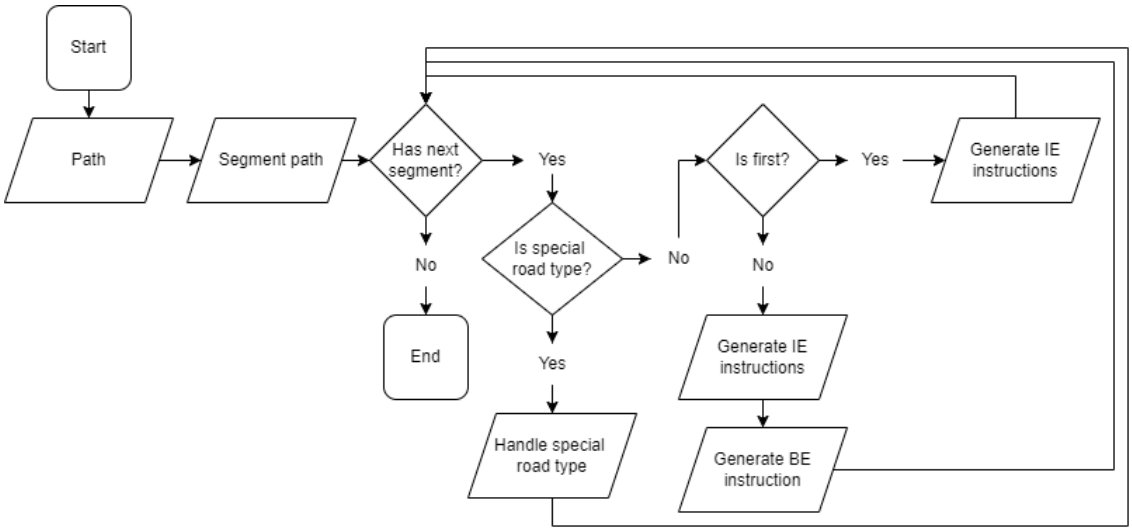

Figure 5.11: Instruction generation

5.2.9 Optimal Route description

By defining optimal maneuver and selection of appropriate aggregated edges, we can generate optimal sequence of instructions describing given path P, which maximizes combined fitness featured in 3.4.

Chapter 6

Development of the app

This chapter describes development of resulting application using approach described in Chapter 4 and Chapter 5. Project is separated into backend and frontend (web UI). Figure 6.1 shows entire process from acquiring the data to Web application and presentation of the result.

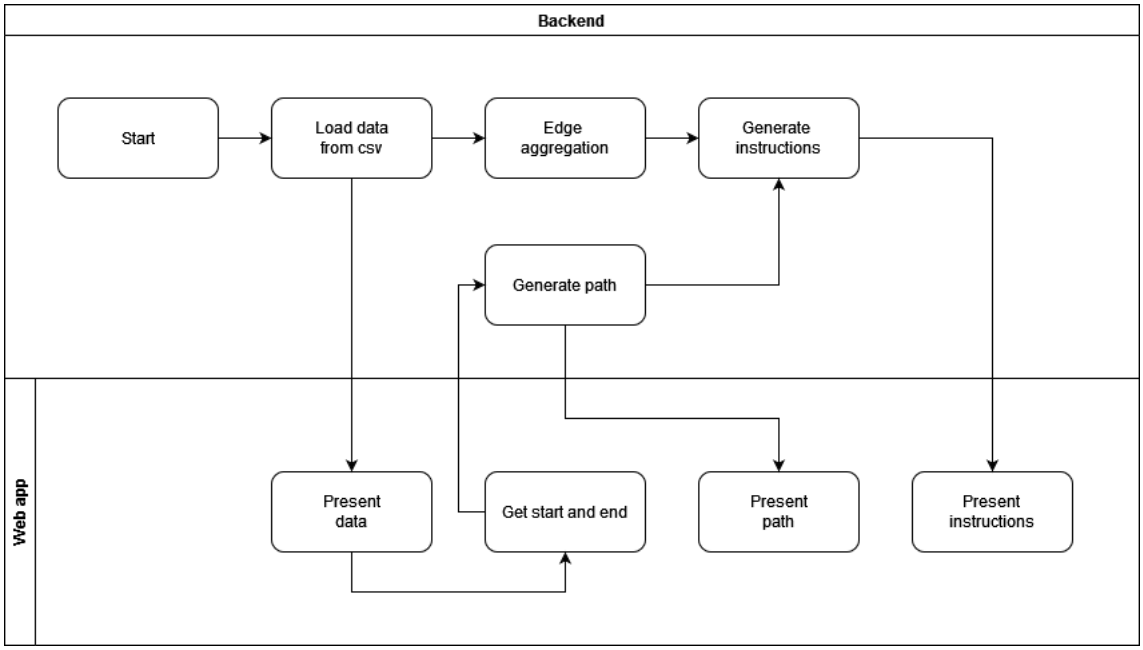

Figure 6.1: Flowchart of application

6.1 Backend

Application is developed in java (jdk 17), where we created custom libraries for 2D linear algebra and edge aggregation. Project uses Jakarta Servlets[1] for easy server deployment. Application can be compiled into WAR file[2] and then deployed into any server supporting this file type or Jakarta Servlets. Application was tested in Tomcat[3] server 10.1.18.

6.1.1 Building enriched routing graph

Application requires 2 csv files to run. First being list of nodes, containing information about node ID and coordinates. Second being list of edges containing information about road type etc. After providing these two files, enriched routing graph is constructed and edge aggregation is commenced. Depending on size of the routing graph and hosting machine, aggregation may take up to several minutes.

6.1.2 Generation of route

Route generation is done by cyclers api[4]. If communication with the cyclers api fails, Dijkstra algorithm is deployed.

6.1.3 Interface

Because of the Jakarta Servlets, we provide simple http POST and GET interface. Returned data contain GeoJSON[5] file to visualize the response on a map.

Nodes

GET method on address '*nodes*' returns a GeoJSON file containing feature collection of points with title containing their ID.

Edges

GET method on address '*edges*' returns a GeoJSON file containing feature collection of lines with an array of features.

[1] `https://jakarta.ee/learn/docs/jakartaee-tutorial/current/web/servlets/servlets.html`
[2] `https://en.wikipedia.org/wiki/WAR_(file_format)`
[3] `https://tomcat.apache.org/`
[4] `https://cyclers.tech/`
[5] `https://geojson.org/`

Route

POST method on address '*route*' requires a data in a format

```
{data: {
    f: fromID,
    t: toID
    }
}
```

where fromID and toID are unique identificator of nodes. This prepares server for route planning.
Get method on address '*route*' returns a sequence of edges in a same format as edges in 6.1.3.

Instructions

POST method on address '*route*' requires a data in a format

```
{data: {
    f: fromID,
    t: toID
    }
}
```

where fromID and toID are unique identificator of nodes. This prepares server for route description.
GET method on address '*route*' returns a feature collection with this structure:

```
{"type":"FeatureCollection",
"features":[
    {"type":"Feature",
    "geometry":{"coordinates":[longitude,latitude],
        "type":"Point"},
    "properties":{
        "title":"maneuver description",
        "fitness":"fitness values for each maneuver",
        "risk":"risk values for each maneuver"}},
    ...
]}
```

Polygons

For visualization of aggregated edges, we generate GeoJSON polygons stored in feature collection. GET method on address '*polygons*' returns a feature collection of polygons.

6.2 Web application

Our demo web application is developed in Jakarta Server Pages(JSP[6]) for easy generation of html code and communication with Jakarta Servlets.

6.2.1 Appearance

Web application is divided into two sections: visualization of the data and informational menu.

Figure 6.2: Web app

6.2.2 Data visualization

GeoJSON data are visualized using an interactive Mapbox map[7], where all the data (except for initial routing graph) are dynamically generated.

Edge interaction

By clicking at any edge in Mapbox map, we are able to see all the edge features in the menu on the left.

Instruction interaction

By clicking at any generated instruction presented the geometrical fitness of the maneuver and ambiguity of presented instruction. These values are shown in the menu.

[6] `https://jakarta.ee/learn/specification-guides/`
[7] `https://www.mapbox.com/`

Edge information

numberOfLanes	1
roadType	FOOTWAY
surface	EXCELLENT
isRoundabout	false
cycleInfrastructure	NONE
isOneway	false
isPavement	true
length	123.22008336193385

Figure 6.3: Edge information

Fitness

SHARP_LEFT	0.2431867667293714
LEFT	0.9836420716866228
SLIGHT_LEFT	0.41934519271031107
STRAIGHT	0.018842722720827796
SLIGHT_RIGHT	0.00008923866438316623
RIGHT	4.454509420351292e-8
SHARP_RIGHT	2.3436033896916814e-12

(a) Fitness

Risk

SHARP_LEFT	0.9185342993710217
LEFT	0.0274669248516195
SLIGHT_LEFT	0.8041462508437285
STRAIGHT	0.22259913292198885
SLIGHT_RIGHT	0.7010911492997375
RIGHT	0.9881661337026991
SHARP_RIGHT	0.9999506342695126

(b) Risk

Figure 6.4: Fitness and ambiguity of an instruction

6.2.3 Menu sections

Navigation

In the navigation section, you can select numbers of two nodes and then the generated path together with generated instructions will be visible on the map.

Styling

User can select one of the selected map styles. User can also select what information should visualize in ‘Map layers’ section. These layers show information ranging from provided data, through edge aggregation to generated instructions.

Chapter 7

Evaluation

In this section we will go through results and evaluate the output of the algorithm.

7.1 Selecting hyper-parameters

In our algorithm, we use hyper-parameters, and in this section we will give these hyper-parameters meaning and explain why we set them to particular value.

7.1.1 Geometrical fitness and ambiguity

In Figure 5.2 you can see normal distribution functions. Each having it's distinct mean, which is appropriate to general direction. We assigned same deviation to each normal distribution function, $N(\mu, 30)$. This ensures that the fitness of an oriented angle deviated more than 90° is really low, at the same time fitness of oriented angle deviated less than 45° is still considerable and for oriented angle deviating less than 30° the fitness is at least 0.5.

When calculating ambiguity we use constant β_s, which is set to '0.7'. This can be interpreted as changing the oriented angle between two edges by 20° and helps with 'ignoring' less significant edges, when traversing infrastructure for motorized vehicles.

Function $d_{degrading}$, also used in ambiguity calculation, is another normal distribution function $N(0, 8)$. This value helps with finding optimal maneuver in more dense parts of a routing graph. By increasing deviation σ of the $d_{degrading}$, the ambiguity calculation becomes more aware of surrounding edges and in parts of graph with higher density of edges, our method becomes unreliable and unusable.

7.1.2 Fitness and ambiguity thresholds

In Definition 5.2.9 we use maneuvers, which have ambiguity lower than γ_r and geometrical fitness higher than γ_f. We want to ensure orientation shown in Figure 5.3. To ensure this requirement, we need to find intersection between two neighboring fitness functions, in our case minimal allowed fitness γ_f would be 32, but we decided to lower this bound to 30, to fight numerical errors.
With similar approach we set γ_r to 75. This helps us with multi-arm intersections without using topological numbering of edges.

7.1.3 Orientation of consecutive edges

For the interest function used for weighted average orientation in Definition 5.2.2 was selected normal distribution function $N(0, 15)$. Other values were too short-sighted or far-sighted.

7.2 Testing of routes

In this section we will showcase results of our algorithm on few selected roads and assess the generation of decision and flexibility of our instruction set. The algorithm was tested in our developed app. All of the shown examples can be recreated in our developed app.

Route 1 Figure 7.1 displays evaluated route.

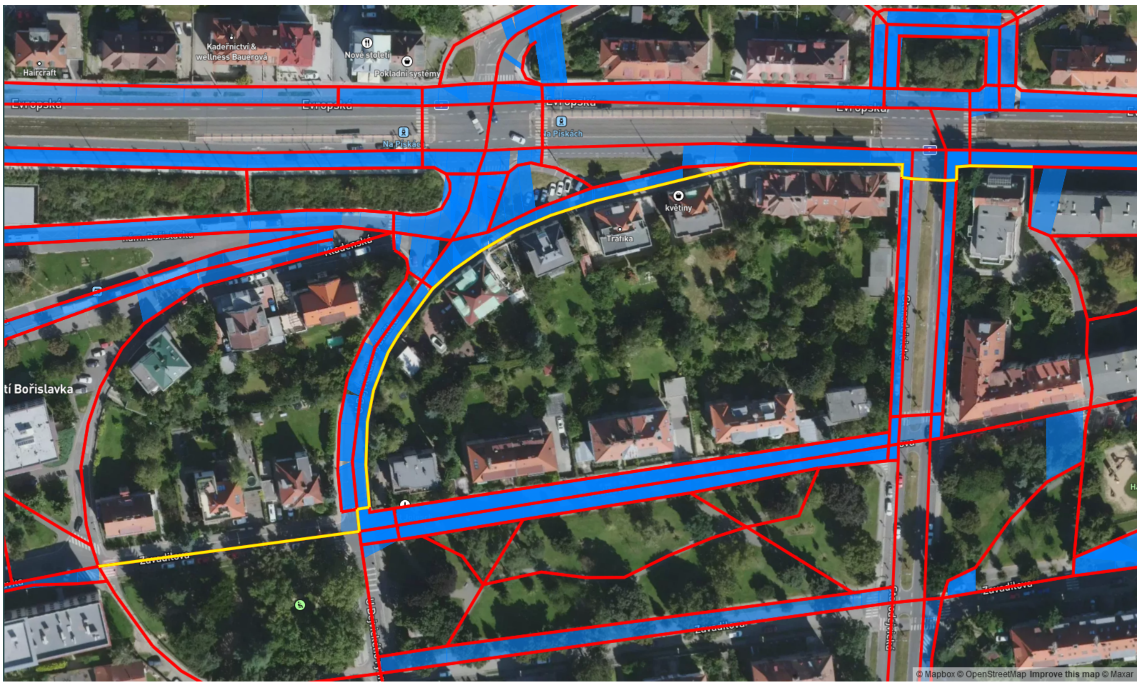

Figure 7.1: Route from 291741 to 247682

Figure B.1 displays first two decision points. First BE instruction being "Turn left onto service road and continue for 8 meters" caused by intersection of two service roads. Risk of this instruction in the direction of the maneuver was 0.07 and fitness 0.94, because no nearby edges with similar description were present. Second being "Get on the footway on the right side of the road" caused by changing sides in an aggregated edge IE instructions do not have ambiguity evaluation, because they are too general.

Figure B.2 displays BE distance instruction "continue for 163 meters on the right side of service road" serving more of a reassuring purpose.

Figure B.3 shows two BE instructions "Continue straight onto crossing and continue for 17 meters on the right side of primary road" and "Continue straight onto crossing and continue for 49 meters on the right side of primary road" both caused by intersection of nodes to the left of the crossing. These instructions should have been represented as one, but unfortunately we did not have time to implement this feature.

If we change direction of the last maneuver we get instruction "Keep on the right side and Turn right" seen in Figure B.4. This consecutive edge was aggregated to both service roads and by great deviation of our orientation was generated this specialized BE instruction. Note that without the aggregation or awareness of nearby edges, no instruction would be generated, because the continuation rule would be applied.

Route 2 Figure 7.2 displays evaluated route. In Figure B.5 you can see two decision

Figure 7.2: Route from 264060 to 26642

points. First being "Turn slight right onto path and continue for 16 meters", which was generated almost purely on geometrical fitness, because no unambiguous roads are nearby. Second being "Choose path in the direction of straight closer to your right and continue for 255 meters", which is a instruction generated by algorithm described in section 5.2.6 because of high ambiguity of these edges.

Route 3 In Figure B.6 we can see a decision point, where our algorithm deemed this intersection as too dangerous and instruction generation failed.

Route 4 In Figure B.7 (where the cyclist is arriving from the bottom) we can see a instruction "Turn slight left onto path and continue for 25 meters" presented after "Get on the righ side of the road", which would not correspond with orientation of the user after joining the footway on the right.

7.3 Aggregation of edges

Our data consisted of 9034 distinct edges each with two possible orientations. 3388 edges were assigned as edges with the highest significance and therefore used as main reference points. 2118 edges were able to reference edges with higher significance. Other edges were not aggregated.

Chapter 8

Discussion

In this section we will discuss the quality of generated instructions and reliability of the algorithm The results presented in previous section should provide insight into the advantages and disadvantages of our algorithm.

Where does the algorithm perform well? The instruction generation heavily relies on edge aggregation and therefore in routing graphs with low variety of edges with different significance, the algorithm might not perform that well and may not select right decision points. Therefore the algorithm performs better in urban areas, where there is abundance of more significant edges.

Where does the algorithm fail? The algorithm in some cases (like in B.7) fails to generate correct instructions, because the algorithm does not take in account orientation of the user after instruction generation from another instruction set. Another problem arises with parts of graph, where there is higher density of edges with same significance connected in quick succession and high branching. In this scenario, the algorithm generates too many instructions, because consecutive edges are short.

Is the instruction set appropriate? We used three different instruction sets (IE, BE and solution for too ambiguous edges). Our instructions are based only on distance and change of direction. We did not use non-distance based instructions like "take third right". Use of these instructions would benefit the description of the route (prioritization of these instruction is mentioned in [4]).

Is ambiguity great for instruction selection of maneuver? In most cases the ambiguity does great job at filtering maneuvers, which could be more likely misinterpreted, however in some cases (like B.5) two roads were too similar and instruction "turn slight right" could be used. But this problem arises, because we used fitness function of maneuver for whole 360°. If we would have restricted fitness function of each maneuver to a particular interval, then we would not have to compare left side with the right side and therefore resolve this case with BE instruction.

Chapter 9

Conclusion

We have separated route description into two stages: edge aggregation and generation of instruction. Edge aggregation can be calculated in advance and therefore not affect speed of instruction generation. We were able to find sequence of appropriate significant edges for given path and later use them as a reference points. Instruction generation is divided into two sections: IE instruction generation and BE instruction generation and therefore allows usage of different sets of maneuvers, where decision points are dependant on aggregated edges.

We proposed solution to get more approximate orientation of a road, based on consecutive edges, and created a metric for ambiguity of two roads from some intersection, based on the geometry and road types. Selection of appropriate BE maneuver is influenced on oriented angle and ambiguity of nearby roads.

Finally we implemented this approach in java application, which can be deployed in server supporting WAR[1] files. And in this application we showcased advantages and shortcomings of using this algorithm.

Proposed algorithm for instruction generation uses similar technique as landmark-based instruction generation. It performs well in some areas, but is not robust enough, because of is using only distances, not referencing surrounding roads more precisely and being shortsighted.

9.1 Future work

As mentioned above, our solution is not robust enough. Generated instructions lack information about approaching a decision point and confirmation of execution of correct maneuver. Information about next decision point is done only by providing distance and not mentioning possible turns in between. Another improvement would be different type of aggregation, where edges or nodes could be aggregated by their proximity, and therefore

[1] `https://en.wikipedia.org/wiki/WAR_(file_format)`

different set of instructions could be used. Mentioning intersection type could help to distinguish the decision point as well.

This thesis should serve as a stepping stone for future works and research on instruction generation in an enriched routing graph.

Bibliography

[1] Michel Denis. "The description of routes: A cognitive approach to the production of spatial discourse". In: *Cahiers de Psychologie Cognitive* 16 (Aug. 1997), pp. 409–458.

[2] Angela Schwering Vanessa Joy Anacta Jakub Krukar. *Landmark-Based Navigation in Cognitive Systems - KI - Künstliche Intelligen*. https://link.springer.com/article/10.1007/s13218-017-0487-7.

[3] Kai Hamburger, Florian Röser, and Markus Knauff. "Landmark selection for route instructions: At which corner of an intersection is the preferred landmark located?" In: *Frontiers in Computer Science* 4 (2022). ISSN: 2624-9898. DOI: 10.3389/fcomp.2022.1044151. URL: https://www.frontiersin.org/articles/10.3389/fcomp.2022.1044151.

[4] Adam Rousell and Alexander Zipf. "Towards a Landmark-Based Pedestrian Navigation Service Using OSM Data". In: *ISPRS International Journal of Geo-Information* 6.3 (2017). ISSN: 2220-9964. DOI: 10.3390/ijgi6030064. URL: https://www.mdpi.com/2220-9964/6/3/64.

[5] Phil Bartie Candela Sanchez-Rodilla Espeso William Mackaness. *Understanding Information Requirements in "Text Only" Pedestrian Wayfinding Systems*. https://link.springer.com/chapter/10.1007/978-3-319-11593-1_16.

[6] Mary Hegarty Daniel R. Montello Kristin L. Lovelace. *Elements of Good Route Directions in Familiar and Unfamiliar Environments*. https://link.springer.com/chapter/10.1007/3-540-48384-5_5.

[7] Martin Raubal Stephan Winter. *Enriching Wayfinding Instructions with Local Landmarks*. https://link.springer.com/chapter/10.1007/3-540-45799-2_17.

[8] Michel Denis Ariane Tom. *Language and spatial cognition: comparing the roles of landmarks and street names in route instructions*. https://onlinelibrary.wiley.com/doi/10.1002/acp.1045. 2004.

Appendix A

Enum types

A Road type

- PRIMARY[1]
- SECONDARY[2]
- TERTIARY[3]
- SERVICE[4]
- RESIDENTIAL[5]
- PATH[6]
- TRACK[7]
- FOOTWAY[8]
- CROSSING[9]
- STEPS[10]
- CYCLEWAY[11]
- UNKNOWN[12]

[1] I. class roads
[2] II. class roads
[3] III. class roads
[4] generally for access to a building, service station, industrial estate, etc., or unclassified roads connecting houses and buildings
[5] living streets (streets where is lowered allowed speed due to contact with pedestrians)
[6] narrow roads in the countryside, mostly unpaved
[7] wide roads in the countryside, mostly unpaved
[8] footway
[9] crossing
[10] stairs
[11] dedicated road/street for cyclists.
[12] uknown

B Cycle infrastructure

- TRACK[13]
- ZONE[14]
- LANE[15]
- SHARROW[16]
- NONE[17]

C Surface

- UNKNOWN
- EXCELLENT[18]
- GOOD[19]
- BAD[20]
- HORRIBLE[21]
- IMPASSABLE[22]

[13] dedicated track for cyclists separated from car traffic
[14] pedestrian zone with allowed access by bike
[15] mandatory cycle lane
[16] advisory cycle lane
[17] no cycling infrastructure is present
[18] brand new asphalt
[19] old asphalt roads or concrete roads, or paving stones with very narrow gaps
[20] cobblestones with bigger gaps, not compacted unpaved roads
[21] roads hardly used for bicycle
[22] roads that should be used only as a last resort

Appendix B

Images

A Routes

Figure B.1: Two consequent decision points

Figure B.2: Distance instruction

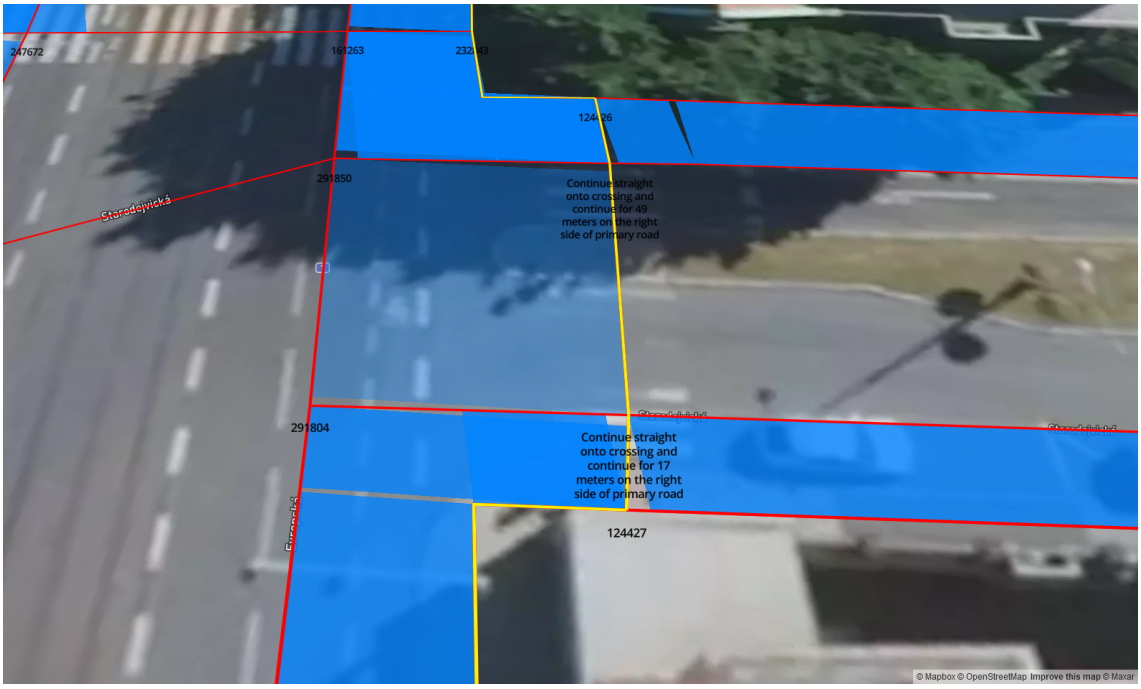

Figure B.3: Continuation

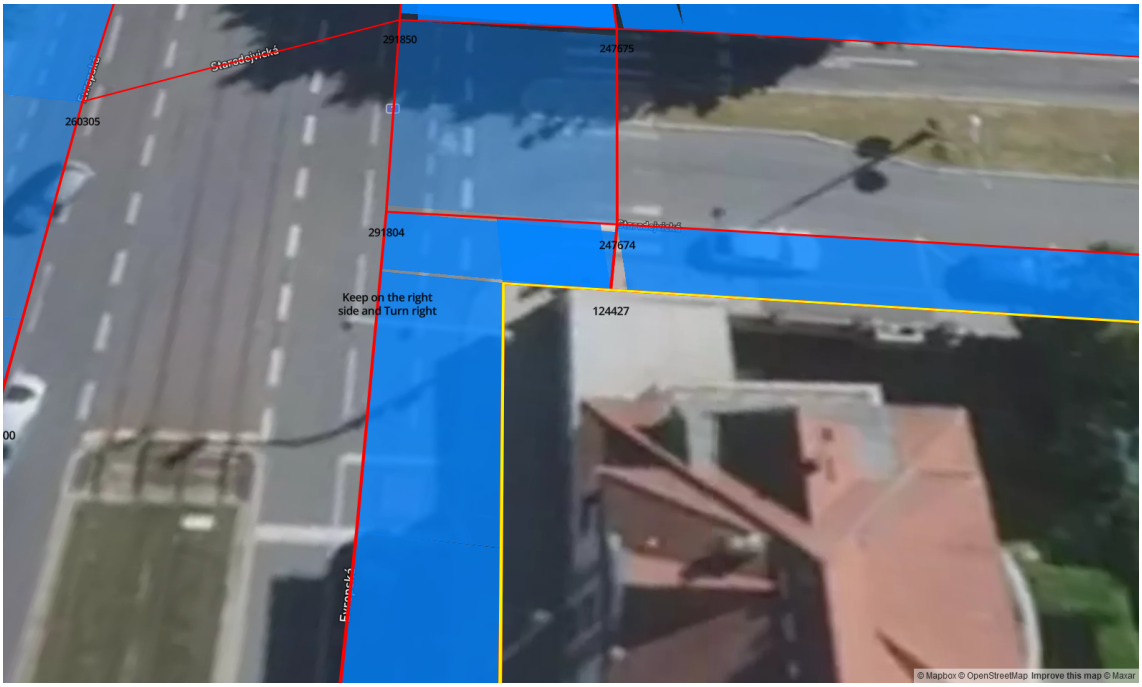

Figure B.4: Decision point caused by aggregated edges

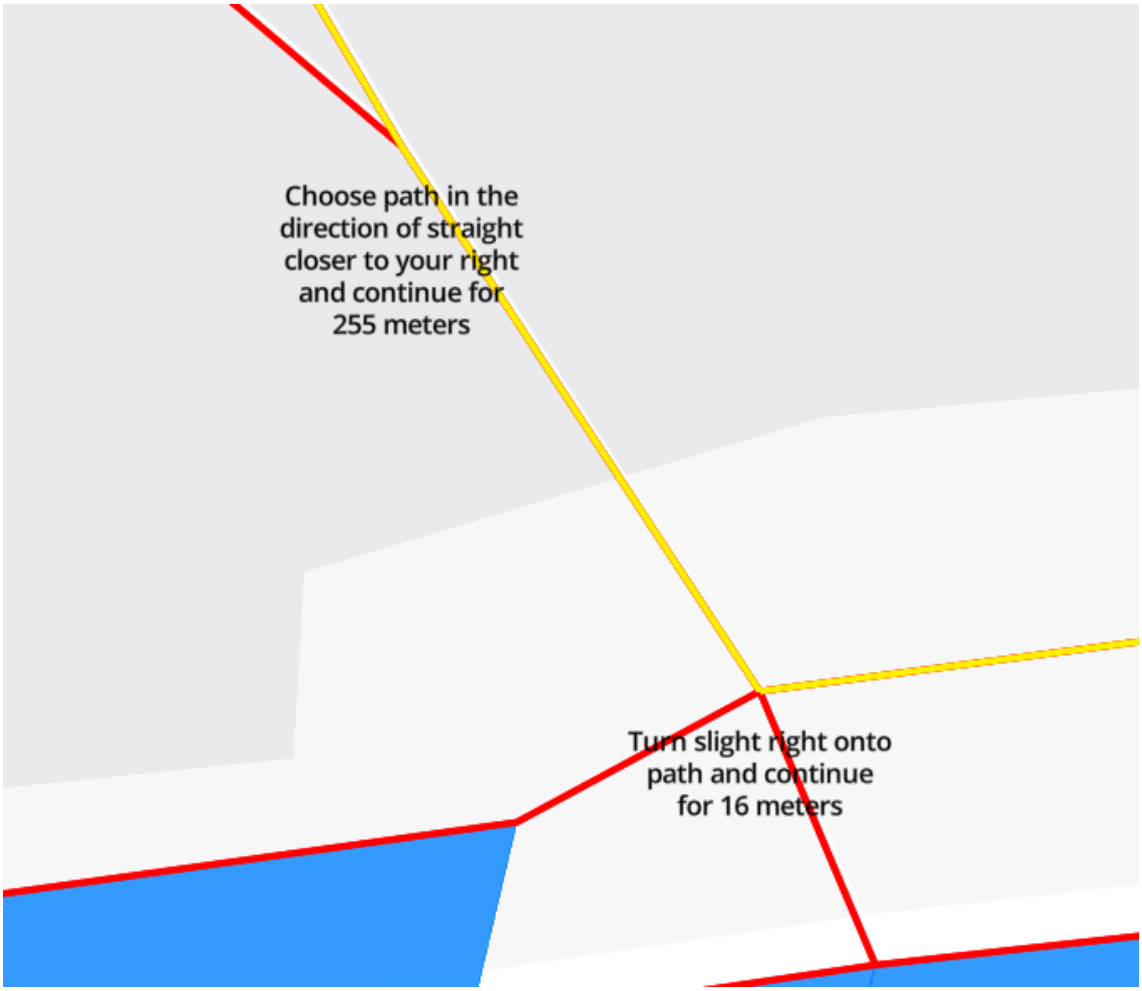

Figure B.5: Similarity of roads

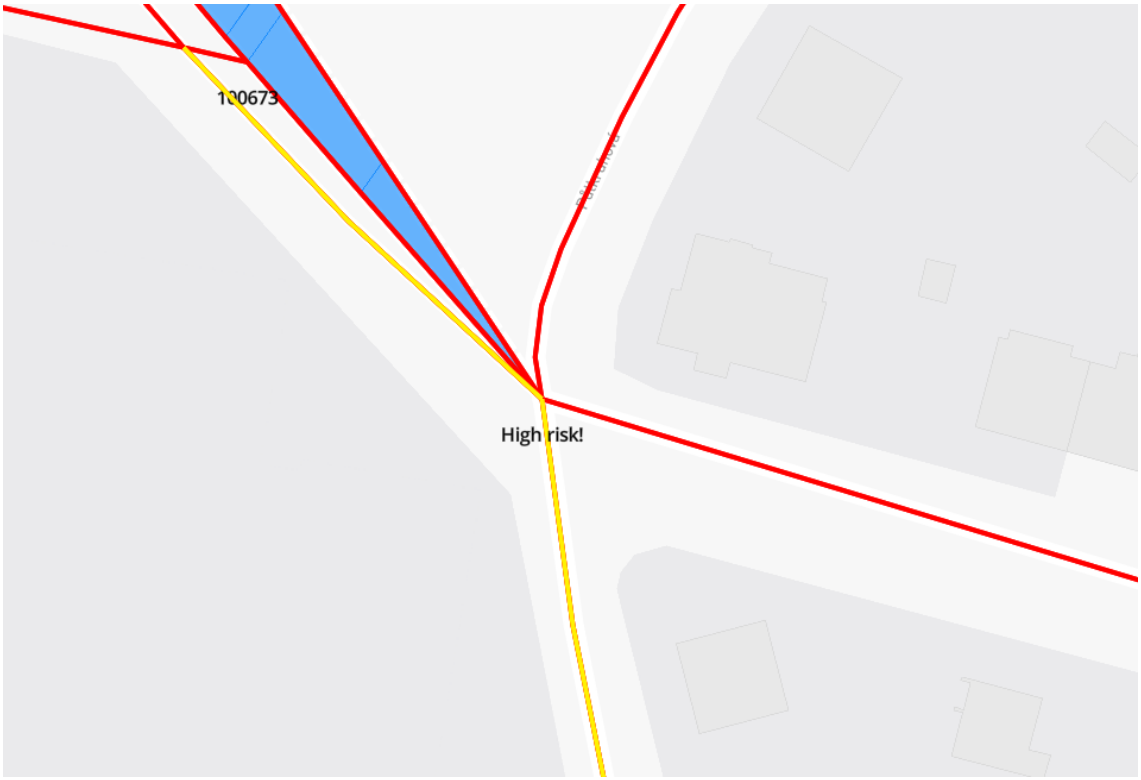

Figure B.6: Route from 254753 to 1162412

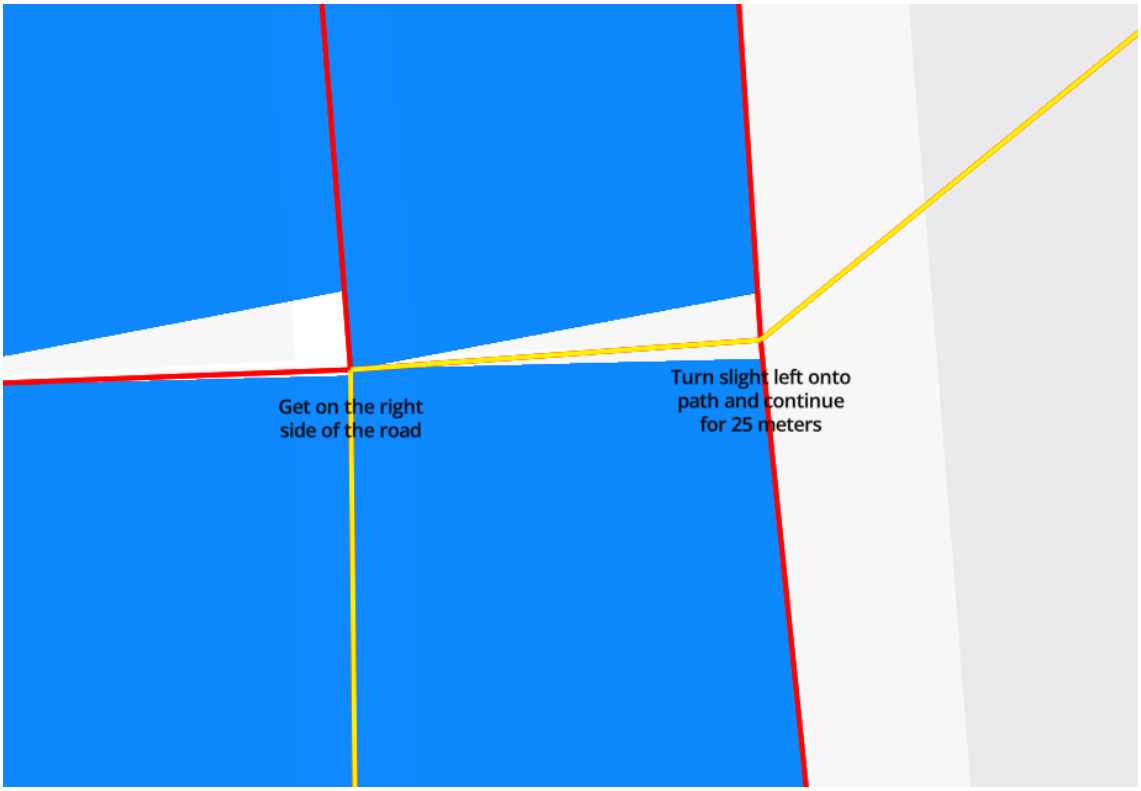

Figure B.7: Route from 102401 to 117269

Appendix C

Little bit of linear algebra

Node can be represented as a point in space and edge is comprised as a sequence of line segments. As coordinates are used latitude and longitude. In this paper, we work only with vectors in two dimensions.

A Line segment

Line segment [1] is a part of a straight line that is bounded by two distinct end points, and contains every point on the line that is between its endpoints. We define an **opposite line segment** $opposite(l)$ to line segment $l = (\vec{start}, \vec{end})$ as:

$$opposite(l) = (\vec{end}, \vec{start})$$

B Angles

B.1 Angle using dot product formula

To calculate cosine of angle between two vectors $(\vec{u}, \vec{v})$ we can use dot product formula [2]:

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|}$$

and then simply to get the angle use:

$$\theta = \arccos \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \cdot \|\vec{v}\|}$$

B.2 Oriented angle between ingoing and outgoing vector

To determine oriented between ingoing vector $\vec{a}$ and outgoing vector $\vec{b}$ of some node we can use law of cosines [3]:

$$\|\vec{a}\| = \|\vec{b}\| + \|\vec{c}\| - 2 \cdot \|\vec{a}\| \cdot \|\vec{b}\| \cdot \cos\theta$$

, where vector $\vec{c} = \vec{a} + \vec{b}$. To get the angle θ we use:

$$\alpha = \arccos\left(\frac{\|\vec{a}\|^2 + \|\vec{b}\|^2 - \|\vec{c}\|^2}{2\|\vec{a}\|\|\vec{b}\|}\right)$$

[1] `https://en.wikipedia.org/wiki/Line_segment`
[2] `https://en.wikipedia.org/wiki/Dot_product`
[3] `https://en.wikipedia.org/wiki/Law_of_cosines`

If the vector $\vec{b}$ is oriented to the right from the perspective of vector $\vec{a}$ then we choose complementary angle $\theta^{'} = 360 - \theta$.

B.3 Perpendicular vector

Vector $\vec{u}$ is perpendicular to vector $\vec{v}$ if $\vec{u} \cdot \vec{v} = 0$. In two dimensions are two perpendicular vectors to $\vec{v} = (x, y)$, first is $\vec{u} = (x, -y)$ and second is $-\vec{u}$, because $\vec{u} \cdot \vec{v} = 0$ and $\vec{u} \cdot -\vec{v} = 0$ for any number.

C Orientation

C.1 Using dot product

We can use dot product to separate space to 2 parts, one above and one bellow hyperplane determined by it's normal vector $\vec{n}$. Let

$$product = \vec{n} \cdot \vec{p}$$

where $\vec{p}$ is a point in space. If the $product > 0$ then the point $\vec{p}$ is above the hyperplane, if the $product < 0$ then the point $\vec{p}$ is bellow the hyperplane, otherwise it lies on the hyperplane.

Definition

We define function:

$$isAbove(\vec{vector}, \vec{point}) = \vec{vector} \cdot \vec{point} > 0$$

where $\vec{vector}$ is normal vector to hyperplane and $\vec{point}$ is point in space.

C.2 Using determinant

Because we use 2 dimensions (longitude and latitude), we can use the determinant [4] of two vectors to tell if one vector is on the left/right of the other one. The determinant gives the signed n-dimensional volume of n-dimensional parallelotope. In two dimensions the sigh represent orientation between these vectors. Let

$$\det M = \det \begin{vmatrix} \vec{a} & \vec{b} \end{vmatrix}$$

If $\det M > 0$ then the vector $\vec{b}$ is on the right of the vector $\vec{a}$, if $\det M < 0$ then the vector $\vec{b}$ is on the left of the vector $\vec{a}$, otherwise vectors $\vec{a}$ $\vec{b}$ have same orientation.

Definitions

We define functions:

$$isOnRight(\vec{vector}, \vec{point}) = \det \begin{vmatrix} \vec{vector} & \vec{point} \end{vmatrix} > 0$$

and

$$isOnLeft(\vec{vector}, \vec{point}) = \det \begin{vmatrix} \vec{vector} & \vec{point} \end{vmatrix} < 0$$

C.3 Perpendicular line sections

In two-dimensional space we can get 2 perpendicular line sections with same starting point and same size. Let's have line segment l with start $\vec{s}$ and direction $\vec{d}$. Set of perpendicular line segments is $perp = \{(\vec{s}^{'}, \vec{d}^{'}) | \vec{s}^{'} = \vec{s} \wedge \vec{d}^{'} \cdot \vec{d} = 0 \wedge \|\vec{d}\| = \|\vec{d}^{'}\|\}$ We define:

[4] https://en.wikipedia.org/wiki/Determinant

Perpendicular line $l = (\vec{s}', \vec{d}')$ is **perpendicular left** if $l \in perp$ and $isOnLeft(\vec{d}, \vec{d}')$.
Perpendicular line $l = (\vec{s}', \vec{d}')$ is **perpendicular right** if $l \in perp$ and $isOnRight(\vec{d}, \vec{d}')$.

D Computation with line segment

We can use formulas in C with line segments with small adjustments. Line segment has a start point $\vec{s}$ and an end point $\vec{e}$ and the direction $\vec{d} = \vec{e} - \vec{s}$. If we use the direction $\vec{d}$ and subtract vector $\vec{s}$ from other vectors in coordinate space then we can use formulas in C.

Definition

Let's have line segment $l = (\vec{s}, \vec{e})$ and point $\vec{p}$. Direction $\vec{d} = \vec{e} - \vec{s}$, opposite direction $\vec{o} = \vec{e} - \vec{s}$, point $\vec{n} = \vec{p} - \vec{s}$ and point $\vec{m} = \vec{p} - \vec{e}$. We define functions:

$$isOnRight(l, \vec{p}) = \det \begin{vmatrix} \vec{d} & \vec{n} \end{vmatrix} > 0$$

$$isOnLeft(l, \vec{p}) = \det \begin{vmatrix} \vec{d} & \vec{n} \end{vmatrix} < 0$$

$$isBetween(l, \vec{p}) = isAbove(\vec{d}, \vec{n}) \wedge isAbove(\vec{o}, \vec{m})$$

E Projection

E.1 Linear space

Orthogonal projection [5] of $\vec{v}$ onto the line spanned by a nonzero $\vec{s}$ is:

$$proj_{\vec{s}}(\vec{v}) = \frac{\vec{v} \cdot \vec{s}}{\vec{s} \cdot \vec{s}} \cdot \vec{s}$$

E.2 Affine space

Orthogonal projection of $\vec{v}$ onto the line spanned by a nonzero $\vec{s}$ shifted by point $\vec{p}$ is:

$$proj_{\vec{s}}(\vec{v} - \vec{p})$$

F Intersection

F.1 Intersection of two two-dimensional lines

To find intersection of two affine spaces we can use GEM [6]. Let l_1 be a parametric line with shift $\vec{p_1}$ and direction $\vec{d_1}$ and l_2 be a parametric line with shift $\vec{p_2}$ and direction $\vec{d_2}$. We try want to find coefficients $(k1, k2)$ of linear combination [7] of vectors $\vec{d_1}$ and $\vec{d_2}$ which equals to vector $\vec{v} = \vec{p_2} - \vec{p_1}$. Then the intersection is $k1 \cdot \vec{d_1} + \vec{p_1}$.

F.2 Getting part of line segment above affine hyperplane

To get part of Line segment we newline to take a look at the end points: star point $\vec{s}$ and end point $\vec{e}$.

- If both points are above or on the affine hyperplane, whole line segment is above and result is original line segment.

[5] `https://textbooks.math.gatech.edu/ila/projections.html`
[6] `https://en.wikipedia.org/wiki/Gaussian_elimination`
[7] `https://en.wikipedia.org/wiki/Linear_combination`

- If both points are bellow , whole line segment is bellow and result is $\emptyset$.
- If one point is above, we can calculate Intersection of two two-dimensional lines $\vec{i}$.
  - If start is above, we get: Line segment with original start and end $\vec{i}$
  - If end is above, we get: Line segment with start $\vec{i}$ and original end

F.3 Strip intersection

We define **stripIntersection(strip, lineSegment)**, where both parameters are line segments and is result of F.2 between two affine hyperplanes defined by strip and opposite(strip).

G Distance of two points on a sphere

To get distance of two points using coordinates on a sphere we can use **Haversine formula** [8]. First point $\vec{p}_1 = (lat_1, lon_1)$ second point $\vec{p}_2 = (lat_2, lon_2)$. After reforming Haversine formula we get that distance

$$d(vecp_1, vecp_2) = 2 \cdot r \cdot \arcsin \sqrt{\sin^2(\frac{lat_2 - lat_1}{2}) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2(\frac{lon_2 - lon_1}{2})}$$

where r is radius of earth and equals 6378.

[8] `https://en.wikipedia.org/wiki/Haversine_formula`