EVALUATION AND REPRESENTATION OF THE RISK OF INTERSECTIONS FOR CYCLISTS AT LANE-LEVEL

A GEOSPATIAL APPROACH

Master Thesis

by
Elena Gaus

supervised by
Dr. Martin Loidl

Department of Geoinformatics

Salzburg, May 2023

Submitted to the Department of Geoinformatics in partial fulfilment of the requirements for the degree of Master of Science (MSc) in Applied Geoinformatics at the Paris-Lodron-University of Salzburg.

ABSTRACT

Cyclists are involved in a disproportionately high number of traffic accidents. Approximately fifty percent of accidents involving cyclists occur at intersections. To proactively counteract these dangers, closer insights in the intersection are necessary. However, the current standard representation of a transport network is the graph model which displays intersections in the form of zero-dimensional nodes. This shape is insufficient to investigate the events that take place at an intersection in more detail. Therefore, it is imperative to process an intersection in a way that the risk for cyclists can be represented and evaluated. This is the goal of this thesis.

Based on the GIP (Intermodal Transport Reference System of Austria) and using SQL scripts, this thesis accomplishes the conversion of the zero-dimensional node into a one-dimensional and two-dimensional representation of the intersection. The result is a lane-accurate resolution of the intersection that reflects turning maneuvers. In this form, the intersection can be evaluated in detail using factors that affect the risk to a bicyclist at an intersection. The indicators are identified through a review of the literature.

The results of this thesis have the ability to evaluate the turning relations of an intersection with respect to their accident risk. This gives cyclists the opportunity to choose a low risk route and actively ensure their safety. Furthermore, it is possible to simulate the influence of traffic measures on the safety of cyclists. The model also allows decision makers to identify and improve high-risk intersections. The two-dimensional representation of the model facilitates the identification of intersection risk zones and is a precursor to the HD map.

Keywords: crossing | intersection plateau | junction | microscale | risk index

ZUSAMMENFASSUNG

Radfahrende sind im Straßenverkehr in verhältnismäßig viele Unfälle verwickelt. Etwa fünfzig Prozent der Unfälle, in die Radfahrende verwickelt sind, finden an Kreuzungen statt. Um den Gefahren proaktiv entgegen wirken, wird sind präzisere Informationen zur Kreuzung nötig. Allerdings ist die derzeitig gängige Darstellung einer Kreuzung im Verkehrsnetz, ein nulldimensionaler Knoten im Knoten-Kanten-Modell, unzureichend um die Geschehnisse in einer Kreuzung näher zu betrachten. Es ist daher dringend nötig, eine Kreuzung so aufzubereiten, dass das Risiko für Radfahrende dargestellt und bewertet werden kann. Dies ist das Ziel dieser Thesis.

Auf Grundlage der GIP (Graphenintegrationsplattform Österreichs) und mittels SQL-Skripten wird in dieser Arbeit die Konvertierung des nulldimensionalen Knotens hin zu einer ein- und zweidimensionalen Repräsentation der Kreuzung bewerkstelligt. Das Ziel ist eine fahrspurgenaue Auflösung der Kreuzung, sodass sie die Abbiegemanöver wiederspiegelt. In dieser Form kann die Kreuzung detailliert bewertet werden mithilfe von Faktoren, die das Risiko für einen Radfahrenden an einer Kreuzung beeinflussen. Die Faktoren werden im Vorfeld durch eine Literaturrecherche identifiziert.

Die Ergebnisse dieser Thesis haben das Potential zum einen die Abbiegerelationen einer Kreuzung bezüglich ihres Unfallrisiko zu bewerten. Das gibt Radfahrenden die Möglichkeit, eine risikoarme Route zu wählen und aktiv für ihre Sicherheit zu sorgen. Zum anderen ist es möglich, den Einfluss von Verkehrsmaßnahmen auf die Sicherheit von Radfahrenden zu simulieren. Und des weiteren ermöglicht das Modell Entscheidungstragenden, risikoreiche Kreuzungen zu identifizieren und zu verbessern. Die zweidimensionale Darstellung gestattet es, die Risikozonen von Kreuzungen zu ermitteln und sie stellt eine Vorstufe zur HD map dar.

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr. Martin Loidl for his dedicated support and attentive supervision, for his constant responsiveness, for his helpful inputs and insightful advice, and last but not least for his patience.
I would also like to thank Christian Werner for his advice, the lively discussions with him that gave me a lot of knowledge and a broader understanding of mobility issues, and for his friendship.
And, of course, a big thank you to the entire Mobility Lab team for their counsel and guidance.

Matthias Fritz für seine steten Ermutigungen und ständige Unterstützung über meine ganze Zeit in Salzburg hinweg und darüber hinaus. Danke dass du immer auf meiner Seite und mein bester Freund bist.
Zu guter Letzt meinen Eltern, die mich immer geliebt und unterstützt haben und mir Unterschlupf boten, wenn ich ihn brauchte.

Federal Ministry
Republic of Austria
Climate Action, Environment,
Energy, Mobility,
Innovation and Technology

FFG

The research leading to these results has received funding from the Mobility of the Future programme. Mobility of the Future is a research, technology and innovation funding programme of the Republic of Austria, Ministry of Climate Action. The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

1D one-dimensional
2D two-dimensional
AOI area of interest
ASFINAG Autobahnen- und Schnellstraßen-Finanzierungs-Aktiengesellschaft
BMV bicycle-motorized vehicle
CRS coordinate reference system
CSV Comma Separated Values
DTM Digital Terrain Model
EPSG European Petroleum Survey Group
ERD entity relationship diagram
GIP Graph Integration Platform
GIS geographic information system
GUI graphical user interface
HD map high-definition map
MV motor vehicles
OEBB Österreichischen Bundesbahnen
ÖVDAT Österreichisches Institut für Verkehrsdateninfrastruktur
OGD open government data
OSM OpenStreetMap
PL/pgSQL procedural language of the database system PostgreSQL

SQL	structured query language
TIN	triangulated irregular network
VRU	vulnerable road users
WFS	web feature service
m	meter
km/h	kilometer per hour

Part I

CONTEXT

1

INTRODUCTION

1.1 MOTIVATION

The streets are becoming increasingly safer, if the index is measured by the number of people injured or killed on the roads, which are falling (Statista Research Department, 2021). However, this is countered by the number of accidents involving cyclists. The share of those accidents is rising, meaning that the number of bicycle injuries decreases slower than the accidents involving cars (OECD/International Transport Forum, 2013). Cyclists are among the vulnerable road users (VRU) on the road as they do not have a physical protection in case of an accident. They have, compared to other road users such as pedestrians and motorists, an increased risk of death (Bouaoun, Haddak, & Amoros, 2015).

Intersections are overly hazardous for cyclists

Intersections are places where various traffic flows meet and mix. Not only do traffic flows from different directions with different destinations meet, but also road users who are guided separately along the road segment. These aspects make intersections points of conflict. About fifty percent of accidents involving cyclists occur at intersections. Crashes at intersections constitute one of the most common and dangerous scenarios in traffic as several researchers point out in their works (Boufous, Rome, Senserrick, & Ivers, 2011; Dozza & Werneke, 2014; Prati, Marín Puchades, Angelis, Fraboni, & Pietrantoni, 2018; Pschenitza, 2017). The risk of an accident happening is composed of various factors. Prati et al. (2018) discovered in their systematic review that the factors *road user(s)* and *infrastructure* are those that have been researched the most.

A node is insufficient if the intersections shall be examined in detail.

The common representation used to display transport infrastructure is the graph model. It represents road segments as graphs and intersections as nodes in which the graphs converge. This model is useful for routing purposes because it is a simple and space-efficient way of storing and analyzing road networks. But for many other purposes, the zero-dimensional representation of an intersection is inadequate. This is because it does not capture the geometry and complexity of an intersection - how do the lanes run? Where do they meet? Where are the corresponding risk zones? A zero-dimensional node cannot answer these questions. It is not able to display the infrastructure of an intersection.

Answering these questions is also of interest from a routing perspective: there is currently a lot of political interest in making cycling more attractive in order to achieve sustainability goals and to make transport safer. However, many state that they avoid cycling due to safety concerns (Manaugh, Boisjoly, & El-Geneidy, 2017; Pearson, Gabbe, Reeder, & Beck, 2023). Having information about which intersection or turning maneuver is particularly risky, can help to avoid these very routes and make cycling safer.

Some indicators have already been developed for evaluating lanes along a segment. These indicators determine the risk of a lane for cyclists (or even pedestrians) and include other aspects such as comfort or bicycle friendliness in their evaluation as well (Lowry, Callister, Gresham, & Moore, 2012; Schmid-Querg, Keler, & Grigoropoulos, 2021; Werner et al., 2023). However, if there is a dangerous intersection between two edges with excellent ratings, then safe routing has effectively failed. Determining the risk of the intersections between, closes the gap. The information about an intersection can therefore be used to evaluate a route holistically and choose the safest way.

In order to solve these questions and meet the requirements it is necessary to refine the intersection's zero-dimensional node to a more precise shape by instead depicting it in one-dimensional and two-dimensional space. The representation of the intersection in its turning relations allows it to be analyzed at a higher level of detail.

Turning relations enable one to depict an intersection precisely

Knowledge of the turning relations in an intersection provides information about the movement patterns of the road users. The risk model developed in this thesis is built on this basis. The one- and two-dimensional representation of the turning relations is derived from the zero-dimensional node of a graph model. The turning relations have a resolution, which is accurate to the lane, and they serve as the basis for the risk model. This way, the risk potential of each possible turning maneuver can be assessed. It is necessary to assess an intersection at this level of detail because this resolution corresponds to the spatial scale at which a cyclist interacts.

The risk of a bicycle-motorized vehicle (BMV) collision is composed of a variety of factors. This model focuses on the factors related to the infrastructure itself, i.e. attributes of the intersection or turning relation, which can be derived primarily from the "geometric conditions". In addition, information such as speed or the presence of cycling infrastructure provide valuable information on the safety of an intersection.

The overall aim to which this work contributes to is the increasement of cyclist safety on the roads. It builds on the assumption that one should know the causes of collisions in order to avoid them. This is also the aim of the European Commission's Road Safety Framework 2021-2030. Knowledge about the dangers of intersections helps cyclists avoid dangerous infrastructure and decision-makers can improve high-risk infrastructure. The aim is to build safe systems in which fatal accidents are prevented (BMK, 2021).

1.2 GENERAL OBJECTIVES

This work was driven by a number of goals:

- discover the possibility of re-engineering an intersection
- investigating the potential offered by the GIP
- building a workflow including data preparation and data modelling
- representing intersections more precisely
- discovering risk factors relevant to intersections
- evaluating intersections from a cyclist's point of view

1.3 RESEARCH QUESTIONS

This work seeks to answer the following research questions and corresponding sub-questions:

REPRESENTATION

How can intersections be represented in a geographic information system (GIS)?

- How can the conversion of a node from 0D to 1D and 2D be automated?
- Which type of representation of an intersection is suitable for which purpose?
- Which details must necessarily be included in the model?

RISK MODELLING

How can the risk of an intersection for cyclists be measured?

- What is the composition of the risk of an intersection and what factors can be used to determine it?
- How can the factors be evaluated and weighted?
- Which factors prove to be important?

RESULTS

How satisfactory is the model?

- How adaptable are the results to the user's priorities?
- Are both global and local factors needed?
- How susceptible is the model to adaptions?
- What sort of information would be needed to better calculate the risk?

2

BACKGROUND AND THEORY

2.1 RISK FACTORS

What are risk factors?

The aim of this work is to assess the risk of intersections. In order to achieve this, an index is calculated from various factors, which draws conclusions about the risk of an intersection. The so-called risk factors that are determined and used in this project are variables that are associated with an increased risk. These factors do not have to be causally linked to the event, i.e. an accident, but can also be correlative. Usually, statistical means are used to determine how strong the association is to the occurrence. In this work, the weighting of the factors is extracted from literature and adapted according to expert knowledge. Risk factors as such are mostly used in the fields of medicine (National Cancer Institute, 2023) and real estate (Risk Factor, 2023).

As mentioned in the Chapter 1, intersections are a dangerous part of the traffic infrastructure for all traffic participants. This work concentrates on the risks encountered by cyclists. Its key focus lies on the event of crashes between cyclists and other traffic participants, mainly motor vehicless (MVs). This is because such an accident has the worst consequences for the cyclist. In addition, these accidents are best documented and recorded by insurances and the police, whereas researchers suspect a large number of accidents involving solely non-motorized vehicles to go unrecorded.

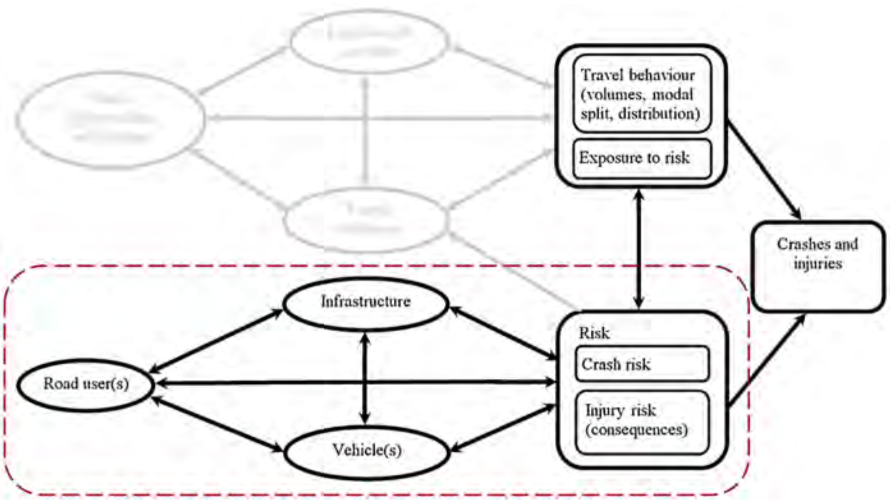

Figure 2.1: Conceptual framework for road safety adapted from Schepers et al. (2014)

framework for road safety

Schepers, Hagenzieker, Methorst, van Wee, & Wegman (2014) developed a conceptual framework (Figure 2.1) for road safety. It holds interacting factors for *exposure (to risk)*, *risk* and *crashes and injuries*. The framework enables researchers to recognize possible outcomes of measures. The arrows hint at how the factors interact, implying that Schepers et al. (2014) assume that crashes are the aftermath of interacting variables. *Exposure* and *risk* are connected because the exposure impacts risk. *risk* lowers when *exposure* raises. The arrow is also pointing in the other direction because a road user can only be subjected to as much risk as is present.

Infrastructure is a pillar of road safety

Accident risk is the result of the combination of three factors, sometimes referred to as the "three pillars of road safety": *Road users*, *vehicles* and *infrastructure*. Pillar one, the *road users*, includes information such as the age, the sex, and the health status of the persons involved in the accident. Pillar two, the *vehicles*, describes the type of the involved vehicles and their condition. The last pillar, *infrastructure*, describes the infrastructure-related attributes of the location where the accident took place. This is the column of *the three pillars of road safety* that this thesis explores and dissects in more detail.

In order to shed sufficient light on the infrastructure factor, it was examined in a literature review. In the process, characteristics of an intersection were identified that have an influence on the risk of an intersection. The characteristics are displayed in Figure 2.2 and will be explained further in the following sub-chapters.

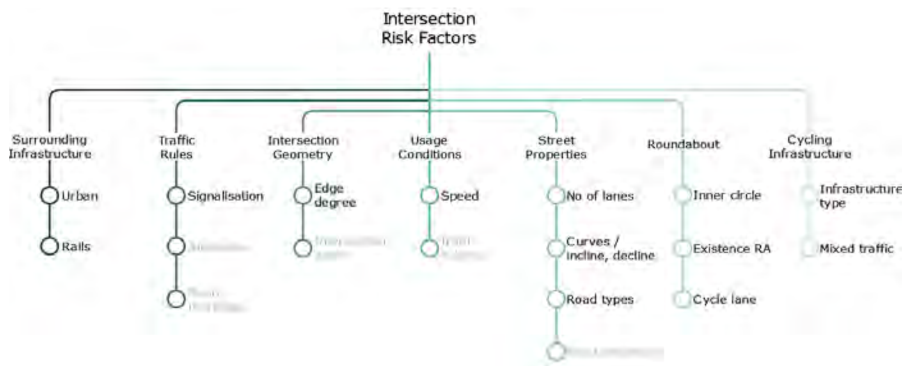

Figure 2.2: Intersections' characteristics contributing to the infrastructure's risk/safety

2.1.1 *Surrounding infrastructure*

Characteristics of an intersection include its location and what surrounds it. Since those factors have an influence on the intersection they are integrated in the risk model as well.

Urban surroundings

In the literature review it was discovered that it matters whether intersections are located in urban or rural areas. Shen, Wang, Zheng, & Yu (2020) found that urban intersections bear a greater risk of fatality. The fatality risk provides information about the severity of an accident - but not about the probability of an accident occurring. So the information is not directly useful for the work's objective, but it does say something about the danger of an intersection.

It matters in which surroundings the intersection is located

This statement is unfortunately not supported by Strauss, Miranda-Moreno, & Morency (2015), who found in their work that there are more accidents in central neighborhoods. However, the volume of cycling is also comparatively higher there. As a result, the overall risk of being involved in an accident is lower. But these statements should be treated with caution because it is questionable what is defined as a "central" neighborhood. Strauss et al. (2015) examine the entire island of Montreal. Accordingly, central neighborhoods are presumably the city core.

It is certain there are more accidents involving cyclists in urban areas than in rural areas. Yet, no clear statement can be made about the relative ratios, i.e. how many accidents happen per cyclist, according to this literature review. Though based on the research of Shen et al. (2020) and an educated guess, the developed model assumes that it positively impacts the risk index if the intersection is located in an urban area.

Rail infrastructure

Closeness to rail infrastructure

This research only looks at road traffic, but some of the road space interacts with railways, which consequently have an impact on the road. Harris et al. (2013) found that there is an increased risk of accidents within 5 meter (m) (diameter) of a streetcar or train track. However, they were only able to make this finding at non-intersection locations. However, Vandenbulcke, Thomas, & Int Panis (2014) confirm in their work the general assumption that trams pose an increased risk. They found a higher risk of accidents at tram crossings and for on-road tram tracks.

2.1.2 *Traffic rules*

Signalization

A number of papers examine the influence of traffic lights at intersections and whether or not they are beneficial to cyclists. The literature review's results are summarized below:

Research on the risk of bicycle crashes at traffic signal intersections is mixed. Studies have found different results (Meuleners et al., 2019). Nevertheless, more scientific research concludes that traffic signal intersections increase the risk of a BMV collision.

Some works conclude that signalization increases risk for cyclists..

Prati et al. (2018) and Chen (2015) have found increased risk of accidents in areas where there are many traffic lights in a confined space. Strauss, Miranda-Moreno, & Morency (2014) discovered in their work the risk for cyclists at signalized intersections is five times higher in comparison to non-signalized intersections. Meuleners et al. (2019) found a similar figure as they determined risk has significantly increased by four for intersections with traffic lights compared to priority control and uncontrolled intersections. Other works also promote the assumption that signalized intersections are 2.5 times more dangerous (Strauss et al., 2015) or overall connected with higher risk (Vandenbulcke et al., 2014).

.. whereas others found it to lower the risk.

Contrarily, Harris et al. (2013) report to have found lesser crash risk at signalized intersections than at priority control intersections. Additionally, bicycle flows are much higher at intersections with traffic signals as they are at non-signalized intersections (Harris et al., 2015). This means the absolute figure of accidents at signalized intersections naturally is higher than at their counterparts. This problem is discussed more closely in Section 6.3.2.
Furthermore, Eluru, Bhat, & Hensher (2008) did research on fatality risks for cyclists. They discovered that less severe crashes take place at signalized intersections.

Manirul Islam, Washington, Kim, & Haque (2022) underline the importance of the signal strategy when discussing risk at intersections. Depending on whether it is a protected or a permitted turning maneuver, the risk for cyclists varies. Because there is no open data covering the signal strategy of Austrian traffic lights, it is not possible at the moment to make investigations in this regard using the model.

Sign posts

The potential of sign posts

There are few publications on the influence of road signs on accident risk. On a macro-scale level, Chen (2015) concluded that the density of road signals and street parking signs correlates positively with the number of bicycle crashes. In their studies in Japan, Sekiguchi, Tanishita, & Sunaga (2022) took a closer look and found that there were more fatal accidents when there were no stop signs at an intersection. This finding is difficult to transfer because of the lack of direct comparability between Austria and Japan. Furthermore, there is a lack of data on the Austrian side to include this factor in the model at all.

In spite of the data desideratum, road signage is mentioned here because it seems plausible that it has the potential to make drivers of MV aware of VRU and thus reduce the likelihood of an accident.

Road markings

As in the section before, for road markings only a japanese study by Sekiguchi et al. (2022) was found. It mainly points out the importance of the central reservation: in case there is no median strip, or a painted line of if there is a rumble strip at an intersection, this increases the risk of a fatal accident. On the other hand, very penetrating colors (high-brightness paint) in the median reduce the risk of a fatal accident.

The problems regarding the road markings are the same as discussed in Section 2.1.2. The transfer from Japan to Austria seems problematic, as rumble strips or median strips in very penetrating colors are rare to non-existent. In addition, there is currently no open government data (OGD) on the subject.

2.1.3 *Intersection geometry*

Edge degree

Edge degree and complexity

In their research, Manirul Islam et al. (2022) found that, in general, more accidents seem to occur at four-legged intersections than at their three-legged counterparts.

This is also supported by Vandenbulcke et al. (2014), who state that complex intersections increase the chance of a bicycle accident. Although the complexity of an intersection is not the same as the edge degree of an intersection, it can still be argued that a complex intersection is likely to bring together a large number of intersections. And also that an intersection becomes more complex the more edges converge in it.

Intersection angle

The impact of intersection angles on the risk of an intersection could only be poorly determined in the very extensive literature search. Using a regression model, Dong, Clarke, Yan, Khattak, & Huang (2014) found that intersections with crossing angles of 90° and more contribute negatively to the accident risk. This figure refers to the whole intersection in their model.

Yet, the paper does not describe where or how the angle was measured. It is therefore not clear between which edges were used to gauge the angle. Therefore, this factor is not included in the model. Nevertheless, it is realistic that the intersection angle has an influence on the accident risk. This is because, on the one hand, it may imply that a driver cannot see all edges due to a limited field of vision. On the other hand, some papers consider that a poorly visible and narrow intersection is a solution to make road users more attentive.

However, there are other influencing variables related to the field of vision that belong to the environmental variables: It is of interest whether there are buildings and trees in the vicinity, as they can influence the field of vision of road users, depending on their location and size.

2.1.4 *Behavior*

Speed

Speed is dangerous

One variable that has been and continues to be studied is speed. There is a clear scientific consensus that driving at high speeds is dangerous - for all parties involved. Both Merlin, Guerra, & Dumbaugh (2020) and Prati et al. (2018) discuss accidents involving MV and cyclists in their papers, and examine the influence of the posted maximum speed. They assert that in zones with higher speed limits there are proportionally more cycling accidents, whereas in zones with lower speed limits they observe a decrease in accidents.

In their studies, several researchers have also come up with concrete figures that they consider to be thresholds for the risk to cyclists:

- Hagel et al. (2014) asserted the risk of collision to be higher in zones with a speed limit *>= 30 km/h*
- Ackery, McLellan, & Redelmeier (2012) found an increased risk for cyclists at 35 mph (= *56 km/h*).

- In the research of Boufous, Rome, Senserrick, & Ivers (2012), about half the cycling accidents occurred on roads with a speed limit of *60 km/h*.
- This figure was confirmed by Meuleners et al. (2019), who found that intersections with speed limits >= 60 km/h were twice as risky for cyclists as intersections with speed limits < 60 km/h.
- Hoque (1990) identified *75 km/h* as a significant threshold above which the risk of MV collisions increases, especially at night.

Speed impacts both the number of crashes and their severity.

In summary, the variable *speed* has been studied on many occasions. The distinct conclusion is that speed is dangerous for cyclists (and everybody else). But not only does the risk of a crash increase constantly with speed, the severity of the injuries suffered by a cyclist also increases proportionally with the speed limit (Boufous et al., 2012).

Traffic volume

The number of road users at an intersection affects the risk those road users are exposed to. Yet, it is not only the number of cars that plays a role, but also the number of cyclists. Whether this influence is positive or negative is not yet clear. According to Harris et al. (2013), a volume of >75 cyclists/h leads to a higher risk of collision at intersections. So they associate more cyclists with higher risk. Leden, Gårder, & Pulkkinen (2000) do not support this assessment - they were told in their expert interviews that the safety of the individual cyclist increases as the total volume of cyclists at an intersection increases.

Safety in numbers for cyclists

This is also supported, without reference to any particular site, by the "safety in numbers" principle that BMK (2021) includes in its Austrian Road Safety Policy. It refers to the observation that in countries where there are more pedestrians and cyclists, comparatively fewer accidents involving these same road users do happen. The individual's risk is therefore reduced.

Positive correlation between accidents and volume of MVs

But of course the volume of MV also has an impact on the accident risk. It has been found that for every 10% increase in motorized traffic, the number of accidents involving cyclists increases by 4.4% (Miranda-Moreno, Strauss, & Morency, 2011). The connection between increased traffic and more accidents involving cyclists has also been reported by other researchers (Harris et al., 2013; Turner, Binder, & Roozenburg, 2009).

However, increased traffic tends to be associated with slower speeds. This in turn has an impact on the severity of accidents - which are less severe (Klop & Khattak, 1999).

2.1.5 *Street properties*

Number of lanes

It is generally accepted that more lanes have a negative impact on risk. However, the studies refer to road segments, not intersections (Abdel-Aty, Chundi, & Lee, 2007; Kaplan, Vavatsoulas, & Prato, 2014). The consensus of these studies is that more lanes are associated with higher risk. Yet, the maximum number of lanes in their studies is three. The reason this model still uses the number of lanes is that more lanes also increase the complexity of an intersection. And as explained in the *intersection grade* section, the complexity of an intersection affects its risk for cyclists.

Kaplan et al. (2014) also point out that more lanes are also associated with higher traffic volumes and speeds. As discussed earlier in this chapter, these are variables that also affect the risk of an intersection.

Incline and decline

The way in which road users approach an intersection also plays a role in the hazards of the intersection. Among other things, this determines how they assess the intersection, whether they can see all other road users, their visibility of the intersection, etc.

Grades on roads are detrimental to bike safety

There is a general consensus that steep gradients and bends contribute to the risk of accidents. Manirul Islam et al. (2022) found that the frequency of accidents increases when approaching a horizontal curve, slope or gradient of more than 5% at an intersection. Harris et al. (2013) have found an increased accident risk for slopes which are greater than 1° (about 1.7%) . Meuleners et al. (2019) do not give exact figures for gradients, but estimate the risk to be more than three times higher for intersections (and also straight lines) with gradients than for those without.

Major roads are more hazardous than minor roads

Road types

It is also of interest which road types are leading towards an intersection. In this respect, the intersection of minor roads is associated with lower risk than the intersection of major roads (Harris et al., 2013). Other authors have also concluded that major roads and multi-lane roads are associated with higher risk and more serious injuries (Merlin et al., 2020), and that smaller roads are less dangerous than major roads (Winters et al., 2012).

This variable, which is so consistently rated by experts as risk increasing, is related to other variables as well. In many cases, major roads are associated with higher *speeds*. Moreover, there is a higher *volume of traffic*. These two variables are already listed separately. However, they are not mutually dependent and one of them can occur without the other. Therefore, it is justified to list the variables separately.

Pavement condition

A damaged road in poor condition increases the risk of an accident. This may be because road users, especially cyclists, are distracted by potholes or bumps, have to avoid them, or fall simply because of the poor surface.

The road surface can be characterized in a number of ways, but Dong et al. (2014) identify the roughness of a road as the most statistically significant factor in terms of risk. They found that a rougher road surface increases the frequency of accidents. This is because a rougher surface is more likely to lead to loss of control when braking and turning, as it reduces the contact area between the tire and the road surface. This is supported by further research that cites sand and gravel as factors that increase risk (Prati et al., 2018). Other sources also mention the road surface and its condition as a factor to consider, also in terms of *bikeability* and *walkability* (Lee, Abdel-Aty, & Shah, 2019; Said, Abou-Zeid, & Kaysi, 2017).

Data desideratum concerning pavement condition

Due to the data desideratum, it is currently not possible to include this factor in the risk model. The GIP has the ability to store information about the type of road surface, but 81.2% of the edges in the selected area of interest (AOI) do not hold this information. Because the road surface and its condition seem to be important in such a model, it is explained here anyway, although it cannot be integrated at the moment.

2.1.6 *Roundabout*

Existence of roundabouts

Roundabouts are a double-edged sword in road infrastructure. On the one hand, they are safer for most traffic participants because they reduce the speed of road users. On the other hand, a roundabout has significantly more points of conflict than an intersection does. At these conflict points, road users come into contact with each other in a similar way to an intersection.

Roundabouts are among the most perilous elements in traffic for cyclists

At roundabouts, cyclists are particularly vulnerable to the following scenario: a cyclist and a car are both at approximately the same point in the roundabout. The car driver wants to turn and the cyclist, who is further out on the roundabout, wants to take the next exit first. If the driver of the MV does not see the cyclist as they turn, a collision can occur. As this scenario can occur considerably more often in a roundabout than in a conventional junction, this tends to make a roundabout more dangerous for cyclists, as many researchers could conclude (Jensen, 2013; Meuleners et al., 2019; Sakshaug, Laureshyn, Svensson, & Hydén, 2010; Vandenbulcke et al., 2014).

Inner circle

The size of the inner circle of a roundabout has an influence on the safety of a roundabout, as it affects both the angle of vision of the users and the speed that can be driven on the roundabout. The literature review showed that an inner circle which is too large, is disadvantageous for cyclists. In their research, Hels and Orozova-Bekkevold (2007) found that more accidents involving cyclists per year were associated with roundabouts with large curves. On the other hand, large roundabouts (30 m diameter) were discovered to reduce the risk of accidents for drivers of MV (Harris et al., 2013).

Other variables that may have a negative impact on cyclists at roundabouts are a high volume of cyclist traffic and MV, a narrow apron (Hels & Orozova-Bekkevold, 2007) and a raised platform. They are not discussed further due to a lack of data and because their importance in the model is considered low.

Cycling infrastructure in a roundabout

Cycling Infrastructure

Most authors do not make general statements about cycling infrastructure at roundabouts. Instead, they look at the type of infrastructure and differentiate between them: if the cycle lane is only 'drawn' by road markings, this increases the risk of accidents for cyclists (Hels & Orozova-Bekkevold, 2007). However, if the cycle lane is separated from the traffic lane, as in a cycle path, low risks for cyclists have been observed (Daniels, Brijs, Nuyts, & Wets, 2009; Harris et al., 2013).

The *Design manual for bicycle traffic* also explicitly advises against cycle lanes, as cyclists are more likely to be overlooked, especially by HGV drivers. If cyclists ride on the outside of a roundabout, they can end up in the lorry's blind spot - which the cycle lanes force them to do. The authors recommend segregated cycle lanes instead (CROW, 2016).

What argues against favoring segregated cycle paths is the concern that spatial separation could also have a negative effect (Sekiguchi et al., 2022): If bikes and MV are kept separate at the edges, drivers of MV may forget to pay attention to the presence of cyclists. If they are reunited at intersections, this may the accident risk may be higher.

2.1.7 *Cycling infrastructure*

Cycling Infrastructure type and existence

The influence of cycling infrastructure at intersections is controversial

In general, infrastructure dedicated to cyclists does reduce accidents (Robartes & Chen, 2017). Reynolds, Harris, Teschke, Cripton, and Winters (2009) found that bicycle-specific facilities like bike lanes, bike routes, and off-road bike paths lower cyclist accidents and injury.

In contrast, other works differentiate between the types of infrastructure. Vandenbulcke et al. (2014) found that it influences the cyclist's risk whether the cycle lane is marked or not. Merlin et al. (2020) concluded in their research that separated cycling facilities indeed do reduce both the number and the fatality of bicyclist crashes. But they also assert that on-street bike lanes do not have a risk-lowering effect. Prati et al. (2018) raise the concern that physically separated bicycle paths might have the opposite effect at intersections. This is because the drives of MVs might forget about the presence of VRUs and get confronted with them at intersections. Furthermore, Chen (2015) and Merlin et al. (2020) emphasize that bicycle routes along off-arterial routes bear less risk and fewer crashes than cycling routes on arterials, which are associated with increased collisions.

Mixed traffic

Mixed traffic means that lanes are shared by different types of road users. This can refer to cyclists and MV as well as cyclists in combination with pedestrians. Teschke et al. (2012) and Harris et al. (2013) found in their studies that mixed traffic with both MV (shared lanes, multi-use lanes) and pedestrians (multi-use paths, sidewalks) increases the risk of a collision. This thesis is supported by Sekiguchi et al. (2022) who finds mixed traffic with pedestrians on footpaths and multi-use paths to be the most dangerous.

Part II

IMPLEMENTATION

3 DATA PREPARATION

This chapter describes the pre-processing of the data. The data preparation is done automatically using structured query language (SQL) scripts. Based on these results the Chapter 4 is carried out. The quality of the data preparation is analyzed in the Chapter 6.

3.1 BACKGROUND AND (DATA) BASIS

3.1.1 *Background*

The motivation of this data preparation is that a node, as used in the graph model, is inadequate for representing an intersection in detail. Depending on the intended use of a model, a zero-dimensional node may be the ideal choice. For example, in wayfinding, where great emphasis is placed on the efficiency and topology of the (road) network. But what if it is necessary to look at an intersection in more detail? A zero-dimensional node is uninformative regarding the structure of an intersection. For this reason, researchers are currently developing high-definition maps (HD maps), which are more granular and produced at great expense. These highly detailed maps are important in the field of automated driving because they contain information about regions that sensors cannot capture.

Two-dimensional representation for determining risk zones

The data preparation performed in this chapter is a step towards HD maps, but cannot compete with its accuracy. Nevertheless, the resulting two-dimensional areas representing the lanes allow to perform a risk assessment of an intersection and to identify risk zones of the intersection. The two-dimensional representation also shows where conflict points are located and where an intersection is rather calm.

One-dimensional representation

In the one-dimensional view the data preparation produces, both the lanes and the turning relations are displayed by means of their centerline. This is a convenient way to communicate information about the infrastructure elements. The lines provide the ability to convey an intersection in a detailed yet clear manner. They are used in this work to represent both the type of lane, its risk, and the direction of digitization and travel.

In this work, both one-dimensional and two-dimensional representations were derived from the data base (see Figure 3.1). The underlying source is the GIP.

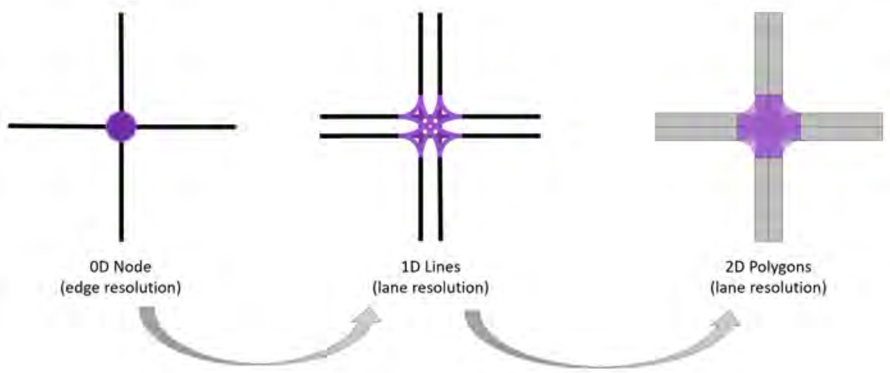

Figure 3.1: Conversion from a zero-dimensional node to a two-dimensional representation of the turnuses

3.1.2 *Data Basis*

The data basis for the intersection representation is the Graph Integration Platform (GIP) of Austria. It is the cooperative transportation network of Austria. The GIP contains all modes of transport (public transport, passenger car traffic, cycling, walking, ...). Its data is maintained decentralized by the individual Austrian federal states and by ASFINAG and OEBB-Infrastruktur AG. The data is recorded and updated separately by each operator. The databases are synchronized in a cycle of about two months (ÖVDAT, 2021). The GIP data of ÖVDAT are published as open government data (Open Data Österreich, 2022). The original data in its current version can be found at the link https://www.data.gv.at/katalog/de/dataset/intermodales-verkehrsreferenzsystem-osterreich-gip-at-beta .

The GIP provides a spatial data set

The GIP is structured on the base of a graph model. Yet, it provides not only nodes and edges, but also usage strips (cross-sectional elements) and turning relations. It is divided into partial datasets. In this work, the partial datasets B, GIP Network, which contains the basic network including nodes, links, usage strips and turning relations, and D, which holds the lookup tables, are used. Subset B is a geospatial dataset (CRS: 4326) provided as a geopackage. Subset D, on the other hand, contains lookup tables that link IDs to the actual nomenclature. They are provided in CSV format.

It provides a graph model, a cross section layer and usage conditions

The base network (level 1) of the GIP are the edges and the nodes. The *edges* correspond to road segments. They are equivalent to the center line of the road. At the beginning and at the end of an edge a *node* is created. Usually three or more edges meet at a node - nodes normally correspond to intersections. However, there are also virtual nodes and link nodes, which are created when a lower-level subnet joins a higher-level subnet, or the node is kept for consistency reasons. An edge is not broken at these node types, instead there are *links* which are very similar in geometry to edges, but terminate at virtual nodes and link nodes.

In this model, both links and edges were used as information sources because they contain different information, but both were needed for the model.

The cross-section layer (level 2) is built on top of this base network. It is the structural layer that represents the cross-section of a segment: a segment has an extent perpendicular to the direction of travel for each cross-section element, the *usage strips* (*GIP jargon: linearuse*) . These can be, for example, sidewalk, roadway, or bicycle lane. Each usage strip has an offset from the edge and a width. They are attributively related to the edge. The *turning relations* (*GIP jargon: turnuse*) are the elements that connect the usage strips across the intersection.

The third layer (level 3) is that of usage conditions. There are legal conditions of use, such as the road traffic act, which prescribes driving permits and maximum speeds. And there are factual usage conditions that describe environmental conditions, such as the road surface or average speeds. This level is integrated in the attributes of the first two levels.

The layers build a cooperative transportation network

The nodes, links, edges, usage strips, and turning relations rely on each other and build together a cooperative transportation network. How the data sets are related to each other is displayed in the entity relationship diagram (ERD) in Figure 3.2.

3.1.3 *Software*

Only open source software was used

This project worked with a PostgreSQL database. It held the data and was also used to execute the algorithms and store intermediate and final results. The version used is PostgreSQL 12.2, compiled by Visual C++ build 1914, 64-bit.

To work with and analyze spatial data, the spatial extension PostGIS was added to PostgreSQL. The version used is Version 3.0 USE_GEOS=1 USE_PROJ=1 USE_STATS=1

The execution of the scripts and the management of the data was mainly performed using the graphical user interface (GUI) pgAdmin. This allows easier graphical access to the database, and it is possible to create ERDs in an automated way.

QGIS displayed the data from the database. This was very useful during the development, as it allowed to visualize the data and made debugging easier. The version used is QGIS 3.22 Białowieża.

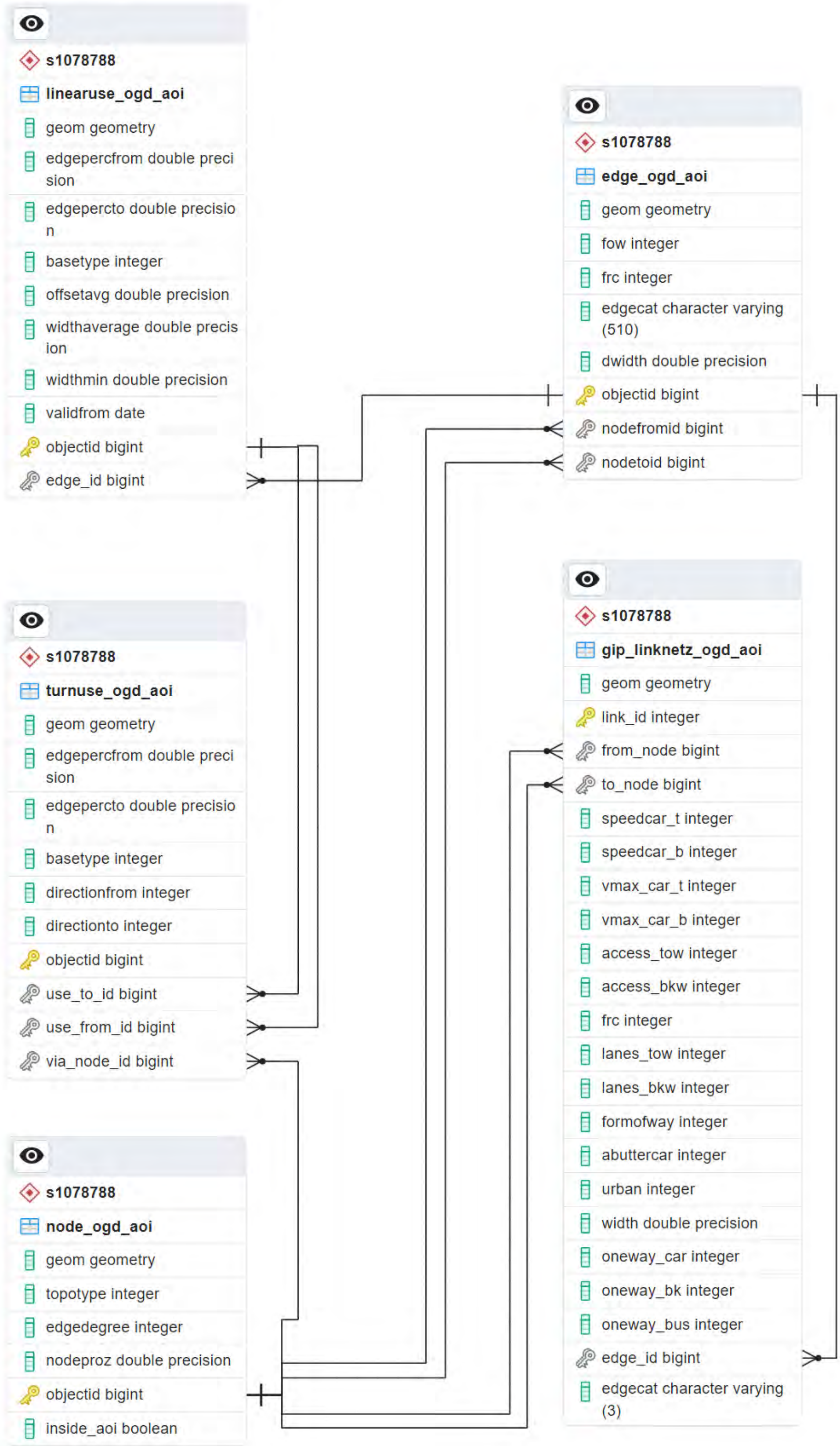

Figure 3.2: Entity-Relationship-Diagram of the GIP's spatial subset B - shortened version

3.1.4 *Area of Interest*

For the project, an area in the southwest of Vienna which includes the suburban districts Mauer, Atzgersdorf, Speising and Hetzendorf was chosen Figure 3.3. This area was elected for the following reasons: It is assumed that the data in Vienna, as Austria's capital, is recorded particularly well and attentively. Furthermore, Vienna offers a large variety of other OGD, which can be additionally used for the model. The southwest of Vienna also has a diverse infrastructure - there are densely built-up areas as well as paths on the outskirts of the city. This makes it possible to test the model under different circumstances.

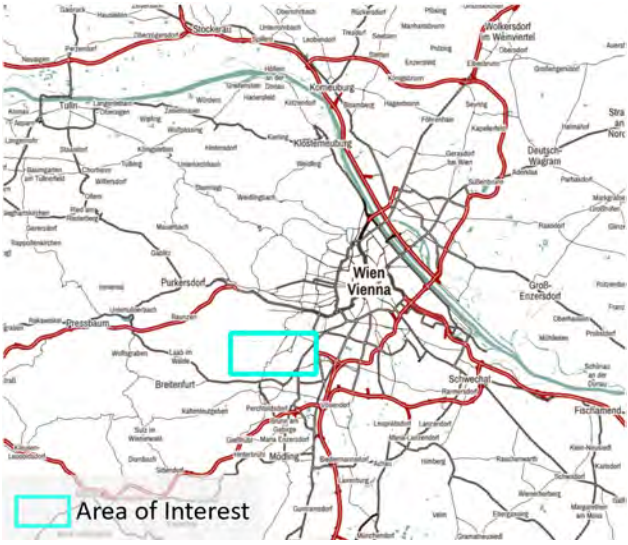

Figure 3.3: The chosen area of interest in the southwest of Vienna, Austria

3.2 METHODS

3.2.1 *Import and Storage*

The import of the basis network was done using QGIS. A connection to the PostgreSQL (PostGIS) database was established in QGIS and the data was transferred using the database tool. The foreign key relationships and indices are created with pgAdmin. The lookup tables are in Comma Separated Values (CSV) format. To import these data sets into PostgreSQL, first the tables were created with the according structures of the lookup tables. Then, pgAdmin was used to import the data. Finally, indices were added to these tables as well and their IDs were connected to the network tables they support.

3.2.2 *Structure*

In this subsection, it is briefly explained how the scripts that perform the conversion from a zero-dimensional node to a 2D representation, as seen in Figure 3.1, build upon each other.

The second level of the GIP builds the base of the model. It holds the cross-section layer, which includes the use strips and the turning relations. They contain the geometries on which the model is built. This data is used because it is closest to the desired result.

three major steps are executed to convert from 0D to 2D

The data preparation consists of three principal steps, which are mainly executed in three functions. Each of these functions is called with the same parameter, which is the *objectid* of the node that is being examined.

First, the segments are prepared by working out the individual lanes, aligning them in the direction of travel, and forming a two-dimensional representation. On this basis, the intersection plateau is created, which represents the turning relations - in 1D and 2D form. Finally, the parking lanes are created, and if they overlap with the lanes, the lanes are trimmed. The steps are illustrated in Figure 3.4. The activity diagram illustrating the lane construction and the plateau building can be found in the Appendix in Figure A.1 and Figure A.2.

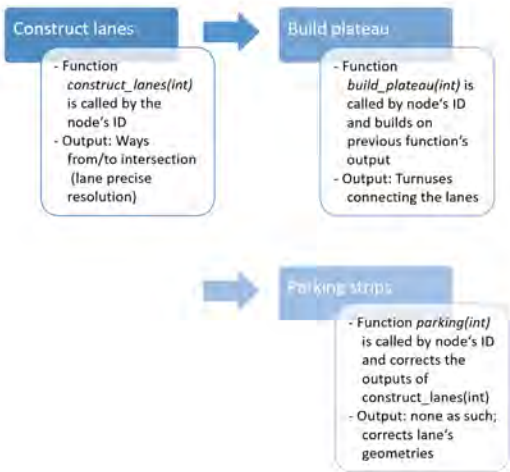

Figure 3.4: The data preparation process: first lane construction, then creating the turnuses and addition of the parking strips

3.2.3 *Construction of lanes*

In this preparation step, . . .

(1) the resolution of the road display is improved from usage strip to lane accuracy.

(2) the direction of the one-dimensional graphs is corrected to fit the lane's direction of travel.

(3) the one-dimensional graphs are converted to two dimensional lanes and turning relations.

Although this work focuses on intersections, the preparation step of constructing the lanes needs to be executed. This is because the GIP's turnuses start and end at the linearuses they connect. But those are not existent in the level of precision that is needed for this work's purpose. They are on hand in the resolution of usage strips, required is lane-level resolution, though. Therefore, the lanes first have to be constructed from the usage strips to then be used as start and end of the turning relations, which are the elements, the work is focused on.

To identify the usage strips that are going to be converted to lanes, the function (Listing A.3) is given the node's ID as parameter. The first step is the determination of the number of usage strips *(linearuses)* in the original data, leading to/from the node. With the help of a loop, each usage strip is individually guided through the following procedure. From the attributes, basic information such as the average width of the usage strip, its ID and the underlying edge's ID, are collected. Furthermore, it is determined to which type the usage strip belongs. This is a very central information because pathways and bike lanes usually only have one lane. Whereas a usage strip representing a car lane (most likely) stands for more than one lane.

Building two-dimensional bike lanes and pathways

Meaning that if the usage strip's type is not *'car'*, the geometry of the strip is taken as it is. To make it a 2D lane, it is buffered with half of its width, which is determined before. Lastly, these results are inserted into the table for lanes, *linearuse_lanes*.

Building the lanes of motorized vehicles

This contrasts with the processing of data when it comes to a usage strip for cars. Additional information about the number of lanes and the driving direction of the single lanes is needed. They are obtained by extracting them from the dataset that holds the edges (*gip_linknetz_ogd*). It contains the information which nodes the edge connects and how many lanes there are in and against the direction of digitization. The lane width can therefore be calculated from the quotient of the usage strip width and the total number of lanes. In for-loops, this information is used to calculate the offset of the lane center line to the original geometry (usage strip). Depending on whether the lane is aligned in or against the digitization direction, it is offset to the left or right. Additionally, it is determined from which node the line comes and to which node it leads - this gives the advantage that the direction of the lane no longer must be derived from the geometry. Finally, the geometries of the lanes' center lines are generated by including the offset and aligning the geometry in the direction of travel. A buffer is formed around this geometry, which represents the actual lane in 2D. The products of this process are inserted into the table *(linearuse_lanes)*.

3.2.4 *Building of turnuses*

The function *build_plateau(integer)* is called after having executed the lane constructing function because it builds upon those outputs as explained above. What it does is to...

(1) create turnuses between lanes where turning relations are.

(2) turning the turnuses in the right direction

(3) converting the turnuses to a two-dimensional representation

This function constructs the turning relations that are the base on which the risk function is built. In order to accomplish this task, the function (Listing A.6) is called with the intersection's node ID as a parameter. The first step here is to determine the number of turning relations (called turnuses in the original data) that are of the bike lane or pathway type. Only these are counted, as the processing of the two road types(cars and bike/pedestrian) differs greatly. The reason for this is that the car lanes are almost completely newly created in the previous step and are not built on the original geometry, as it is the case for bicycle lanes and footpaths.

Building the turnuses for car lanes

Therefore, the turning relations for the car lanes also need to be newly built. For this purpose, an auxiliary table is created in which all possible combinations of (car) lanes are generated. Duplicates are avoided, as is a combination of lanes derived from the same usage strip, i.e. parallel running lanes. In the previous step, attributes were added to the lane objects that describe from which node to which node the lane leads. With the help of these attributes, it is now determined whether two lanes have the same direction of travel (or would collide with each other). If this is the case, the start and end points of the turning relation are determined using the two lanes between which it is located. Since this turning relation is generated and not directly based on GIP data, it is checked whether such a connection also exists in the original data. For this purpose, the IDs of the start and end nodes of the original and generated turning relations are compared.

Next, the geometry of the turning relation gets bent into a curve. A dedicated methodology has been developed for this purpose. First the apex of the curve is found using the function *get_intermediate_point(geometry,* *integer)*. It builds a circle around the start and the endpoint of the turning relation. The radius used to draw the circle is the length of the turning relation *0.505. This number was chosen because it produces two points of intersection that form a slight curve when one of them is tied in. As you can see in Figure Figure 3.5, the point that is closer to the node of the intersection is chosen.

The two-dimensional lane is formed by creating a circular buffer around each of the start and end points, the radius of which is adapted to the width of the adjacent lane. The buffer radius of the apex point corresponds to the mean value of the start and end. A convex hull is then placed around these buffers so that a planar lane with a curve is created. In addition to the two-dimensional lane, a curved line is created, as shown in Figure Figure 3.5. This is achieved with the function in Listing A.9. It is given a line string consisting of start, apex, and end points. In the function, curved elements and further points are inserted between these points using PostGIS functions. The outcomes are stored in the table *iplateau*.

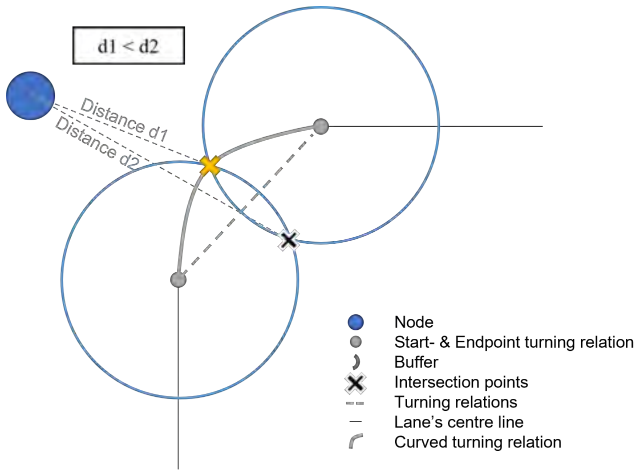

Figure 3.5: Graphic display of the function get_intermediate_point(geometry, integer)

Building the turnuses for bike lanes and pathways

The two-dimensional representation of the turning relations for cyclists and pedestrians, on the other hand, is based more on the existing geometries. With the help of a for-loop, each of them is processed individually. Here again, a distinction must be made between two types of turning relations in order to model the course of the road as correctly as possible: in case a footpath merges with another footpath or bike path, the original turning relation can be used. If, however, a bike path joins a usage strip of the type *car*, this poses a problem in that the original turning relation leads to the original usage strip. Yet, it is highly probable that there are two lanes at this location after the first script has been run. Meaning, the geometries no longer meet.

If the turning relation connects two cycle paths or footpaths, the function *help_connect(...)* (Listing A.7) is called with information regarding the geometry and width of the two paths, and the start and end points of the relation. The function assigns the widths of the lanes to the beginning and end of the turning relation so that they connect to the usage strips as faultlessly as possible. As with the car lanes, the apex point is then determined, the buffers are created around the points, and a convex hull is built around it. The outcomes are inserted into the result table.

A bicycle lane leading into a car lane requires special handling

If a bicycle lane merges into a car lane, it is first determined into how many lanes the usage strip of the type *car* has been converted. According to the number of lanes, a for loop is initiated, which has the purpose of duplicating the turning relation so many times as there are connections from the cycle track to each car lane. Furthermore, the point of the turning relation that joins the road is moved, so that it is located on the starting point of the lane centerline. If the loop is repeated because of a second car lane, the point is moved to the start point of the center line of that lane. Great emphasis is put on the right digitalization direction of the turnuses: Since they connect to a car lane, it is obvious that the bicyclists also need to drive in the car lane's direction. For this reason, the turnuses were oriented according to the direction of travel of the car lane. Placing such a value on the direction of travel has no effect on the two-dimensional model, which is consists of polygons. But it makes the one-dimensional model closer to reality and capable of routing.

3.2.5 *Parking strips*

In the preparation step of the parking strips...

(1) the parking strips are created and stored.

(2) the lanes are corrected by clipping.

Parking lanes are used to correct car lane's geometry

The function *parking(integer)* processes the parking lanes along a road and corrects the width of its adjacent lanes. This is done firstly because parking lanes have been highlighted as a significant danger to cyclists in several papers regarding the risk of segments. Therefore it cannot be ruled out that they bear risks for intersections as well. Second, the GIP includes the width of the parking lane in the average width of the traffic lane. This creates a false visual impression and can lead to errors in the risk modeling.

In order to transform the parking lanes to a two-dimensional representation, the function (Listing A.5) first loops through all parking strips belonging to an edge that is connected to the node being analyzed. Parking strips are stored as linearuses and are characterized by the attribute *basetype*. A buffer is created around such a parking strip, which is equal to half of the average width. The created two-dimensional geometry is stored in the table *parking_strips*. The loop ends here, and if there are more parking strips at the intersection, they will go through the same process.

The second part of the function is the clipping which pursues the goal of correcting driving lanes that are too broad. Therefore lanes, which were created beforehand, are clipped by the parts that overlap the parking lane. This is done by first looping through all previously constructed lanes in *linearuse_lanes*. Each parking lane is tested for overlaps with the lane. If this is the case, the geometry of the lane is trimmed and corrected.

Actually, the possibility of overlapping is already handled in the *construct_lanes* function: Depending on the existence of a parking lane along the edge, the average width or the minimum width is used to calculate the buffer. However, since these widths refer to the entire edge, there may be inaccuracies and in this case the clipping part of the *parking(integer)* function takes over.

3.2.6 *Preliminary Result*

In the data preparation step, the data of the GIP was prepared to fit the requirements of the risk model. The result are one-dimensional and two-dimensional representations of an intersection, which are derived automatically. The data processing is done by means of specially written functions in the procedural language of the database system PostgreSQL (PL/pgSQL).

The preliminary results are the lanes, the intersection plateau with its turning relations, and the parking lanes. They are based on the spatial data extract of GIP and are further linked to it, as can be seen in Figure 3.6.

The geometries produced by the data preparation and on which further work is based are shown in Figure 3.7 in a one-dimensional representation and in Figure 3.8 in a two-dimensional representation.

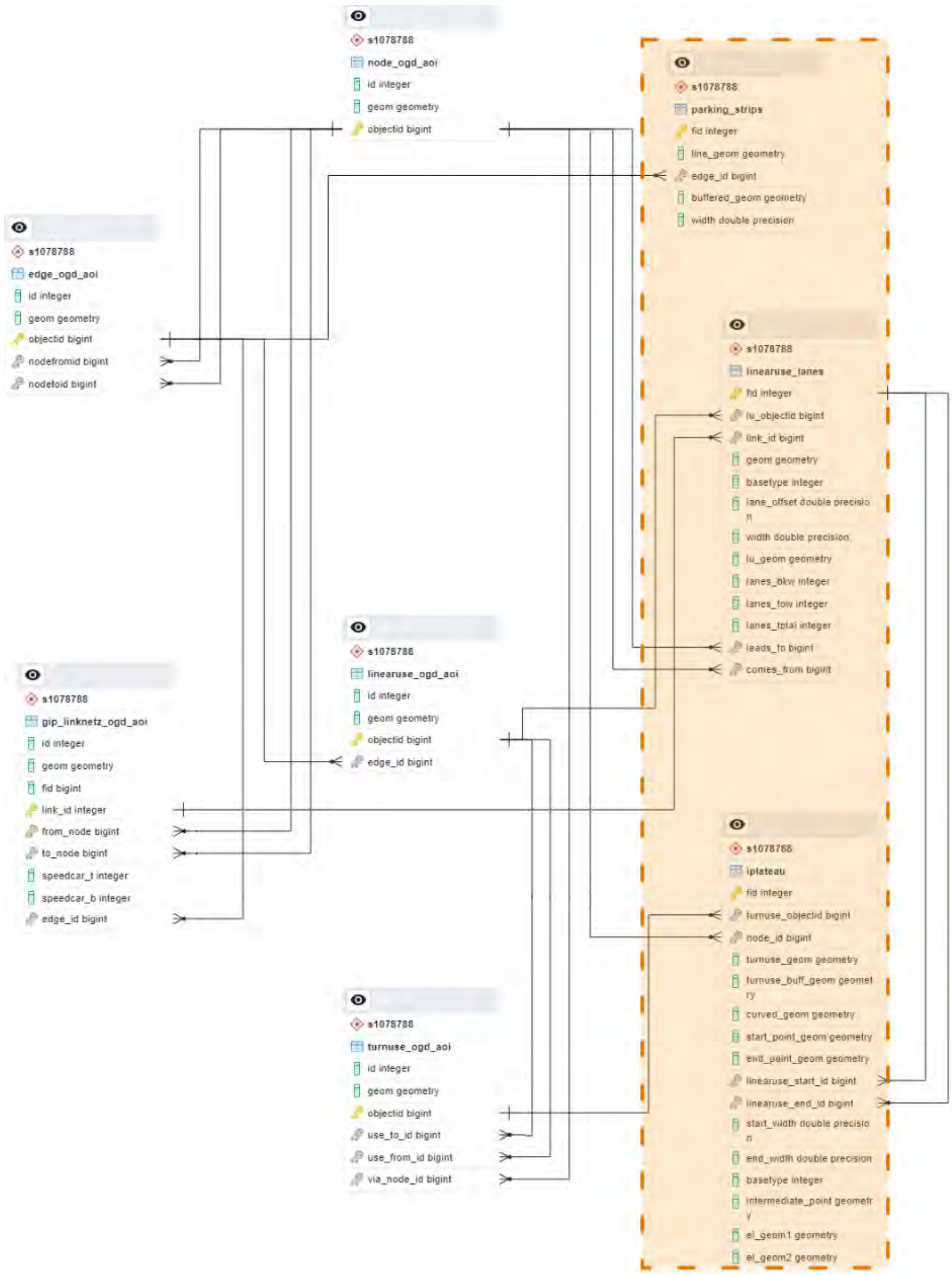

Figure 3.6: Entity-Relationship-Diagram displaying the outputs of the data preparation. The resulting tables are highlighted in orange. The GIP tables were compressed to be able to display the diagram on one page.

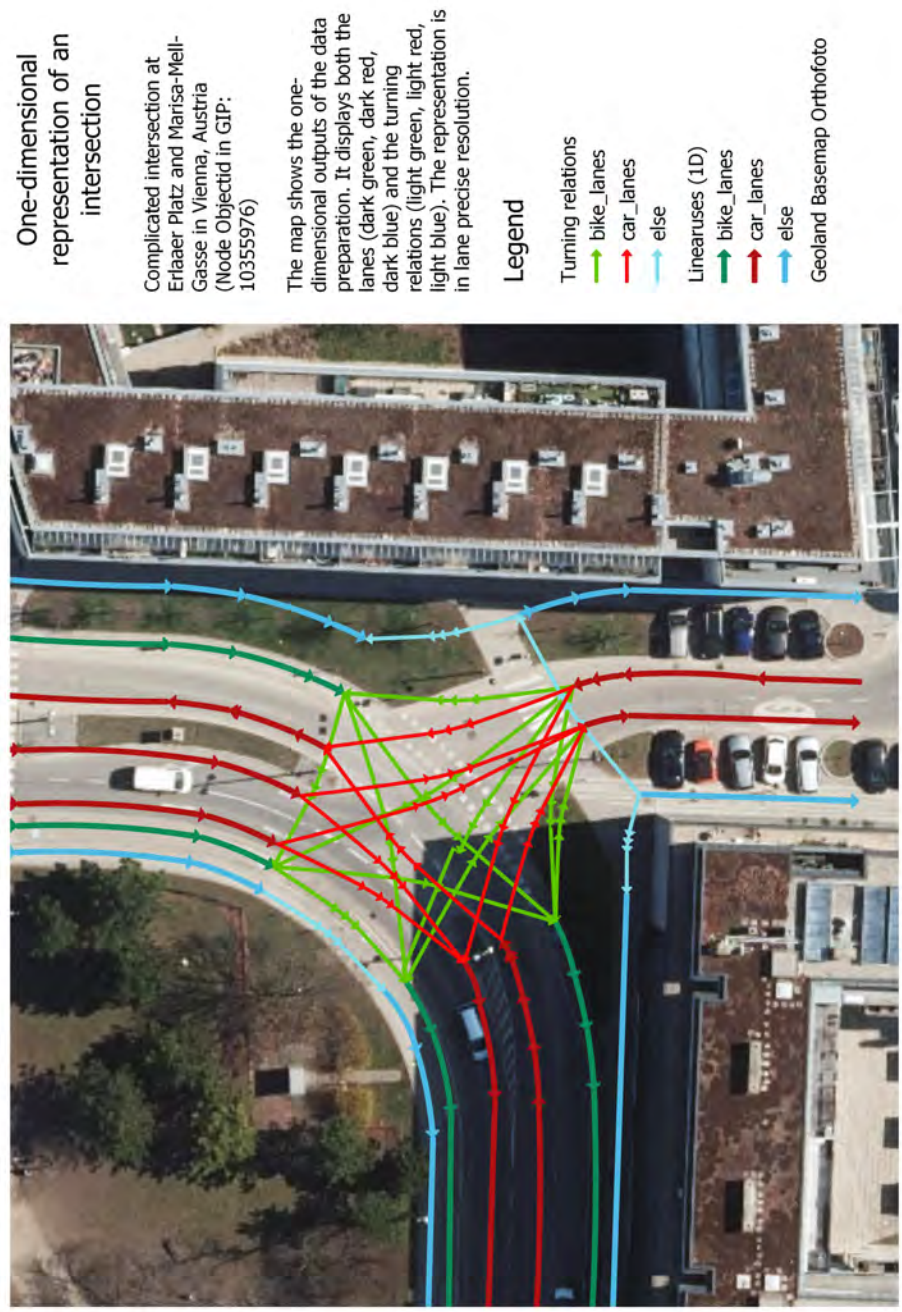

Figure 3.7: One-dimensional representation of intersection of lane-accuracy. Note: the digitizing direction corresponds to the direction of travel

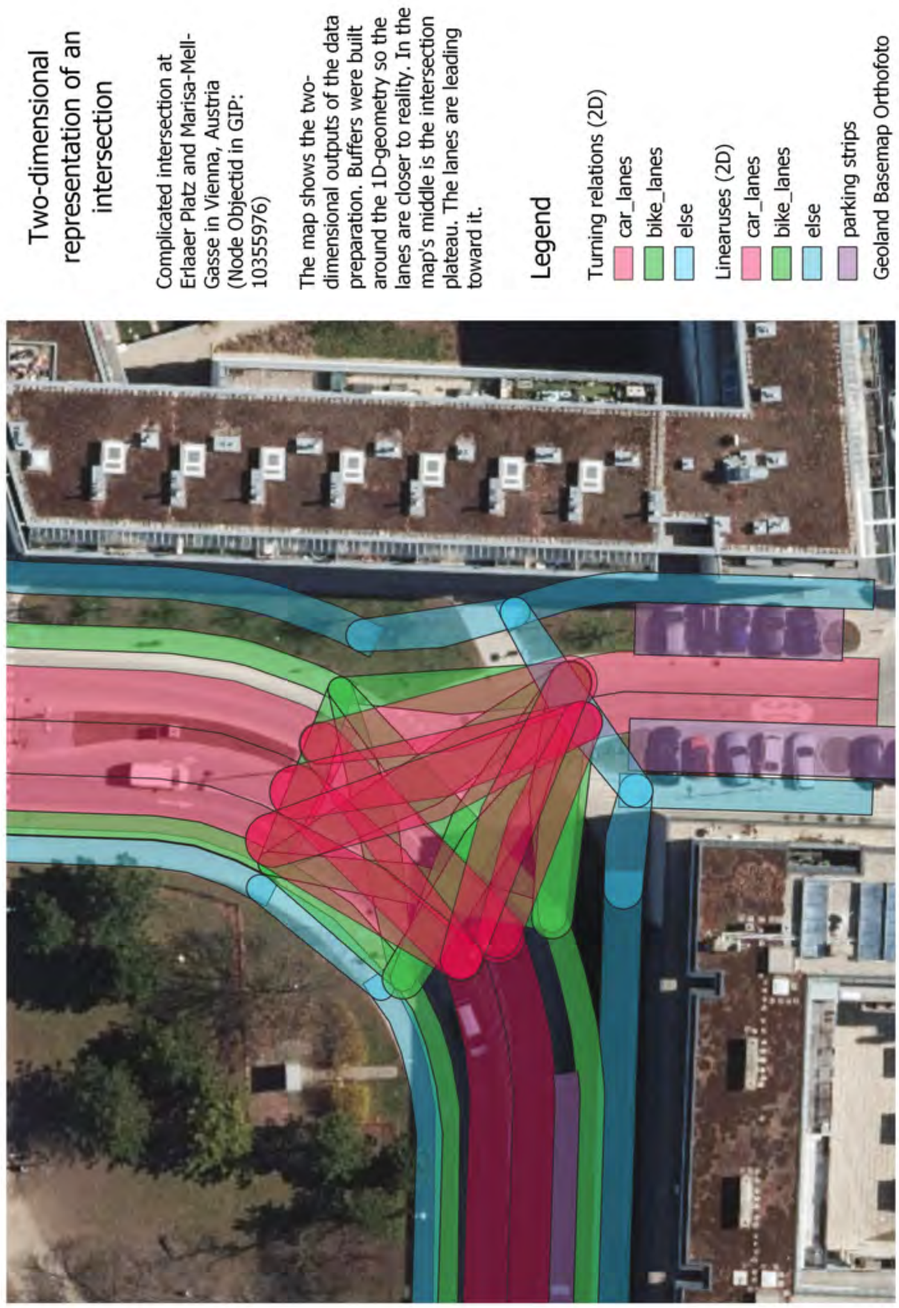

Figure 3.8: Two-dimensional representation of intersection of lane-accuracy.

4 RISK MODELING

This chapter explains how the turning relations are evaluated with respect to their risk. In Chapter 3 the turning relations were derived from GIP data. Thus, more precise information about the intersection's geometry is known now. Based on this information, the risk evaluation is performed. On top, attributive data of the GIP is integrated in the model.

The risk modeling consists of three major steps

The risk model is divided into three broad steps: First, the data regarding an intersection are obtained (Section 4.1). In the next stage, these data are normalized to a uniform scale (Section 4.2) and, at last, the factors are weighted, and the risk index is calculated (Section 4.3).

The potential of calculating a risk index for turning relations

The development of such a risk index for bicyclists at intersections has the potential to be valuable for bicycle routing. At this point in time, it is primarily the edges that are tested and evaluated for their bikeability. This ignores how the distance between two edges is bridged. If bridging the gap takes a lot of time, is difficult, or is risky, this reduces the safety of bikeability of a route. However, knowing which characteristics of an intersection increase the risk of a BMV collision can improve more than just bicycle routing: For one thing, decision makers can use this knowledge to actively improve hazardous intersections so that accidents are less likely to occur in the first place. Additionally, this information can be used to plan and implement lower-risk intersections. Knowing the dangers of an intersection also gives cyclists a tool to ensure their own safety. They can avoid the intersection or proceed with extra caution.

4.1 OBTAINING INFORMATION ABOUT INTERSECTION

Factors concerning the entire intersection and the single turning relation are used

In order to be able to evaluate the risk, the model is in need for information about the turning relations. So in the first step of risk modeling, data about the turning relations is obtained and stored in a table. There are two supertypes of factors: *global* and *local* variables. The global type is information that relates to the entire intersection. Examples are edge degree, urban environment or proximity to railway infrastructure. Local information, on the other hand, is related to a single turning relation - the value applies only to one turn action. Although some information may be related to an entire edge (and thus likely to multiple turning relations), it is handled as a local variable in this model, e.g. the slope of the edge leading into the intersection.

To retrieve the information necessary for the risk modeling, the function *get_risk_factors(node_id)* was developed (see Listing A.11). This function serves as a hub because it retrieves all the information and stores it in a table called *risk_value_table*. In this table there is an entry for each turning relation with the corresponding values. In the case of global variables, these are identical for one intersection. But if it is a local variable, the values can diverge.

As said, *get_risk_factors(node_id)* is a hub where the information converges. This is possible because the function calls what is referred to as helper functions. Each of the helper functions is written for a specific purpose and returns the requested information. This process is illustrated in Figure 4.1.

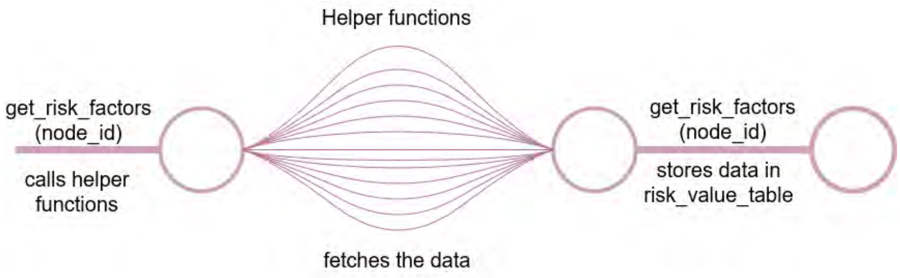

Figure 4.1: The steps of obtaining the data which is used to assess a turning relation's risk.

The following subchapters describe how to obtain the global and local variables. The corresponding scripts can be found in the appendix in Listing A.11 and Listing A.12.

Not all factors identified in the literature review (Section 2.1) could be implemented in the risk model. This is due to missing data due to an open data desideratum. In Figure 2.2, the factors for which no data source could be found are grayed out. It was emphasized to work mainly with the data of the GIP to maintain the consistency of the data.

4.1.1 *Global variables*

URBAN ENVIRONMENT

In the literature review (Section 2.1.1), it was found that it is relevant whether an intersection is located in an urban or rural environment. This information is stored in each link under the attribute *urban* which indicates whether an intersection is in a built-up area. The function *check_urban(node_id)* is called to query if any of the links are associated with the node whose risk is being determined. If this is the case for any of the links, the function returns *true*, otherwise it returns *false*.

TRAIN NEARBY?

Another factor connected to the environment of the intersection is rail traffic. As stated in Section 2.1.1, the risk of accidents increases if there is rail infrastructure within a radius of 5 meters. In order to find any rails within this radius, the function *check_rails(node_id)* is exerted. It checks if a track intersects the 5 m buffer around an intersection and returns the according value.

TRAFFIC LIGHTS

To include traffic lights, another layer was added

As expounded in Section 2.1.2, some researchers estimate the presence of traffic lights to be very important regarding a cyclist's risk. To obtain information about the traffic lights, an external source was used: The city of Vienna provides open government data on signalization. The two layers that are provided on the matter[12] were merged beforehand in QGIS and clipped to the AOI. Then, the resulting layer was imported into PostGIS. There, a buffer of 15 meters was created around the traffic light layer. Within this buffer, the nearest intersection-node was looked for. If indeed there is a nearest neighbor found within the buffer, the node's ID is inserted into the traffic light layer. Due to these preparations, when checking whether the intersection is a traffic light intersection, it is only necessary to check whether there is a tuple in the traffic light table that refers to the ID of the intersection under investigation.

The other two factors related to the risk category traffic rules, sign posts and road markings, were not included in the model because of a lack of data.

EDGE DEGREE

Complex intersection with a high edge degree were discovered to pose a risk to cyclists (Section 2.1.3). Out of this reason the edge degree is determined. This can be easily executed because the degree is an attribute of the node. In case, any problems occur nonetheless, the edges connected to an intersection are counted.

SPEED

Speed is one of the most significant and undisputed risk factors. As elucidated in Section 2.1.4, the MV's speed plays a major role in the safety of all road users. Therefore, though it is possible for roads with different speed limits to meet at an intersection, the speed limit is treated as a global variable. The alternative would be to compare only the speed limits of the two lanes that the turning relation connects.

1 Source as of March 2023: https://www.data.gv.at/katalog/dataset/699a5a9a-348c-4a46-b99e-a23b59f27721

2 Source as of March 2023: https://www.data.gv.at/katalog/dataset/c5cf2502-7572-4fd1-a836-48b335a2d47d

For the model, the first version is applied: the maximum speed allowed in the intersection is used for the risk evaluation. This is because in an intersection it is almost inevitable to interact with other road users. Therefore, the speeds that these road users travel by have an impact on the other turning relations. The GIP provides two different types of speed: the average speed and the maximum speed that is/can be driven. Again, the maximum speed is assumed.

The function querying the speed, *get_speed(node_id)*, first checks what the speed limit of each link. Across links, the highest figure is chosen. If there is no information about the maximum speed, the average speed is employed. If still no speed is found, it is derived from the street type.

STREET TYPES

Although the street type belongs unlike the factor speed to the street properties, they are treated similarly: if a major street is involved in the intersection, this affects all turning relations negatively because major streets bear greater risk for cyclists (see Section 2.1.5). The *check_streettypes(node_id)* function checks the road type of all edges associated with the given intersection. If a major road is connected to the intersection, it returns *true*. So a classification into major and minor roads is already done at this point. It is performed in this step because the references in the literature search (Section 2.1.5) are consistently dividing the road types into these two classes. Meaning, no information is lost performing this categorization.

EXISTENCE ROUNDABOUT

As stated in Section 2.1.6, roundabouts are generally a source of danger for cyclists because it entails more points of conflict than a usual intersection does. To check whether the node under investigation is part of a roundabout, the function *check_roundabout(node_id)* is passed the node's ID. It then checks if the node is connected to a *link* of the type roundabout. The factor is treated as a global variable because the literature does not specify whether the "normal" road leading into and out of the roundabout is also associated with increased risk for cyclists.

The inner circle's radius is calculated with a chevron cut

ROUNDABOUT'S INNER CIRCLE

Another aspect about roundabouts that is relevant regarding cyclists' risk is the size of the inner circle of a roundabout (see Section 2.1.6). For this reason, the inner circle's radius is included as a factor in the risk model. This radius has to be calculated because it is not available as an attribute. Therefore, the calculation of a chevron cut is performed: To find the center of the circle, three points, start, end and center of the edge, are extracted. The distance between the points is computed and buffers are drawn around the three points using this figure. The intersection points of the buffers are determined, and lines are drawn between these points. Where the lines meet is approximately the geometric center of

the intersection. The size of the inner circle of the roundabout can now be determined by calculating the distance from the center point to the start or end point. Since the edge refers to the median of a road, half of the road's width is subtracted.
The result of the calculation is stored as a global variable, which is applied to all turning relations or tuples in the table *risk_value_table*.

CYCLING INFRASTRUCTURE IN ROUNDABOUT

Several literature sources suggest that bicycle infrastructure within a roundabout has a positive impact on cyclist safety (Section 2.1.6). To obtain the corresponding data, the function *check_cyclinfra_in_ra(node_id)* is used to check if there is a bike lane along the edge belonging to a roundabout. In GIP, bike infrastructure is only mapped if it is separate from the car lane. If separate bike infrastructure exists, the function returns *true*.
This information is treated as a global variable because it is assumed that if there is bicycle infrastructure in a roundabout, it is present in every segment of the roundabout.

4.1.2 *Local variables*

Local variables are those factors that can vary from one turning relation to another, although the relations belong to the same intersection. They are an essential part of the model as they are the characteristics of the relation that allows it to differ from other relations. The local factors are the key to the risk modelling on lane-level.

NUMBER OF LANES (CAR LANES)

The number of car lanes crossed indicates the type of maneuver, the intersection's complexity and edge degree

It is of great interest, how many conflict points a turning relation is involved in. On the one hand, they indicate how many possibilities there are for collisions with other road users. On the other hand, the number of conflict points indicate whether it is a left or a right turn maneuver. If it is a left-turn action, there are more other turning relations crossed than in a right-turn action, where no other relation may be crossed at all. And the more conflict points there are, the higher the risk of an accident when using the turning relation.
Additionally, this variable is related to the edge degree and complexity of the intersection as well: if many other turning relations are crossed, this may imply that there is a large number of edges converging at the node. Again, this would indicate an increased risk.

The function *car_lanes_crossed(turn_relation_id)* is executed to return this local variable as it iterates separately over each turning relation belonging to the intersection. Using a for-loop, it counts the other turning relations designated to MVs that the turning relation under examination crosses. This calculated figure is in the result table (*risk_value_table*). This process is performed for each turning relation belonging to the intersection plateau.

NUMBER OF LANES (BICYCLE LANES AND PATHWAYS)

Although the focus of this risk model is on BMV collisions, crossing bicycle and pedestrian paths are also examined. This is because, on the one hand, accidents only involving VRUs do happen a lot - researchers estimate the number of unreported cases to be high. On the other hand, the quantity of infrastructure measures designated to VRUs indicates the number of VRUs using this infrastructure. And more additional road users can increase the risk of accidents (see Section 2.1.4).

Determining the number of crossing bicycle and pedestrian paths is performed the same way as for MV traffic (Section 4.1.2). The only difference is not the turning relations of MVs are counted, but those of cyclists and pedestrians. These two groups of road users are combined here. The reason for this is the focus of the work, which lies on collisions between motor vehicles and cyclists, so pedestrians as a group may be neglected.

The slope is calculated on base of a DTM

SLOPE

The slope at which an edge approaches an intersection is a relevant factor in a risk model because as the slope increases, so does the risk potential. As elaborated in Section 2.1.5, the gradient affects the safety of road users even if it is only 1%. To determine the slope, an additional layer is needed because the GIP does not provide sufficient and reliable data in this regard. For this purpose, the Digital Terrain Model (DTM) of Vienna [3] was included: The download format used was .tif. The original data is divided into several map sheets [4] and the ones corresponding to the AOI were downloaded and imported into the database using PostGIS' *raster2pgsql* tool.

The *check_gradient_slope(edge_id)* function determines the height of the start and end points of the edges. For this, the values of the DTM at the coordinate of the start and end point is queried. The slope is calculated from these elevation values and the distance between the start and end point. This is a local variable that is queried for each edge. The result of the calculation is transferred to those turning relations of the table *risk_value_table* that run parallel to the edge.

CYCLING INFRASTRUCTURE

In Section 2.1.7, the impacts of cycling infrastructure on cyclists' risk are debated. The referenced scientists consider the type of cycling infrastructure important. Therefore, the helper function

3 Source: https://www.wien.gv.at/ma41datenviewer/public/start.aspx
4 Map Sheets: 43/3, 43/4, 44/3, 44/4, 53/1, 53/2, 54/1, 54/2

check_cycl_infra(turn_relation_id, lane_id, node_id) is used to determine whether the turning relation is bike infrastructure and what type it is. The function first checks if the turning relation has a bicycle base type. If this is the case, the information is stored in the result table.
If this is not the case, further investigation is carried out by determining the base type of the lane before and after the turning relation.
If no results can be obtained this way, the table *bikehike* is included, a table that belongs to dataset A of the GIP. Since this layer is also part of the GIP, it can be linked to the geometric dataset B used in this work by already existing foreign keys. *bikehike* contains information about the bike infrastructure along the edges. To obtain information about the turning relation, the type of the lanes before and after the turn lane is queried and stored in case it is cycling infrastructure.

MIXED TRAFFIC

Another factor that is applied on the level of a single turning relation, is *mixed traffic*. As noted in the literature review (Section 2.1.7), mixed traffic lanes increase the risk of bicycle accidents. The GIP provides information on mixed traffic. However, the most detailed information is not in one of the geometry layers, but in the *bikehike* layer of dataset A.

check_mixed_traffic(linearuse_lanes_fid, node_id) is a local function that retrieves information about mixed traffic. It is called twice by the parent function *get_risk_factors* in a for-loop that iterates through the turning relations: for the lane before and the one *after* the turning relation. The value returned by the function refers to the lanes connected to the turning relation.
The function itself does the following: The tuple of the table *bikehike* that is connected to the examined lane is found (see Listing A.12). *bikehike* holds information about mixed traffic and what type of mixed traffic it is. This information is returned by the function. Since this data is retrieved for the lanes leading to and coming from the turning relation, both are stored in the result table (*risk_value_table*).

4.2 NORMALIZATION

In order to combine and compare the information collected about the intersection and the individual turning relations, they have to be normalized. For this purpose, the collected data are projected on a scale of 0 to 1. A value of 0 represents high risk and low safety for cyclists. The value 1 means low risk and higher safety for cyclists.

Normalization serves the purpose of being able to combine the accumulated data

The function *normalize_factors()* (Listing A.13) performs the normalization based on the result table *risk_value_table* of the previous section. Using a for-loop, it iterates through each tuple or turning relation and assigns the information on the scale of 0 to 1. The results of the normalization are stored in the table *normalized_risk_values*, which is similar to *risk_value_table*: it contains the *fid* of the turning relation, the ID of the node and the normalized values of the risk factors.

In Figure 4.2 and Figure 4.3 it is shown how the data were normalized. In the case of bicycle infrastructure and mixed traffic, not all values could be presented due to the abundance of information. However, in the code (Listing A.13) they can be fully explored.

In the following, Figure 4.2 and Figure 4.3 are closer analyzed as they contain a nearly complete overview of the normalization:

SURROUNDING INFRASTRUCTURE

An intersection is, as described in Section 2.1.1, less hazardous for cyclists if it is located in an urban area. Therefore, an urban intersection is classified as posing small risk (normalized value = 1), whereas rural intersections are classified as high risk (normalized value = 0) in the normalization step. Similarly, the factor holding information on whether rails are nearby is normalized: if yes, this corresponds to high risk (value = 1), otherwise it is estimated nonhazardous for cyclists.

Booleans are assigned either 0 or 1

TRAFFIC RULES

Although the impact of traffic lights is not entirely clear (see Section 2.1.2) it was decided to assign a signalized intersection a normalized value of 0, and an unsignalized intersection a value of 1. This assignment was chosen because the majority of the referenced literature found increased risk at intersections with traffic lights.

GEOMETRY

In contrast, the principle of assigning normalized values to the number of legs is unambiguous. The higher the edge degree, the more complex the intersection and the higher the accident risk (see Section 2.1.3). Therefore, an edge degree of two and minor is related to a normalized value of 1, whereas six and more is assigned 0.

USAGE CONDITIONS

Correlation: Speed and number and severity of accidents

Concerning the meaning of speed for the road users' risk, the referenced scientist in Section 2.1.4 are in complete agreement. They see a positive correlation between speed and the number and severity of injuries. Hence, the normalized value reflects the researchers' conclusion and is the lowest (0) if the speed limit is set 0 km/h and the highest (1) if the speed limit is 100 km/h and beyond.

STREET PROPERTIES

As explicated in the previous Section 4.1.2, the accident risk and the number of turning relations, that the relation under examination crosses, correlate as well. Therefore, the normalized value is set to 0 if a turning relation does not cross another relation designated to MVs. However, the value rapidly increases in case a relation is crossed.

The same principle is applied to the number of crossed turning relations for bikes and pedestrians. Yet, the risk of an accident when colliding with another VRU was estimated lower. Out of this reason, the normalized risk value does not increase as rapidly its motorized counterpart does.

The category street properties holds on top of the crossed turning relations the factors gradient and street type. Reviewing Section 2.1.5, it can be summarized that steep slopes increase the risk of accident for the road users. On the basis of the figures found in the literature review, the assignment scheme for the intensity of the gradient was created. It classifies edges with a lower steepness than 1% to be of low risk (normalized value = 1), 3% of medium risk, and all of the values above are graded as higher risk.

The third factor, street type, is already classified beforehand into minor and major roads. This is because it was found in Section 2.1.5, that an intersection connected to major roads bears higher risk for cyclists. Accordingly, if this is the case, the intersection is assigned a normalized value of 1.

ROUNDABOUT

One of the greater risks for cyclists in traffic is roundabouts, as was found in several sources (Section 2.1.6). Hence, if the intersection is connected a roundabout, the normalized value given is 0, otherwise it is 1.

However not only the presence of a roundabout is of interest but also

the radius of its inner circle. The bigger this figure is, the riskier the roundabout is for cyclists. Correspondingly, a small radius of at most 3 m is graded with a high normalized value of 1, whereas a radius of at least 20 m is assigned the lowest value.
Furthermore, it is considered relevant by scientists if a roundabout has designated cycling infrastructure. Due to a lack of data, only the existence of cycling infrastructure can be observed, but not its type or the roundabout's structure. So, the assignment of the normalized values is generalized to whether cycling infrastructure is present, which is associated with decreasing risk.

Both cycling infrastructure and mixed traffic enable a very precise assignment and assessment

CYCLING INFRASTRUCTURE

The category bicycle infrastructure rates the suitability of the infrastructure for bicyclists.
The first factor of the category refers to the presence of bicycle infrastructure. However, because the type of the infrastructure is also provided, this factor can be graduated. Physically separated bike lanes are assigned a low-risk value of 1, whereas lanes for cars are normalized toward a low value. The other types are normalized according to their suitability for cyclists (see Listing A.14).
The second factor of the category is mixed traffic. It was determined in Section 2.1.7 that mixed traffic has an amplifying effect on the accident risk of cyclists. Accordingly, bike lanes that are exclusively accessible to bicyclists are assigned the value 1, whereas zones of encounter are assigned a low value. Other types are normalized according to the degree of mixed traffic, similar to the bicycle infrastructure factor (see Listing A.14).

The two factors belonging to the category also have in common that the normalized value is partly derived from the lanes adjacent to the turning relation. If the types of the lanes differ, the normalized value that indicates a higher risk is chosen.

4.3 INDEX CALCULATION

After the collected data has been normalized in the previous function and thus also evaluated (e.g. existence of traffic circle was evaluated with 0 as disadvantageous), the last step is the weighting of the individual factors and the calculation of the risk index. In this step, it is determined how much influence an indicator has on the risk assessment of a turning relation. And finally, the index is computed.

The user can insert their own values or use the default weighting scheme

To weight the factors, there are two options: the user can determine their own weighting scheme and insert the according parameters to the weighting function (Listing A.14). The other option is to apply the default values (Listing A.15) These are shown in the table below.

Table 4.1: The default values used for index calculation

Indicator	Default Weight
Speed	0.3
Roundabout	0.2
Cycling infrastructure	0.2
No. of turning relations crossed (MVs)	0.15
Gradient	0.1
No. of turning relations crossed (VRUs)	0.05

The default weighting scheme

These default values are derived from an extensive literature research that is summarized in Section 2.1.
On the factor *speed* the greatest emphasize is put on because it is identified as one of the most important factors regarding the safety of road users. *Roundabouts* are known to bear increased risk for cyclists as they augment the number of conflict points. Researchers found they cause significantly more accidents involving cyclists than usual intersections do. In contrast, *cycling infrastructure* can have a positive impact on the risk index, depending on its type. Separated cycle lanes protect cyclists from MV traffic. The *number of turning relations (MVs) crossed* is included with a moderate weight because it determines the conflict points and thus the number of possibilities for crashes of a turning relation. The fifth factor, *gradient*, is integrated because researchers found it to increase the risk of an intersection up to three times. The final factor, *number of turning relations (VRUs) crossed*, is included with a minor weight because most accidents involving cyclists are BMV collisions and yet, it is weighted because scientist suspect a high number of unreported cases.

Calculation of risk index

The equation below is used to calculate the index for the turning relations. The weights are applied the same way to each turning relation, but the indicators differ in the local variables. This can result in different index values for the turning relations.

$$index = \frac{\sum_{i=1}^{n} indicator_i * weight_i}{\sum_{i=1}^{n} weight_i}$$

However, the function *weighting(...)* (Listing A.14) is open to inputs regarding the weights. It is given the weights of the factors as parameters. This way, the model is interactive.

The resulting table is called *weighted_turnuse*. It contains the turning relations of an intersection, the corresponding *fid*s and the geometries of the relations.

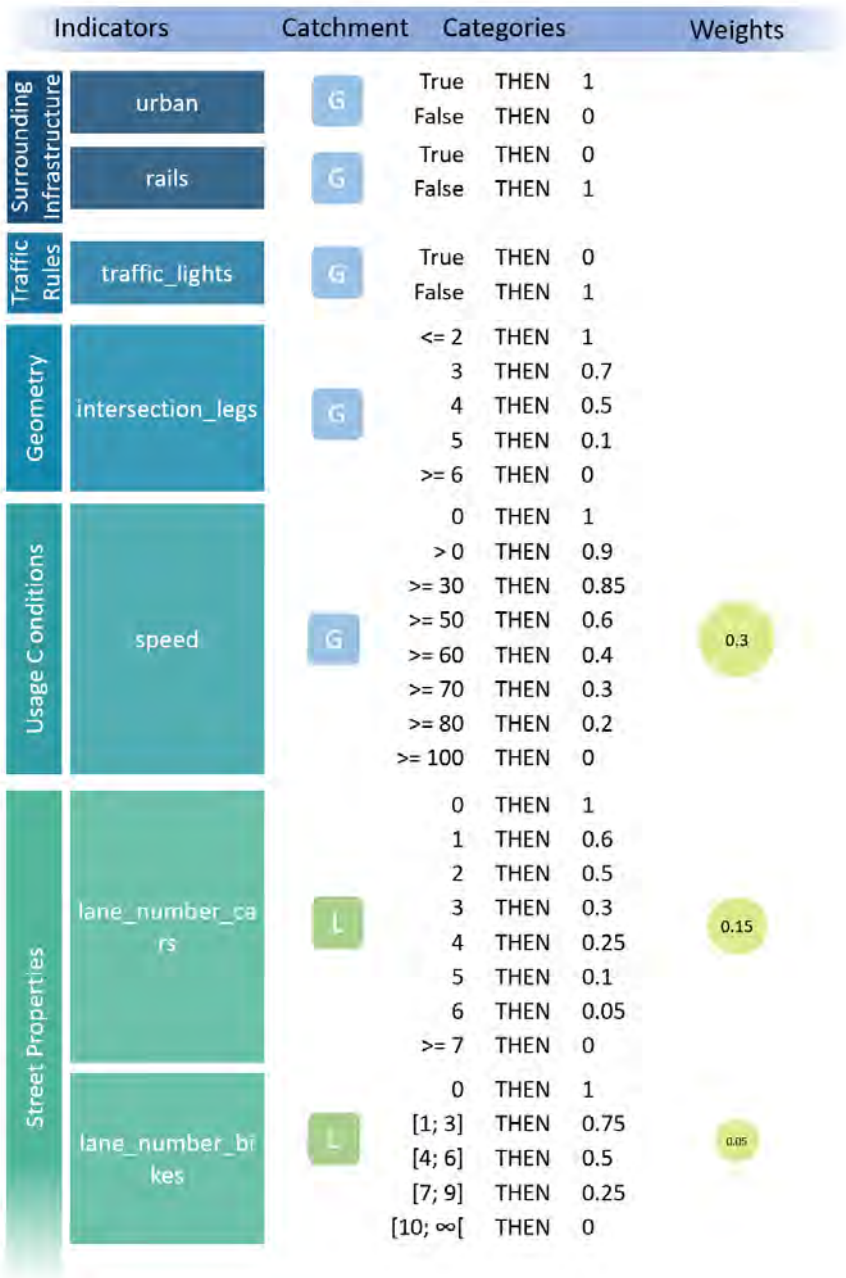

Figure 4.2: The indicators used for the turning relation's risk assessment, whether it is a Global/Local indicator, how the values get normalized and the default weights. Part 1.

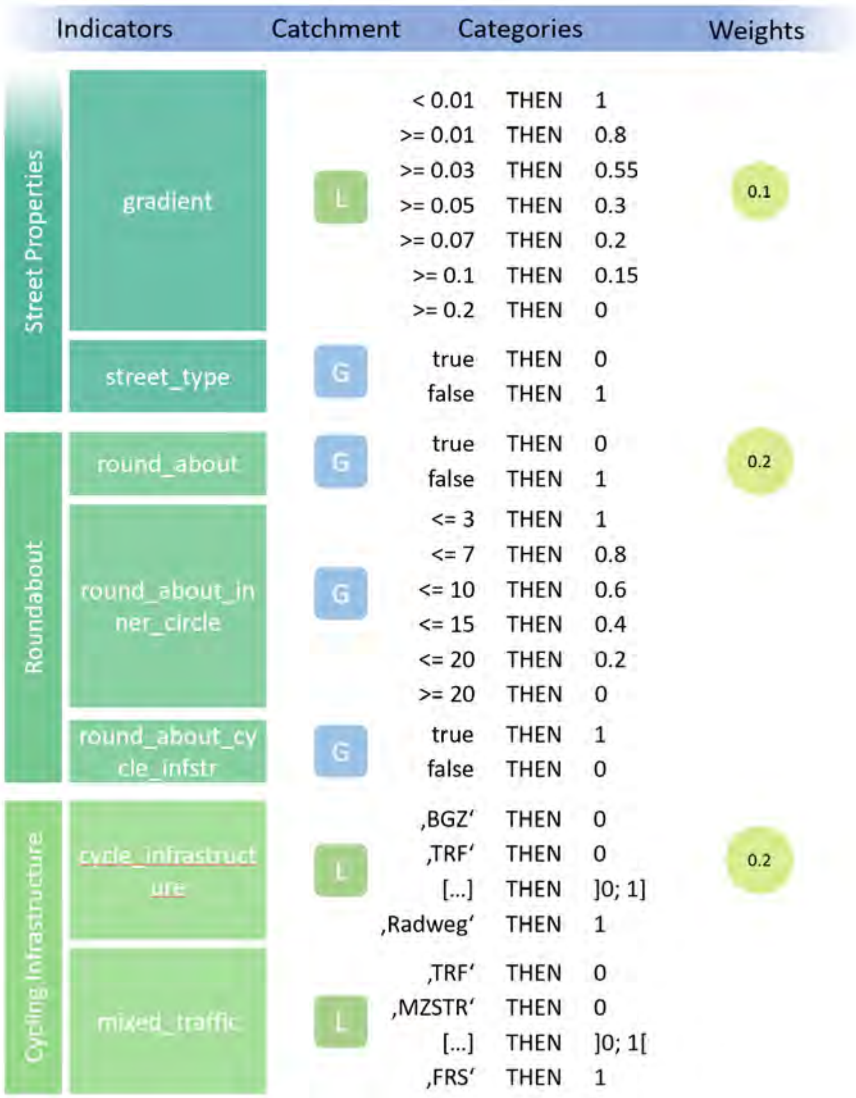

Figure 4.3: The indicators used for the turning relation's risk assessment, whether it is a Global/Local indicator, how the values get normalized and the default weights. Part 2.

5 RESULT

The main result of this work is a model which assesses the risk of the turning relations of intersections. The model is applicable to Austria as it is based on GIP data - however the area of interest used is limited to the south-west of Vienna. The model can be consulted to determine the safest intersections for cyclists in the routing process. It can also be applied to simulate the influence of factors on an intersection. As the underlying dataset is regularly updated by the Austrian government, it is expected that the risk model will continue to be useful in the long term. With one exception, the model uses the original data without prior adjustment. This makes it easier to update the data in the model.

To demonstrate the potential benefits of the model, proof-of-concept demonstrations are given below. An intersection is automatically decomposed into its turning relations. These are enriched with information and then evaluated in terms of their risk using factors weighted with different degrees of importance (see Figure 5.8). The results of this risk analysis are presented in the form of maps of three intersections with different characteristics. This allows in-depth interpretation and comparability. The advantage of this type of modeling is that each turn can be assessed individually, and the variable weighting allows the assumed influence of the factors to be controlled and adjusted.

In Chapter 4 a lot of information about the intersection is obtained. However, not all of these factors affect the intersection risk index the same. Therefore, they are weighted. The default weights, which were applied in different variations to the demonstration results, place value on the following factors: speed, presence of roundabouts and bicycle infrastructure, crossed car and bicycle lanes, and gradient (see Figure 4.2 and Figure 4.3). On the one hand, these values were chosen because experts emphasize their influence. On the other hand, there is scientific consensus about their influence, and they are both local and global variables.

The demonstration examples shown in the maps experiment with the weighting of factors, both locally and globally. Other situations are also simulated: How does a traffic light or increased speed affect the risk index?

The three intersections used to present the results were chosen because they are located in the AOI and bring together MVs as well as bicyclists and other road users. *Intersection A* is a one-way intersection. *Inter-*

section B is a bit more complex, as it also includes a two-lane roadway and bicycle lanes that merge with automobile lanes. *Intersection C* is a classic four-way intersection with bike facilities. More information about the intersections can be found in the table below. It summarizes the characteristics of the three intersections. Its purpose is to provide an overall impression of them and to describe the baseline situation from which the factors will be evaluated, and the data adapted to the research question.

Table 5.1: Detailed information on the intersections of the result demonstration

	Intersection A	Intersection B	Intersection C
node id	10354202	10355976	10355123
basetype	1, 2, 7, 35	1, 2, 7, 21, 22	1, 2, 7, 22
urban	yes	yes	yes
rails	no	no	no
traffic light	no	no	no
node degree	4	3	4
speed	30	30	50
gradient [%]	0.013 - 0.039	0.002 - 0.010	0.005 - 0.025
street type	minor	minor	minor
roundabout	no	no	no

As shown in Section 5.1.1 and Section 5.1.2, both one- and two-dimensional models were developed as a basis for risk modeling during data preparation. The two-dimensional representation of the intersection has the advantage of identifying entire zones where road users may encounter each other. It is also a precursor to the HD map. The one-dimensional view, on the other hand, can indicate the direction of travel. In addition, information can be presented more concisely. Since this model focuses on clear communication and presentation of risk indicators, the one-dimensional representation was chosen.

The workflow from data preparation to data modeling provides as a final result the turning relations between the lanes at the intersection under study. Each of these relations has a risk value in which the variables of the relation and their weights are aggregated. By changing the weights, the risk values are re-evaluated. The model is interactive and is based on the user's priorities in terms of what they consider to be the riskiest factors.

This interactive model runs in a PostgreSQL database. For this purpose, a database structure was built, which is shown in the appendix as entity relationship diagram in Figure A.3. Note that only the most relevant tables are shown, and only those attributes that are important for the structure of the system are included.

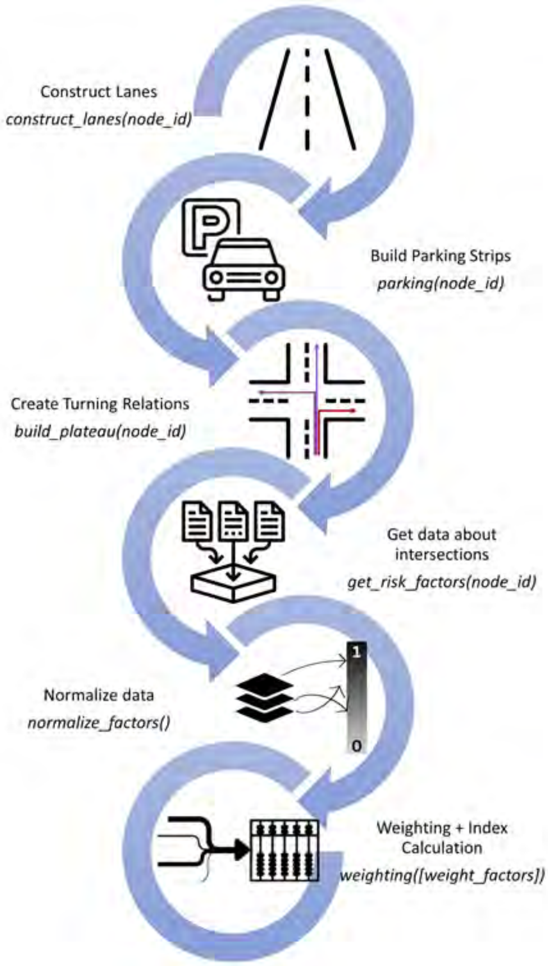

Figure 5.1: The workflow of the project visualized: from lanes to turning relations over data collection and normalization to the calculated risk index.

5.1 MAPS

Below are visualized outputs from the risk assessment for three different intersections with different characteristics and weighting schemes.
The maps are discussed in Section 6.1.

5.1.1 *Data Preparation Status - 1D*

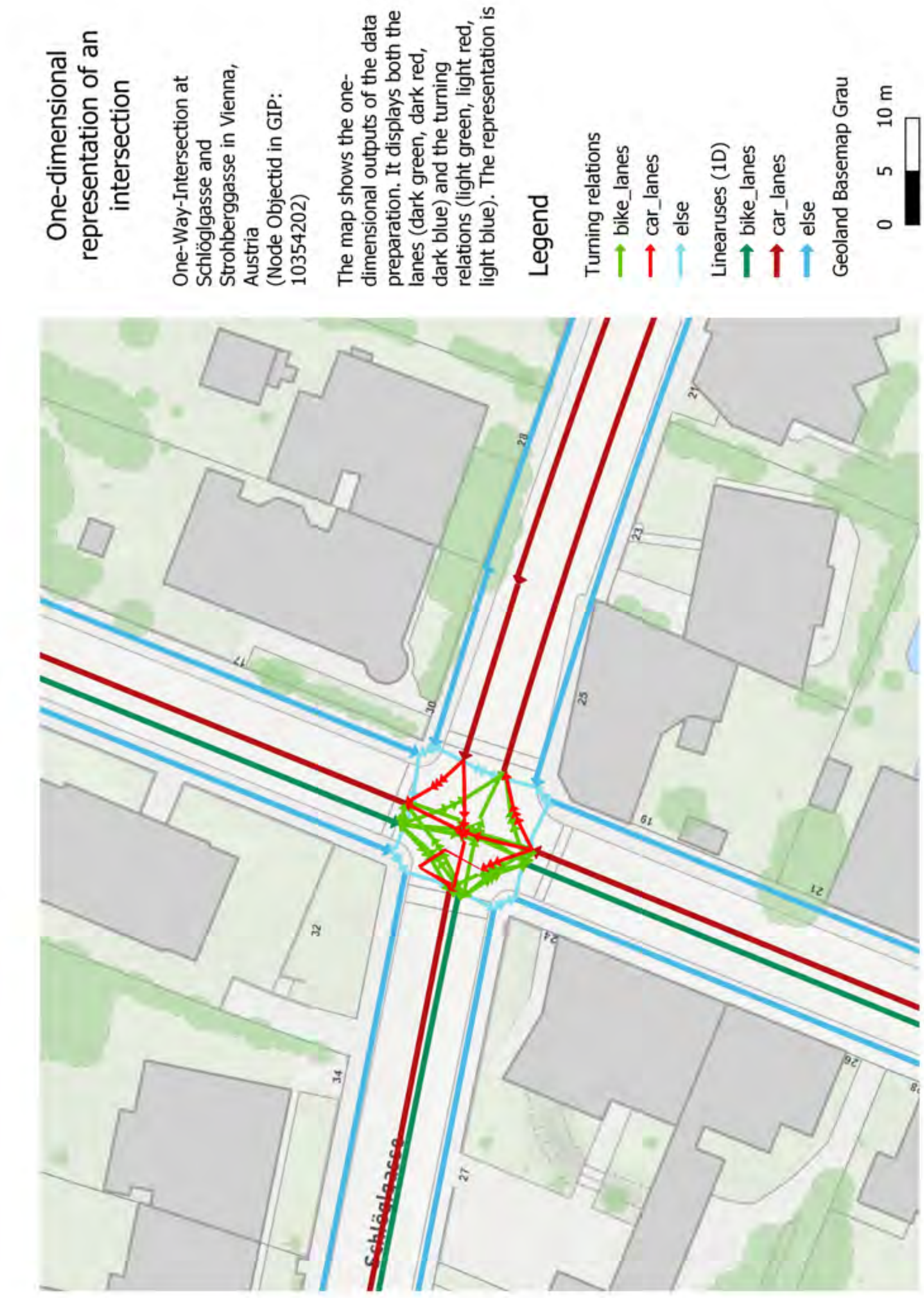

Figure 5.2: Intersection A - the results of the data preparation displayed one-dimensionally. *Note: the red turning relation which does not quite look right is simply a display error of QGIS*

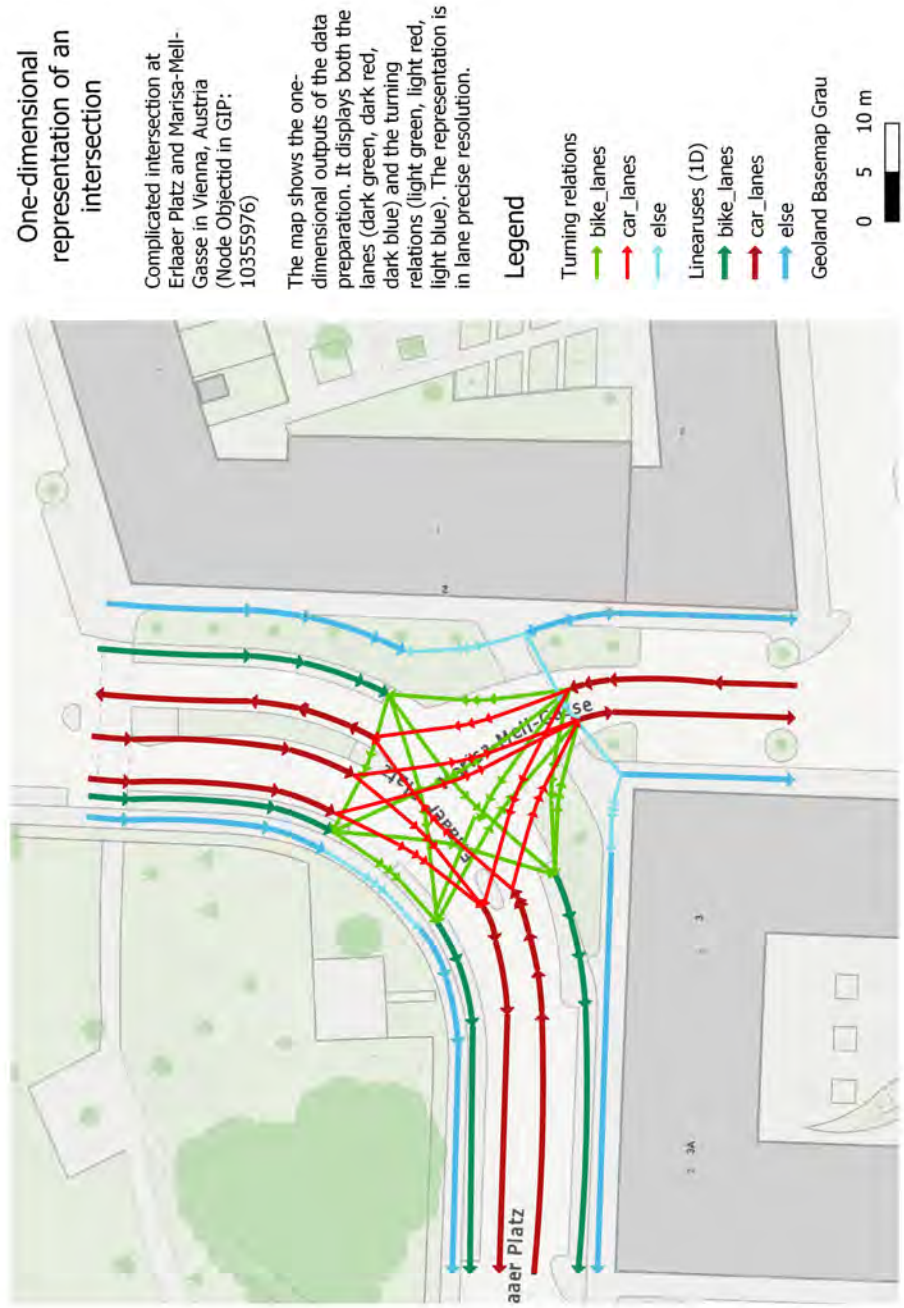

Figure 5.3: Intersection B - the results of the data preparation displayed one-dimensionally

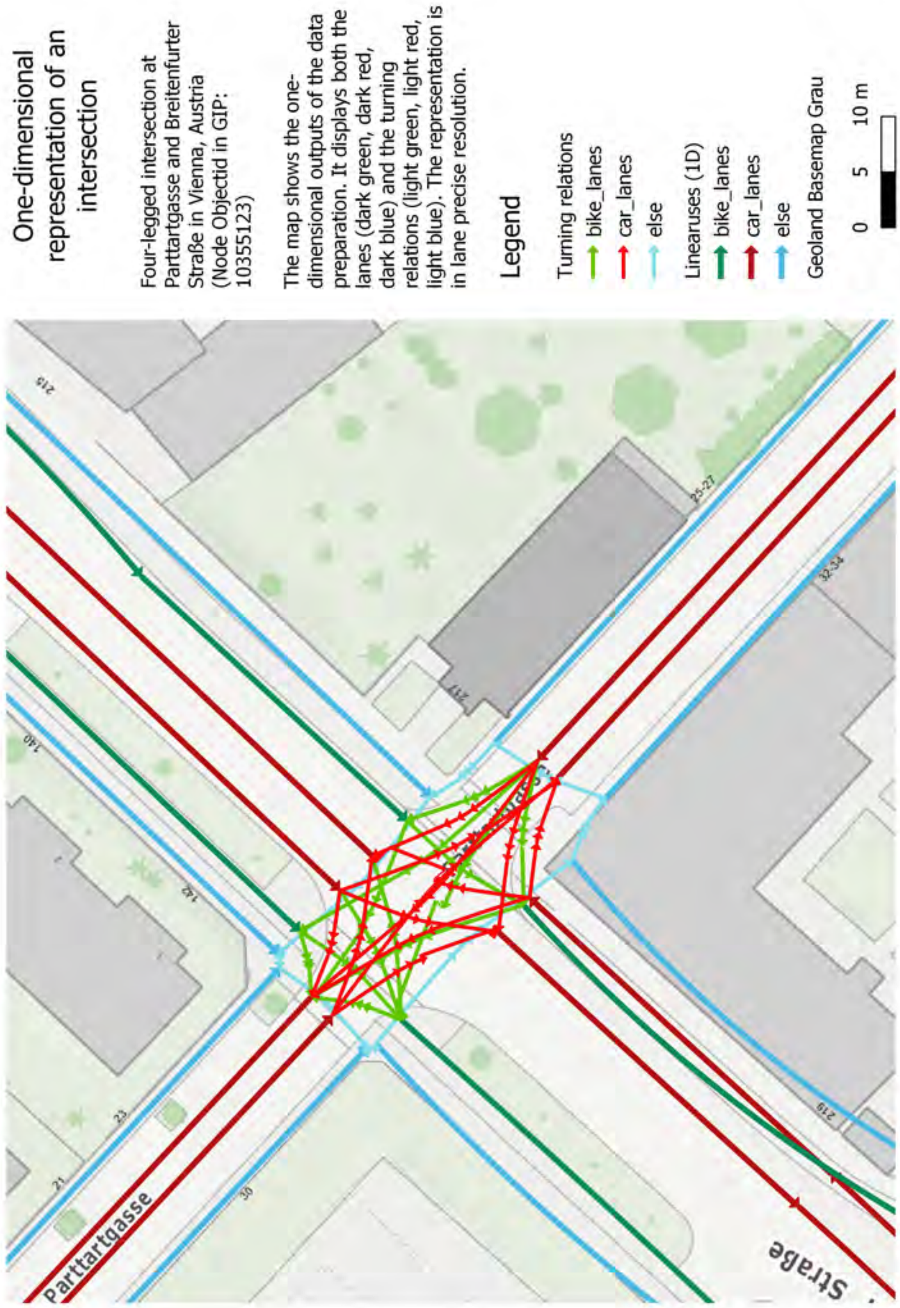

Figure 5.4: Intersection C - the results of the data preparation displayed one-dimensionally

5.1.2 *Data Preparation Status - 2D*

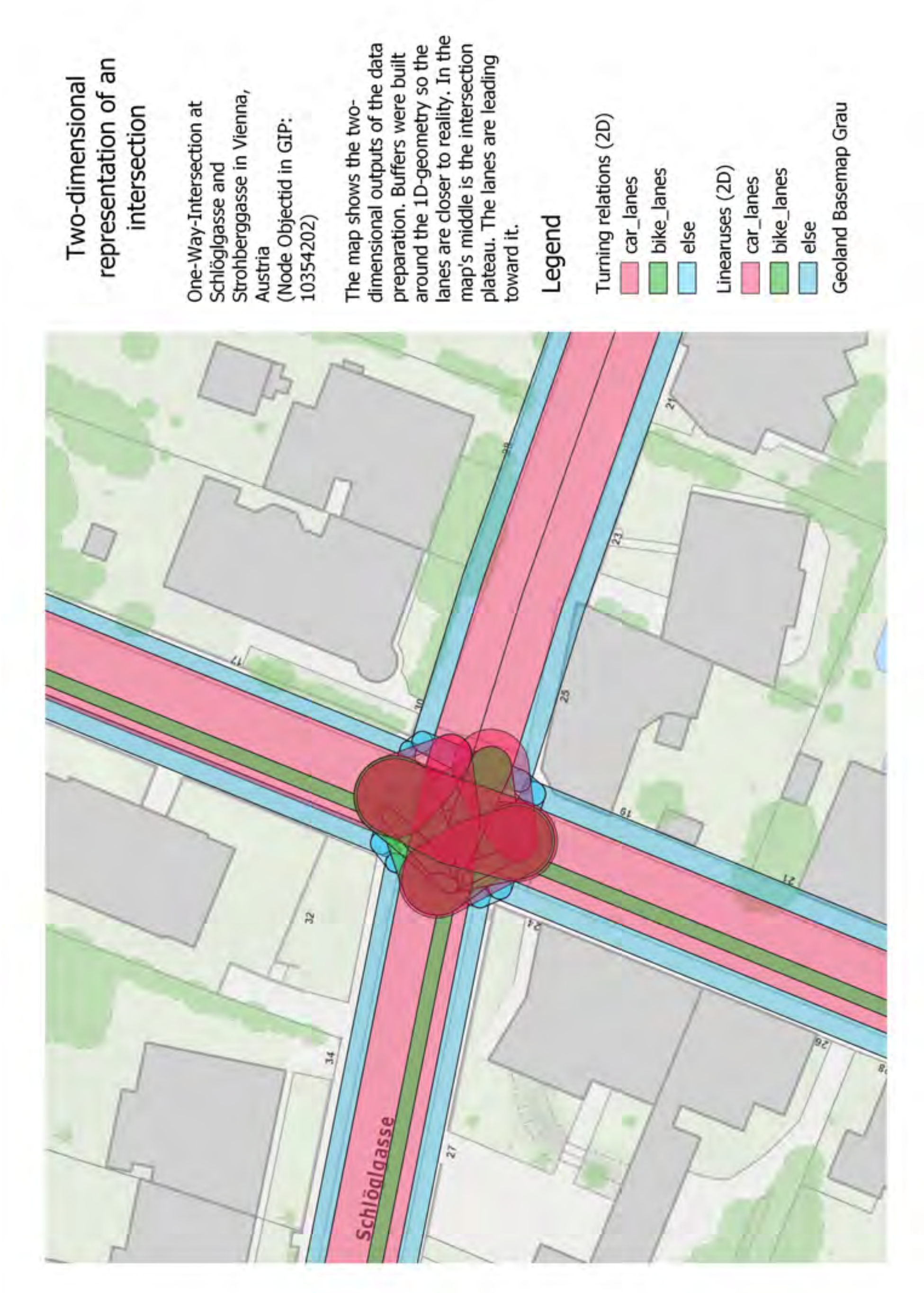

Figure 5.5: Intersection A - the results of the data preparation displayed two-dimensionally

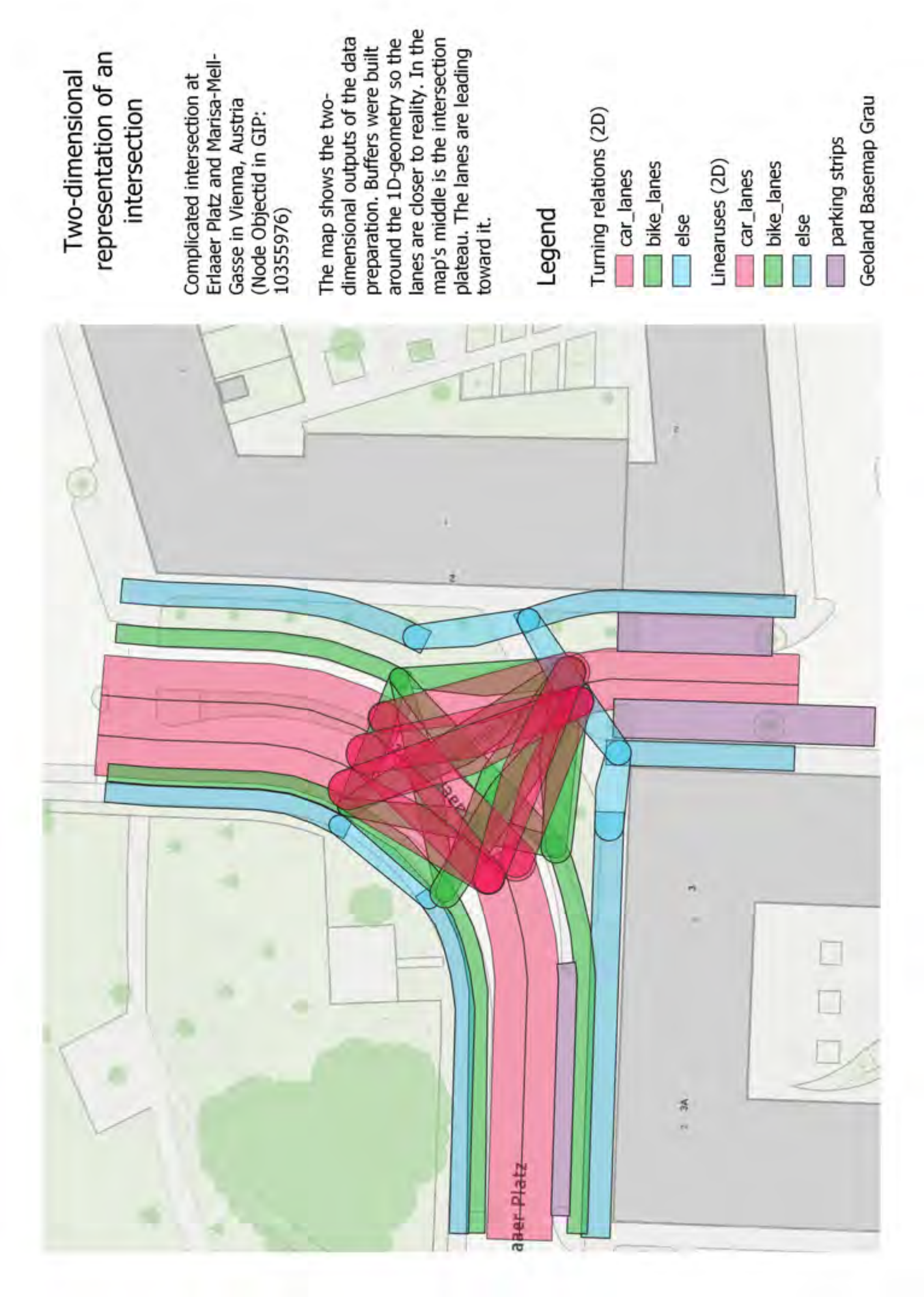

Figure 5.6: Intersection B - the results of the data preparation displayed two-dimensionally

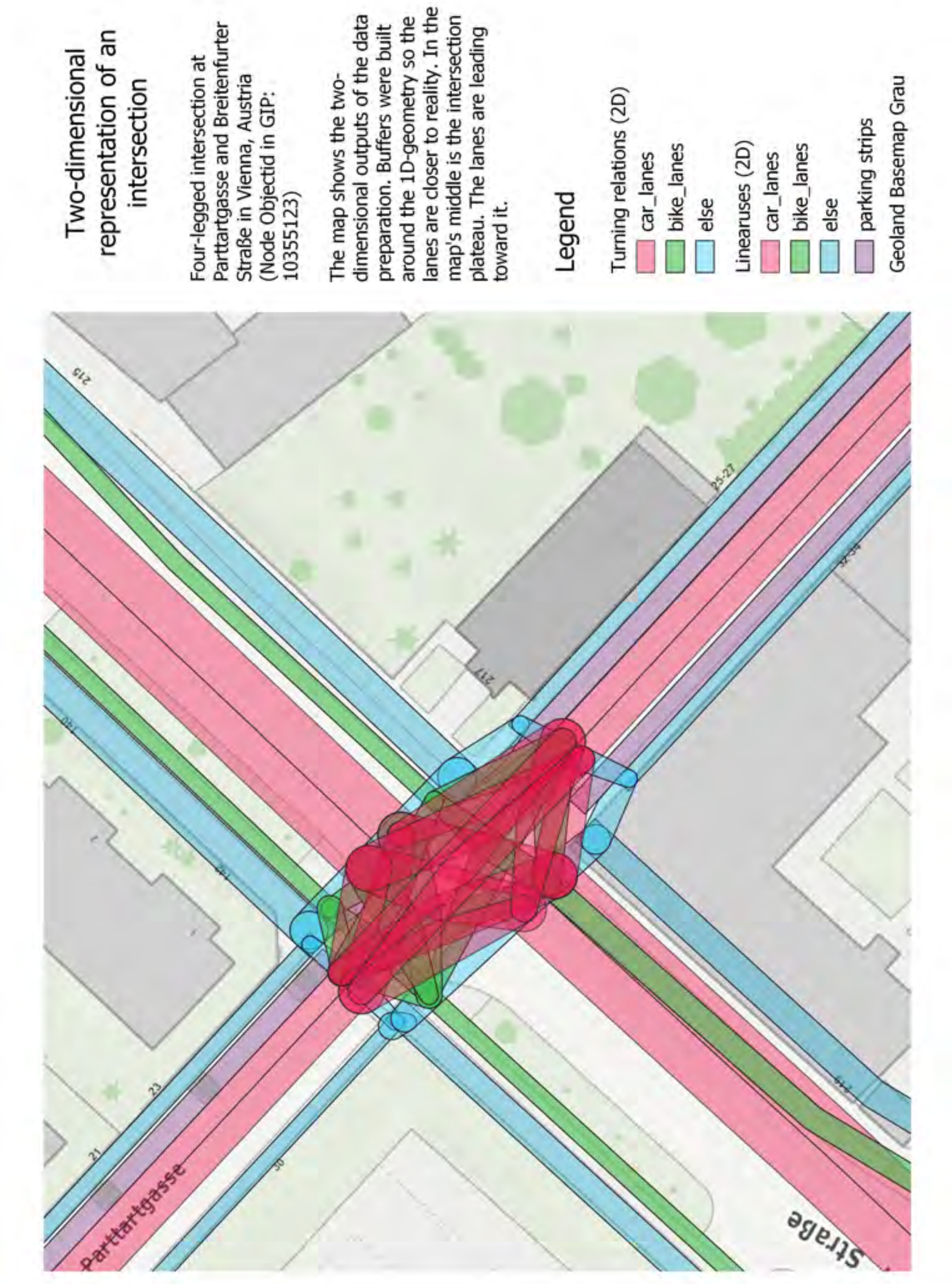

Figure 5.7: Intersection C - the results of the data preparation displayed two-dimensionally

5.1.3 *Weighting by default values*

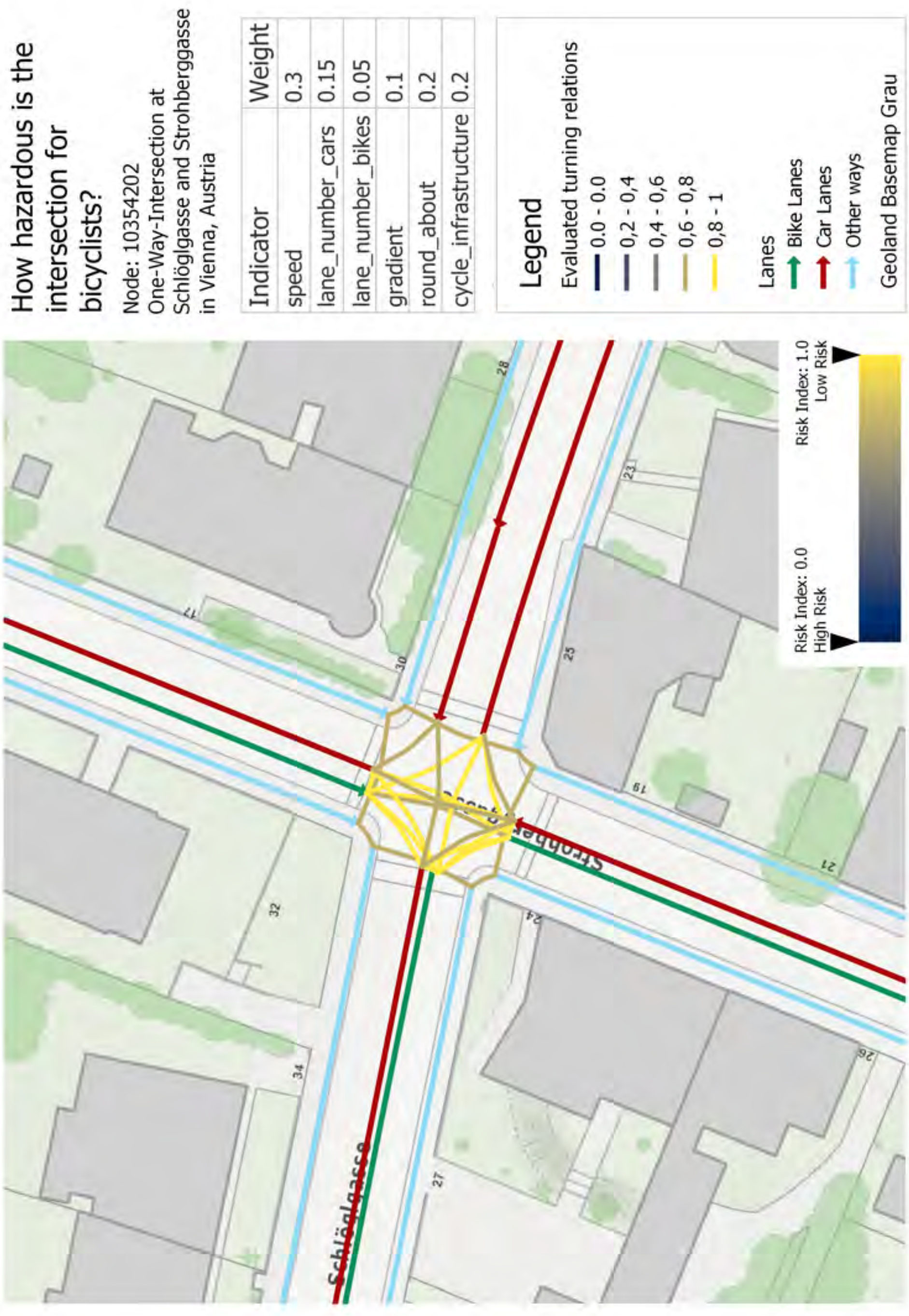

Figure 5.8: Intersection A - the turning relations are weighted by the default values which lay emphasis on speed, the existence of roundabouts and cycling infrastructure, the steepness, and how many car and bike lanes are being crossed by the relation under examination.

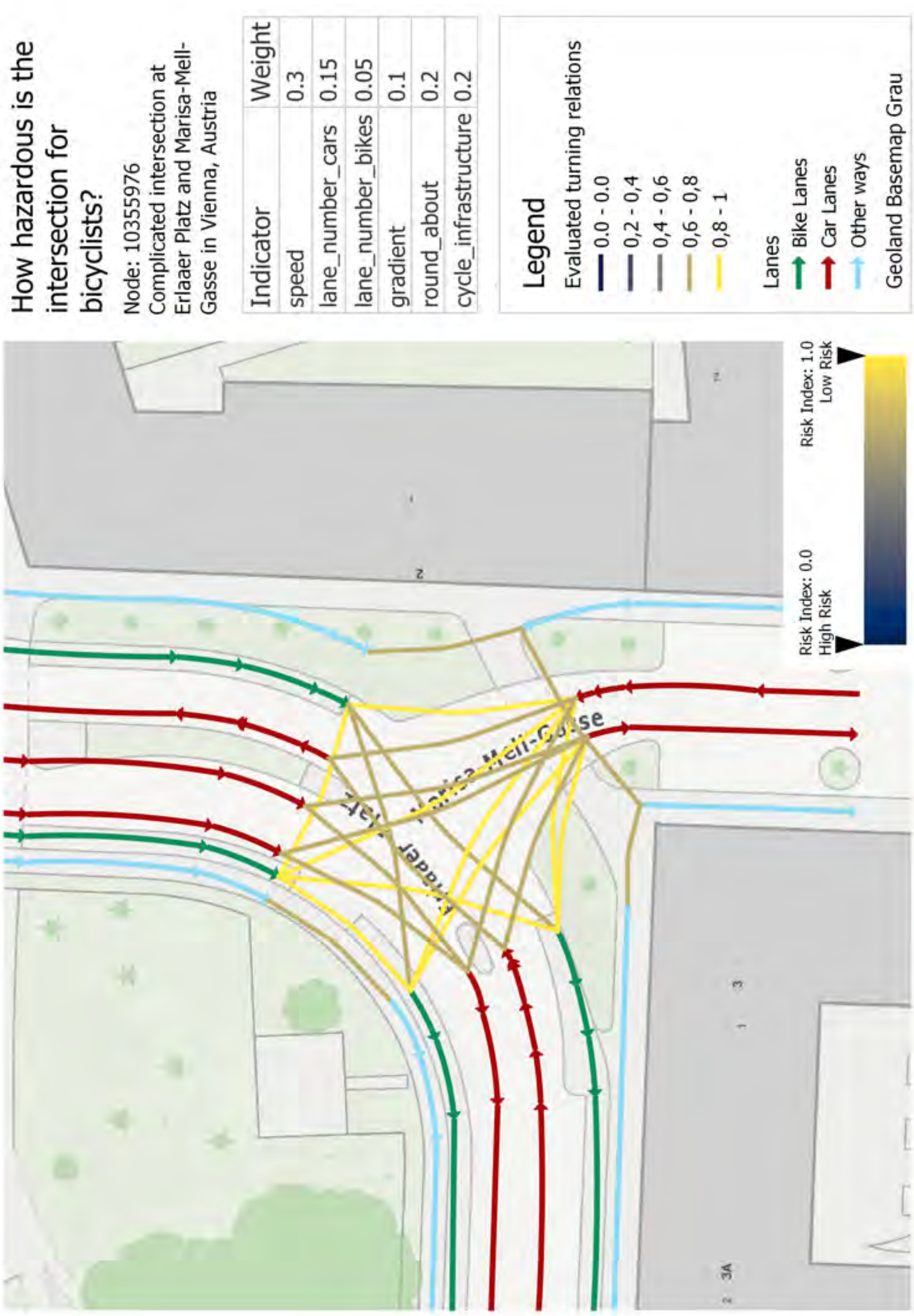

Figure 5.9: Intersection B - the turning relations are weighted by the default values.

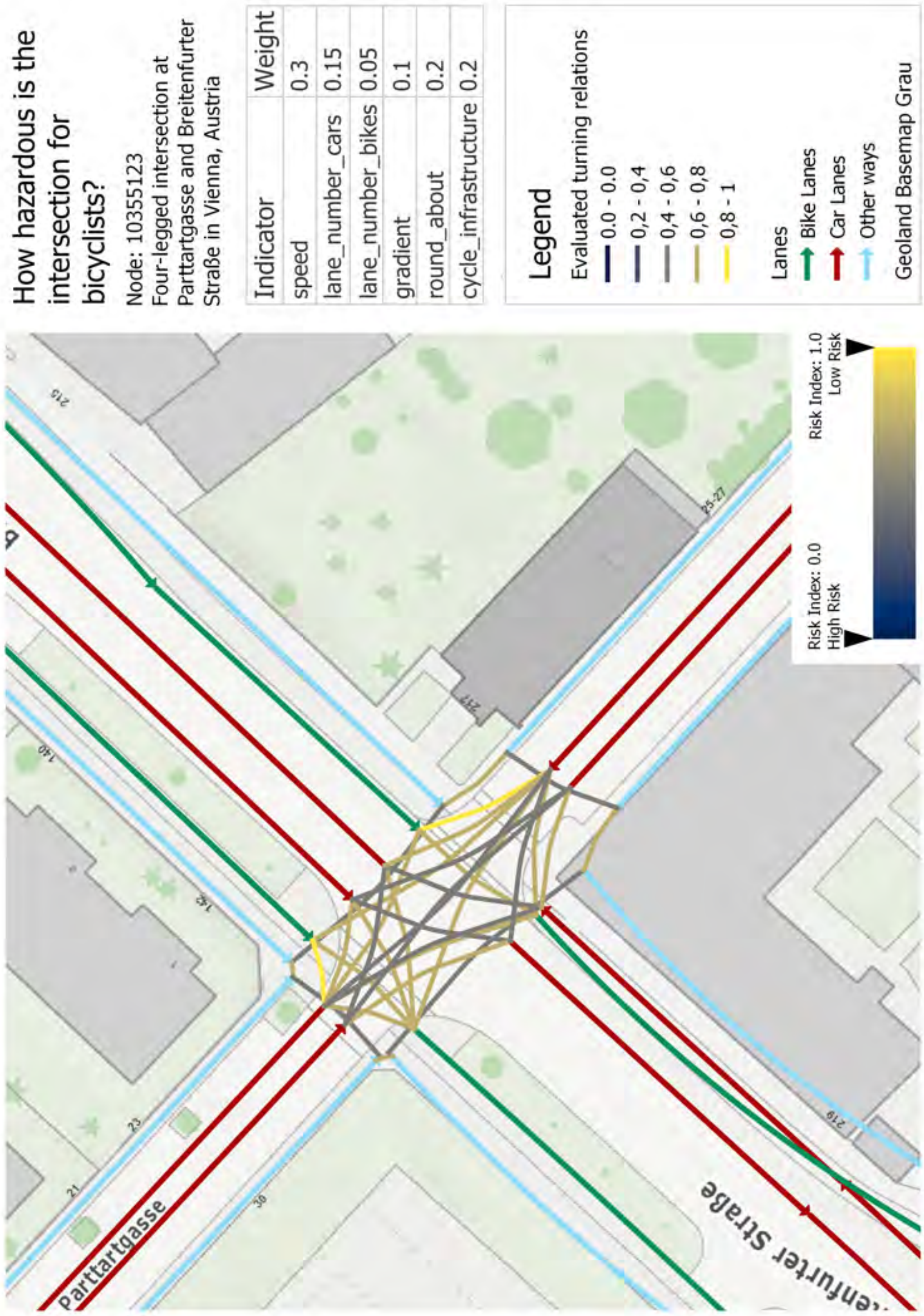

Figure 5.10: Intersection C - the turning relations are weighted by the default values.

5.1.4 *Weighting the local factors*

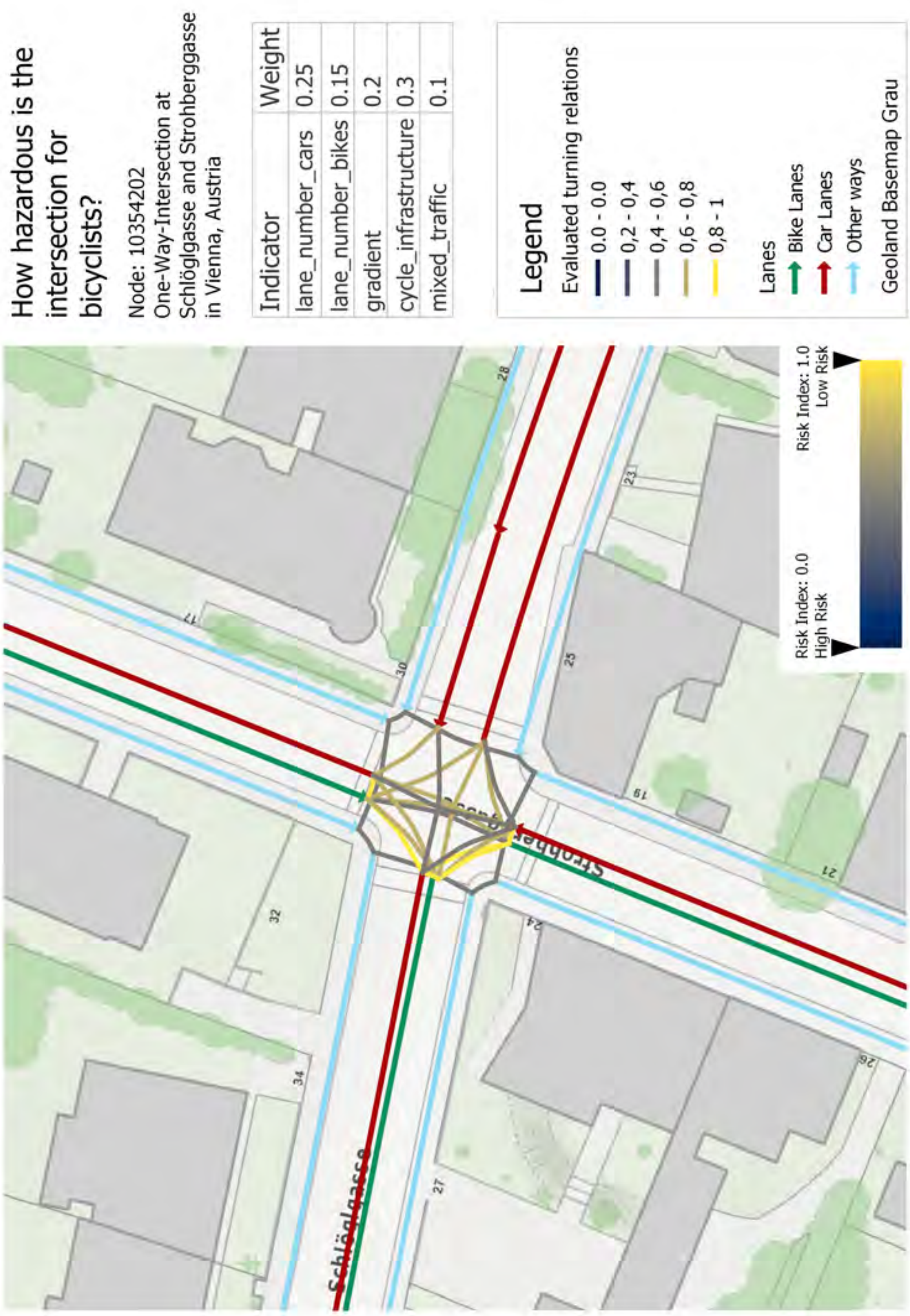

Figure 5.11: Intersection A - the turning relations are weighted with an emphasis on the factors that differ within an intersection as they depend on the turning relation.

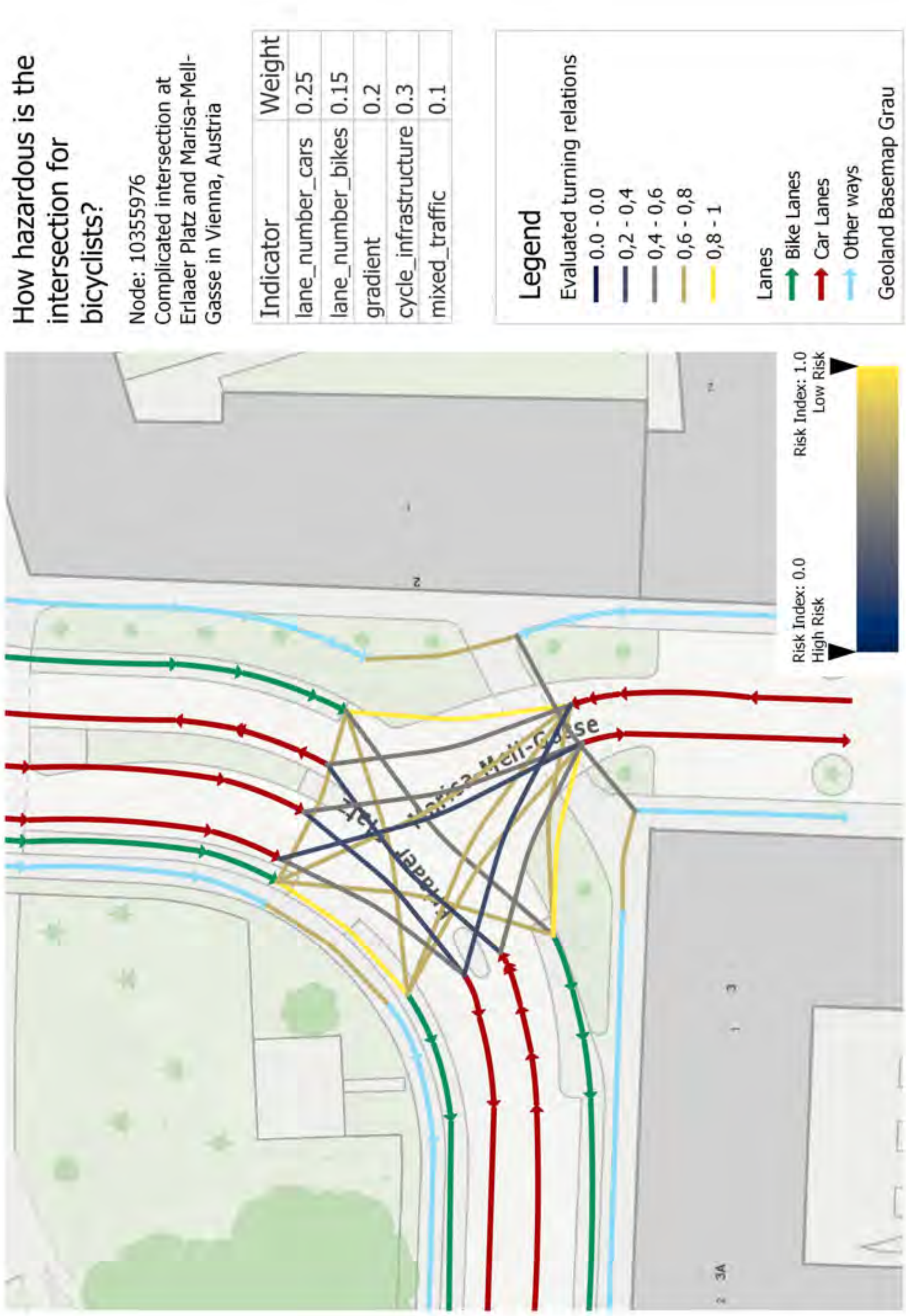

Figure 5.12: Intersection B - the turning relations are weighted with an emphasis on the local factors.

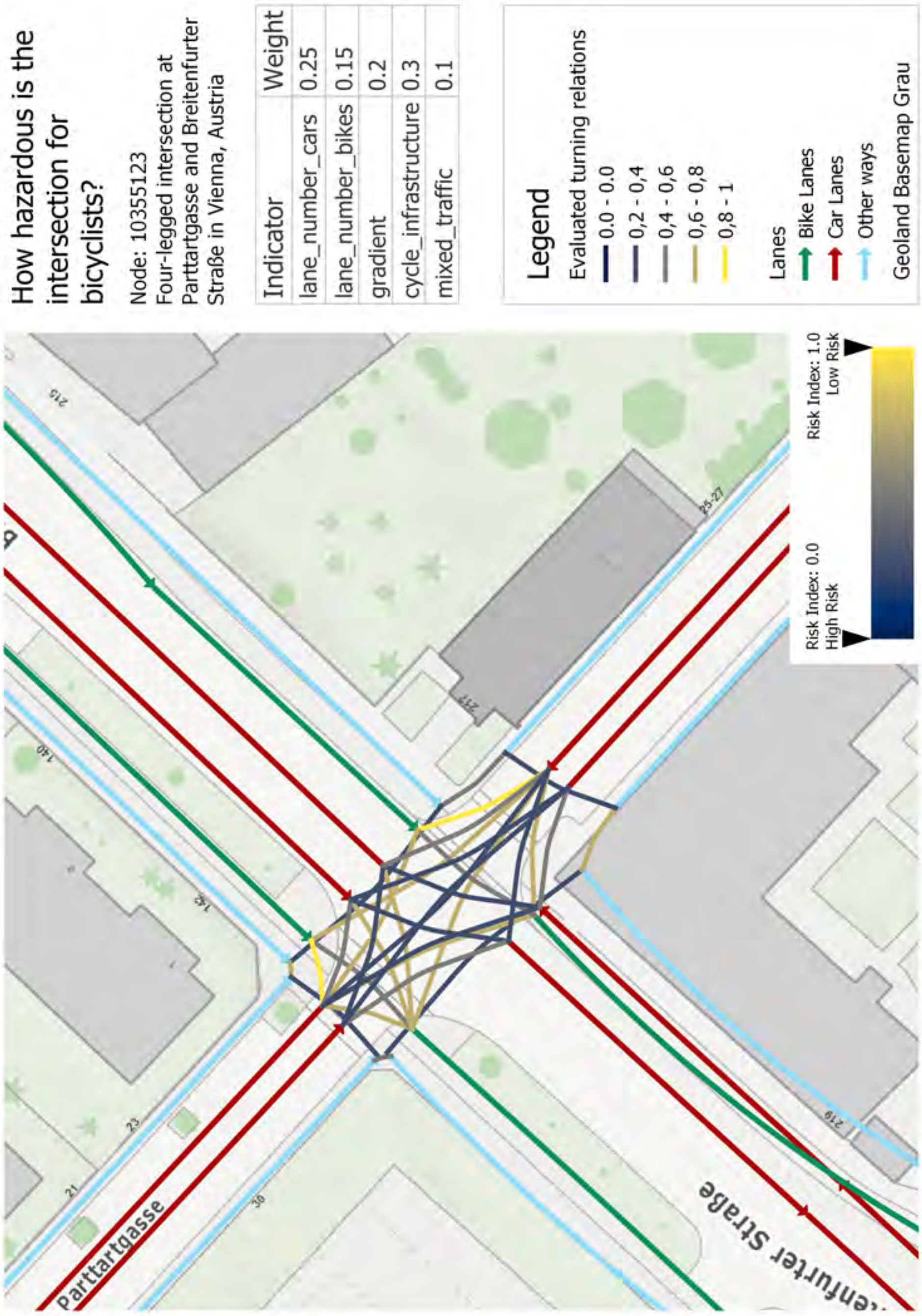

Figure 5.13: Intersection C - the turning relations are weighted with an emphasis on the local factors.

5.1.5 *Equal weighting of the factors*

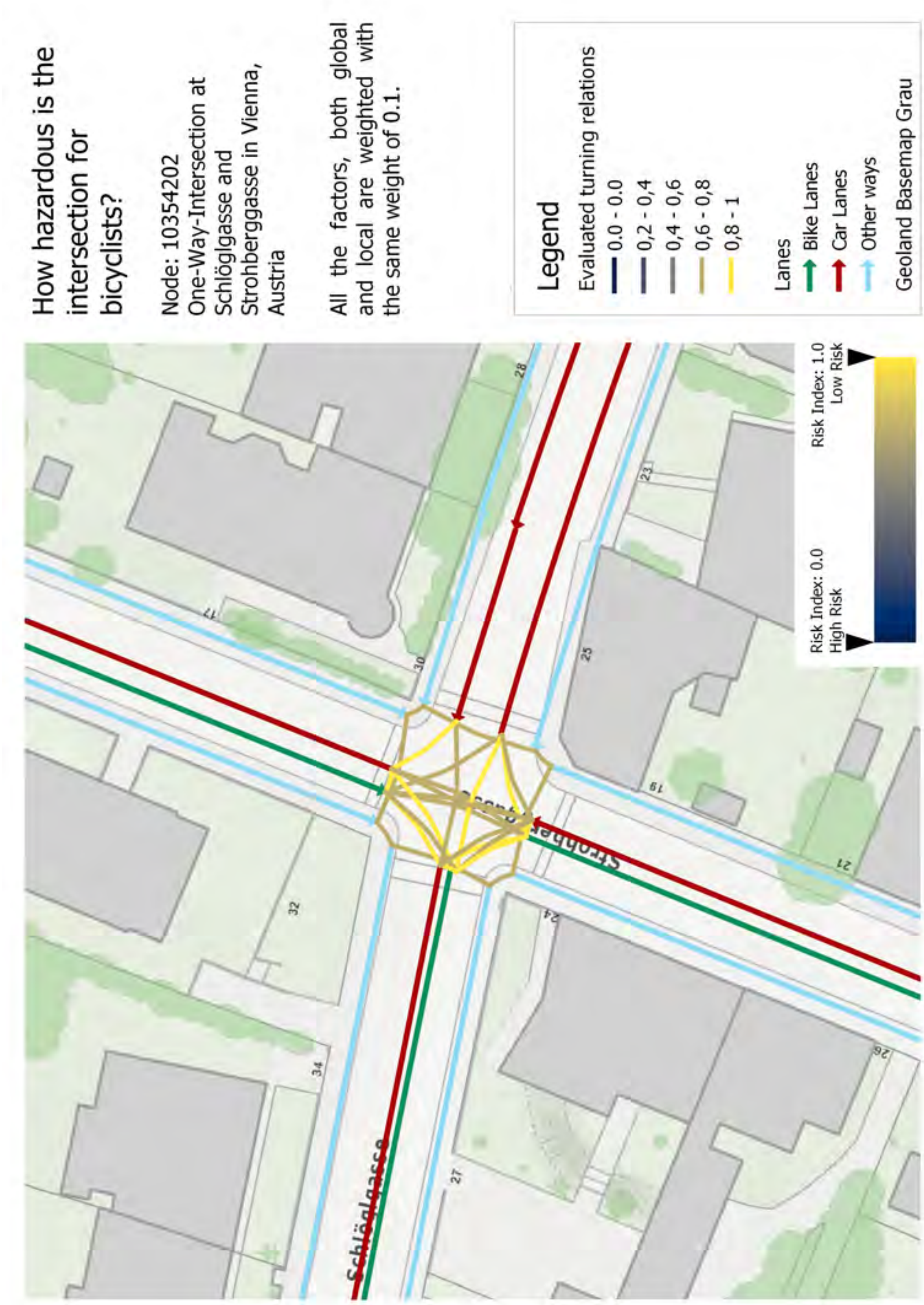

Figure 5.14: Intersection A - all factors are weighted evenly.

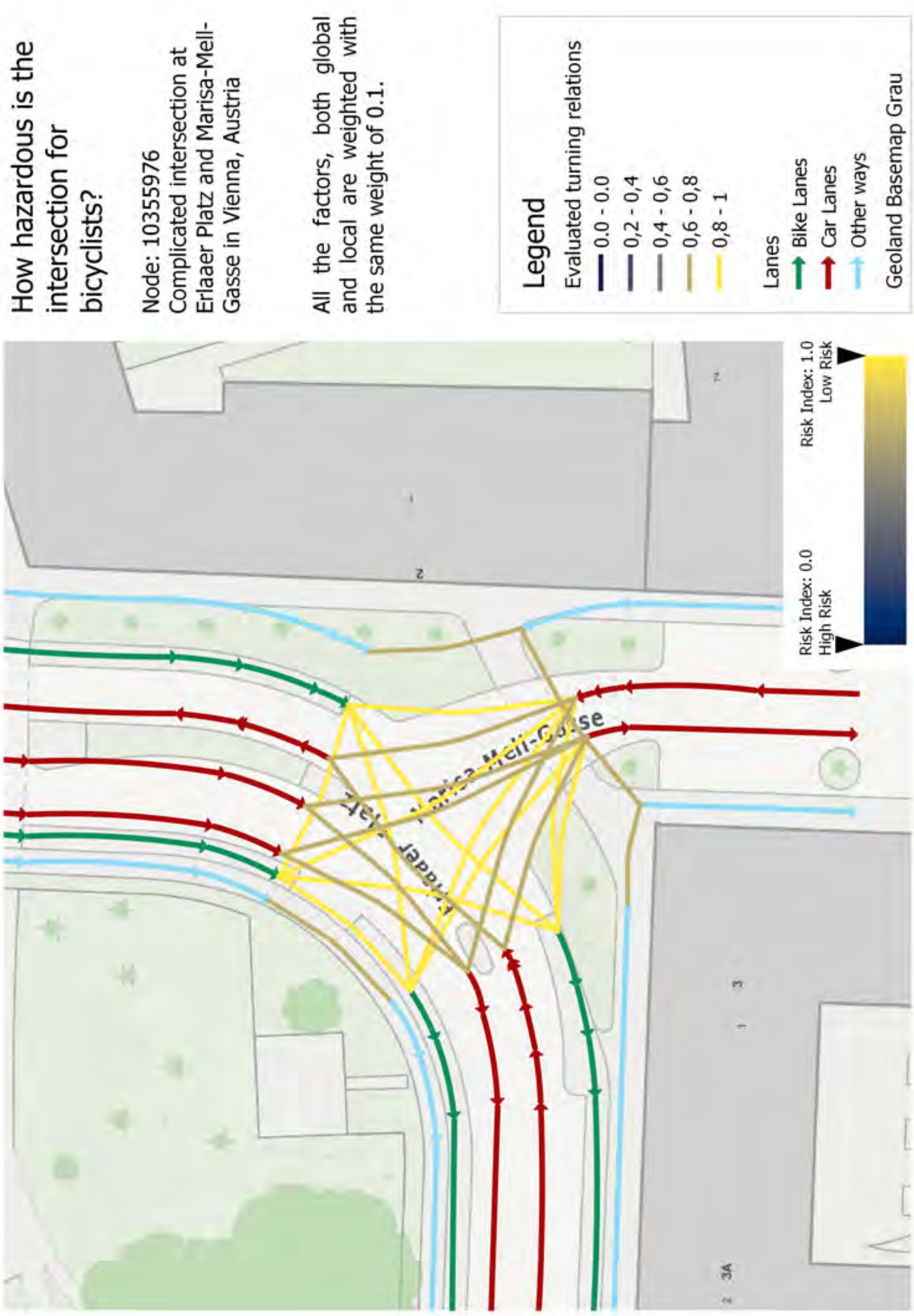

Figure 5.15: Intersection B - all factors are weighted evenly.

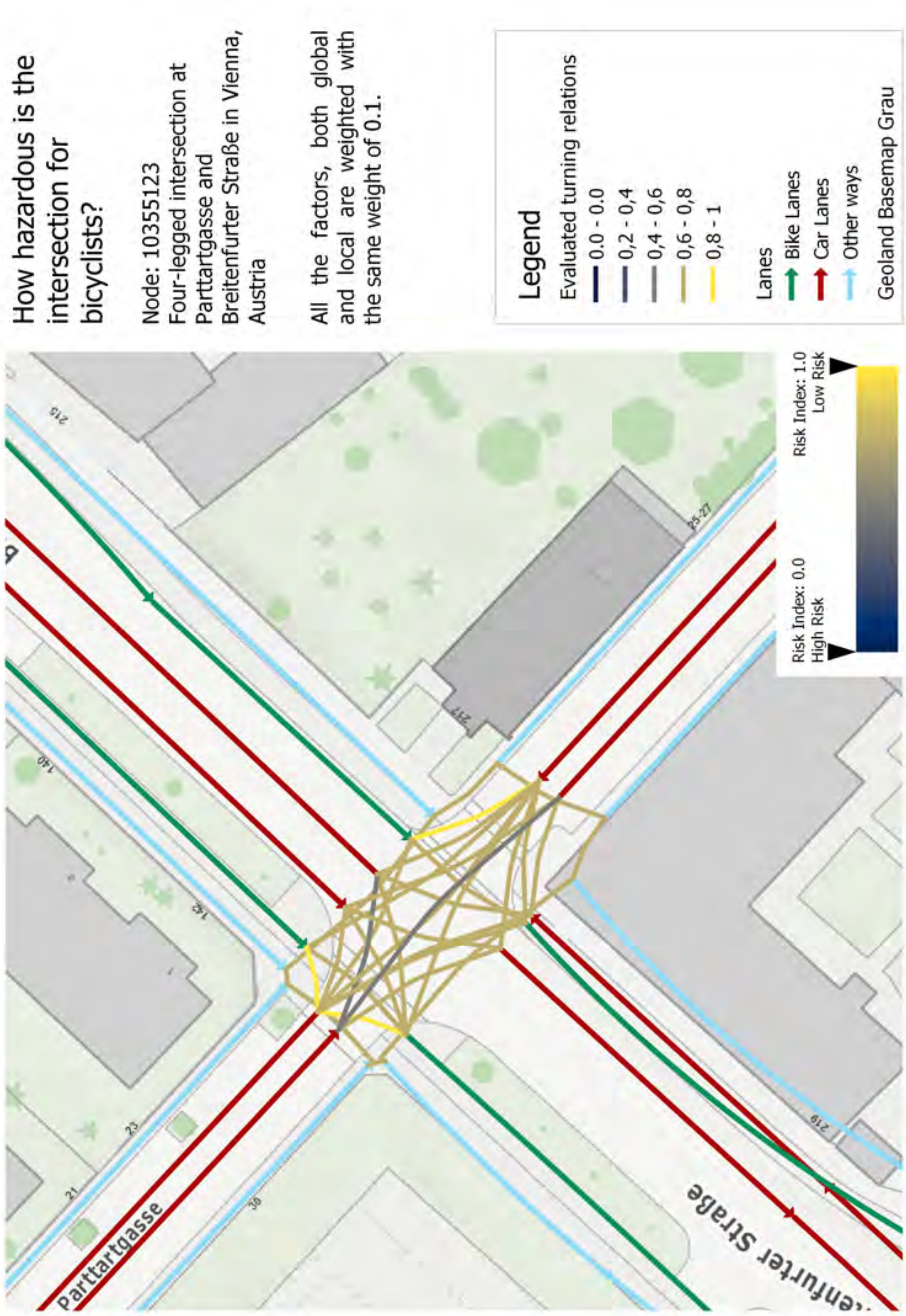

Figure 5.16: Intersection C - all factors are weighted evenly.

5.1.6 *Simulation of a traffic signal*

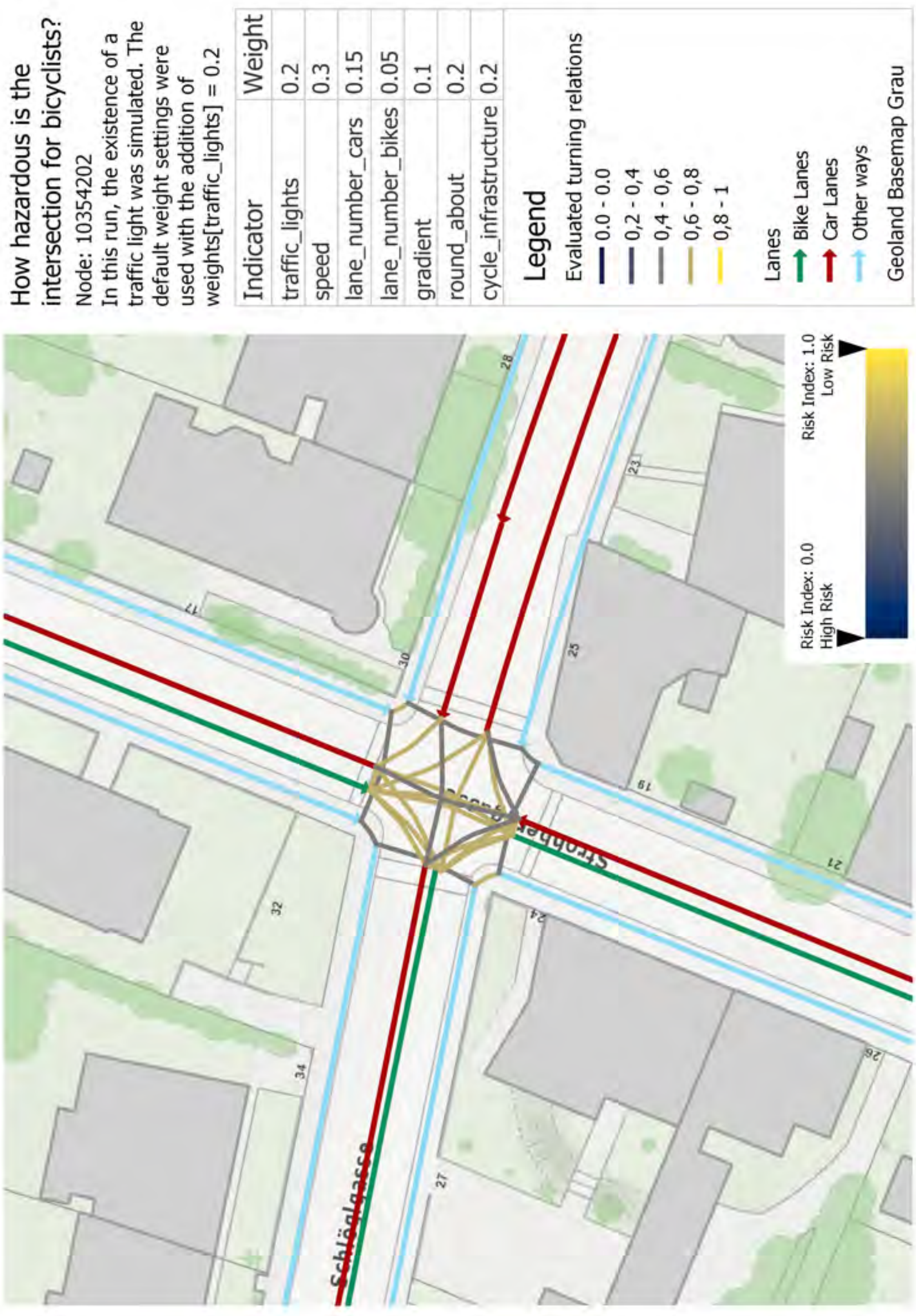

Figure 5.17: Intersection A - the table holding the data concerning the intersection's risk factors was manipulated. A traffic light was "inserted" and the weighting was adopted so the index calculation pays attention to the traffic light's "existence"

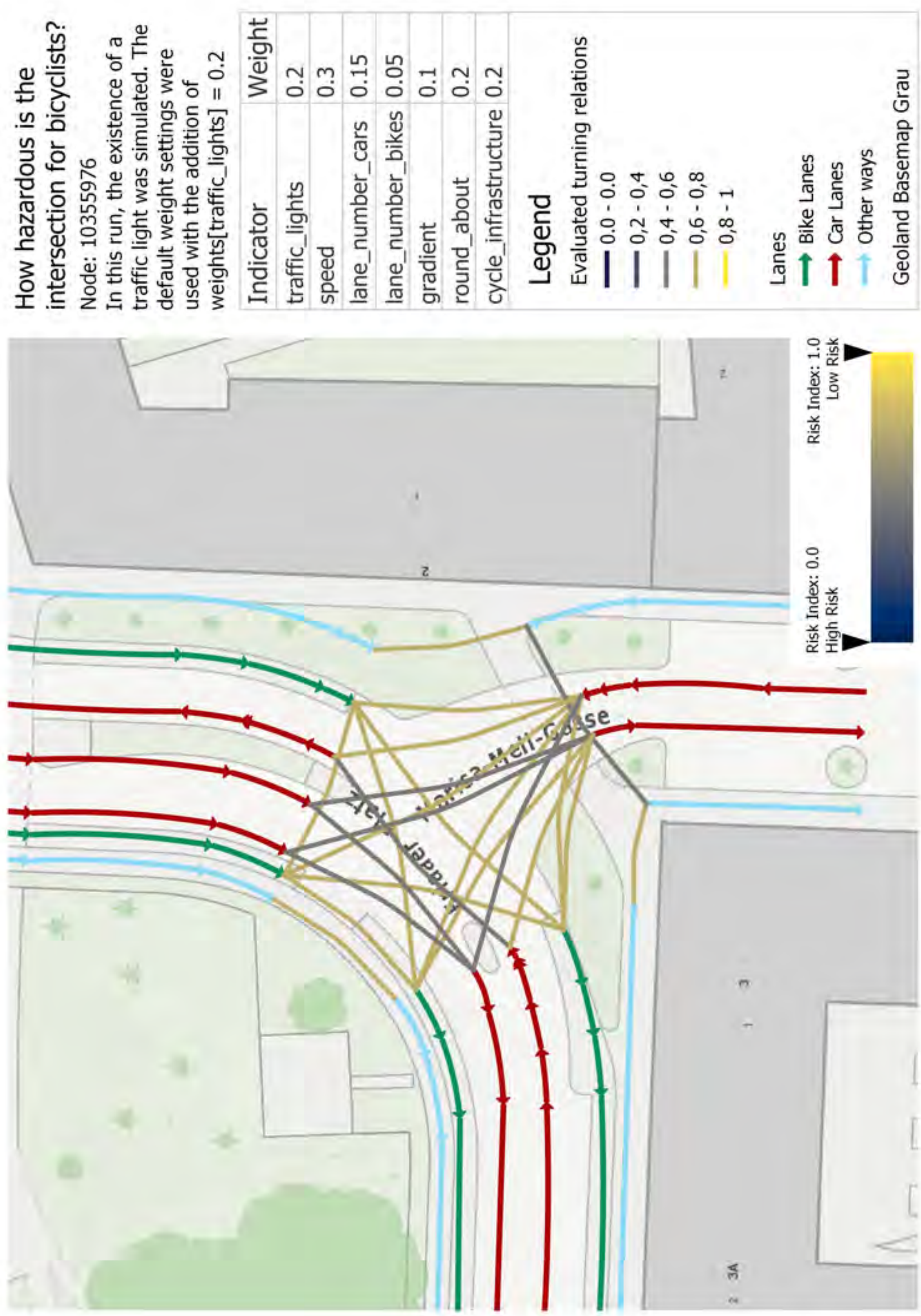

Figure 5.18: Intersection B - the existence of a traffic light was simulated, the weighting was adopted accordingly and the index calculation included the signalization.

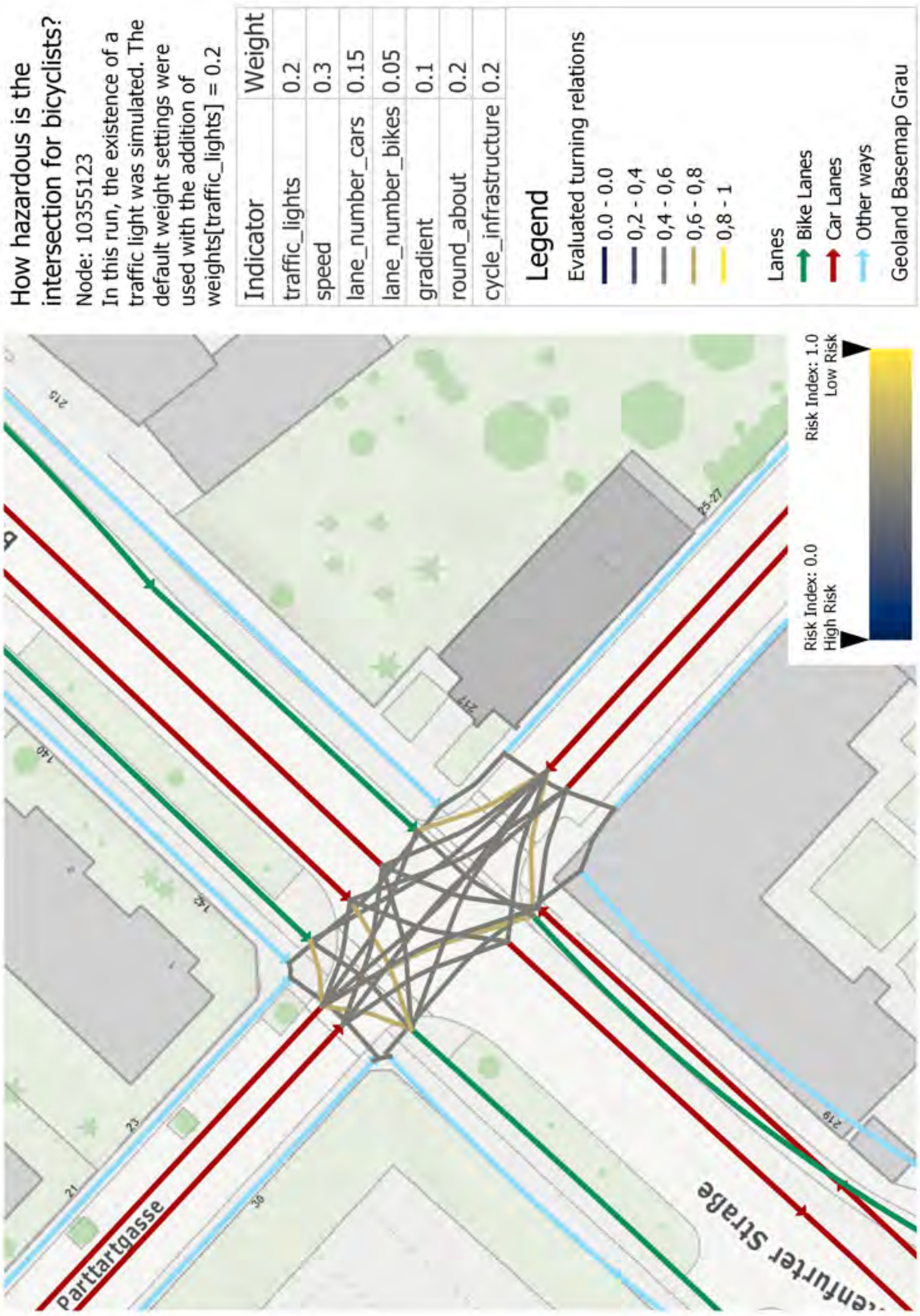

Figure 5.19: Intersection C - the existence of a traffic light was simulated, the weighting was adopted accordingly and the index calculation included the signalization.

5.1.7 *Comparison of speeed limits*

Note: The maximum speed of the intersections A and B are 30 km/h. Intersection C has a speed limit of 50 km/h

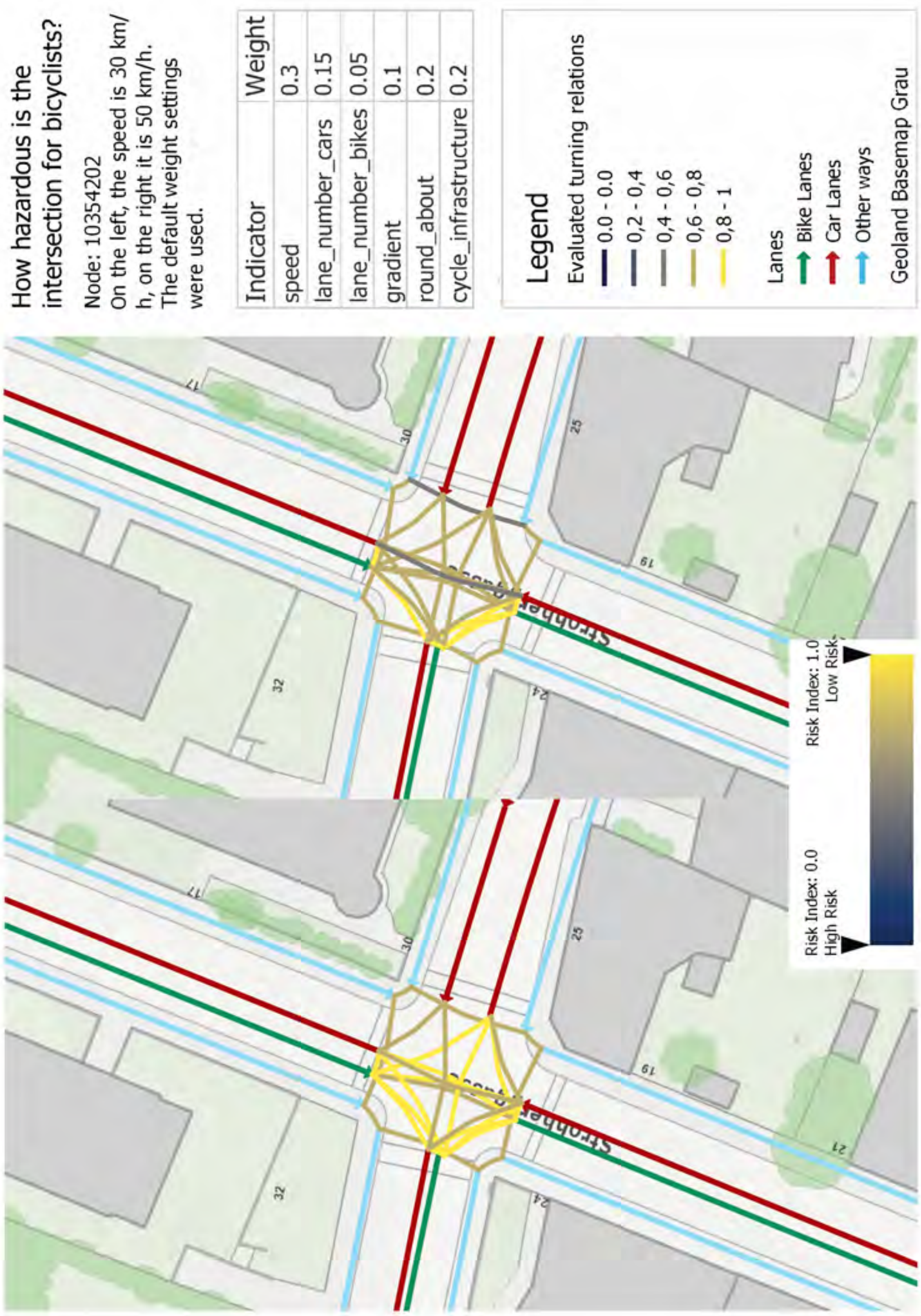

Figure 5.20: Intersection A - on the left the native speed of 30 km/h and on the right the simulated speed of 50 km/h both weighted by the default values.

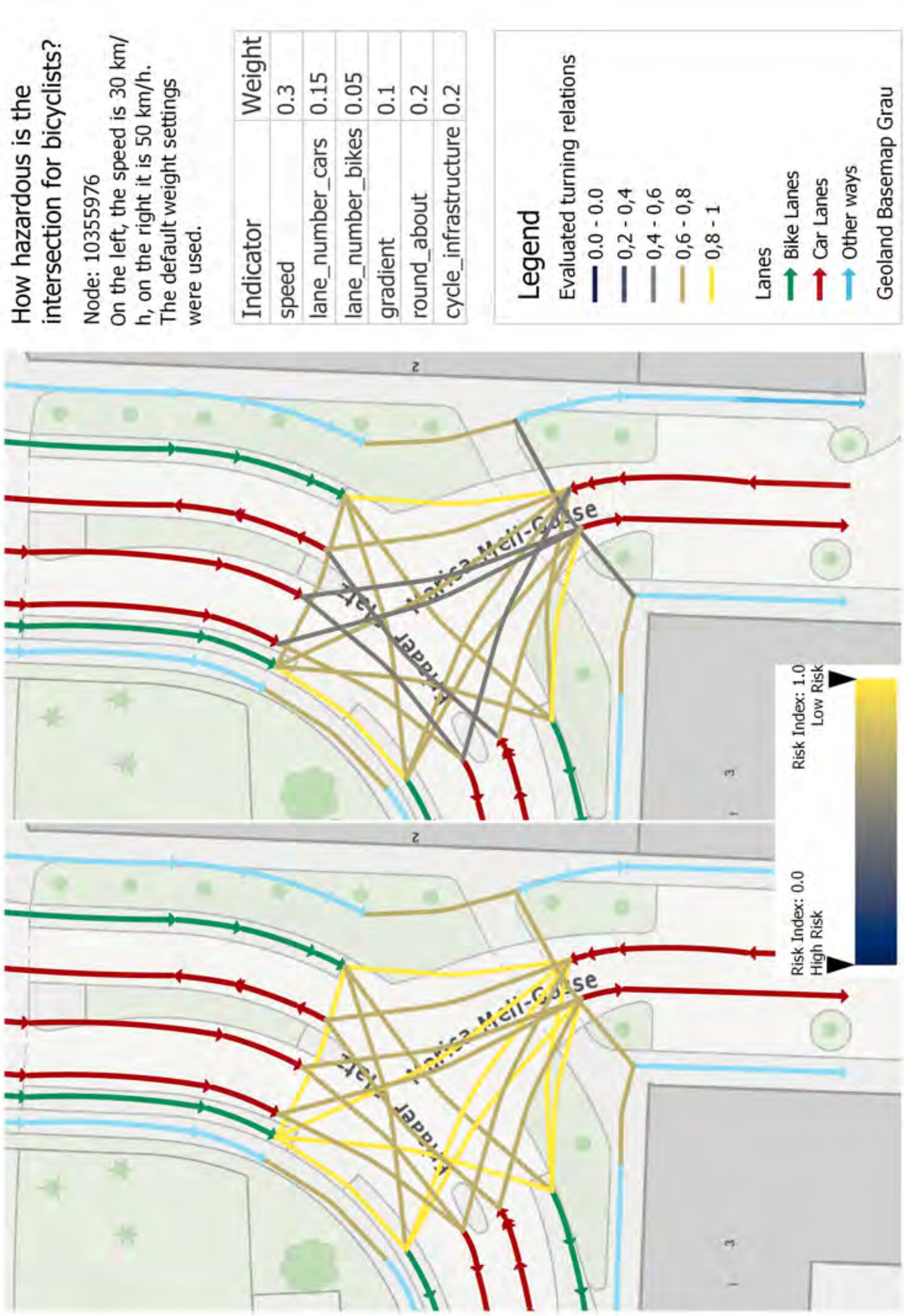

Figure 5.21: Intersection B - on the left the native speed of 30 km/h and on the right the simulated speed of 50 km/h both weighted by the default values.

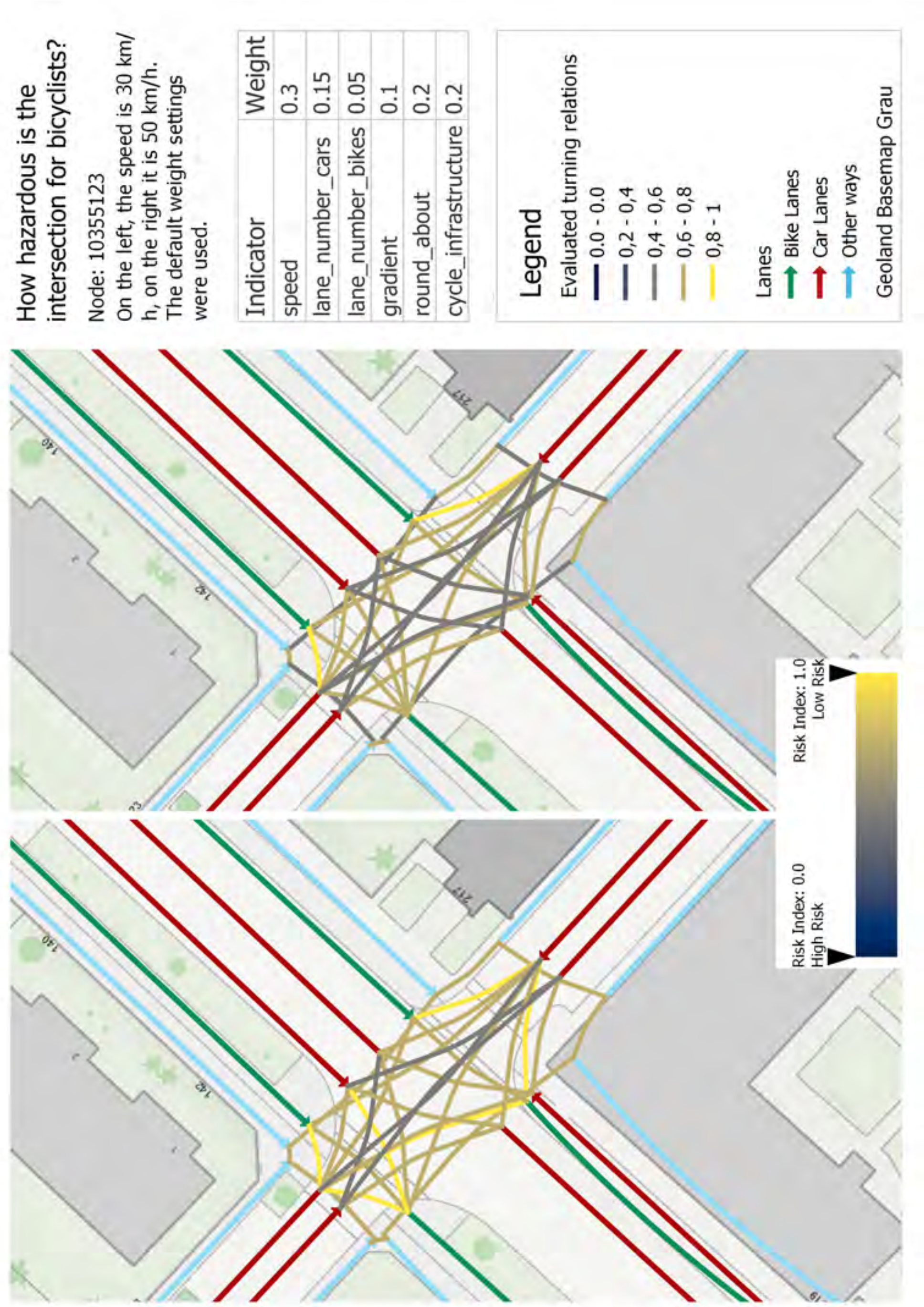

Figure 5.22: Intersection C - on the left the simulated speed of 30 km/h and on the right the native speed of 50 km/h both weighted by the default values.

Part III

DISCUSSION AND CONCLUSION

6 DISCUSSION AND OUTLOOK

This chapter explores the results presented in the Chapter 5 chapter. Those are map representations of the risk analysis of individual turning relations for bicyclists. The discussion regarding the data basis and the implementation of the model are conducted separately afterwards. In addition, the relevance of the work is discussed and finally the research questions are revisited and answered.

6.1 INTERPRETATION OF MAPS

In the previous Chapter 5 exemplary results are shown in maps. These are now explained and interpreted.

6.1.1 *Unweighted 1D lanes*

In the maps of Section 5.1.1 the 1D representation is shown. The focus of this representation is on the structure and layout of the intersections. The intersections are represented in lane-level resolution using turning relations instead of a node. And the edges and use strips have also been broken up and instead the lanes and paths are displayed separately. To still be able to distinguish the different types of use, lanes are categorized by their *basetype*: Bicycle infrastructure [1] is shown with the color green. Lanes for motor vehicles are displayed in red. And everything else, which are mostly pedestrian lanes, is drawn in blue. In order to visually differentiate turning relations and lanes, lighter shades were chosen for the turning relations.

In this development phase, special care was taken to ensure that the geometries are topologically correct and that turning relations and lanes associated with MV infrastructure are directed in the direction of travel. In addition, the bend in the turning relation, which was incorporated to bring the model closer to reality, is visible and resembles reality in most cases.

This representation makes it possible to identify points of conflict and to understand the traffic flows and turning maneuvers.

1 Basetypes that are considered to be part of the bicycle infrastructure: id in (2, 22, 23, 31, 33, 35, 36): Bikeway, Cyclist crossing, Protective path and cyclist crossing, Bike lane with adjacent sidewalk, Cycle lane, One-way cycle lane, Sidewalk and bike lane

6.1.2 *Unweighted 2D lanes*

In Section 5.1.2 the results are the 2D representations of the intersections. The development from the previous visualizations lies in the fact that the width of the lanes was included in the geometry. Thus, a two-dimensional representation of lanes and turning relations is formed. Additionally, parking lanes (purple) are integrated into this representation, as can be seen in Figure 5.6 and Figure 5.7. This can be particularly beneficial when exploring and evaluating segments.

The potential of this representation lies in the possibility to detect where the lanes cross and thereby be able identify risk zones. There are several methods for forming this type of intersection representation. What makes the method used in this work special is that it is an automated methodology based on open data. It is an inexpensive, effective, and fast way to work up an intersection two-dimensionally with lane-accuracy. And it is possible to employ it for continuing explorations of the road in GIS. In addition, the 2D representation has the potential to be further developed into the format of an HD map. Thus, it could also be used for autonomous driving.

6.1.3 *Risk Index with Default Values*

Design decisions

In Section 5.1.3, the turning relations are evaluated and assigned the risk index. For the presentation of these results the 1D representation was chosen. The reason for this is that lines are able to present the results more clearly than areas could. The disadvantage of the 1D version is that the visual information on width and overlap zones are lost. This causes the representation to decrease in informative value regarding the model and the risks of the turning relations. Nonetheless, this deficit was accepted in order to communicate the risk index of the turning relations precisely. Additionally, the 2D embodiment of the lanes is static and therefore does not contain any new information when the risk factors are changed. Consequently, the dynamic risk indices are given priority over the static spatial risk zones.

The color scheme of the lanes was left as it was. The colors of the turning relations, however, represent the risk index calculated for the turning relation. Dark blue are the relations that pose a high risk for cyclists, the yellow ones are considered safer for cyclists as they have a low risk value.

Weighting

The risk factors were weighted using the default weighting. Their values are shown to the right of the maps (Figure 5.8, Figure 5.9, Figure 5.10). The default weighting puts emphasis in descending order on the maxi-mum speed in the intersection, the presence of a traffic circle and bike infrastructure, the number of MV lanes crossed, and with what slope a turning relation is approached and how many bike lanes are crossed by the relation. Speed and the presence of a traffic circle are global factors that apply to the entire intersection. The others are local factors whose values may vary depending on the turning relation.

At this point the reader is again referred to Table 5.1, which lists basic information about the intersections under study.

What influenced the risk index?

Intersections A and B have a speed limit of 30 km/h, whereas that of intersection C is 50 km/h. This information is considered to be the decisive factor, which is why the risk at intersection C is generally rated higher than the other two junctions (see Figure 5.8 - Figure 5.10).
Since none of the intersections are part of a traffic circle, this factor has a positive impact on each of the three intersections in the evaluation.
It is also noticeable that the turning relations adjacent to bicycle infrastructure are rated better. This can be seen in Figure 5.8, where there appear to be very similar left turn actions, but they have been rated differently - due to their (non)affiliation with bicycle infrastructure.
The effect of crossing many other turning relations is well seen in Figure 5.10: The relations connecting car lanes to the north-west and those to the north-east have the same apparent conditions: they connect the same roads and are both car infrastructure. Nonetheless, the left-turn action from north-west to north-east is evaluated as riskier than the right-turn action. The reason is that the left-turn action crosses more other relations. Thus, exactly what was anticipated occurred: although the model does not know that right and left turn actions are involved, it evaluates the relations accordingly.

6.1.4 *Weighting of the local factors*

Choice of factors

In order to highlight the differences between the individual turning relations of an intersection, their risk index is calculated based only on the local factors. Slope, crossed other lanes, bicycle infrastructure and mixed traffic are included in the calculation. This means that any commonality between the relations of an intersection is excluded from the calculation.
For this reason, the full scale of 0 to 1 is exhausted, as general positive characteristics, such as not being a traffic circle, are not weighted.
In reality, this weighting would not be applied because it ignores important factors such as the speed. But it is suitable to emphasize the diversity of turning relations in comparison to a node. The relations differ and can be evaluated in a very nuanced way, although they originate from the same intersection.

Not every factor can / is displayed

However, these maps present a problem in terms of visualization: From the maps and the table (Table 5.1) not all information used for risk calculation can be obtained by the user: The slope of the lanes is not shown - the implementation would perhaps be possible via shading or isohypses. However, the differences in elevation are quite small, so the

representation might prove difficult.
It is also not possible to see whether mixed traffic is involved. Reorganizing the classification of the lane representation might solve this problem.

6.1.5 *Equal weighting of all factors*

In the maps of Section 5.1.5 all factors are weighted equally. It can be observed that the turning relations were evaluated more homogeneously compared to the maps of Section 5.1.4. Apart from Figure 5.16, where the risk index of the relations spans three classes, in the other two maps only the classes from 0.6 - 1 are occupied.

Why are the intersections evaluated similarly?

The consistently rather positive evaluations of all turning relations can be explained by some characteristics of the intersections under investigation: they are neither a traffic light intersection, nor are the intersections part of a roundabout. Moreover, the intersections are located in urban areas, there are no railroad lines in their vicinity, and no street classified as major road merges with any of the intersections. Thus, the intersections have positive characteristics that go unnoticed under the default values, but have a positive effect on the calculation of the index with the current weighting.

Why are the turning maneuvers evaluated differently?

The reason why the turning relations of an intersection differ in the risk index are the local factors. Section 5.1.5 shows not only equal weighting of the factors, but also what influence local variables have on the risk index. It can be clearly seen that an intersection should not be evaluated as a grand whole. Local circumstances, such as the type of turning action or whether bicycle infrastructure can be used, have an influence on individual turning relations that should not be underestimated.

It may be possible to evaluate an intersection as a single element, a node, in terms of its risk to bicyclists. However, the consideration of turning relations provides much more detailed information and makes it possible to evaluate an intersection in a more differentiated and precise manner.

6.1.6 *Simulation of a modified traffic measure*

In Section 5.1.6 the presence of a traffic light is simulated. Such a test is suitable to estimate the influence of a potential road traffic measure. Moreover, this example allows to observe the influence of an additional factor and to test the sensitivity of the model.

The weighting of the model is adjusted by assigning a weight of 0.2 to the factor *traffic_light*. The other factors are weighted according to the default weighting scheme. The results are shown in Figure 5.17 to Figure 5.19.

Not every change in the risk index is being displayed

Since the presence of a traffic light is treated as a global variable, it has approximately the same effect on each turning relation - depending on how much information is missing for a turning relation whose weight is then not counted. The impact of the traffic light is most noticeable when comparing default weights (Figure 5.8) with current weights (Figure 5.17). Traffic lights are classified as a risk for cyclists in the normalization step. Accordingly, signalized intersections are found to be more risky - where without the traffic lights the values rank between 0.4 and 1.0, with traffic lights they are between 0.4 and 0.8. Under these circumstances, generally negative effects of the classification can be noticed: Looking at the results table (*weighted_turnuses*), it can be seen that the values within a class differ by about 0.14 between the options "with traffic light" and "without traffic light". However, this is partly not shown because of the classification.

6.1.7 *Speed Comparisons*

What if the maximum speed limit was set to 30 km/h?

In Section 5.1.7 the effects of speed limits 30 km/h and 50 km/h are compared. These two figures were chosen because of a current debate initiated by researchers from the transport departments of the Universities of Vienna, BOKU, and Innsbruck (Aigner, 2023). There are calls for the speed limit to be restricted to 30 km/h in built-up areas. The rationale is, among other things, that the number and severity of accidents decreases with lowered speed.
This study also allows an analysis of how strongly the model responds to a change in a factor. Since speed is the most heavily weighted factor, the model can be expected to visibly adjust the risk indices.

Intersections A and B already have a speed limit of 30 km/h, accordingly a limit of 50 km/h is simulated for them. The opposite is true for intersection C. The results can be seen in Figure 5.20, Figure 5.21, and Figure 5.22.

As suspected, the change in the risk indices is clearly visible in all three maps: the values of the individual turning relations have each worsened or improved by about 0.07 (intersection C).
This means, on the one hand, that speed also has a major influence on the risk of an intersection according to the model. And on the other hand, that the model is sensitive enough to react to the change of a single parameter.

6.1.8 *Conclusion of map interpretation*

The model is able to evaluate each turning relation of an intersection with respect to its risk for cyclists. The factors used are from the infrastructure domain - environmental variables or driver-specific characteristics are not collected. Accordingly, this model specifically identifies infrastructure risk.

The weights used to calculate the risk indices can be adjusted by the user. Accordingly, the accuracy of the results is not discussed, but how sensitive the model is to changes in the weights or the data. Changes to weights are described in Section 6.1.3, Section 6.1.4, and Section 6.1.5; changes to data are performed in Section 6.1.6 and Section 6.1.7.
As noted in the referenced sections, the model is sensitive to both weight and data changes. The sensitivity of the model depends on the number of weighted variables. The fewer factors used in the risk calculation, the more sensitive the model is to changes in weight or factor values.
Because the risk calculation is a linear calculation in which no dependencies have been integrated, a change in a global weight has an equivalent effect on the risk index of each turning relation.

Using global AND local variables is important

Furthermore, the inclusion of both local and global factors has the effect that the resulting risk indices vary within a certain range and do not use the complete scale from zero to one.
In contrast, the risk indices of the turning relations vary greatly in the map that explores the weighting of local factors only (Section 6.1.4). The reason for this is that with the absence of global factors, the common characteristics of the relations of an intersection are not considered. This leads to the conclusion that it is important to include both global and local properties in the risk calculation.

6.2 DISCUSSION OF DATA

As data basis mainly the GIP is being used, for data on traffic lights two layers of the OGD-platform of Austria (see Section 4.1.1) and concerning the gradients a DTM of the city of Vienna was included. Thus, all data come from sources published as open government data.

6.2.1 *Signalization*

Intersection signaling data was aggregated from two layers because it is provided in separate datasets in the original data based on whether it is equipped with acoustic detection. Since the structures of the data sets are the same, there was no loss of information during aggregation.

The data are collected and maintained by the City of Vienna and are accordingly only available in the province of Vienna - not Austria-wide. One point in the layer corresponds to one traffic light - i.e. there is one tuple per signalized intersection in the data set. Thus, the geometric accuracy can be estimated to about 10 m. On the one hand, this resolution has the advantage that an intersection can be classified fairly accurately as "signalized" or "non-signalized". On the other hand, it is also very uninformative with respect to the questions of which turning relation is directly affected by a traffic signal and what type of signal strategy it is - information that would be valuable for risk analysis (see Section 2.1.2).

Because the layers were included statically, the database must be updated regularly in the model. This is because changes made in reality should also be adjusted in the model as quickly as possible to keep the model close to reality. The original data itself is updated on a weekly basis.

One can argue that OSM Data might have served better in this case: They have higher geometric resolution, can compete in temporal resolution and completeness with the OGD Data and are more detailed in attribution. However, the (OSM) attributes have the disadvantage of being non-uniform and inconsistent. In the end, it was decided to use the OGD data, as it is easier to handle and will suffice for my purposes for now. Should the AOI be expanded, the traffic light data source will need to be adapted to this circumstance and reconsideration should be given to whether OpenStreetMap (OSM) data shall be used.

6.2.2 *Digital Terrain Model*

The DTM is used to determine the slope at which an edge approaches the node under investigation. The data source used is the Digital Terrain Model of the city of Vienna. It is a raster format that is updated quarterly and has a geometric resolution of 1 m. The DTM is also included statically, which is highly useful in this case, since it involves large amounts of data. If this data were to be queried dynamically, the consequence would be to slow down the entire workflow. In addition, the height, especially in the road sector, does not change very often. Accordingly, regular updating of this data is only necessary at long intervals.

The alternative to the DTM raster format would have been an triangulated irregular network (TIN) of the same area. It was decided against this option to avoid the step of interpolation. In addition, the breaklines are of little interest to the model, whereas a raster with a resolution of 1 m averaged the height of the points in that 1-meter square.

6.2.3 *Data Basis GIP*

The GIP is the basic framework on which the model is built - both in data preparation and risk analysis. The GIP is maintained by several stakeholders (federal states, Autobahnen- und Schnellstraßen-Finanzierungs-Aktiengesellschaft (ASFINAG), Österreichischen Bundesbahnen (OEBB)), but follows a detailed standard on how to enter the infrastructure data. The data is continuously maintained and updated in a decentralized way (by the above mentioned stakeholders) and merged every two months to cover the whole of Austria with up-to-date data. The geometric resolution of the geodata of the GIP is emphasized less than the topological correctness. The accuracy of acquisition should allow a reasonable representation at a scale of 1:2000 (ÖVDAT, 2021, p. 25). This has corresponding implications for the model, whose geometry is derived from the GIP and whose accuracy is located in the meter range.

Minimum Standards

The GIP describes in its standard description (ÖVDAT, 2021) how the data are structured. In this context, the minimum standards are what must be recorded as a bare minimum in the GIP. Accordingly, the factors that are below the minimum standards are consistently present. This cannot be guaranteed for the other factors. For this very reason, Vienna was chosen for an AOI because the city is known to place great emphasis on the maintenance of the GIP.

The GIP for the data preparation

Regarding the usage strips, the minimum standard is to record the roadway and the railroad track - the recording of bicycle and pedestrian paths, etc. is part of the standards. This means that the GIP most likely does not capture every real usage strip, as it is not an immutable obligation to enter other than roadways and railroads (ÖVDAT, 2021, p. 56). Thus, the model cannot be relied upon to include all lanes that the user deems relevant, since the GIP itself focuses on MVs.

Using the GIP for risk modeling

To carry out the risk analysis, information from the GIP is used as well. The factors employed by the model are twofold - calculated and adopted. The calculated factors were derived from the geometries, the adopted factors are obtained directly from the GIP. Adopted factors, which are among the minimum standards, are railroad lines, speed, higher-ranking roads and roundabouts. The attributes *urban*, *intersection_legs*, *mixed_traffic*, and *cycling_infrastructure* are not minimum standards. To avoid missing data, the last three are being double-checked: if the first method of data collection does not yield results, at least a second one will take effect. This hopefully covers most contingencies and the model can process as much data as possible provided by the GIP.
For the factor *urban* no backup is implemented, because on the one hand there is no link in the AOI that does not have a value for this attribute. On the other hand, implementing this safeguard would mean having to integrate another external data source. This is additional work especially in the maintenance of the model.

Why the GIP was used.

The GIP is not the only provider of transportation infrastructure data, OSM provides comparable data. For this reason, the question arose whether to use GIP or OSM for this model. Ultimately, GIP was chosen because it is government-managed. It is maintained by professionals who operate according to a standard catalog. It is also updated periodically and is topologically verified. Its downsides are that it is partly incomplete because not everything is covered and because it cannot cover every contingency. In addition, its update cycle is not as agile as OSM and it only contains data on Austria, which greatly limits the application of the model in other countries.

When using the GIP, it must be ensured that it is not only updated on the content level. Changes and adjustments in its structure may also occur. This has the consequence that the model must be adapted. However, this also allows the model to "grow" with the GIP, and further developments always have the potential to bring progress. New developments could simplify the workflow or bring in additional information.

6.3 DISCUSSION OF METHODS

6.3.1 *Data Preparation*

In the chapter Chapter 3 the capabilities of the model to generate one- and two-dimensional representations of intersections automatically based on the geodata of the GIP were already discussed. In the following, the limitations and weaknesses of the model are discussed in more detail.

Application spectrum of the tool

Currently only one node is processed per run

The possible applications are currently limited in several respects: First, there is the consideration of the number of nodes that can be processed. The workflow is set up in such a way that only one intersection can be passed to the functions. The results of the functions are stored in tables until the next function uses them. It is automatically assumed that the tables only contain the data for one intersection. If this was not the case, errors would occur. This restriction is very helpful for the development of the model. However, if one wants to process an entire network, adjustments will have to be made.

Intersections without car lanes cause problems

This work focuses on risk modeling of BMV-collisions at intersections. Accordingly, the focus of the data preparation was to automatically decompose intersections into their turning relations where both MVs and bicycles enter. In most cases, the preparation of such a classical intersection goes smoothly. However, the scripts are not set up for this case that there are no car lanes at the intersection. The *Bike2CAV* project focuses on the interaction of motorists and cyclists, so this limitation plays only a minor role for the project.

Correctness, consistency, and completeness issues.

Geometric accuracy of turning relations

The geometric accuracy of the lanes is estimated to be approximately 1-3 meters. This inaccuracy comes from the fact that the data base GIP was partly digitized with aerial photos only and puts more emphasis on correct topology than geometry. Especially at intersection C in Figure 5.4, where car and bike lane intersect due to the very roughly drawn GIP in the southwest, it can be seen that the correct location was neglected. From the comparison with the base map it can be seen that the geometries of the GIP do not correspond to reality.
The curvature of the turning relations, which tie to car lanes, are created automatically. Partly the result is close to reality, partly it clearly misses it, as can be seen in Figure 5.4, where two turning relations visibly curve towards the center of the intersection, although this does not correspond to their actual course.

Assumption: the number of lanes refers to car lanes

.

The number of lanes embodied by a usage strip is not directly linked to them in the GIP. This information is obtained from another table, but this table relates this information to the entire link, not to a specific usage strip. The assumption was made that the number of lanes *always* refers to usage strips of cars. In several tests this assumption has been confirmed and in the vast majority of cases the results are very close to reality and useful for further use.
There could be exceptions, though: If the link refers to a multi-lane bike lane, valuable information about the bike infrastructure is lost. This is because the model does not consider that it could be a bike lane that has left and right lanes.

In addition, the GIP has physically separated car lanes entered as individual usage strips. The model would determine the number of lanes along the associated link, as described above, and apply that value to each usage strip of type *car*. So the result would be x times as many lanes along the link as there are individual (car) usage strips associated with the link.

Assumption regarding multi-lane one-way street

While developing the workflow, it was noticed that there exist multi-lane one-way streets in Vienna. The problem associated with this is that the usage strip is usually located between the outbound and inbound directions. Since the lanes all go in the same direction, there is no such centerline. The GIP has not yet described this special case in its standard description. Therefore, the assumption was made that the usage strip is drawn in the geometric center of the road, and when compared with aerial photographs, the assumption yields useful results. However, due to the fact that in the GIP is not officially regulated, alternative approaches, depending on the surveyor, cannot be excluded and these segments should be considered with care.

Parking lanes lead to contradictory information on width

To correctly represent two-dimensional segments, parking lanes are subtracted from travel lanes. However, this results in the width of the lane, which is also attributively recorded in the table, no longer matching the geometry properly. In this case, the minimum width is passed as an attribute instead of the average width. However, the lane is no longer the same width everywhere due to the cutting. As a result, there is no longer a correct, uniform width, only an approximation that fits appropriately for much of the length.

---

completeness

As in any model, not all real-world conditions could be brought into this one. In the case of geometric data preparation, emphasis was placed on infrastructure associated with cars, bicyclists, and pedestrians. The modeling of other means of transport was omitted as far as possible. Also, more special occurrences such as tunnels, bridges, or underpasses are not treated separately in the model. There are several reasons for these choices: The model is limited to cars, bicyclists, and pedestrian infrastructure to avoid making it more complicated than it needs to be for its dedicated purposes. In addition, tunnels, bridges, etc. are extremely rarely part of intersections and are not identified as risk factors (associated with intersections) in the literature.

consistency

Because of the automated data preparation, the results are extremely consistent. However, where inconsistent action had to be taken are the coordinate reference system (CRS) into which the geometry columns are mapped:

- the original data is provided in EPSG 4326. This is not changed to keep the updating of the data straightforward.
- the tables used in the model use EPSG 31259. The reason for this is that this CRS can be used to calculate in meters without having to transform.
- the geometry column of the result table *weighted_turnuses* uses EPSG 3857. This was chosen to be able to integrate the layer easily into web applications.

Although the different CRS can be explained and justified, it is against the rules of user-friendliness to use three different CRS in one project.

Structure of the intersection plateau

The locality that the work focuses on is the intersection plateau. In this work, an intersection is defined based on the GIP such that it begins where the usage strips end. Accordingly, the intersection plateau is the area that can be spanned between the ends of the usage strips.

To prepare the intersection plateau, a one-dimensional (and two-dimensional) version of the intersection is formed from a zero-dimensional node in the data preparation step. The lines of the one-dimensional representation resemble edges connecting the lanes. Thus, the model is similar to a graph model: both the lanes and the turning relations could be interpreted as edges. What is missing are the graphs that connect the edges.

It was decided against this option because it is not useful in this model, as it is focused on risk analysis. On the map, it could negatively affect the meaningfulness of the model. However, if the model is to be used for routing, adding these nodes could be beneficial.

Another reason why the nodes were not implemented in the current state of the model is that the model is already coherent, both topologically and attributively.
The lanes have the information from which node they come and to which they lead. And the turning relations possess attributes that refer

to the lanes they link. So the elements of the model refer to each other and are linked, leaving no doubt about the structure of the intersection. In topological-geometric terms, the turning relations and the lanes are linked, since their start and end points are identical when they are connected.

6.3.2 *Data Modeling*

In the risk modeling step, information about an intersection and its turning relations is obtained, processed, weighted, and finally evaluated by means of a risk index.
Risk modeling works almost exclusively without problems. The reason for this is that the data collection has been double-checked in case of doubt (see Section 6.2.3). In addition, the collected values are not dependent on each other, so that one error does not trigger any others.

General Remarks

Various literature sources

In order to find relevant risk factors for cyclists, a literature review was conducted in advance. For this purpose, different sources were consulted, which in turn used different survey methods for their findings. This has the advantage that the works confirm each other and give more backbone to the factors and their classification (positive or negative influence on the risk index).
However, the sources used were not assessed for validity and for the correctness of their experimental design. For example, for before and after comparisons regarding a factor, such as a traffic signal, it is possible that the change also attracted more road users.
In addition, the literature review ignored the fact that the papers were from different countries. Researchers often use the infrastructure in their region/country as AOI. The road space of different countries can vary greatly, which in turn can affect how much risk a factor poses. The extent to which road space is similar across countries and where there are general differences was not given attention in this paper. Although it is an interesting issue, it was not explored further as it is ultimately a research question in its own right and would have been beyond the scope.

Accepted inaccuracies

Another inaccuracy that was accepted is the simplification of the model regarding factors: For example, for signalizations the signal strategy matters (Manirul Islam et al., 2022), for bike infrastructure it is beneficial to know what type of bike lane it is, and regarding roundabouts the construction is of interest and the integration of the bike infrastructure (Daniels et al., 2008). The model is generalizing in these and other respects. Remedies could be improved by additional, more detailed data. But more precise information from the literature would also help to process conditions in more detail and to evaluate factors in a more differentiated way.

How certain can one be of the factors?

As mentioned earlier, the factors used in the model are drawn from other work. Some of these factors were found to be significant based on before and after comparisons. In these cases, the comparison is made between how many accidents occurred before a measure was implemented and how many occurred after the measure was implemented. The measures studied are often the construction of a roundabout or traffic signals. This type of comparison tends to find an increased number of accidents after the measure has been implemented. However, the possibility that a measure makes an intersection more attractive to road users causing them to use the intersection more should be considered. Because often, as is the case for this work, no figures on road users are available, only absolute values can be compared, although it is the percentage of road users injured that is of most interest.
In work that argues perceived risk or perceived stress at an intersection, it is important to remember that these are subjective perceptions. What is quantified in numbers is at best the greatest commonality.

Discussing specific factors

"Urban" is dependent on definition

In the model, an intersection that is located in an urban area is identified as being less risky for bicyclists. However, the question arises as to what "urban" actually means. In the GIP it means populated areas, but in some literature source it means inner-city areas. There is a lack of a uniform definition to be able to evaluate the factor unerringly. A possible solution would be to implement more differentiated gradations of the factor. This would allow to distinguish the inner-city location from the outskirts and to evaluate it differently.

Traffic lights are difficult to classify

The majority of sources classify a traffic signal as a risk factor. However, a traffic signal is a classic example of the impact of measures just discussed: Due to the traffic light, the intersection may be perceived as a better alternative and therefore traffic volume could be higher there - which of course would increase the number of accidents. In addition, traffic lights are generally placed where busy areas are. Accordingly, it is questionable how it should be classified in my model. Ultimately, the factor was classified as a risk factor because most of the literature reviewed identified it as a significant hazard for cyclists.

Not all roundabouts might be recognized

In the GIP, roundabouts must be included according to the minimum standard only from a roundabout roadway of >= 50 m. This could lead to the problem that smaller roundabouts are not recorded and the model cannot include this factor for this reason, and calculates the risk index with incorrect information. This problem could be solved by including another data source whose minimum standard is higher.

Calculating the gradient takes time

To calculate the factor *gradient*, the height of the start and end point of an edge are queried. This is done by accessing the DTM, which was imported locally into the database. This query slows down the entire workflow: actually, the entire workflow (without *gradient*) can be run in under a second. With this factor, it is about 24 seconds. For this comparison, *intersection A* was used. The solution to the performance problem could be in greater expertise in working with raster data in databases. With more proficiency, it is certainly possible to minimize the time required.

Functionality of the risk modeling

More effective data collection possible?

Although data modeling runs mostly smoothly, the question arises whether it could be implemented more efficiently: The script *get_risk_factors(node_id)* queries the information for all factors, regardless of whether that factor is used and weighted at all or not. Especially if there is more than one intersection, it has a positive effect on the runtime of the scripts if fewer factors have to be retrieved. However, this change would also require that whenever the weighting is adjusted to include another factor in the index calculation, the information on that factor must be obtained and the values normalized. In contrast, with the previous full data collection, only the calculation needs to be redone. Thus, the agile method would have a lower time cost, while the more static method has the advantage of being stable and consistent in execution. Which method is more advantageous should be decided depending on the intended use and the available resources.

.

In the normalization step, evaluation is performed

Although the weighting and evaluation of the factors is only carried out in the last step, a kind of evaluation already takes place in the normalization step. This is because the personal impressions gained in the literature research are also incorporated here. A user who does not share my assessment and evaluates the situation differently must therefore already intervene in the normalization step to evaluate a factor differently than it was done in the model.

6.4 RELEVANCE

This section explains how the developed model can be of relevance and ranks its value.

The model can be of use in planning and routing

The results produced by the model are, first, one- and two-dimensional representations of the intersection and, second, an index that conveys the risk of each turning relation to bicyclists. The one-dimensional representation of the intersection is paired with the risk index. This feature can find application in bicycle routing, as it allows bicyclists to avoid hazardous intersections, knowing in advance of the risk.
In addition, the model can be used in roadway planning and improvement: By identifying particularly high-risk intersections and turning relations, improvements can be undertaken. The model can also be used to simulate the impact of new measures.

In the two-dimensional representation, coarse risk zones can be identified, and through further development (or adoption of the data preparation approach of this model), this model could tie into HD maps and eventually find application in car routing.

The advantage of the methodology used in this work is that it builds on GIP. This makes it possible to apply the methodology to all areas covered by the GIP, i.e. Austria. Unlike other methods with similar objectives, this one is very efficient as it is based on already existing open government data and does not require a separate data collection and is also extremely time efficient.

On-going use?

Of course, the question also arises whether this work will be used further or whether it will end with the submission of this thesis. The Mobility Lab Salzburg plans to continue research approaches in this direction. In addition, recently mainly the edges have been researched. There are currently already some approaches to these and in order to cover the majority of the road space, the nodes are now of great interest, so this work may also arouse the curiosity of other research institution in the field of transportation research.
Furthermore, the papers and code material will be published under the MIT license to counteract oblivion.

Infrastructure is only part of the risk

As was discussed in Section 2.1, infrastructure exerts only a partial influence on road safety. Many other factors, such as other road users, their vehicle, and traffic behavior, factor into the risk a bicyclist faces on the road. Nonetheless, it is important to explore and quantify the infrastructure. This is because it brings researchers one step closer to identifying and minimizing the hazards of intersections.

Cyclist don't stick to cycling infrastructure

As research (Breum, Kostic, & Szell, 2022) has shown, bicyclists do not always adhere to the routes provided for them because those do not always promote uncomplicated, uninterrupted travel. If cyclists do not adhere to routes, what is the point of evaluating them at all? For one thing, the model's turning relations cover a wide range of options - including those that are not part of the cycling infrastructure. For another, the objective. should be to provide infrastructure that is accepted and used by bicyclists.
Ultimately, the goal is to create safe infrastructure. But if it still doesn't appeal to users and satisfy their needs, it's not a good solution either. Cyclists are not exclusively concerned with their safety, but also with matters such as time spent and comfort.

6.5 ANSWERING RESEARCH QUESTIONS

This thesis has worked towards answering the research questions that were posed in the Chapter 1. In the following section, they are answered in brief. In doing so, reference is made in part to other relevant chapters of the thesis that deal with answers or comments on the research question.

6.5.1 *Representation*

How can the conversion of a node from 0D to 1D and 2D be automated?

Based on the GIP, the car lanes leading to the analyzed node are converted into a lane-specific representation. Based on this, the turning relations between the car lanes are formed. The paths for cyclists and pedestrians are already available in one-dimensional version. By creating buffers around the 1D versions of the turning relations, the two-dimensional representation is created. A more detailed description can be found in Section 3.2.4.

Which type of representation of an intersection is suitable for which purpose?

The zero-dimensional representation is excellent for wayfinding.
The one-dimensional representation is well suited to show the structure of the intersection and information about the individual turning relations.
To represent risk zones of the lanes in the intersection, the two-dimensional representation is particularly suitable.

Which details must necessarily be included in the model?

The answer is formulated in generic terms and can be applied to both the lanes and the turning relations formed in the data preparation.

- To display the results of data preparation, it needs the geometry (1D and 2D) and the base type.

- For the unique identification of the tuple an ID is assigned. The tables of the risk modeling then also refer to this unique identifier.
- To link the elements with the surrounding elements, their IDs are stored to provide not only topological integrity, but also attributive links.
- IDs to the source tables to infer back to the original GIP tuples. This is important because information for risk modeling is obtained from the GIP layers.

6.5.2 *Risk Modeling*

What is the composition of the risk of an intersection and what factors can be used to determine it?

The role of infrastructure with respect to the risk of an intersection for bicyclists, and what the relevant factors are, are listed in Section 2.1.

How can the factors be evaluated and weighted?

The information obtained about an intersection is first normalized, then the factors are weighted with a user-specified weighting, and finally the risk index for the turning relations is calculated. A more detailed description can be found in Section 4.2 and Section 4.3.

Which factors prove to be important?

In the literature review it was found that experts find *speed* to be the most important factor. It impacts the safety of all road users and they estimate traffic to be safer if the speed limit is set low. A comparison of intersections with different speed limits is provided in Section 2.1.4.
Roundabouts are hazardous for cyclists as they hold more points of conflict. They are known to increase the likelihood of a cyclist accident significantly. So, although roundabouts are advantageous to most other road users, cyclists certainly do not benefit from them.
Overall, it was found that local factors are essential to this model. If they were not included, the decomposition of the intersection into its turning relations would be unnecessary, as all relations would be evaluated equally. Local factors highlight the differences within an intersection and demonstrate the importance of looking at an intersection more closely.

Furthermore, Section 4.2 explains why the default values are considered important.

6.5.3 *Results*

How adaptable are the results to the user's priorities?

The weighting function (Listing A.14) is designed to allow the user to insert their own values. Accordingly, it is possible for the user to decide for themselves how important they consider each factor to be.
However, none of the functions allow the user to determine whether a

factor is classified as high risk or low risk. To influence this, the user must make adjustments in the normalization script (Listing A.13).

Are both global and local factors needed?

Yes, in order to accurately evaluate an intersection it is necessary to use both global factors that affect the whole intersection and to include local factors that are individual to the turning relations.
Further discussion of this can be found in Section 6.1.8.

How susceptible is the model to adaptions?

In Section 6.1 the result maps are interpreted. It was found that the model perceives and reflects both changes in weighting and adjustment of the data. As discussed in Section 6.1.8, the strength of the impact depends on the number of factors used in the risk calculation and the extent of the change made.

What sort of information would be needed to better calculate the risk?

In Figure 2.2, the factors identified in the literature review as relevant to the risk calculation are presented. The factors shown in gray are those that are not incorporated into the model. In most cases, they were not included in the model because data for them were not available.

To better estimate the risk to bicyclists, more precise information on bicycle infrastructure is particularly relevant: Especially knowing the characteristics of the bicycle infrastructure in the intersection would be beneficial. Traffic volume is also a very interesting attribute: with information regarding bicycle traffic volume, it is possible to estimate whether "safety in numbers" is present. If there is a higher volume of car traffic, this would mean a higher risk for cyclists. While traffic volume is not an infrastructure-related attribute, there is static data on it (motorists/h) that can be easily incorporated and would add value to the model.
In addition, detailed information on the signal strategy would be advantageous: it could help to assess whether cyclists are exposed to additional hazards or whether the traffic lights protect cyclists.

6.6 OUTLOOK

In the course of this work, many more thoughts came up that could not be considered, explored or executed yet. This section addresses these ideas and possible further developments in regard to the risk model.

DATA

For one, the model could be improved in respect to the data it uses. Due to the unavailability of some data sources, not all factors that were identified to be relevant in regard to the safety of an intersection are included in the risk calculation. The lacking data is depicted in gray in Figure 2.2. In some cases, data exists but it is not as precise as the matter demands (signalization strategy, structure of roundabout, urban surrounding, etc.). Possibilities to obtain the lacking data might be the use of external data sources such as OSM or the cooperation with partners who are in possession of the data needed.

In the current version of the model, the data is included statically. It would be beneficial to make this more dynamic, for example by using a web feature service (WFS). This would have the advantage that the data is maintained by the data provider and is up to date. It would also make the model lighter.

VALIDATION

The normalization and the default weighting scheme were carried out based on an extensive literature review. Nevertheless, it would be interesting to validate them, especially the weighting. Several possibilities may be considered for this purpose: Expert interviews could be conducted to get their opinion on the weighting. Another possibility is to interview cyclists about their personal impressions and conduct a survey. And last but not least, own statistical evaluations can be made. A validation would also offer the possibility to relate the importance of the factors solely to one country. Due to the fact that the literature review is based on work of international origin, it is likely that the normalization and weighting will slightly differ in each country.

USER-FRIENDLINESS

At the moment, only persons with experience regarding spatial data and databases are able to use the tool as it consists of SQL scripts and the data is stored in tables. It could be made more user-friendly by creating a graphical user interface which enables users to operate the tool without having to see a single line of code nor having to download anything. The tool's interface could be a map on which the nodes are displayed. By clicking on one of them, the 1D and 2D are calculated. To enable the user to test different weighting combinations, there could be slide controls on the side.

FURTHER DEVELOPMENTS OF THE TOOL

Momentarily, only the turning relations of an intersection are evaluated. To effectively make use of the model, it should be integrated into a tool that evaluates the segments of the road. This way, the majority of the road space can be assessed regarding its risk.
Although it needs to be considered that when selecting a route, cyclists are interested in more characteristics of their chosen way than only safety. Therefore, additional factors could be included to cover more concerns of the cyclists. Possible factors are cohesion, directness, comfort, and attractiveness.
But the model could also be developed even further in the direction of factors related to collisions: currently, the model only evaluates the risk of accidents happening. However, it could also measure and find factors regarding the severeness of accidents.

---

7 CONCLUSION

In this work, the risk potential of intersections for cyclists was determined. For this purpose, a methodology was developed that decomposes an intersection into its turning relations and then calculates the risk of the relations based on various factors.

The detailed consideration of the intersection is of great importance for several reasons. Firstly, intersections are conflict points where road users meet, which is why a large number of traffic accidents occur at intersections. With accurate information about the risks at the intersection, decision-makers can take measures to improve safety. In addition, cyclists can proactively ensure their own safety by looking for safe intersections when planning a route. The best way to implement safe routing is to integrate the evaluated turning relations into a system that evaluates segments for safety. Turning relations can bridge the gap between two segments, allowing a route to be assessed holistically in terms of its risks. Accident prevention is also supported by the two-dimensional representation of intersections. By extracting risk zones, it is possible to mark dangerous areas and intersections as such, giving road users the opportunity to take appropriate precautions.

The contribution and innovation of this thesis is the automated and effective conversion from a zero-dimensional node to the one-dimensional and two-dimensional representation of the intersection and the evaluation of the risk potential of intersections for cyclists. The thesis differs from the status quo in that it allows the conversion of the node with comparatively few resources, whereas similar results are obtained using methods such as GPS tracking, laser scanning or video recording. In addition, the evaluation of the junction in terms of its risk is somewhat novel, as most previous work has focused on segment evaluation. Comparable work that also assesses intersection risk performs this analysis at a coarser scale, such as assessing entire intersections or grid cells. The more detailed lane-level consideration, combined with the use of global and local risk factors, allows for a more precise analysis of intersections.

8 REFERENCES

Abdel-Aty, M., Chundi, S. S., & Lee, C. (2007). Geo-spatial and log-linear analysis of pedestrian and bicyclist crashes involving school-aged children. *Journal of Safety Research*, *38*(5), 571–579. https://doi.org/10.1016/j.jsr.2007.04.006

Ackery, A. D., McLellan, B. A., & Redelmeier, D. A. (2012). Bicyclist deaths and striking vehicles in the USA. *Injury Prevention : Journal of the International Society for Child and Adolescent Injury Prevention*, *18*(1), 22–26. https://doi.org/10.1136/injuryprev-2011-040066

Aigner, F. (2023). 30/80/100 – Geschwindigkeit senken, Lebensqualität erhöhen! Retrieved from https://www.tuwien.at/tu-wien/aktuelles/news/news/30-80-100-geschwindigkeit-senken-lebensqualitaet-erhoehen

BMK (2020). Assistenzsysteme im Auto: Kleine Helfer, große Wirkung? Retrieved from https://infothek.bmk.gv.at/fahrer-assistenzsysteme-verkehrssicherheit-vernetzung/

BMK (2021). *Österreichische Verkehrssicherheitsstrategie 2021-2030.* Vienna: Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie: Bundesministerium für Klimaschutz, Umwelt, Energie, Mobilität, Innovation und Technologie.

Bouaoun, L., Haddak, M. M., & Amoros, E. (2015). Road crash fatality rates in France: A comparison of road user types, taking account of travel practices. *Accident Analysis & Prevention*, *75*, 217–225. https://doi.org/10.1016/j.aap.2014.10.025

Boufous, S., Rome, L. de, Senserrick, T., & Ivers, R. (2011). Cycling crashes in children, adolescents, and adults-a comparative analysis. *Traffic Injury Prevention*, *12*(3), 244–250. https://doi.org/10.1080/15389588.2011.563333

Boufous, S., Rome, L. de, Senserrick, T., & Ivers, R. (2012). Risk factors for severe injury in cyclists involved in traffic crashes in Victoria, Australia. *Accident; Analysis and Prevention*, *49*, 404–409. https://doi.org/10.1016/j.aap.2012.03.011

Breum, S. M., Kostic, B., & Szell, M. (2022). Computational Desire Line Analysis of Cyclists on the Dybbølsbro Intersection in Copenhagen. *Findings.* Retrieved from https://findingspress.org/article/56683-computational-desire-line-analysis-of-cyclists-on-the-dybbolsbro-intersection-in-copenhagen

Chen, P. (2015). Built environment factors in explaining the automobile-involved bicycle crash frequencies: A spatial statistic approach. *Safety Science*, *79*, 336–343. https://doi.org/10.1016/j.ssci.2015.06.016

CROW (Ed.) (2016). *Record: Vol. 28. Design manual for bicycle traffic.* Ede: Crow.

Daniels, S., Brijs, T., Nuyts, E., & Wets, G. (2009). Injury crashes with bicyclists at roundabouts: Influence of some location characteristics and the design of cycle facilities. *Journal of Safety Research*, *40*(2), 141–148. https://doi.org/10.1016/j.jsr.2009.02.004

Daniels, S., Nuyts, E., & Wets, G. (2008). The effects of roundabouts on traffic safety for bicyclists: An observational study. *Accident Analysis & Prevention*, *40*(2), 518–526. https://doi.org/10.1016/j.aap.2007.07.016

Dong, C., Clarke, D. B., Yan, X., Khattak, A., & Huang, B. (2014). Multivariate random-parameters zero-inflated negative binomial regression model: An application to estimate crash frequencies at intersections. *Accident; Analysis and Prevention*, *70*, 320–329. https://doi.org/10.1016/j.aap.2014.04.018

Dozza, M., & Werneke, J. (2014). Introducing naturalistic cycling data: What factors influence bicyclists' safety in the real world? *Transportation Research Part F: Traffic Psychology and Behaviour*, *24*, 83–91. https://doi.org/10.1016/j.trf.2014.04.001

Eluru, N., Bhat, C. R., & Hensher, D. A. (2008). A mixed generalized ordered response model for examining pedestrian and bicyclist injury severity level in traffic crashes. *Accident Analysis & Prevention*, *40*(3), 1033–1054. https://doi.org/10.1016/j.aap.2007.11.010

Gaus, E. (2023). GitHub Repository Master Thesis. Retrieved from https://github.com/-PostElefant/master-thesis

Hagel, B. E., Romanow, N. T. R., Morgunov, N., Embree, T., Couperthwaite, A. B., Voaklander, D., & Rowe, B. H. (2014). The relationship between visibility aid use and motor vehicle related injuries among bicyclists presenting to emergency departments. *Accident; Analysis and Prevention*, *65*, 85–96. https://doi.org/10.1016/j.aap.2013.12.014

Harris, M. A., Reynolds, C. C. O., Winters, M., Cripton, P. A., Shen, H., Chipman, M. L., . . . Teschke, K. (2013). Comparing the effects of infrastructure on bicycling injury at intersections and non-intersections using a case-crossover design. *Injury Prevention : Journal of the International Society for Child and Adolescent Injury Prevention*, *19*(5), 303–310. https://doi.org/10.1136/injuryprev-2012-040561

Hels, T., & Orozova-Bekkevold, I. (2007). The effect of roundabout design features on cyclist accident rate. *Accident Analysis & Prevention*, *39*(2), 300–307. https://doi.org/10.1016/j.aap.2006.07.008

Hoque, M. M. (1990). An analysis of fatal bicycle accidents in Victoria (Australia) with a special reference to nighttime accidents. *Accident Analysis & Prevention*, *22*(1), 1–11. https://doi.org/10.1016/0001-4575(90)90002-3

Jensen, S. U. (2013). Safety Effects of Converting Intersections to Roundabouts. *Transportation Research Record: Journal of the Transportation Research Board*, *2389*(1), 22–29. https://doi.org/10.3141/2389-03

Kaplan, S., Vavatsoulas, K., & Prato, C. G. (2014). Aggravating and mitigating factors associated with cyclist injury severity in Denmark. *Journal of Safety Research*, *50*, 75–82. https://doi.org/10.1016/j.jsr.2014.03.012

Klop, J. R., & Khattak, A. J. (1999). Factors Influencing Bicycle Crash Severity on Two-Lane, Undivided Roadways in North Carolina. *Transportation Research Record: Journal of the Transportation Research Board*. (1674), 78–85. https://doi.org/10.3141/1674-11

Leden, L., Gårder, P., & Pulkkinen, U. (2000). An expert judgment model applied to estimating the safety effect of a bicycle facility. *Accident Analysis & Prevention*, *32*(4), 589–599. https://doi.org/10.1016/S0001-4575(99)00090-1

Lee, J., Abdel-Aty, M., & Shah, I. (2019). Evaluation of surrogate measures for pedestrian trips at intersections and crash modeling. *Accident Analysis & Prevention*, *130*, 91–98. https://doi.org/10.1016/j.aap.2018.05.015

Lowry, M. B., Callister, D., Gresham, M., & Moore, B. (2012). Assessment of Communitywide Bikeability with Bicycle Level of Service. *Transportation Research Record: Journal of the Transportation Research Board*, *2314*(1), 41–48. https://doi.org/10.3141/2314-06

Manaugh, K., Boisjoly, G., & El-Geneidy, A. (2017). Overcoming barriers to cycling: understanding frequency of cycling in a University setting and the factors preventing commuters from cycling on a regular basis. *Transportation*, *44*(4), 871–884. https://doi.org/10.1007/s11116-016-9682-x

Manirul Islam, S., Washington, S., Kim, J., & Haque, M. (2022). A comprehensive analysis on the effects of signal strategies, intersection geometry, and traffic operation factors on right-turn crashes at signalised intersections: An application of hierarchical crash frequency model. *Accident; Analysis and Prevention*, *171*. https://doi.org/10.1016/j.aap.2022.106663

Merlin, L. A., Guerra, E., & Dumbaugh, E. (2020). Crash risk, crash exposure, and the built environment: A conceptual review. *Accident; Analysis and Prevention*, *134*, 105244. https://doi.org/10.1016/j.aap.2019.07.020

Meuleners, L. B., Stevenson, M., Fraser, M., Oxley, J., Rose, G., & Johnson, M. (2019). Safer cycling and the urban road environment: A case control study. *Accident; Analysis and Prevention*, *129*, 342–349. https://doi.org/10.1016/j.aap.2019.05.032

Miranda-Moreno, L. F., Strauss, J., & Morency, P. (2011). Disaggregate exposure measures and injury frequency models of cyclist safety at signalized intersections. *Transportation Research Record: Journal of the Transportation Research Board*, *2236*(1), 74–82.

National Cancer Institute (2023, March 11). Definition of risk factor - NCI Dictionary of Cancer Terms. Retrieved from https://www.cancer.gov/publications/dictionaries/cancer-terms/def/risk-factor

OECD/International Transport Forum (2013). *Cycling, Health and Safety*. OECD Publishing/ITF. Retrieved from https://www.gip.gv.at/assets/downloads/10JahreGip.pdf https://doi.org/10.1787/9789282105955-en

ÖVDAT (2021). *Standardbeschreibung der Graphenintegrationsplattform (GIP): Intermodaler Verkehrsgraph*. Klagenfurt

Pearson, L., Gabbe, B., Reeder, S., & Beck, B. (2023). Barriers and enablers of bike riding for transport and recreational purposes in Australia. *Journal of Transport & Health*, *28*, 101538. https://doi.org/10.1016/j.jth.2022.101538

Poggenhans, F., Pauls, J.-H., Janosovits, J., Orf, S., Naumann, M., Kuhnt, F., & Mayr, M. (2018). Lanelet2: A high-definition map framework for the future of automated driving. In *2018 IEEE Intelligent Transportation Systems Conference: November 4-7, Maui, Hawaii* (pp. 1672–1679). Piscataway, NJ: IEEE. https://doi.org/10.1109/ITSC.2018.8569929

Prati, G., Marín Puchades, V., Angelis, M. de, Fraboni, F., & Pietrantoni, L. (2018). Factors contributing to bicycle–motorised vehicle collisions: a systematic literature review. *Transport Reviews*, *38*(2), 184–208. https://doi.org/10.1080/01441647.2017.1314391

Pschenitza, M. (2017). Kreuzungsunfälle: Berichte der ADAC Unfallforschung: ADAC Unfallforschung im ADAC Technik Zentrum Landsberg/Lech. Retrieved from https://www.adac.de/-/media/pdf/rund-ums-fahrzeug/kreuzungsunfalle_1886kb.pdf

Reynolds, C. C. O., Harris, M. A., Teschke, K., Cripton, P. A., & Winters, M. (2009). The impact of transportation infrastructure on bicycling injuries and crashes: A review of the literature. *Environmental Health : A Global Access Science Source*, *8*, 47. https://doi.org/10.1186/1476-069X-8-47

Risk Factor (2023, March 12). Find Your Property's Climate Risks - Homepage | Risk Factor. Retrieved from https://riskfactor.com/

Robartes, E., & Chen, T. D. (2017). The effect of crash characteristics on cyclist injuries: An analysis of Virginia automobile-bicycle crash data. *Accident; Analysis and Prevention*, *104*, 165–173. https://doi.org/10.1016/j.aap.2017.04.020

Said, M., Abou-Zeid, M., & Kaysi, I. (2017). Modeling Satisfaction with the Walking Environment: The Case of an Urban University Neighborhood in a Developing Country. *Journal of Urban Planning and Development*, *143*(1), 5016009. https://doi.org/10.1061/(ASCE)UP.1943-5444.0000347

Sakshaug, L., Laureshyn, A., Svensson, A., & Hydén, C. (2010). Cyclists in roundabouts–different design solutions. *Accident; Analysis and Prevention*, *42*(4), 1338–1351. https://doi.org/10.1016/j.aap.2010.02.015

Salzburg Research Forschungsgesellschaft m.b.H. (2020). *Bike2CAV: Entwicklung und Validierung von Methoden zur Kollisionsvermeidung von RadfahrerInnen durch Fahrzeug-zu-X-Kommunikation.* Retrieved from https://projekte.ffg.at/projekt/3802159

Schepers, P., Hagenzieker, M., Methorst, R., van Wee, B., & Wegman, F. (2014). A conceptual framework for road safety and mobility applied to cycling safety. *Accident; Analysis and Prevention*, *62*, 331–340. https://doi.org/10.1016/j.aap.2013.03.032

Schmid-Querg, J., Keler, A., & Grigoropoulos, G. (2021). The Munich Bikeability Index: A Practical Approach for Measuring Urban Bikeability. *Sustainability*, *13*(1), 428. https://doi.org/10.3390/su13010428

Sekiguchi, Y., Tanishita, M., & Sunaga, D. (2022). Characteristics of Cyclist Crashes Using Polytomous Latent Class Analysis and Bias-Reduced Logistic Regression. *Sustainability*, *14*(9), 5497. https://doi.org/10.3390/su14095497

Shen, J., Wang, T., Zheng, C., & Yu, M. (2020). Determinants of Bicyclist Injury Severity Resulting from Crashes at Roundabouts, Crossroads, and T-Junctions. *Journal of Advanced Transportation*, *2020*, 1–12. https://doi.org/10.1155/2020/6513128

Statista Research Department (2021, July). Statistiken zu Verkehrsunfällen in Österreich. *Statista*. Retrieved from https://de.statista.com/themen/4400/verkehrsunfaelle-in-oesterreich/#dossierContents___outerWrapper

Strauss, J., Miranda-Moreno, L. F., & Morency, P. (2014). Multimodal injury risk analysis of road users at signalized and non-signalized intersections. *Accident Analysis & Prevention*, *71*, 201–209. https://doi.org/10.1016/j.aap.2014.05.015

Strauss, J., Miranda-Moreno, L. F., & Morency, P. (2015). Mapping cyclist activity and injury risk in a network combining smartphone GPS data and bicycle counts. *Accident; Analysis and Prevention*, *83*, 132–142. https://doi.org/10.1016/j.aap.2015.07.014

Teschke, K., Harris, M. A., Reynolds, C. C. O., Winters, M., Babul, S., Chipman, M., . . . Cripton, P. A. (2012). Route infrastructure and the risk of injuries to bicyclists: A case-crossover study. *American Journal of Public Health*, *102*(12), 2336–2343. https://doi.org/10.2105/AJPH.2012.300762

Turner, S., Binder, S., & Roozenburg, A. (2009). Cycle Safety: Reducing the Crash Risk.

Vandenbulcke, G., Thomas, I., & Int Panis, L. (2014). Predicting cycling accident risk in Brussels: A spatial case-control approach. *Accident; Analysis and Prevention*, *62*, 341–357. https://doi.org/10.1016/j.aap.2013.07.001

Werner C., Wendel R., Kaziyeva D., Stutz P., van der Meer L., Zagel B., & Loidl M. (2023, March 31). plus-mobilitylab/netascore: v1.0.0 “Core” (Version v1.0.0). Version v1.0.0. Zenodo. https://doi.org/10.5281/zenodo.7789961

Winters, M., Babul, S., Becker, H. J. E. H., Brubacher, J. R. [Jeffrey R.], Chipman, M., Cripton, P., . . . Teschke, K. (2012). Safe Cycling: How Do Risk Perceptions Compare With Observed Risk? *Canadian Journal of Public Health*, *103*(S3), S42-S47. https://doi.org/10.1007/BF03403834

Part IV

APPENDIX

A

DATA AND CODE

A.1 FIGURES

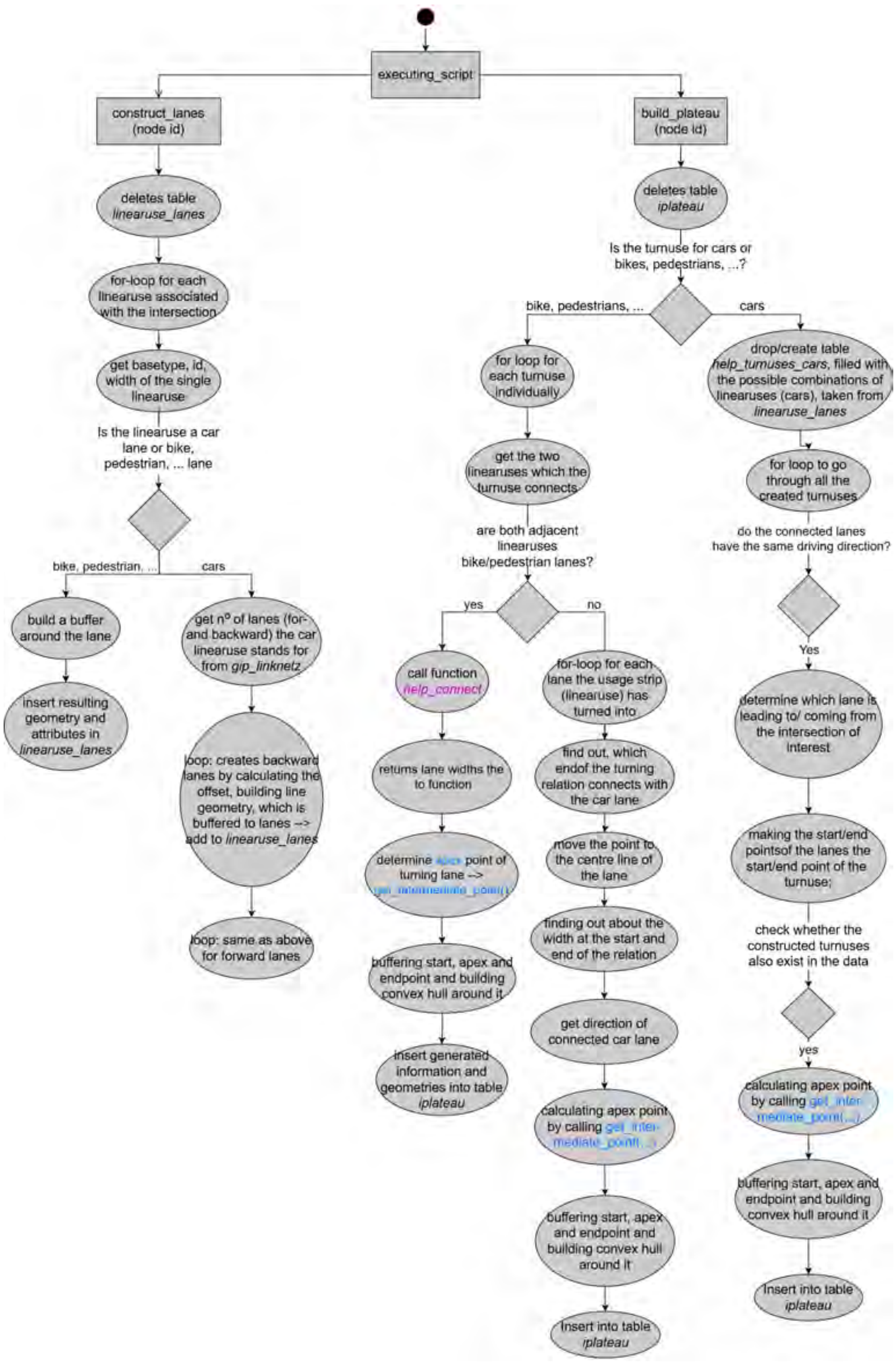

Figure A.1: Activity-Diagram of the functions `construct_lanes(node_id)` and `build_plateau(node_id)`. The figure illustrates the steps taken in the data preparation in order to create one- and two-dimensionale lanes and turning relations. Note: the colored functions are illustrated in Figure A.2

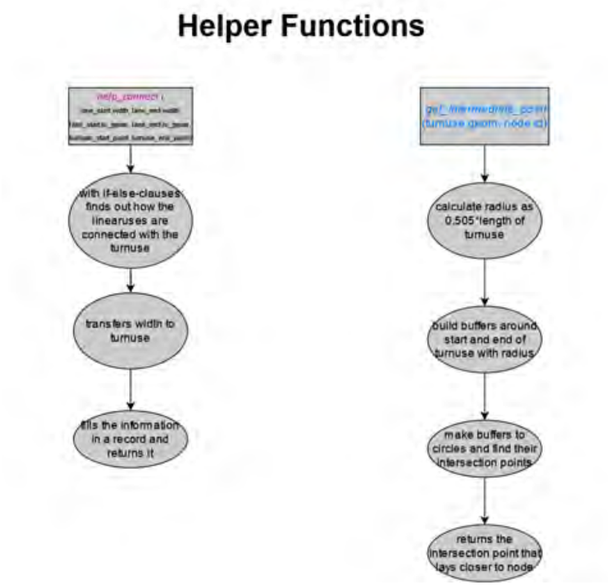

Figure A.2: Activity-Diagram of the functions `help_connect(...)` and `get_intermediate_point(geom, node_id)`. They are helper functions which are called in the process of building the intersection plateau.

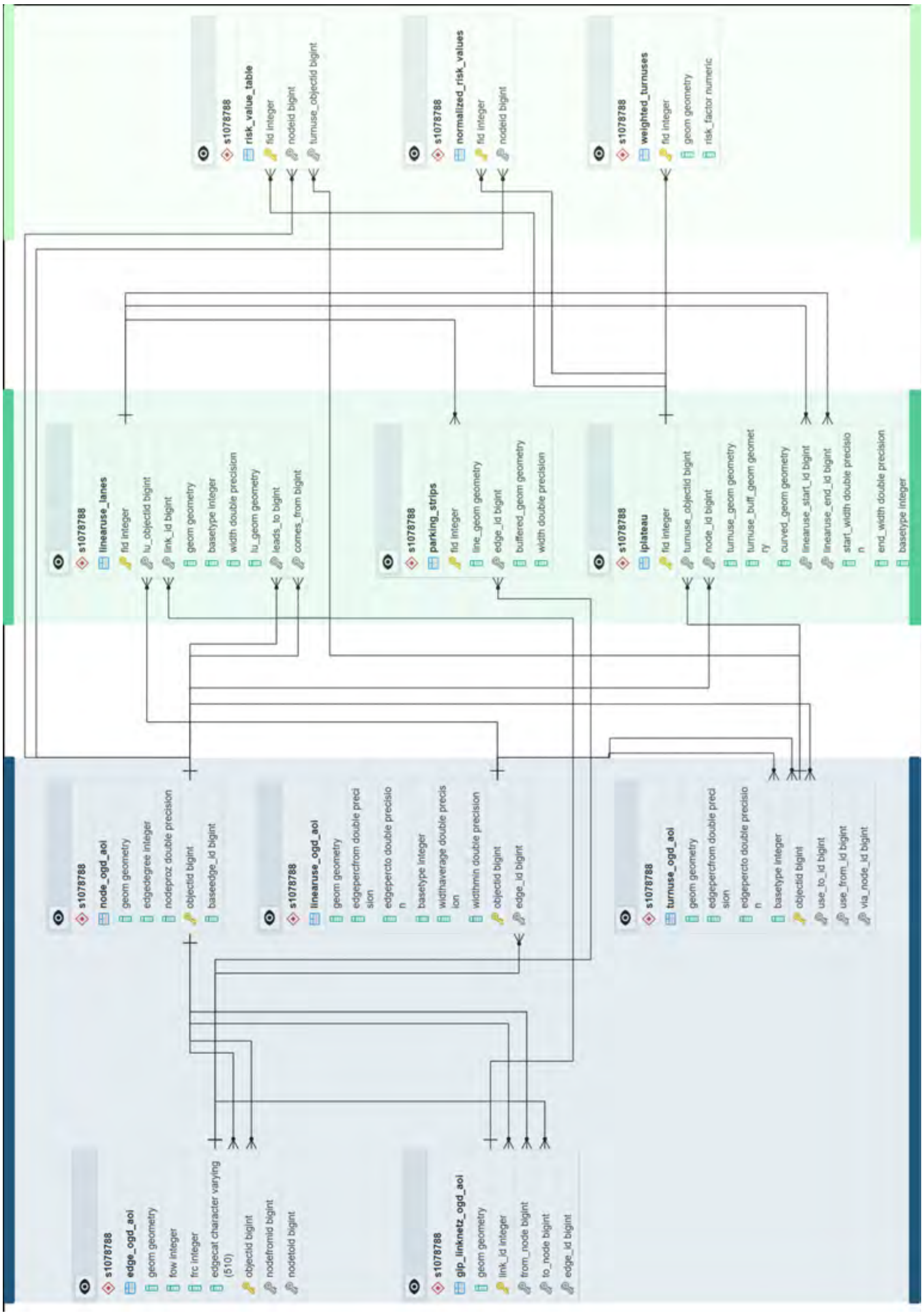

Figure A.3: This ER-Diagram showcases the backbone of the model. It displays the most relevant tables of the system and the attributes, that "hold it all together" and connect the tables. On the left (blue) are the source data from the GIP. In the middle (green) are outputs of the data preparation - the geometries of the intersection. And on the right (light green) are the risk-related tables. In the lower right is the result table that only holds the risk index, fid and geometry of the turning relations.

A.2 CODE

A.2.1 *License Information for Code*

The code developed in the purview of this thesis is licensed under a MIT License.

The code is free: you can use it for commercial and private use, you are allowed to distribute it and you may modify it. The code can be used under the terms of the MIT License as published by the Open Source Initiative.

For more details, have a look at the Open Source Initiative's website, which can be accessed at: https://opensource.org/license/mit/

A.2.2 *Actual Code*

The scripts are provided in a printed version here in order to "have it all in one place" and so it is documented on a longer term. However, it can be found in a digital GitHub repository. In case you want to use the code, it is advised to refer to the digital version.

At the time of publication, the scripts are available in the GitHub repository of the author, Gaus (2023): https://github.com/PostElefant/master-thesis

If the repository is no longer online, the Mobility Lab Salzburg can help.

This script prepares necessary tables for the risk model.

Listing A.1: The environment for the risk model is created.

```
SET search_path = 's1078788', 'public';
DROP FUNCTION IF EXISTS setup();
CREATE OR REPLACE FUNCTION setup()
RETURNS void AS $$

   --DECLARE
   BEGIN

      -- 1st step table: create table that holds the linearuse_lanes for storing
      constructed lanes
      DROP TABLE IF EXISTS linearuse_lanes CASCADE;
      CREATE TABLE linearuse_lanes(fid serial PRIMARY KEY,
         lu_objectid bigint,
         link_id bigint,
         geom geometry,
         basetype integer,
         lane_offset double precision,
         width double precision,
         lu_geom geometry,
         lanes_bkw integer,
         lanes_tow integer,
         lanes_total integer,
         leads_to bigint,
         comes_from bigint,
         CONSTRAINT linearuselanes_luobjectid_fkey FOREIGN KEY(lu_objectid) REFERENCES
      linearuse_ogd_aoi(objectid),
         CONSTRAINT linearuselanes_linkid_fkey FOREIGN KEY(link_id) REFERENCES
      gip_linknetz_ogd_aoi(link_id),
         CONSTRAINT linearuselanes_leadsto_fkey FOREIGN KEY(leads_to) REFERENCES
      node_ogd_aoi(objectid),
         CONSTRAINT linearuselanes_comesfrom_fkey FOREIGN KEY(comes_from) REFERENCES
      node_ogd_aoi(objectid),
         CONSTRAINT linearuselanes_basetype_fkey FOREIGN KEY (basetype) REFERENCES
      lut_basetype(id)
      );


      --2nd step table: it holds the results of the turnuse creation
      DROP TABLE IF EXISTS iplateau CASCADE;
      CREATE TABLE iplateau(
         fid SERIAL PRIMARY KEY,
         turnuse_objectid bigint,
         node_id bigint,
         turnuse_geom geometry,
         turnuse_buff_geom geometry,
         curved_geom geometry,
         start_point_geom geometry,
         end_point_geom geometry,
         linearuse_start_id bigint,
         linearuse_end_id bigint,
         start_width double precision,
         end_width double precision,
         basetype integer,
         intermediate_point geometry,
         el_geom1 geometry,
         el_geom2 geometry,
         CONSTRAINT iplateau_turnuseobjectid_fkey FOREIGN KEY (turnuse_objectid) REFERENCES
      turnuse_ogd_aoi(objectid),
         CONSTRAINT iplateau_nodeid_fkey FOREIGN KEY (node_id) REFERENCES
      node_ogd_aoi(objectid),
         CONSTRAINT iplateau_lustart_fkey FOREIGN KEY (linearuse_start_id) REFERENCES
      linearuse_lanes(fid),
         CONSTRAINT iplateau_luend_fkey FOREIGN KEY (linearuse_end_id) REFERENCES
      linearuse_lanes(fid),
         CONSTRAINT iplateau_basetype_fkey FOREIGN KEY (basetype) REFERENCES
      lut_basetype(id)
      );


      --3rd step: it holds the parking lanes
      DROP TABLE IF EXISTS parking_strips CASCADE;
      CREATE TABLE parking_strips(
         fid serial primary key,
         line_geom geometry,
         edge_id bigint,
         buffered_geom geometry,
         width double precision,
         CONSTRAINT parkingstrips_fid_fkey FOREIGN KEY(fid) REFERENCES linearuse_lanes(fid),
```

```
        CONSTRAINT parkingstrips_edgeid_fkey FOREIGN KEY (edge_id) REFERENCES edge_ogd_aoi(objectid)
    );

    -- 4th step: table that holds the data that is obtained with the helper functions.
    DROP TABLE IF EXISTS risk_value_table;
    CREATE TABLE IF NOT EXISTS risk_value_table (
        fid integer PRIMARY KEY,
        nodeid bigint,
        turnuse_objectid bigint,
        basetype integer,
        urban boolean,
        rails boolean,
        traffic_lights boolean,
        intersection_legs integer,
        intersection_angle integer,
        speed integer,
        traffic_volume integer,
        lane_number_cars integer,
        lane_number_bikes integer,
        gradient real,
        barriers integer,
        street_type boolean,
        street_condition integer,
        round_about_inner_circle integer,
        round_about boolean,
        round_about_cycle_infstr boolean,
        mixed_traffic_f varchar(10),
        mixed_traffic_t varchar(10),
        cycle_infrastructure_f varchar(45),
        cycle_infrastructure_t varchar(45),
        CONSTRAINT riskvaluetable_fid_fkey FOREIGN KEY (fid) REFERENCES iplateau(fid),
        CONSTRAINT riskvaluetable_nodeid_fkey FOREIGN KEY (nodeid) REFERENCES node_ogd_aoi(objectid),
        CONSTRAINT riskvaluetable_turnuseobjectid_fkey FOREIGN KEY (turnuse_objectid) REFERENCES turnuse_ogd_aoi(objectid)
    );

    -- 5th step: the normalized values (0-1) are stored in here.
    DROP TABLE IF EXISTS normalized_risk_values;
    CREATE TABLE normalized_risk_values (
        fid integer PRIMARY KEY,
        nodeid bigint,
        urban numeric,
        rails numeric,
        traffic_lights numeric,
        intersection_legs numeric,
        speed numeric,
        lane_number_cars numeric,
        lane_number_bikes numeric,
        gradient numeric,
        street_type numeric,
        round_about_inner_circle numeric,
        round_about numeric,
        round_about_cycle_infstr numeric,
        mixed_traffic numeric,
        cycle_infrastructure numeric,
        CONSTRAINT normalizedriskvalues_fid_fkey FOREIGN KEY(fid) REFERENCES iplateau(fid),
        CONSTRAINT normalizedriskvalues_nodeid_fkey FOREIGN KEY (nodeid) REFERENCES node_ogd_aoi(objectid)
    );

    --6th step: the finished lanes (weighted and normalized in 1D)
    DROP TABLE IF EXISTS weighted_turnuses;
    CREATE TABLE weighted_turnuses (
        fid integer PRIMARY KEY,
        geom geometry(Linestring, 3857),
        risk_factor numeric,
        CONSTRAINT weightedturnuses_fid_fkey FOREIGN KEY(fid) REFERENCES iplateau(fid)
    );



  END;
$$ LANGUAGE 'plpgsql';
```

The execution script calls the functions that build the risk model. They are given an integer parameter which is the node's (intersection's) id.

Listing A.2: Script executes functions that build the risk model (`executing_script.sql`)

```sql
-- this script executes the functions that make the workflow
-- the script is where all the threads come together.
-- the single parameter given is the node's objectid.
SET search_path = 's1078788', 'public';

-- create tables in which the results will be stored in
SELECT setup();

-- build 1D and 2D representation of the lanes leading to the intersection
SELECT construct_lanes(10355976);

-- create the intersection plateau by creating the turning relations
SELECT build_plateau(10355976);

-- construct the parking lanes and clip them from the driving lanes
SELECT parking(10355976);

-- get information and data about the data
SELECT get_risk_factors(10355976);

-- normalize the data to a scale from 0 (very risky) to 1 (safe)
SELECT normalize_factors();

-- Weighting of the factors, which are then used to calculate the risk index.
-- the parameters correspond to the weights for the factors. Th
-- In the following order, the parameters correspond to the weights:
-- urban, rails, traffic lights, edge degree, speed, # crossed car lanes,
-- # crossed bike lanes, gradient, street type, round about (ra), ra inner circle,
-- ra cycle infrastructure, cycling infrastructure, mixed traffic
SELECT weighting(0, 0, 0, 0, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0, 0, 0.2, 0)

-- when all functions have been called, the script add_foreignkeys.sql should be run
-- to create foreign keys between the permanent and the constantly regenerated tables.

-- Example Intersections:
-- Intersection A One-way streets: objectid = 10354202
-- Intersection B complicated junction (3-legged): objectid = 10355976
-- Intersection C classic 4-legged junction: objectid = 10355123

-- Other Weighting Schemes:
-- 1 Default-Values:
-- SELECT weighting(0, 0, 0, 0, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0, 0, 0.2, 0)
-- 2 local indicators:
-- SELECT weighting(0, 0, 0, 0, 0, 0.2, 0.2, 0.2, 0, 0, 0, 0, 0.3, 0.1)
-- 3 all values same:
-- SELECT weighting(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
-- 4 signalisation sim. with def values:
-- UPDATE risk_value_table SET traffic_lights = true; SELECT normalize_factors();
-- SELECT weighting(0, 0, 0.2, 0, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0, 0, 0.2, 0)
-- 5 speed: 30 km/h, weighting with def. values:
-- UPDATE risk_value_table SET traffic_lights = false, speed = 30; SELECT normalize_factors();
-- SELECT weighting(0, 0, 0, 0, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0, 0, 0.2, 0)
-- 6 speed: 50 km/h, weighting with def. values:
-- UPDATE risk_value_table SET speed = 50; SELECT normalize_factors();
-- SELECT weighting(0, 0, 0, 0, 0.3, 0.1, 0.1, 0.1, 0, 0.2, 0, 0, 0.2, 0)
```

This is the function construct_lanes(integer). Its task is to convert the linearuses to one- and two-dimensional representations of the lanes leading to or coming from the intersection.

Listing A.3: Convert linearuses to 1D and 2D lanes (`construct_lanes.sql`)

```sql
SET search_path = 's1078788', 'public';
DROP FUNCTION IF EXISTS construct_lanes(integer);
CREATE OR REPLACE FUNCTION construct_lanes(integer)
RETURNS void AS $$
  DECLARE
     node_id ALIAS FOR $1;
    lnes_tow INTEGER;
    lnes_bkw INTEGER;
    lnes_total INTEGER;
    oneway_cr INTEGER;
    width DOUBLE PRECISION;
    single_lane_width DOUBLE PRECISION;
    basetyp INTEGER;
    linearuse_objectid BIGINT;
    linearuse linearuse_ogd_aoi%ROWTYPE;
    links gip_linknetz_ogd_aoi%ROWTYPE;
    linkn_id BIGINT;
    no_linearuses INTEGER;
    edg_id BIGINT;
    links_geo GEOMETRY;
    lineuse_geo GEOMETRY;
    lineuse_buff_geom GEOMETRY;
    offset_lane_bkw DOUBLE PRECISION := 0;
    offset_lane_tow DOUBLE PRECISION := 0;
    node_edgedegree integer;
    one_way BOOL := false; -- boolean that is set to true in case it is a one-way street
    one_way_direction BOOL; -- which direction leads the one way street? true=tow, false=bkw

  BEGIN
   -- get number of linearuses connected to the node
    no_linearuses := (SELECT COUNT(l.*) FROM linearuse_ogd_aoi l, edge_ogd_aoi e
                     WHERE l.edge_id = e.objectid
                           AND (e.nodefromid = node_id OR e.nodetoid = node_id));
    -- get number of edges connected to the node
    node_edgedegree := (SELECT n.edgedegree FROM node_ogd_aoi n
                   WHERE n.objectid = node_id);

    RAISE NOTICE 'No linearuses: %', no_linearuses;

   --formerly: creating table (linearuse_lanes) that holds the information of the lanes
   --now: the table is created in the setup-script.
   -- there, former created information is deleted so it can be newly filled again
   DELETE FROM linearuse_lanes CASCADE;


   -- LOOP through the linearuses
   FOR q IN 1..no_linearuses LOOP
      linearuse :=NULL;
       -- get one linearuse of the intersection
      SELECT INTO linearuse l.* FROM linearuse_ogd_aoi l, edge_ogd_aoi e
      WHERE l.edge_id = e.objectid AND (
         e.nodefromid = node_id OR e.nodetoid = node_id)
      ORDER BY l.objectid
      LIMIT 1 OFFSET (q-1);


      -- find out about its basetype
      basetyp := linearuse.basetype;
      RAISE NOTICE 'Basetyp: %',basetyp;

      -- get basic information
      linearuse_objectid := linearuse.objectid;
      edg_id := linearuse.edge_id;



--- if it is a driving lane (basetype=1) we have to deal with it differently: ----
      IF(linearuse.basetype=1) THEN

      -- we determine the width of the street based on
      -- whether there is a parking strip next to it or not
      -- because: the parking strip is part of the roadway.
      -- as such, they are part of the street and its width (widthaverage)
      -- if there is a parking strip, the "true" width of the lanes is stored in widthmin
         IF (parking_strip_nearby(edg_id) = true) THEN -- refers to other function
            width := linearuse.widthmin;
            RAISE NOTICE 'parking strip nearby is true.\nTherefore width is %', width;
         ELSE
```

```
            width := linearuse.widthaverage;
            RAISE NOTICE 'parking lane nearby is false.\nTherefore width is %', width;
         END IF;


      -- check whether it is a deadend
      -- because if it is, the query has to be slightly different
         IF (node_edgedegree > 1) THEN
         -- select the link that the recent linearuse is connected with into var links
         SELECT INTO links li.*
            FROM linearuse_ogd_aoi lu, gip_linknetz_ogd_aoi li, node_ogd_aoi n,
               edge_ogd_aoi e, turnuse_ogd_aoi tu
            WHERE lu.edge_id=e.objectid
            AND li.edge_id = e.objectid
            AND (li.from_node = n.objectid OR li.to_node = n.objectid)
            AND n.objectid=node_id
            AND lu.objectid = linearuse_objectid
            AND (lu.objectid = tu.use_to_id OR lu.objectid = tu.use_from_id)
            -- the next line leaves out elements that don't belong to the junction
            -- (parking strips, traffic islands, ...)
            AND tu.via_node_id=n.objectid;

         -- IF IT IS a dead end...
         ELSIF (node_edgedegree = 1) THEN
         SELECT INTO links li.*
            FROM linearuse_ogd_aoi lu, gip_linknetz_ogd_aoi li, node_ogd_aoi n, edge_ogd_aoi
      e, turnuse_ogd_aoi tu
            WHERE lu.edge_id=e.objectid
            AND li.edge_id = e.objectid
            AND (li.from_node = n.objectid OR li.to_node = n.objectid)
            AND n.objectid=node_id
            AND lu.objectid = linearuse_objectid
            AND (lu.objectid = tu.use_to_id OR lu.objectid = tu.use_from_id);
         END IF;

         RAISE NOTICE 'link id %', links.link_id;
         RAISE NOTICE 'lu_objectid %', linearuse_objectid;

         -- storing information about the link id, number of driving lanes
         -- ... and if it is a one-way street
         linkn_id := links.link_id;
         lnes_tow := links.lanes_tow; -- # linearuses directed in the digitalization dir.
         RAISE NOTICE 'lanes tow: %', lnes_tow;
         lnes_bkw := links.lanes_bkw; -- # linearuses directed against the dig. dir.
         RAISE NOTICE 'lanes bkw: %', lnes_bkw;
         oneway_cr := links.oneway_car;
         RAISE NOTICE 'oneway: %', oneway_cr;
         -- geometry
         links_geo := links.geom;

         --finding out, how many (car) lanes there are in total
         IF (lnes_tow > 0 AND lnes_bkw > 0) THEN
            lnes_total := lnes_tow+lnes_bkw;
         ELSIF (lnes_tow > 0 AND lnes_bkw <= 0) THEN
            lnes_total := lnes_tow;
            one_way := true;
            one_way_direction := true; -- one way street is in direction tow
         ELSIF (lnes_tow <= 0 AND lnes_bkw > 0) THEN
            lnes_total := lnes_bkw;
            one_way := true;
            one_way_direction := false; -- one way street is in direction bkw
         END IF;
         RAISE NOTICE 'total number of lanes: % and width: % for linkuse %',
            lnes_total, width, linearuse_objectid;

         --getting the width of a single driving lane
         single_lane_width := width/lnes_total;

         --setting the offset variables back to zero
         offset_lane_bkw := 0;
         offset_lane_tow := 0;

         --in case it is not oneway streets
         IF (one_way = false) THEN

            --finding out about the single lane's position and buffering
            --first for the direction bkw
            FOR j IN 0..lnes_bkw-1 LOOP
               lineuse_buff_geom := NULL;
               lineuse_geo := NULL;
               -- calculate the lane's offset
               IF (j=0) THEN
```

```
            -- if it is the first round, the offset is half the lane's width
            offset_lane_bkw := offset_lane_bkw + (single_lane_width/2);
            RAISE NOTICE 'offset_lane_bkw % in pass i= %', offset_lane_bkw, j;
         ELSE
            -- if it is the >=2nd round, the offset is the lane's full width
            offset_lane_bkw := offset_lane_bkw + single_lane_width;
            RAISE NOTICE 'offset_lane_bkw % in pass i= %', offset_lane_bkw, j;
         END IF;

         -- create the 2D geometry: the original 1D geometry of linearuse_ogd_aoi is
         -- taken, then it is transformed into a local coordinate reference system,
         -- after that, the geometry is shifted by the determined offset
         -- finally, the lane is made 2d by creating a buffer around it
         -- (size of lanewidth/2)
         lineuse_buff_geom := ST_Buffer(ST_OffsetCurve(ST_Transform(linearuse.geom,
31259),
                                            offset_lane_bkw, 'quad_segs=4 join=round'),
                                 single_lane_width/2,'endcap=square');

         -- create the 1D geometry: the oridinal 1D geometry of linearuse_ogd_aoi is
         -- taken, then it is transformed into a local coordinate reference system,
         -- after that, the geometry is shifted by the determined offset
         -- finally, its direction is reversed.
         -- this has to be done because ST_offsetCurve turns the original geometry
         lineuse_geo := ST_Reverse(ST_OffsetCurve(ST_Transform(linearuse.geom, 31259),
                                        offset_lane_bkw, 'quad_segs=4 join=round')
                         );

         -- insert the values and calculated geometries into the table linearuse_lanes
         INSERT INTO linearuse_lanes(lu_objectid, link_id, basetype, lane_offset,
                              width, lu_geom,geom, lanes_bkw, lanes_tow,
                              lanes_total, leads_to, comes_from)
            VALUES (linearuse_objectid, linkn_id, basetyp, offset_lane_bkw,
                  single_lane_width, lineuse_geo, lineuse_buff_geom, lnes_bkw, lnes_tow,
                  lnes_total, links.from_node, links.to_node);

      END LOOP;


      -------- and now basically the same for the direction tow ----------
      FOR k IN 0..lnes_tow-1 LOOP
         lineuse_buff_geom := NULL;
         lineuse_geo := NULL;
         IF (k=0) THEN
            offset_lane_tow := offset_lane_tow - (single_lane_width/2);
            RAISE NOTICE 'offset_lane_tow % at pass i= %', offset_lane_tow, k;

         ELSE
            offset_lane_tow := offset_lane_tow - single_lane_width;
         END IF;

         lineuse_buff_geom := ST_Buffer(ST_OffsetCurve(ST_Transform(
            linearuse.geom, 31259), offset_lane_tow, 'quad_segs=4 join=round'),
                                 single_lane_width/2, 'endcap=square');
         lineuse_geo := ST_Reverse(ST_OffsetCurve(ST_Transform(linearuse.geom, 31259),
                                        offset_lane_tow, 'quad_segs=4 join=round')
                         );

         RAISE NOTICE 'geom: %',lineuse_buff_geom;

         INSERT INTO linearuse_lanes(lu_objectid, link_id, basetype,
                              lane_offset, width, lu_geom,
                              geom, leads_to, comes_from)
            VALUES (linearuse_objectid, linkn_id, basetyp, offset_lane_tow,
                  single_lane_width, lineuse_geo, lineuse_buff_geom,
                  links.to_node, links.from_node);

      END LOOP;


   ELSE -- in case it is a one way street ---------------------


      FOR p IN 0..lnes_total-1 by 1 LOOP

         IF (p = 0) THEN
            IF (lnes_total % 2 = 0) THEN
             -- if it is of an even number of lanes,
             -- an initial offset of half a lane's width is needed
               offset_lane_tow := single_lane_width / 2;
            END IF;
         ELSE
```

```
                IF (p % 2 = 0) THEN
                    offset_lane_tow := offset_lane_tow + (single_lane_width * p);
                ELSIF (p % 2 = 1) THEN
                    offset_lane_tow := offset_lane_tow + (single_lane_width * p * (-1));
                END IF;
            END IF;

            RAISE NOTICE 'In pass % there is an Offset of %', p, offset_lane_tow ;

            lineuse_buff_geom := ST_Buffer(ST_OffsetCurve(ST_Transform(linearuse.geom,
    31259),
                                            offset_lane_tow, 'quad_segs=4 join=round'),
                                single_lane_width/2,'endcap=square');


            -- problem is, that ST_OffsetCurve changes the direction of lines
            -- that were created with the help of negative distances.
            -- In a one way street we naturally want all the streets to head in the
            -- same direction. Therefore an if-else building needs to be constructed.

             -- if the linearuse geometry of the one way street is headed in the same
             -- direction as the edge it belongs to, ...
            IF (lnes_tow > 0) THEN
            -- ... we need to reverse the geometry of those lanes
            -- with a negative distance
                IF (offset_lane_tow < 0) THEN
                    lineuse_geo := ST_Reverse(ST_OffsetCurve(ST_Transform(linearuse.geom,
    31259),
                                            offset_lane_tow,
                                            'quad_segs=4 join=round'));
                 -- ... for the others, we don't need to reverse them
                ELSE
                    lineuse_geo := ST_OffsetCurve(ST_Transform(linearuse.geom, 31259),
                                        offset_lane_tow, 'quad_segs=4 join=round');
                END IF;

            -- if the linearuse geometry of the one way street is headed in the
            -- opposite direction of the edge it belongs to...
            ELSIF (lnes_bkw > 0) THEN
            -- ... the ones we do not need to reverse the lane's geometry because
            -- of the negative distance they are reversed anyway
                IF (offset_lane_tow < 0) THEN
                    lineuse_geo := ST_OffsetCurve(ST_Transform(linearuse.geom, 31259),
                                        offset_lane_tow, 'quad_segs=4 join=round');
                 -- however, those geometries that have a positive distance
                 -- to the original linearuse, need to be turned now
                ELSE
                    lineuse_geo := ST_Reverse(ST_OffsetCurve(ST_Transform(
                        linearuse.geom, 31259), offset_lane_tow, 'quad_segs=4 join=round'));
                END IF;
            END IF;

            -- to keep the attributive integrity, leads_to and comes_from
            -- need to be adapted f the oneway streets are in the same
            -- direction as the link (gip_linknetz)
            IF (one_way_direction = TRUE) THEN
                INSERT INTO linearuse_lanes(lu_objectid, link_id, basetype, lane_offset,
                                width, lu_geom, geom, leads_to, comes_from)
                    VALUES (linearuse_objectid, linkn_id, basetyp, offset_lane_tow,
                        single_lane_width, lineuse_geo, lineuse_buff_geom,
                        links.to_node, links.from_node);
            ELSIF (one_way_direction = FALSE) THEN
                INSERT INTO linearuse_lanes(lu_objectid, link_id, basetype, lane_offset,
                                width, lu_geom, geom, leads_to, comes_from)
                    VALUES (linearuse_objectid, linkn_id, basetyp, offset_lane_tow,
                        single_lane_width, lineuse_geo, lineuse_buff_geom,
                        links.from_node, links.to_node);
            END IF;

        END LOOP;

    END IF;


-----------------------------------------------------------------------------------------


    -- IF IT IS NOT A DRIVING LANE, THEN CHECK WHETHER
    -- IT IS AN OTHER RELEVANT BASETYPE OF LINEARUSE
    -- path ways: 7, 21, 37, 41    -- cycle lanes: 2, 22, 23, 31, 33, 35, 36
    ELSIF (linearuse.basetype IN (2, 7, 21, 22, 23, 35, 36)) THEN

```

```
        -- get width of the geometry
        width := linearuse.widthaverage;
        -- 2D geometry: build buffer around it
        lineuse_buff_geom := ST_Buffer(ST_Transform(linearuse.geom, 31259),
                                width/2,'endcap=square');
        -- 1D geometry: transform it to local crs
        lineuse_geo := ST_Transform(linearuse.geom,31259);

        -- insert results in table linearuse_lanes
        -- note: leads_to and comes_from are attributes that are not really needed
           -- here, as sidewalks do not have a driving direction
        INSERT INTO linearuse_lanes(lu_objectid, link_id, basetype, lane_offset,
                             width, lu_geom, geom, leads_to, comes_from)
              VALUES (linearuse_objectid, linkn_id, basetyp, 0, width, lineuse_geo,
                    lineuse_buff_geom, links.to_node, links.from_node);


     END IF;


  END LOOP;

 END;
$$ LANGUAGE 'plpgsql';
```

The function determines whether the minimal or the average width is used to construct the two-dimensional lanes.

Listing A.4: Determine width attribute used to construct car lanes. (`parking_strip_nearby.sql`)

```
-- checks, whether along the edge is a parking lane
-- function returns boolean

-- is given integer (the edge's id)
DROP FUNCTION IF EXISTS parking_strip_nearby(bigint);
CREATE OR REPLACE FUNCTION parking_strip_nearby(bigint)
RETURNS bool AS $$

  DECLARE
   edg_id ALIAS FOR $1;
   count_linearuses integer;
   bt integer;

  BEGIN
   -- get the linearuses along the edge
   count_linearuses := (SELECT COUNT(lu.*) FROM linearuse_ogd_aoi lu
                   WHERE lu.edge_id = edg_id);

   -- loop through them
   FOR q IN 1..count_linearuses LOOP
      bt := (SELECT lu.basetype
            FROM linearuse_ogd_aoi lu
            WHERE lu.edge_id = edg_id
            ORDER BY lu.objectid
            OFFSET q - 1
            LIMIT 1);

      -- if the examined linearuse is indeed a parking strip, return true
      IF (bt = 8) THEN
         RETURN true;
      END IF;

   END LOOP;

   -- if none of the linearuses was a parking strip, return false
   RETURN false;

  END;
$$ LANGUAGE 'plpgsql';
```

The parking function detects the parking strips, builds a buffer around it and trims overlaps of linearuse-lanes.

Listing A.5: Get parking strip, convert it to 2D, trim overlaps of lanes (`parking.sql`)

```
-- this function is used for constructing the parking lanes
-- it does it for the parking lanes that are along segments leading to
   -- the intersection under investigation
-- first, it makes the parking lanes 2-dimensional by building a buffer
-- then, it clips the parking strip's geometry from the driving lanes

SET search_path = 's1078788', 'public';
DROP FUNCTION IF EXISTS parking(integer);
CREATE OR REPLACE FUNCTION parking(integer)
RETURNS void AS $$
  DECLARE
   node_id ALIAS FOR $1;
   parking_linearuse linearuse_ogd_aoi%ROWTYPE;
   width double precision;
   no_parking_strips integer;
   lu_edgeid bigint;
   buffered_geom geometry;

   no_linearuse_lanes integer;
   lu_lane linearuse_lanes%ROWTYPE;
   park_strip parking_strips%ROWTYPE;
   lane_wo_park geometry; -- lane without geometry

  BEGIN
   -- get number of parking lanes
   no_parking_strips := (SELECT COUNT(lu.*)
                    FROM linearuse_ogd_aoi lu, edge_ogd_aoi e
                    WHERE lu.edge_id = e.objectid
                     AND lu.basetype = 8
                    AND (e.nodefromid = node_id OR e.nodetoid = node_id));

   -- formerly, the table parking_strips was created here.
   -- however, this was outsourced to the setup() script
   DELETE FROM parking_strips;

   -- loop through all parking strips that are connected to edges leading to the
     intersection
   FOR q in 1..no_parking_strips LOOP

      -- take one parking strip and store it in the variable parking_linearuse
      SELECT INTO parking_linearuse lu.*
      FROM linearuse_ogd_aoi lu, edge_ogd_aoi e
      WHERE lu.edge_id = e.objectid
         AND lu.basetype = 8
         AND (e.nodefromid = node_id
            OR e.nodetoid = node_id)
      ORDER BY lu.objectid
      LIMIT 1
      OFFSET (q-1);

      RAISE NOTICE 'line geom: %', parking_linearuse.geom;

      lu_edgeid := parking_linearuse.edge_id;

      -- build a buffer around the parking strip in question.
      -- Calculate the buffer's width from the strip's attribute widthaverage
      buffered_geom := ST_Buffer(ST_Transform(parking_linearuse.geom, 31259),
      parking_linearuse.widthaverage/2 ,'endcap=square');

      -- store the geometry that was just created in the table parking_strips.
      -- Attributes: original geometry, buffered geometry, the geom's width,
      -- the edge's ID it belongs to
      INSERT INTO parking_strips (line_geom, buffered_geom, width, edge_id)
      VALUES (ST_Transform(parking_linearuse.geom, 31259), buffered_geom,
            parking_linearuse.widthaverage, lu_edgeid);


   END LOOP;



   ------ CLIPPING -------
   -- the clipping part clips elements of the linearuse_lanes table.
   -- this is done because it happened that the road's width included the parking strip.
   -- This widens the lane immensely and distorts the model.
   -- For this reason, this function cuts out the parking strip from the lane.

```

```
    no_linearuse_lanes := (SELECT COUNT(*) FROM linearuse_lanes);

    -- go through all lanes
    FOR r in 1..no_linearuse_lanes LOOP

        -- store it in the variable lu_lane
        SELECT INTO lu_lane lul.*
        FROM linearuse_lanes lul
        ORDER BY lul.fid
        LIMIT 1
        OFFSET (r-1);

        -- go through all parking strips
        FOR s in 1..no_parking_strips LOOP

            -- store it in the variable park_strip
            SELECT INTO park_strip ps.*
            FROM parking_strips ps
            ORDER BY fid
            LIMIT 1
            OFFSET (s-1);

            -- if the lane and the parking strip intersect...
            IF (ST_Intersects(lu_lane.geom, park_strip.buffered_geom)) THEN

                -- ... build the difference (lane - parking strip). store it in the var lane_wo_park
                lane_wo_park := ST_Difference(lu_lane.geom, park_strip.buffered_geom);

                -- update the table linearuse_lanes by replacing the old with the new geometry
                UPDATE linearuse_lanes
                SET geom = lane_wo_park
                WHERE fid = lu_lane.fid;


            END IF;

        END LOOP;

    END LOOP;

  END;
$$ LANGUAGE 'plpgsql';
```

This script holds the function which created the intersection plateau. They connect the lanes (be it pathways, cycle lanes or car lanes) with each other.

Listing A.6: Creating turning relations with the help of this SQL-function (build_plateau.sql)

```
-- this function builds the 1D and 2D version of the intersection node
-- it builds upon the 1D results of the lane construction script
-- the outputs of this function serve as the base of the risk model

SET search_path = 's1078788', 'public';
DROP FUNCTION IF EXISTS build_plateau(integer);
CREATE OR REPLACE FUNCTION build_plateau(integer)
RETURNS void AS $$
  DECLARE
   node_id ALIAS FOR $1;
   no_turnuses_side INTEGER;
   turnuse turnuse_ogd_aoi%ROWTYPE;
   turnuse_car help_turnuses_cars%ROWTYPE;
   --table that holds the turnuse options of the intersection for cars
   var_available_turnuses available_turnuses%ROWTYPE;
   node node_ogd_aoi%ROWTYPE;
   turnuse_id bigint;
   lane_start linearuse_lanes%ROWTYPE;
   lane_end linearuse_lanes%ROWTYPE;
   lane_start2 linearuse_lanes%ROWTYPE;
   lane_end2 linearuse_lanes%ROWTYPE;
   turnuse_start_point geometry;
   turnuse_end_point geometry;
   turnuse_start_point2 geometry;
   turnuse_end_point2 geometry;
   start_twidth double precision; -- width the turnuse buffer has at its start
   end_twidth double precision; -- width the turnuse buffer has at its end
   lane_at_tustart boolean; --boolean that is true, if the start lane is at the startpoint
      of the turnuse. the bool is false if the endpoint is the startpoint of the turnuse
   turnuse_buff_geom geometry;
   turnuse_basetype integer;
   no_turnuses_car integer;
   no_available_turnuses integer;
   turnuse_car_point1 geometry;
   turnuse_car_point2 geometry;
   intermediate_point geometry;
   elongated_l_geom geometry;
   elongated_ll_geom geometry;
   intersection_point_elongated_lls geometry;
   record_hc record;
   no_actual_carlanes integer;
   linearuse_lane_car linearuse_lanes%ROWTYPE;
   ll_start_fid bigint;
   ll_end_fid bigint;


  BEGIN

   -- get number of turnuses in the intersection that are for pedestrians/cyclists
   no_turnuses_side := (SELECT COUNT(t.*) FROM turnuse_ogd_aoi t
                WHERE t.via_node_id = node_id
                  AND basetype IN (2, 7, 21, 22, 23, 33, 35, 36, 37, 41));
   RAISE NOTICE 'no_turnuses_side: %', no_turnuses_side;

   SELECT INTO node n.* FROM node_ogd_aoi n WHERE n.objectid=node_id;

    -- the if-clause stretches over the whole script.
    -- it just reassures that no error is thrown because there simply are..
    -- ..no turnuses when the node is at the end of a one-way-street
   IF (node.edgedegree > 1) THEN

   --formerly, the iplateau was created here. this was moved into the setup-script
   DELETE FROM iplateau;

--------

-- loop through the turnuses of pedestrians and cyclists
FOR q in 1..no_turnuses_side LOOP

   SELECT INTO turnuse t.* FROM turnuse_ogd_aoi t
   -- Basetypes for bicycle infrastructure: 2, 22, 23, 31, 33, 35, 36
   -- Basetypes for pedestrians: 7, 21, 37, 41
   WHERE t.via_node_id = node_id
      AND basetype IN (2, 7, 21, 22, 23, 31, 33, 35, 36, 37, 41)
   ORDER BY t.objectid
   LIMIT 1
   OFFSET (q-1);
```

```

  turnuse_id := turnuse.objectid;
  turnuse_basetype := turnuse.basetype;

 IF (turnuse.basetype IN (2, 7, 21, 22, 23, 31, 33, 35, 36, 41)) THEN
  lane_start := NULL;
  lane_end := NULL;
  turnuse_start_point := NULL;
  turnuse_end_point := NULL;

  -- store linearuse_lane that connects to the turnuse in variable
  SELECT INTO lane_start l.*
  FROM linearuse_lanes l, turnuse_ogd_aoi t
  WHERE t.use_from_id=l.lu_objectid AND t.objectid=turnuse_id;

  -- store 2nd linearuse_lane connected to turnuse in 2nd variable
  SELECT INTO lane_end l.*
  FROM linearuse_lanes l, turnuse_ogd_aoi t
  WHERE t.use_to_id=l.lu_objectid AND t.objectid=turnuse_id;

  -- store the start point of the turnuse geometry in a variable
  turnuse_start_point := ST_Transform(ST_StartPoint(turnuse.geom),31259);
  -- store the end point of the turnuse geometry in a variable
  turnuse_end_point := ST_Transform(ST_EndPoint(turnuse.geom),31259);


-- there exist bike turnuses that connect a bike line with a car lane.
-- the problem is: the car lane most likely is not anymore at the place it used to be
-- this is because with the lane building in teh previous step, it was moved,
-- and also most likely it was duplicated.
-- so the turnuse connecting bike and car lane needs to be recalculated.
-- this is done in the following part of the code
-------- getting into the  if-clause that connects bike with car lanes (if needed)

  IF(((lane_start.basetype=1 AND lane_end.basetype<>1)
    OR (lane_start.basetype<>1 AND lane_end.basetype=1)) IS TRUE) THEN

    no_actual_carlanes := (SELECT COUNT(ll.*) FROM linearuse_lanes ll
                         WHERE ll.lu_objectid = lane_start.lu_objectid);

    FOR k in 1..no_actual_carlanes LOOP

      --getting the bikelanes that merge into car lanes
      --change their start- or endpoints so that they go..
      -- ..into the car lanes - not in the middle of them
      IF((lane_start.basetype=1 AND lane_end.basetype<>1) IS TRUE) THEN

        -- Generally: the tupels of the table linearuse_lanes are being...
        -- ...handled individually into the variable lane_start2. (it is...
        -- ...just a transfer of the data from one variable to another...
        -- ...to prevent the variables from being set incorrectly)
        SELECT INTO lane_start2 l.* FROM linearuse_lanes l
        WHERE l.lu_objectid = lane_start.lu_objectid
        ORDER BY l.fid asc LIMIT 1 OFFSET (k-1);

        RAISE NOTICE 'car lane towards cycle lane';
          RAISE NOTICE '(Car) edge now corresponds to % lanes',
          (SELECT COUNT(e.*) FROM linearuse_lanes e
           WHERE e.lu_objectid = lane_start.lu_objectid);

        -- this if-block is simply for discovering the connections of the turnuse.
        -- to which linearuses is it connected? at which linearuse does it start, ..
        -- ..at which does it end? (especially important for car lanes)
        IF (lane_start2.leads_to = node_id) THEN -- does lane_start2 (of which we know that it's a car lane) lead to the intersection and is therefore the beginning of the turnuse?
          ll_start_fid := lane_start2.fid;
          ll_end_fid := lane_end.fid;

          -- I know that in lane_start the car lane is stored. I also know that if we..
          -- ..are in this if-statement, the lane leads towards the node we are inter-
          -- ested in. Therefore, the start of the turnuse is the end of the carlane
          turnuse_start_point2 := (SELECT ST_EndPoint(ll.lu_geom) FROM linearuse_lanes ll
                                 WHERE ll.fid = ll_start_fid);

          -- in lane_end, the bike lane is stored.
          -- we are not interested in the direction of the bike lane, nor are we sure..
          -- ..whether it's directed correctly. Hence, we still use ST_Closest_Point
          turnuse_end_point2 := (SELECT ST_ClosestPoint(ll.lu_geom,
                                         (SELECT ST_Transform(n.geom, 31259)
                                          FROM node_ogd_aoi n
```

```
                                WHERE n.objectid = node_id))
                         FROM linearuse_lanes ll WHERE ll.fid = ll_end_fid);


      -- turn the whole procedure around as in this version, the car lane (lane_start)..
      -- .. is coming from the node. Therefore it is the starting point of the turnuse
      ELSE
         ll_start_fid := lane_end.fid;
         ll_end_fid := lane_start2.fid;

         turnuse_start_point2 := (SELECT ST_ClosestPoint(ll.lu_geom,
                                        (SELECT ST_Transform(n.geom, 31259)
                                         FROM node_ogd_aoi n
                                         WHERE n.objectid = node_id)
                                        )
                          FROM linearuse_lanes ll
                          WHERE ll.fid = ll_start_fid);

         turnuse_end_point2 := (SELECT ST_StartPoint(ll.lu_geom)
                         FROM linearuse_lanes ll
                         WHERE ll.fid = ll_end_fid);
      END IF;

   -- if the car lane is stored in lane_end and the bicycle lane is lane_start
   ELSIF((lane_start.basetype<>1 AND lane_end.basetype=1) IS TRUE) THEN
      SELECT INTO lane_end2 l.* FROM linearuse_lanes l
      WHERE l.lu_objectid = lane_end.lu_objectid ORDER BY l.fid LIMIT 1 OFFSET (k-1);

         RAISE NOTICE 'Bicycle lane towards car lane';
         RAISE NOTICE 'Edge corresponds now to % lanes',
         (SELECT COUNT(e.*) FROM linearuse_lanes e
          WHERE e.lu_objectid = lane_end.lu_objectid);

      -- this if-block is simply for discovering the connections of the turnuse.
      -- is the linearuse lane_end2 (of which we know that it's a car lane), ..
      -- ..the linearuse leading to the intersection?
         IF (lane_end2.leads_to = node_id) THEN
            ll_start_fid := lane_end2.fid; -- car lane is start of turnuse
         ll_end_fid := lane_start.fid; -- bike lane is end of turnuse

         -- the car lane is stored in lane_end.
         -- in this if-part, the car lane is the lane, the turnuse starts from
         turnuse_start_point2 := (SELECT ST_EndPoint(ll.lu_geom)
                          FROM linearuse_lanes ll WHERE ll.fid = ll_start_fid);
         turnuse_end_point2 := (SELECT ST_ClosestPoint(ll.lu_geom,
                                       (SELECT ST_Transform(n.geom, 31259)
                                        FROM node_ogd_aoi n
                                        WHERE n.objectid = node_id))
                         FROM linearuse_lanes ll WHERE ll.fid = ll_end_fid);

      ELSE -- the other way around: the turnuse ends in the car lane
         ll_start_fid := lane_start.fid; -- bike lane is start of turnuse
         ll_end_fid := lane_end2.fid; -- car lane is end of car lane

         turnuse_start_point2 := (SELECT ST_ClosestPoint(ll.lu_geom,
                                        (SELECT ST_Transform(n.geom, 31259)
                                         FROM node_ogd_aoi n WHERE n.objectid = node_id))
                          FROM linearuse_lanes ll WHERE ll.fid = ll_start_fid);
         turnuse_end_point2 := (SELECT ST_StartPoint(ll.lu_geom)
                         FROM linearuse_lanes ll WHERE ll.fid = ll_end_fid);
      END IF;

   END IF;

   elongated_l_geom := NULL;
   elongated_ll_geom := NULL;

--------------------------------------------------------------------------------------------
   -- finding out about the widths at the two different ends of the turnuse geometry
   IF(ST_Distance(ST_Transform(turnuse_start_point2,31259),
            ST_Transform(ST_StartPoint(lane_start2.lu_geom),31259))<0.1) THEN
      start_twidth := lane_start.width;
      lane_at_tustart := TRUE;
      RAISE NOTICE 'Startpoint und lane_start';


   ELSIF(ST_Distance(ST_Transform(turnuse_start_point2,31259),
             ST_Transform(ST_EndPoint(lane_start2.lu_geom),31259))<0.1) THEN
      start_twidth := lane_start.width;
```

```
            lane_at_tustart := TRUE;
            RAISE NOTICE 'Endpoint und lane_start';


         ELSIF(ST_Distance(ST_Transform(turnuse_start_point2,31259),
                       ST_Transform(ST_StartPoint(lane_end2.lu_geom),31259))<0.1) THEN
            start_twidth := lane_end.width;
            lane_at_tustart := FALSE;
            RAISE NOTICE 'Startpoint und lane_end';


         ELSIF(ST_Distance(ST_Transform(turnuse_start_point2,31259),
                       ST_Transform(ST_EndPoint(lane_end2.lu_geom),31259))<0.1) THEN
            start_twidth := lane_end.width;
            lane_at_tustart := FALSE;
            RAISE NOTICE 'Endpoint und lane_end';


         ELSE RAISE NOTICE 'I do not know what is happening here -- help connect';
         END IF;



      ----------------------------------------------------------------------------------


         -- depending on which end of the turnuse could not be attributed with the width, now gets it
         IF (lane_at_tustart = TRUE) THEN
            end_twidth := lane_end.width;
         ELSIF (lane_at_tustart = FALSE) THEN
            end_twidth := lane_start.width;
         ELSE
            RAISE EXCEPTION 'Width of lanes could not be connected to turnuse.';
         END IF;


         --calculating the apex point by calling the function get_intermediate_point()
         intermediate_point := NULL;
         SELECT INTO intermediate_point get_intermediate_point(
            ST_MakeLine(turnuse_start_point2, turnuse_end_point2),
            node_id);


         --building the turnuse buffer by using a convex hull
            turnuse_buff_geom := ST_ConvexHull( ST_Collect( ST_Collect(
                ST_Buffer(turnuse_start_point2, start_twidth/2),
               ST_Buffer(intermediate_point, (start_twidth + end_twidth)/4)),
                ST_Buffer(turnuse_end_point2, end_twidth/2)
            ));

         RAISE NOTICE 'TS2: %, Intermed. Pt.: %, TE2: %, Curve: %',
            ST_AsText(turnuse_start_point2), ST_AsText(intermediate_point),
            ST_AsText(turnuse_end_point2), ST_CurveToLine(
               CreateCurve(ST_MakeLine(turnuse_start_point2,
                                 ST_MakeLine(intermediate_point, turnuse_end_point2)
                                 )
                        ));

         --storing the results in table
         INSERT INTO iplateau(turnuse_objectid, turnuse_geom, turnuse_buff_geom,
                         start_point_geom, end_point_geom, linearuse_start_id,
                         linearuse_end_id, start_width, end_width, basetype,
                         curved_geom, intermediate_point, el_geom1, el_geom2)
         VALUES (turnuse_id, ST_Transform(ST_MakeLine(turnuse_start_point2,
                                        turnuse_end_point2),31259),
               turnuse_buff_geom, turnuse_start_point2, turnuse_end_point2, ll_start_fid,
               ll_end_fid, start_twidth, end_twidth, turnuse_basetype,
               ST_CurveToLine(CreateCurve(ST_MakeLine(
                  turnuse_start_point2, ST_MakeLine(intermediate_point, turnuse_end_point2)))),
               intersection_point_elongated_lls, elongated_l_geom, elongated_ll_geom);

      END LOOP;

------------------------the "normal" bike turnuses ----------------------------------

   ELSE --meaning every turnuse that is between two normal bike lanes (no car lane involved)

   elongated_l_geom := NULL;
   elongated_ll_geom := NULL;
```

```

  --CALLING THE FUNCTION HELP_CONNECT
  RAISE NOTICE 'lanestart width: %, laneend width: %, lanestart geom: %,
  laneend geom: %, tu start geom: %, tu end geom: %',
  lane_start.width, lane_end.width, ST_AsText(lane_start.lu_geom),
  ST_AsText(lane_end.lu_geom), ST_AsText(turnuse_start_point),
  ST_AsText(turnuse_end_point);

  record_hc := help_connect(lane_start.width, lane_end.width, lane_start.lu_geom,
                      lane_end.lu_geom, turnuse_start_point, turnuse_end_point);

  --storing the results in variables
  start_twidth := record_hc.start_twidth;
  lane_at_tustart := record_hc.lane_at_tustart;
  --elongated_l_geom  := record_hc.elongated_l_geom;
  --elongated_ll_geom  := record_hc.elongated_ll_geom;


  --depending on which end of the turnuse could
  --not be attributed with the width, now gets it
  IF (lane_at_tustart = TRUE) THEN
     end_twidth := lane_end.width;
  ELSIF (lane_at_tustart = FALSE) THEN
     end_twidth := lane_start.width;
  ELSE
     RAISE EXCEPTION 'Width of lanes could not be connected to turnuse.';
  END IF;

  -- store the linearuses, the turnuse borders to, in variables
  ll_start_fid := lane_start.fid; -- it is not determined, if in "lane_start" really the start of the lane is stored. Because we are dealing with bike lanes, it just might not matter as the cyclists might drive in both directions
  ll_end_fid := lane_end.fid;

  -- calculate the apex point of the curved turnuse
  intermediate_point := NULL;
  SELECT INTO intermediate_point get_intermediate_point(ST_MakeLine(
     turnuse_start_point, turnuse_end_point), node_id);


  -- building convex hull around the buffered start-, apex-, and endpoint
  turnuse_buff_geom := ST_ConvexHull( ST_Collect( ST_Collect(
       ST_Buffer(turnuse_start_point, start_twidth/2),
     ST_Buffer(intermediate_point, (start_twidth + end_twidth)/4)),
       ST_Buffer(turnuse_end_point, end_twidth/2)
   ));


  --storing the results in table
  INSERT INTO iplateau(turnuse_objectid, turnuse_geom, turnuse_buff_geom,
                  start_point_geom, end_point_geom, start_width, end_width, basetype,
                  curved_geom, intermediate_point, el_geom1, el_geom2,
                  linearuse_start_id, linearuse_end_id)
  VALUES (turnuse_id, ST_Transform(turnuse.geom,31259), turnuse_buff_geom,
        turnuse_start_point, turnuse_end_point, start_twidth, end_twidth, turnuse_basetype,
        ST_CurveToLine(CreateCurve(ST_MakeLine(turnuse_start_point, ST_MakeLine(
           intermediate_point, turnuse_end_point)))),
        intersection_point_elongated_lls, elongated_l_geom, elongated_ll_geom,
        ll_start_fid, ll_end_fid);

  END IF; --ending if-else for the bike-car-lane problem
 END IF; --ending if-else that checks whether the turnuse.basetype = 2, 7, ...
END LOOP; --ending the loop that goes through all the existing turnuses with the basetypes = 2, 7, ...




 ------------------------------------ ONLY CARS ----------------------------------------


  --creating a table by generating all the possible combinations of the
  -- linearuses (for cars)- so the table holds all the possible turnuses
  DROP TABLE IF EXISTS help_turnuses_cars CASCADE;
  CREATE  TABLE help_turnuses_cars AS
     SELECT DISTINCT l.fid as l_fid, ll.fid as ll_fid, l.lu_objectid as l_lu_objectid,
     ll.lu_objectid as ll_lu_objectid, l.lu_geom as l_lu_geom,
     ll.lu_geom as ll_lu_geom, l.width as l_width, ll.width as ll_width,
     l.basetype as basetype, l.leads_to as l_leads_to, l.comes_from as l_comes_from,
     ll.leads_to as ll_leads_to, ll.comes_from as ll_comes_from, n.objectid as node_id
     FROM linearuse_lanes l, linearuse_lanes ll, node_ogd_aoi n
```

```
      WHERE l.basetype=1 AND ll.basetype=1
         AND l.lu_objectid <> ll.lu_objectid
         AND l.fid < ll.fid -- this way opposite duplicates get caught. 2-9 and 9-2 would be the same for me
         AND n.objectid = node_id;

   --how many possible combinations are there?
   no_turnuses_car := (SELECT COUNT(*) FROM help_turnuses_cars);

   -- create table that holds all the turning relations that are avalable in GIP data
   DROP TABLE IF EXISTS available_turnuses CASCADE;
   CREATE TABLE available_turnuses AS
      SELECT t.* FROM turnuse_ogd_aoi t WHERE t.via_node_id=node_id
      AND t.basetype=1; --is used for determining all the turnuse options that are "available" for cars

   -- get number of availabe turning relations
   no_available_turnuses := (SELECT COUNT(*) FROM available_turnuses);

    --looping through all the possible turnuses
    FOR p IN 1.. no_turnuses_car LOOP

      --transferring always in each loop one tuple of the possible turnuses
      turnuse_car:=NULL;
      SELECT INTO turnuse_car htc.* FROM help_turnuses_cars htc
      ORDER BY l_fid, ll_fid
      LIMIT 1
      OFFSET (p-1);


    --the following if-clause makes sure that the turnuses do not connect lanes
    -- with each other that have different driving directions
    IF(turnuse_car.l_leads_to <> turnuse_car.ll_leads_to
      AND turnuse_car.l_comes_from <> turnuse_car.ll_comes_from) THEN

      ---------------------------------------------------------------------------

      IF (turnuse_car.l_leads_to = node_id)  THEN
         turnuse_car_point1 := ST_EndPoint(turnuse_car.l_lu_geom);
         --creating the elongated version of the first segment of the linearuse curve in order to determine on which side the curvature of the turnuse needs to be
         elongated_l_geom := elongate_linearuse(ST_Transform(ST_StartPoint(turnuse_car.l_lu_geom),31259), ST_Transform(ST_PointN(turnuse_car.l_lu_geom,2),31259));
      ELSE
         turnuse_car_point2 := ST_StartPoint(turnuse_car.l_lu_geom);
         --creating the elongated version of the last segment of the linearuse (for intermediate point)
         elongated_l_geom := elongate_linearuse(ST_Transform(ST_EndPoint(turnuse_car.l_lu_geom),31259), ST_Transform(ST_PointN(turnuse_car.l_lu_geom,-2),31259));
      END IF;


      IF (turnuse_car.ll_leads_to = node_id) THEN
         turnuse_car_point1 := ST_EndPoint(turnuse_car.ll_lu_geom);
         --RAISE NOTICE 'Loop %; Es wurde ll.start mit einer Distanz von %', p, ST_Distance(ST_Transform(node.geom,31259), ST_Transform(ST_StartPoint(turnuse_car.ll_lu_geom),31259));
         elongated_ll_geom := elongate_linearuse(ST_Transform(ST_StartPoint(turnuse_car.ll_lu_geom),31259), ST_Transform(ST_PointN(turnuse_car.ll_lu_geom,2),31259));
      ELSE
         turnuse_car_point2 := ST_StartPoint(turnuse_car.ll_lu_geom);
         --RAISE NOTICE 'Loop %; Es wurde ll.end mit einer Distanz von %', p, ST_Distance(ST_Transform(node.geom,31259), ST_Transform(ST_EndPoint(turnuse_car.ll_lu_geom),31259));
         elongated_ll_geom := elongate_linearuse(ST_Transform(ST_EndPoint(turnuse_car.ll_lu_geom),31259), ST_Transform(ST_PointN(turnuse_car.ll_lu_geom,-2),31259));
      END IF;

      ---------------------------------------------------------------------------
      -- i want the turnuse to hold the FIDs of the linearuse_lanes which it connects
      -- we know where it starts and ends because we have l_comes_from, l_leads_to and
      -- the node_id. the geometries of the linear uses are assigned accordingly.
      -- by re-connecting the ids, it is possible to get the fid
      -- Note: it is an unfortunate coincidence that both variable begin with "ll".
      -- they are not connected in any other way than explained above.
```

```
      IF (turnuse_car.l_leads_to = node_id) THEN -- if linearuse 1 leads to the node, it is also the start point of the turnuse...
         ll_start_fid := turnuse_car.l_fid;
         ll_end_fid := turnuse_car.ll_fid; -- ... and the start point of linearuse 2 is the end point of the turnuse
      ELSIF (turnuse_car.ll_leads_to = node_id) THEN
         ll_start_fid := turnuse_car.ll_fid;
         ll_end_fid := turnuse_car.l_fid;
      ELSE
         ll_start_fid := 0;
         ll_end_fid := 0;
      END IF;


      --construct the start- and end point of each buffer in order
      -- to build a convex hull around them in the end
      turnuse_start_point := NULL;
      turnuse_end_point := NULL;

      turnuse_start_point := ST_Transform(turnuse_car_point1, 31259);
      turnuse_end_point := ST_Transform(turnuse_car_point2, 31259);


      --check whether the constructed turnuse also exists in the data
      FOR o IN 1..no_available_turnuses LOOP

         SELECT INTO var_available_turnuses at.* FROM available_turnuses at
            ORDER BY fid
            LIMIT 1
            OFFSET (o-1);

         IF ((turnuse_car.l_lu_objectid = var_available_turnuses.use_from_id
             AND turnuse_car.ll_lu_objectid = var_available_turnuses.use_to_id)
               OR(turnuse_car.l_lu_objectid = var_available_turnuses.use_to_id
                  AND turnuse_car.ll_lu_objectid = var_available_turnuses.use_from_id)) THEN


            --calculate the intersection point of the elongated linearuses... it will be needed to get the intermediate point
            RAISE NOTICE 'Startpoint: %, Endpoint: %', turnuse_car_point1, turnuse_car_point2;
            intersection_point_elongated_lls := NULL;
            RAISE NOTICE 'elongate1: %, elongate2: %', ST_AsText(elongated_l_geom),ST_AsText(elongated_ll_geom);
            intersection_point_elongated_lls := ST_Intersection(ST_Transform(elongated_l_geom,31259), ST_Transform(elongated_ll_geom,31259));
            --get intermediate point of the constructed turnuse in order to round off the edges
            intermediate_point := NULL;
            --calling the function get_intermediate_point and store its result in the
            -- variable intermediate_point
            SELECT INTO intermediate_point get_intermediate_point(ST_Transform(
               ST_MakeLine(turnuse_car_point1, turnuse_car_point2),31259), node_id);

            -- build the 2D geometry with the help of buffers and convex hulls
            turnuse_buff_geom := ST_ConvexHull( ST_Collect( ST_Collect(
                ST_Buffer(turnuse_start_point, turnuse_car.l_width/2),
               ST_Buffer(intermediate_point, (turnuse_car.l_width+turnuse_car.ll_width)/4)),
                ST_Buffer(turnuse_end_point, turnuse_car.ll_width/2)
            ));

            -- store result in the table iplateau
            -- the 1D versions (straight and curved turning relations)
            -- are created in this insert-statement on the fly
            INSERT INTO iplateau(turnuse_geom, start_point_geom, end_point_geom,
                                start_width, end_width, basetype, turnuse_buff_geom,
                                curved_geom, intermediate_point, el_geom1, el_geom2,
                                linearuse_start_id, linearuse_end_id)
            VALUES (ST_Transform(ST_MakeLine(turnuse_car_point1, turnuse_car_point2),31259),
                  turnuse_car_point1, turnuse_car_point2, turnuse_car.l_width,
                  turnuse_car.ll_width, 1, turnuse_buff_geom,
                  ST_CurveToLine(CreateCurve(ST_MakeLine(turnuse_start_point, ST_MakeLine(
                     intermediate_point, turnuse_end_point)))), intermediate_point,
                  elongated_l_geom, elongated_ll_geom,ll_start_fid, ll_end_fid);
         END IF;
      END LOOP;

    END IF;
   END LOOP;
```

```
    -- the turnuses belonging to the car lanes (basetype = 1) need to be assigned
    -- the objectid of the original turnuses they belong to.
    -- this has to be done separately because those turnuses are not directly
    -- derived from the original turnuse.
    -- however, this information is needed in the risk analysis.
    UPDATE iplateau
    SET turnuse_objectid = (SELECT t.objectid
                      FROM turnuse_ogd_aoi t, linearuse_lanes ll, linearuse_lanes ll2,
                      linearuse_ogd_aoi l, linearuse_ogd_aoi l2
                         WHERE linearuse_start_id = ll.fid AND linearuse_end_id = ll2.fid
                            AND ll.lu_objectid = l.objectid AND ll2.lu_objectid = l2.objectid
                            AND ((l.objectid = t.use_to_id AND l2.objectid = t.use_from_id)
                            OR (l.objectid = t.use_from_id AND l2.objectid = t.use_to_id)))
    WHERE iplateau.basetype = 1
       AND iplateau.turnuse_objectid IS NULL;



    END IF; -- if clause that asks whether the edgedegree is > 1.

  END;
$$ LANGUAGE 'plpgsql';
```

Listing A.7: Creating turning relations with the help of this SQL-function (help_connect.sql)

```sql
--this function does the following:
--assigns the (lane)width to the startpoint of the turnuse
--finds out, which lane(linearuse) is at the startpoint of the turnuse

DROP FUNCTION IF EXISTS help_connect(double precision, double precision,
                            geometry, geometry, geometry, geometry);
CREATE OR REPLACE FUNCTION help_connect(double precision, double precision,
                              geometry, geometry, geometry, geometry)
RETURNS record AS $$
  DECLARE
   lane_start_width ALIAS FOR $1;
   lane_end_width ALIAS FOR $2;
   lane_start_lugeom ALIAS FOR $3;
   lane_end_lugeom ALIAS FOR $4;
   turnuse_start_point ALIAS FOR $5;
   turnuse_end_point ALIAS FOR $6;
   start_twidth double precision;
   lane_at_tustart boolean;
   elongated_l_geom geometry;
   elongated_ll_geom geometry;
   rec record;

BEGIN


   IF(ST_Distance(ST_Transform(turnuse_start_point,31259),
      ST_Transform(ST_StartPoint(lane_start_lugeom),31259))<0.1) THEN
      start_twidth := lane_start_width;
      lane_at_tustart := TRUE;
      --the whole elongated stuff is to find out where the linearuses would meet. this is
      used to determine in which direction the turnuse needs to curve
      elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,2),31259));
      IF(ST_Distance(ST_Transform(turnuse_end_point,31259),
      ST_Transform(ST_StartPoint(lane_end_lugeom),31259))<0.1) THEN
         elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,2),31259));
      ELSE
         elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,-2),31259));
      END IF;
      RAISE NOTICE 'Startpoint und lane_start';
      SELECT start_twidth, lane_at_tustart, elongated_l_geom, elongated_ll_geom INTO rec;
      RETURN rec;

   ELSIF(ST_Distance(ST_Transform(turnuse_start_point,31259),
      ST_Transform(ST_EndPoint(lane_start_lugeom),31259))<0.1) THEN
      start_twidth := lane_start_width;
      lane_at_tustart := TRUE;
      elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,-2),31259));
      IF(ST_Distance(ST_Transform(turnuse_end_point,31259),
      ST_Transform(ST_StartPoint(lane_end_lugeom),31259))<0.1) THEN
         elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,2),31259));
      ELSE
         elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,-2),31259));
      END IF;
      RAISE NOTICE 'Endpoint und lane_start';
      SELECT start_twidth, lane_at_tustart, elongated_l_geom, elongated_ll_geom INTO rec;
      RETURN rec;

   ELSIF(ST_Distance(ST_Transform(turnuse_start_point,31259),
      ST_Transform(ST_StartPoint(lane_end_lugeom),31259))<0.1) THEN
      start_twidth := lane_end_width;
      lane_at_tustart := FALSE;
      elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,2),31259));
```

```
      IF(ST_Distance(ST_Transform(turnuse_end_point,31259),
      ST_Transform(ST_StartPoint(lane_start_lugeom),31259))<0.1) THEN
         elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,-2),31259));
      ELSE
         elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,2),31259));
      END IF;
      RAISE NOTICE 'Startpoint und lane_end';
      SELECT start_twidth, lane_at_tustart, elongated_l_geom, elongated_ll_geom INTO rec;
      RETURN rec;

   ELSIF(ST_Distance(ST_Transform(turnuse_start_point,31259),
      ST_Transform(ST_EndPoint(lane_end_lugeom),31259))<0.1) THEN
      start_twidth := lane_end_width;
      lane_at_tustart := FALSE;
      elongated_ll_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_end_lugeom),31259),
      ST_Transform(ST_PointN(lane_end_lugeom,-2),31259));
      IF(ST_Distance(ST_Transform(turnuse_start_point,31259),
      ST_Transform(ST_EndPoint(lane_end_lugeom),31259))<0.1) THEN
         elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_StartPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,2),31259));
      ELSE
         elongated_l_geom :=
      elongate_linearuse(ST_Transform(ST_EndPoint(lane_start_lugeom),31259),
      ST_Transform(ST_PointN(lane_start_lugeom,-2),31259));
      END IF;
      RAISE NOTICE 'Endpoint und lane_end';
      SELECT start_twidth, lane_at_tustart, elongated_l_geom, elongated_ll_geom INTO rec;
      RETURN rec;

   ELSE
      RAISE NOTICE 'help, whats happening?';
      RAISE NOTICE 'Values are assumed now to prevent errors!';

      start_twidth := lane_end_width;
      lane_at_tustart := FALSE;
      SELECT start_twidth, lane_at_tustart, NULL, NULL INTO rec;
      RETURN rec;

   END IF;



END;
$$ LANGUAGE 'plpgsql';
```

Determining the point in the middle of the turning relation that helps curving the geometry.

Listing A.8: Determining the intermediate point of turning relation (`get_intermediate_point.sql`)

```sql
--Calculate radius from distance between points
--create buffer with this radius around start and end point
--see where the two buffers meet
--select the point which is closer to the edge point
--take the three points, start-, intermediate- and endpoint as input for function CreateCurve
--then create a ConvexHull around this line, and the buffers around the start and end points.

DROP FUNCTION IF EXISTS get_intermediate_point(geometry, integer, geometry);
CREATE OR REPLACE FUNCTION get_intermediate_point(geometry, integer, geometry)
RETURNS geometry AS $$
   DECLARE
      turnuse_geom ALIAS FOR $1;
      node_id ALIAS FOR $2;
      intersection_p_linearuses ALIAS FOR $3; --point geometry
      distance_p1 double precision;
      distance_p2 double precision;
      intersection_p1 geometry;
      intersection_p2 geometry;
      side_p1 double precision;
      side_p2 double precision;
      side_int_p double precision;
      x double precision;
      y double precision;
      node_geom geometry;

BEGIN
   RAISE NOTICE 'srid turnuse_geom: %', st_srid(turnuse_geom);

WITH
   --get the radius of the Exterior Rings that are to be built
   --by determining the distance between start and endpoint of turnuse
   radius AS (
      SELECT ST_Distance(ST_Transform(ST_StartPoint(turnuse_geom),31259),
                  ST_Transform(ST_EndPoint(turnuse_geom),31259))*0.505 as t),

   --building buffers with the radius r and extracting their exterior rings
   -- to get circles around each point.
   buffers AS (
      SELECT ST_ExteriorRing(ST_Buffer(ST_Transform(ST_StartPoint(turnuse_geom),
                                    31259),r.t)) as st,
      ST_ExteriorRing(ST_Buffer(ST_Transform(ST_EndPoint(turnuse_geom),31259),r.t))
      as en
         FROM radius r),

   --finding out, where the circles around the points intersect
   intersectionpoints AS(
      SELECT (ST_Dump(ST_Intersection(st , en))).geom
      FROM buffers), --delivers multipoints

   --dividing the multipoints and arranging it to a table
   -- that holds the fid, the first and the second point
   dividedpoints AS (
      SELECT DISTINCT ip.geom as geom1, ip2.geom as geom2
      FROM intersectionpoints ip, intersectionpoints ip2
      WHERE ST_AsText(ip.geom) < ST_AsText(ip2.geom)
    )

   --selecting the points in variables
   SELECT INTO distance_p1, distance_p2, intersection_p1, intersection_p2,
   side_p1, side_p2, side_int_p, node_geom
      --take the elongated linearuse to determine the intermediate point
      ST_Distance(p.geom1,intersection_p_linearuses),
      ST_Distance(p.geom2, intersection_p_linearuses),
      p.geom1, p.geom2,
      side_of_line(turnuse_geom, p.geom1), side_of_line(turnuse_geom, p.geom2),
      side_of_line(turnuse_geom, intersection_p_linearuses),
      ST_Transform(n.geom,31259)

   FROM dividedpoints p, node_ogd_aoi n WHERE n.objectid = node_id;


   RAISE NOTICE 'intersection_p1: %, intersection_p2: %,
   sol1: %, sol2: %, solInt: %',
   ST_AsText(intersection_p1), ST_AsText(intersection_p2),
   side_p1, side_p2, side_int_p;
```

```

    IF(ROUND(side_p1) = ROUND(side_int_p)) THEN
       RAISE NOTICE 'ip1: %', ST_AsText(intersection_p1);
       RETURN intersection_p1;
    ELSIF(ROUND(side_p2) = ROUND(side_int_p)) THEN
       RAISE NOTICE 'ip2: %', ST_AsText(intersection_p2);
       RETURN intersection_p2;
    ELSIF(side_int_p IS NULL) THEN
       x := ST_X(intersection_p2)+(ST_X(intersection_p1)-ST_X(intersection_p2))/2;
       RAISE NOTICE 'X: %', x;
       y := ST_Y(intersection_p2)+(ST_Y(intersection_p1)-ST_Y(intersection_p2))/2;
       RAISE NOTICE 'Y: %', y;
       RETURN ST_PointFromText('POINT('||x||' '||y||')', 31259);
    ELSE
       RAISE NOTICE 'NODE is used';
       RETURN node_geom;


    END IF;

   END;
$$ LANGUAGE 'plpgsql';
```

This function takes a linestring of 3 points and turns it into a compound curve, making the turning relation look curved.

Listing A.9: Creating a curve from a linestring. (`createcurve.sql`)

```sql
--Credits: Gabor Farkas (https://gaborfarkas.github.io/)
--Source:
    https://gis.stackexchange.com/questions/56835/how-to-perform-sia-or-bezier-line-smoothing-in-postgis
CREATE OR REPLACE FUNCTION CreateCurve(geom geometry, percent int DEFAULT 40)
    RETURNS geometry AS
$$
DECLARE
    result text;
    p0 geometry;
    p1 geometry;
    p2 geometry;
    intp geometry;
    tempp geometry;
    geomtype text := ST_GeometryType(geom);
    factor double precision := percent::double precision / 200;
    i integer;
BEGIN
    IF percent < 0 OR percent > 100 THEN
        RAISE EXCEPTION 'Smoothing factor must be between 0 and 100';
    END IF;
    IF geomtype != 'ST_LineString' OR factor = 0 THEN
        RETURN geom;
    END IF;
    result := 'COMPOUNDCURVE((';
    p0 := ST_PointN(geom, 1);
    IF ST_NPoints(geom) = 2 THEN
        p1:= ST_PointN(geom, 2);
        result := result || ST_X(p0) || ' ' || ST_Y(p0) || ',' || ST_X(p1) || ' ' || ST_Y(p1)
     || '))';
    ELSE
        FOR i IN 2..(ST_NPoints(geom) - 1) LOOP
            p1 := ST_PointN(geom, i);
            p2 := ST_PointN(geom, i + 1);
            result := result || ST_X(p0) || ' ' || ST_Y(p0) || ',';
            tempp := ST_LineInterpolatePoint(ST_MakeLine(p1, p0), factor);
            p0 := ST_LineInterpolatePoint(ST_MakeLine(p1, p2), factor);
            intp := ST_LineInterpolatePoint(
                ST_MakeLine(
                    ST_LineInterpolatePoint(ST_MakeLine(p0, p1), 0.5),
                    ST_LineInterpolatePoint(ST_MakeLine(tempp, p1), 0.5)
                ), 0.5);
            result := result || ST_X(tempp) || ' ' || ST_Y(tempp) || '),CIRCULARSTRING(' ||
     ST_X(tempp) || ' ' || ST_Y(tempp) || ',' || ST_X(intp) || ' ' ||
            ST_Y(intp) || ',' || ST_X(p0) || ' ' || ST_Y(p0) || '),(';
        END LOOP;
        result := result || ST_X(p0) || ' ' || ST_Y(p0) || ',' || ST_X(p2) || ' ' || ST_Y(p2)
     || '))';
    END IF;
    RETURN ST_SetSRID(result::geometry, ST_SRID(geom));
END;
$$
LANGUAGE 'plpgsql' IMMUTABLE;
```

Executing this script, relationships in the form of foreign keys are created. The tables concerned are the ones dropped and created in the process of data preparation.

Listing A.10: Creating foreign keys between data preparation tables (`add_foreignkeys.sql`)

```
-- -- Conventions naming: {tablename}_{columnname(s)}_{suffix}
--
-- table available_turnuses
ALTER TABLE available_turnuses ADD CONSTRAINT availableturnuses_basetype_fkey
FOREIGN KEY (basetype) REFERENCES lut_basetype (id);

ALTER TABLE available_turnuses ADD CONSTRAINT availableturnuses_vianodeid_fkey
FOREIGN KEY (via_node_id) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE available_turnuses ADD CONSTRAINT availableturnuses_usetoid_fkey
FOREIGN KEY (use_to_id) REFERENCES linearuse_ogd_aoi(objectid);

ALTER TABLE available_turnuses ADD CONSTRAINT availableturnuses_usefromid_fkey
FOREIGN KEY (use_from_id) REFERENCES linearuse_ogd_aoi(objectid);


-- table help_turnuses_cars
ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_lfid_fkey
FOREIGN KEY (l_fid) REFERENCES linearuse_lanes(fid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_llfid_fkey
FOREIGN KEY (ll_fid) REFERENCES linearuse_lanes(fid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_lluobjectid_fkey
FOREIGN KEY (l_lu_objectid) REFERENCES linearuse_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_llluobjectid_fkey
FOREIGN KEY (ll_lu_objectid) REFERENCES linearuse_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_basetype_fkey
FOREIGN KEY (basetype) REFERENCES lut_basetype(id);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_lleadsto_fkey
FOREIGN KEY (l_leads_to) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_lcomesfrom_fkey
FOREIGN KEY (l_comes_from) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_llleadsto_fkey
FOREIGN KEY (ll_leads_to) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_llcomesfrom_fkey
FOREIGN KEY (ll_comes_from) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE help_turnuses_cars ADD CONSTRAINT helpturnusescars_nodeid_fkey
FOREIGN KEY (node_id) REFERENCES node_ogd_aoi(objectid);


-- iplateau
ALTER TABLE iplateau ADD CONSTRAINT iplateau_turnuseobjectid_fkey
FOREIGN KEY (turnuse_objectid) REFERENCES turnuse_ogd_aoi(objectid);

ALTER TABLE iplateau ADD CONSTRAINT iplateau_nodeid_fkey
FOREIGN KEY (node_id) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE iplateau ADD CONSTRAINT iplateau_linearusestartid_fkey
FOREIGN KEY (linearuse_start_id) REFERENCES linearuse_lanes(fid)
ON DELETE SET NULL;

ALTER TABLE iplateau ADD CONSTRAINT iplateau_linearuseendid_fkey
FOREIGN KEY (linearuse_end_id) REFERENCES linearuse_lanes(fid)
ON DELETE SET NULL;

ALTER TABLE iplateau ADD CONSTRAINT iplateau_basetype_fkey
FOREIGN KEY (basetype) REFERENCES lut_basetype(id);


-- linearuse_lanes
ALTER TABLE linearuse_lanes ADD CONSTRAINT linearuselanes_luobjectid_fkey
FOREIGN KEY (lu_objectid) REFERENCES linearuse_ogd_aoi(objectid);

ALTER TABLE linearuse_lanes ADD CONSTRAINT linearuselanes_linkid_fkey
FOREIGN KEY (link_id) REFERENCES gip_linknetz_ogd_aoi(link_id);

ALTER TABLE linearuse_lanes ADD CONSTRAINT linearuselanes_basetype_fkey
FOREIGN KEY (basetype) REFERENCES lut_basetype(id);

```

```sql
ALTER TABLE linearuse_lanes ADD CONSTRAINT linearuselanes_leadsto_fkey
FOREIGN KEY (leads_to) REFERENCES node_ogd_aoi(objectid);

ALTER TABLE linearuse_lanes ADD CONSTRAINT linearuselanes_comesfrom_fkey
FOREIGN KEY (comes_from) REFERENCES node_ogd_aoi(objectid);


-- parking_strips
ALTER TABLE parking_strips ADD CONSTRAINT parkingstrips_edgeid_fkey
FOREIGN KEY (edge_id) REFERENCES edge_ogd_aoi(objectid);


-- risk_value_table
ALTER TABLE risk_value_table ADD CONSTRAINT riskvaluetable_turnuseobjectid_fkey
FOREIGN KEY (turnuse_objectid) REFERENCES turnuse_ogd_aoi(objectid);

ALTER TABLE risk_value_table ADD CONSTRAINT riskvaluetable_basetype_fkey
FOREIGN KEY (basetype) REFERENCES lut_basetype(id);

ALTER TABLE risk_value_table ADD CONSTRAINT riskvaluetable_nodeid_fkey
FOREIGN KEY (nodeid) REFERENCES node_ogd_aoi(objectid);
```

This function collects the information needed for building the risk model. It calls helper function which retrieve the data and then stores the obtained data in a table.

Listing A.11: Getting the data used for the risk model (`get_risk_factors.sql`)

```
DROP FUNCTION IF EXISTS get_risk_factors(integer);
CREATE OR REPLACE FUNCTION get_risk_factors(integer)
RETURNS void AS $$
  DECLARE
    node_id ALIAS FOR $1;
    no_turnuses integer;
    no_edges integer;
    v_edge_id bigint;
    turnuse_id integer;
    v_traffic_lights boolean;
    v_intersection_legs integer;
    v_speed integer;
    v_round_about boolean;
    v_urban boolean;
    v_rails boolean;
    v_gradient real;
    v_no_car_lanes integer;
    v_no_bike_lanes integer;
    v_cyclinfra_in_ra boolean;
    v_road_type boolean;
    v_inner_circle_ra double precision;
    v_mixed_traffic_f varchar(10);
    v_mixed_traffic_t varchar(10);
    v_cycle_infrastructure_f varchar(45);
    v_cycle_infrastructure_t varchar(45);

  BEGIN

---- 0 - DELETE OLD VALUES FROM TABLE
    -- table in which all the values are stored that are obtained in this function
    DELETE FROM risk_value_table;

    -- fill fid, turnuse_objectid and basetype with the values of the table iplateau
    no_turnuses := (SELECT COUNT(*) FROM iplateau);
    FOR q IN 1..no_turnuses LOOP
       INSERT INTO risk_value_table (fid, turnuse_objectid, basetype)
       VALUES (
          (SELECT fid FROM iplateau ORDER BY fid LIMIT 1 OFFSET q-1),
          (SELECT turnuse_objectid FROM iplateau ORDER BY fid LIMIT 1 OFFSET q-1),
          (SELECT basetype FROM iplateau ORDER BY fid LIMIT 1 OFFSET q-1)
       );
    END LOOP;

    UPDATE risk_value_table SET nodeid = node_id;


------- 1 - SURROUNDING INFRASTRUCTURE
    -- 1.1 - urban | checks whether one of the links which are leading to the intersection
       is declared urban in the GIP
    -- Notice: in the area of interest, all of the links are urban
    v_urban := check_urban(node_id);
    UPDATE risk_value_table SET urban = v_urban;
    RAISE NOTICE 'Urban?: %', v_urban;

    -- 1.2 - trains | checks whether there are any rails in the radius of the intersection
    v_rails := check_rails(node_id);
    UPDATE risk_value_table SET rails = v_rails;
    RAISE NOTICE 'Rails?: %', v_rails;

------- 2 - TRAFFIC RULES
    -- 2.1 - traffic lights | compares the node id that is given with a table which
       contains all traffic lights in the AOI. I've inserted the intersection's id that the
       traffic light belongs to in the table. So what is done is simply checking whether the
       node id can be found in the TL table.
    v_traffic_lights := check_traffic_lights(node_id);
    UPDATE risk_value_table SET traffic_lights = v_traffic_lights;
    RAISE NOTICE 'Traffic Lights?: %', v_traffic_lights;


------- 3 - INTERSECTION GEOMETRY
    -- 3.1 - number of legs of an intersection
    v_intersection_legs := (SELECT edgedegree FROM node_ogd_aoi n WHERE n.objectid = node_id);
    UPDATE risk_value_table SET intersection_legs = v_intersection_legs;
    RAISE NOTICE 'Edge degree: %', v_intersection_legs;

------- 4 - BEHAVIOR
    v_speed := get_speed(node_id);
```

```
    UPDATE risk_value_table SET speed = v_speed;
    RAISE NOTICE 'Highest speed: %', v_speed;


  ------- 5 - STREET CHARACTERISTICS
    -- 5.1 - number of lanes: implemented in the local factors segment
    -- 5.2 - gradient, slope implemented in teh local factors segment.
    -- 5.3 - physical barriers:
    -- 5.4 - road types:
    -- 5.4: Road Type
    v_road_type := check_streettypes(v_edge_id); -- get info of minor/major road
    UPDATE risk_value_table SET street_type = v_road_type;

  ------- 6 - ROUNDABOUT
    -- 6.1 - the radius of the inner circle - is in the if-clause below.
    -- 6.2 - is it a roundabout?
    v_round_about := check_roundabout(node_id);
    UPDATE risk_value_table SET round_about = v_round_about;
    RAISE NOTICE 'Roundabout?: %', v_round_about;


    IF v_round_about IS TRUE THEN -- the function is only called if one of the parts of the
      intersection belongs to a roundabout.
      -- 6.1 - the radius of the inner circle
      v_inner_circle_ra := check_inner_circle_ra(node_id);
      UPDATE risk_value_table SET round_about_inner_circle = v_inner_circle_ra;
      -- 6.3 - is there cycling infrastructure in the roundabout?
      v_cyclinfra_in_ra := check_cyclinfra_in_ra(node_id);
      UPDATE risk_value_table SET round_about_cycle_infstr = v_cyclinfra_in_ra;
    END IF;


  ------- 7 - CYCLING INFRASTRUCTURE
    -- 7.1 mixed traffic - is called in get local factors
    -- 7.2 cycling infrastructure - is called in get local factors

  ------- GET THE LOCAL FACTORS ----------------------------------------
    -- get the number of edges
    no_edges := (SELECT COUNT(*) FROM (SELECT COUNT(e.*) FROM iplateau i, linearuse_ogd_aoi
      l, linearuse_lanes ll, edge_ogd_aoi e
           WHERE i.linearuse_start_id = ll.fid
             AND ll.lu_objectid = l.objectid
             AND l.edge_id = e.objectid
           GROUP BY e.objectid) as no_of_linuses_belonging_to_edge);

    -- loop through edges and get edge-related factors
    FOR q IN 1..no_edges LOOP
      v_edge_id := (SELECT e.objectid FROM edge_ogd_aoi e WHERE e.nodefromid = node_id OR
      e.nodetoid = node_id ORDER BY e.objectid LIMIT 1 OFFSET q-1);

      -- 5.2: Gradient
      v_gradient := check_gradient_slope(v_edge_id); -- get gradient of edge

      UPDATE risk_value_table SET gradient = v_gradient
      FROM iplateau, linearuse_lanes, linearuse_ogd_aoi
         WHERE risk_value_table.turnuse_objectid = iplateau.turnuse_objectid
         AND iplateau.linearuse_start_id = linearuse_lanes.fid
         AND linearuse_lanes.lu_objectid = linearuse_ogd_aoi.objectid
         AND linearuse_ogd_aoi.edge_id = v_edge_id;

    END LOOP; --end loop through edges


    -- loop throught turnuses and get turnuse-related factors
    FOR q IN 1..no_turnuses LOOP
      turnuse_id := (SELECT fid FROM iplateau ORDER BY fid LIMIT 1 OFFSET q-1);

      -- 5.1.1: Number of crossed car lanes
      v_no_car_lanes := car_lanes_crossed(turnuse_id);
      UPDATE risk_value_table SET lane_number_cars = v_no_car_lanes WHERE fid = turnuse_id;

      -- 5.1.2: Number of crossed pathways and cycle infrastructure
      v_no_bike_lanes := bike_lanes_crossed(turnuse_id);
      UPDATE risk_value_table SET lane_number_bikes = v_no_bike_lanes WHERE fid = turnuse_id;

      -- 7.1 mixed traffic: get it for the linearuse before and the linearuse after the
      turnuse
      v_mixed_traffic_f := check_mixed_traffic((SELECT linearuse_start_id FROM iplateau WHERE
      fid = turnuse_id), node_id); -- before turnuse
      UPDATE risk_value_table SET mixed_traffic_f = v_mixed_traffic_f WHERE fid = turnuse_id;
      v_mixed_traffic_t := check_mixed_traffic((SELECT linearuse_end_id FROM iplateau WHERE
      fid = turnuse_id), node_id);
```

```
      UPDATE risk_value_table SET mixed_traffic_t = v_mixed_traffic_t WHERE fid = turnuse_id;

      -- 7.2 cycling infrastructure: get it for the linearuse before and the linearuse after the turnuse
      v_cycle_infrastructure_f := check_cycl_infra(turnuse_id, (SELECT linearuse_start_id FROM iplateau WHERE fid = turnuse_id), node_id); -- before turnuse
      UPDATE risk_value_table SET cycle_infrastructure_f = v_cycle_infrastructure_f WHERE fid = turnuse_id;
      v_cycle_infrastructure_t := check_cycl_infra(turnuse_id, (SELECT linearuse_end_id FROM iplateau WHERE fid = turnuse_id), node_id); -- after turnuse
      UPDATE risk_value_table SET cycle_infrastructure_t = v_cycle_infrastructure_t WHERE fid = turnuse_id;

   END LOOP; --end loop through turnuses

  END;
$$ LANGUAGE 'plpgsql';
```

This script contains multiple functions, the helper functions. They are called by `get_risk_factors.sql` and return information about an intersection/ a turning relation.

Listing A.12: Helper Functions: Return a specific information about the intersection (`helper_functions.sql`)

```
------- 1 - ENVIRONMENT ---------------------------------------------
---------- 1.1 URBAN INTERSECTION? ----------------------------
DROP FUNCTION IF EXISTS check_urban(integer);
CREATE OR REPLACE FUNCTION check_urban(integer)
RETURNS boolean AS $$
  DECLARE
   node_id ALIAS FOR $1;
   urban boolean;
   no_urban_links integer;

  BEGIN
   -- get number of links that are part of a round about (formofway = 4)
   no_urban_links := (SELECT COUNT(l.*) FROM gip_linknetz_ogd_aoi l
                   WHERE (l.from_node = node_id OR l.to_node = node_id) AND l.urban = 1);

   IF (no_urban_links > 0) THEN
      urban = TRUE; -- if one or more links with fow = 4 were found and counted in
      no_ra_links, the function returns true
   ELSE
      urban = FALSE; -- else, false is returned
   END IF;

   return urban; -- return whether the intersection belongs to a round about or not.

  END;
$$ LANGUAGE 'plpgsql';


------- 1.2 TRAIN IN ENVIRONMENT? -----------------------------
DROP FUNCTION IF EXISTS check_rails(integer);
CREATE OR REPLACE FUNCTION check_rails(integer)
RETURNS boolean AS $$
  DECLARE
   node_id ALIAS FOR $1;
   rails boolean;
   no_train_links integer;
   radius integer;

  BEGIN
   radius := 5; -- according to Harris et al. (2013)
   -- get number of edges (which are rails) that are in the
   -- radius of the intersection which was determined
   no_train_links := (
                   SELECT COUNT(e.*)
                   FROM edge_ogd_aoi e, node_ogd_aoi n
                   WHERE n.objectid = node_id AND
                         (e.frc = 101 OR e.frc = 102 OR e.frc = 103) AND
                         ST_Intersects(ST_Buffer(ST_Transform(n.geom, 31256), radius),
                                   ST_Transform(e.geom, 31256))
                );

   IF (no_train_links > 0) THEN
      rails = TRUE; -- if one or more edges are trail-related, true is returned
   ELSE
      rails = FALSE; -- else, false is returned
   END IF;

   return rails; -- return whether the intersection belongs to a round about or not.

  END;
$$ LANGUAGE 'plpgsql';


------- 2 - TRAFFIC RULES ------------------------------------------

------- 2.1.1 - TRAFFIC LIGHTS NEAREST NEIGHBOR --------------
-- helps to connect traffic lights and intersection
-- this function looks for the nearest neighbor of traffic lights in the AOI
-- (within a 15 meter radius). If it found a nearest neighbor, it writes the
-- intersection id into the traffic light's table in the attribute node_id.
DROP FUNCTION IF EXISTS nearest_neighbour_tl();
CREATE OR REPLACE FUNCTION nearest_neighbour_tl()
RETURNS void AS $$
  DECLARE
    no_lights INTEGER;
    l_node_id BIGINT;

```

```
  BEGIN

   no_lights := (SELECT COUNT(*) FROM rd_traffic_lights);

   FOR q in 1..no_lights LOOP

      l_node_id := (SELECT n.objectid
      FROM rd_traffic_lights tl, node_ogd_aoi n
      WHERE tl.id = q AND ST_DWithin(ST_Transform(n.geom, 31256), ST_Transform(tl.geom, 31256), 15)
      ORDER BY ST_Transform(n.geom, 31256) <-> ST_Transform(tl.geom, 31256)
      LIMIT 1);

      UPDATE rd_traffic_lights tl SET node_id = l_node_id WHERE tl.id = q;

   END LOOP;

  END;
$$ LANGUAGE 'plpgsql';

-------------- ADDITIONAL traffic light function -------------
---- 2.1 - check whether there is a traffic light "connected" to the intersection
DROP FUNCTION IF EXISTS check_traffic_lights(integer);
CREATE OR REPLACE FUNCTION check_traffic_lights(integer)
RETURNS boolean AS $$
  DECLARE
   nodeid ALIAS FOR $1;
   tl boolean;

  BEGIN

   IF ((SELECT COUNT(*) FROM rd_traffic_lights tl WHERE tl.node_id = nodeid) > 0) THEN
      tl = TRUE;
   ELSE
      tl = FALSE;
   END IF;

   return tl; -- return whether the intersection belongs to a round about or not.

  END;
$$ LANGUAGE 'plpgsql';



------- 3 - GEOMETRY --------------------------------------------

------- 3.1 - intersection legs: can be easily determined with a single
-- SQL-Statement. Therefore, it is directly executed in get_global_risk

------- 3.2 - angle of intersection: is classified as of little importance and the
-- literature is unclear in what is meant when speaking about intersection angle


------- 4 - BEHAVIOR --------------------------------------------

------- 4.1 - SPEED: get the highest speed that "goes into" the intersection
DROP FUNCTION IF EXISTS get_speed(integer);
CREATE OR REPLACE FUNCTION get_speed(integer)
RETURNS integer AS $$
  DECLARE
   node_id ALIAS FOR $1;
   speed integer;
   no_links integer;
   tmp_speed integer;
   frc integer; -- road category (function of road) is used in case no speed could be found

  BEGIN
   no_links := (SELECT COUNT(l.*) FROM gip_linknetz_ogd_aoi l WHERE l.from_node = node_id OR l.to_node = node_id);
   speed := -2;

   FOR q in 1..no_links LOOP
      -- check both the lanes in and against the driving direction for the maximum
      -- speed, store it temporarily in tmp_speed.
      -- if it is the highest value that is found, it gets stored in "speed"
      tmp_speed := (SELECT vmax_car_t FROM gip_linknetz_ogd_aoi l
                  WHERE l.from_node = node_id OR l.to_node = node_id
                  ORDER BY link_id LIMIT 1 OFFSET (q-1));
      IF (speed < tmp_speed) THEN
         speed := tmp_speed;
      END IF;

```

```
      tmp_speed := (SELECT vmax_car_b FROM gip_linknetz_ogd_aoi l
                    WHERE l.from_node = node_id OR l.to_node = node_id
                    ORDER BY link_id LIMIT 1 OFFSET (q-1));
      IF (speed < tmp_speed) THEN
         speed := tmp_speed;
      END IF;

   END LOOP;

   -- in case, none of the connected links do have attributes regarding the
   -- maximum speed (meaning that "speed" is still -2), the average speed is used:
   IF (speed = -2) THEN
      FOR q in 1..no_links LOOP
         tmp_speed := (SELECT speedcar_t FROM gip_linknetz_ogd_aoi l
                       WHERE l.from_node = node_id OR l.to_node = node_id
                       ORDER BY link_id LIMIT 1 OFFSET (q-1));
         IF (speed < tmp_speed) THEN
            speed := tmp_speed;
         END IF;

         tmp_speed := (SELECT speedcar_b FROM gip_linknetz_ogd_aoi l
                       WHERE l.from_node = node_id OR l.to_node = node_id
                       ORDER BY link_id LIMIT 1 OFFSET (q-1));
         IF (speed < tmp_speed) THEN
            speed := tmp_speed;
         END IF;
      END LOOP;
   END IF;

   -- if this also does not get any results, we'll have to rely on the road type.
   IF (speed = -2) THEN
      FOR q in 1..no_links LOOP -- loop through the link connected to the intersection
         -- load the road category in the variable "frc"
         frc := (SELECT l.frc FROM gip_linknetz_ogd_aoi l WHERE l.from_node = node_id OR
      l.to_node = node_id ORDER BY link_id LIMIT 1 OFFSET (q-1));
         CASE
--- the cases don't necessarily produce the same results as the speed attributes
            WHEN frc IN (-1, 10, 20, 21, 22, 24, 25, 31, 45, 46, 47, 48, 101, 102, 103, 115,
      200, 300)
               THEN tmp_speed := 0;
            WHEN frc IN (107)
               THEN tmp_speed := 10;
            WHEN frc IN (105, 106, 301)
               THEN tmp_speed := 30;
            WHEN frc IN (7, 8, 11, 12, 98, 99)
               THEN tmp_speed := 50;
            WHEN frc IN (5, 6)
               THEN tmp_speed := 70;
            WHEN frc IN (2, 3, 4)
               THEN tmp_speed := 100;
            WHEN frc IN (0, 1)
               THEN tmp_speed := 130;
         END CASE;
         IF (tmp_speed > speed) THEN speed := tmp_speed; END IF;
      END LOOP;
   END IF;

   RETURN speed;

  END;
$$ LANGUAGE 'plpgsql';



------------- STREET CHARACTERISTICS --------------------------
-- 5.1.1 - Number of lanes -> returns the number of turn relation (type car)
-- that are crossed
DROP FUNCTION IF EXISTS car_lanes_crossed(integer);
CREATE OR REPLACE FUNCTION car_lanes_crossed(integer)
RETURNS integer AS $$
  DECLARE
   turnuse_fid ALIAS FOR $1;
   no_car_lanes integer;
   car_cross_counter integer;
   original_turnuse iplateau%ROWTYPE;
   turnuses_that_crosses geometry;
   turnuses_id integer;

  BEGIN
   car_cross_counter = 0;
   no_car_lanes := (SELECT COUNT(*) FROM iplateau
                  WHERE basetype = 1 AND fid <> turnuse_fid);
```

```
    SELECT INTO original_turnuse ip.* FROM iplateau ip WHERE ip.fid = turnuse_fid;

    FOR q IN 1..no_car_lanes LOOP
       turnuses_that_crosses := (SELECT turnuse_geom FROM iplateau
                                  WHERE basetype = 1 ORDER BY fid LIMIT 1 OFFSET q-1);
       turnuses_id := (SELECT fid FROM iplateau WHERE basetype = 1
                   ORDER BY fid LIMIT 1 OFFSET q-1);

       IF ST_Crosses(original_turnuse.turnuse_geom, turnuses_that_crosses) IS TRUE THEN
          car_cross_counter := car_cross_counter + 1;
          RAISE NOTICE 'Crossed car turnuse. FID = %', turnuses_id;
       END IF;
    END LOOP;

     -- returns the number of car lanes the bike lane's crossing
    return car_cross_counter;

   END;
$$ LANGUAGE 'plpgsql';


-- 5.1.2 - Number of lanes
-- -> returns the number of bike/pedestrian turnuses crossed
DROP FUNCTION IF EXISTS bike_lanes_crossed(integer);
CREATE OR REPLACE FUNCTION bike_lanes_crossed(integer)
RETURNS integer AS $$
   DECLARE
    turnuse_fid ALIAS FOR $1;
    no_bike_lanes integer;
    bike_cross_counter integer;
    original_turnuse iplateau%ROWTYPE;
    turnuses_that_crosses geometry;
    turnuses_id integer;

   BEGIN
    bike_cross_counter = 0;
    no_bike_lanes := (SELECT COUNT(*) FROM iplateau
                  WHERE basetype <> 1 AND fid <> turnuse_fid);
    SELECT INTO original_turnuse ip.* FROM iplateau ip WHERE ip.fid = turnuse_fid;

    FOR q IN 1..no_bike_lanes LOOP
       turnuses_that_crosses := (SELECT turnuse_geom FROM iplateau
                                  WHERE basetype <> 1 ORDER BY fid LIMIT 1 OFFSET q-1);
       turnuses_id := (SELECT fid FROM iplateau WHERE basetype <> 1
                   ORDER BY fid LIMIT 1 OFFSET q-1);

       IF ST_Crosses(original_turnuse.turnuse_geom, turnuses_that_crosses) IS TRUE THEN
          bike_cross_counter := bike_cross_counter + 1;
          RAISE NOTICE 'Crossed bike turnuse. FID = %', turnuses_id;
       END IF;
    END LOOP;


    return bike_cross_counter; -- returns the number of car lanes the bike lane's crossing

   END;
$$ LANGUAGE 'plpgsql';



-- 5.2 - Gradient
DROP FUNCTION IF EXISTS check_gradient_slope(bigint);
CREATE OR REPLACE FUNCTION check_gradient_slope(bigint)
RETURNS real AS $$
   DECLARE
    edge_id ALIAS FOR $1;
    no_linked_nodes integer;
    links gip_linknetz_ogd_aoi%ROWTYPE;
    distance real;
    height1 real;
    node_id2 bigint;
    node1 node_ogd_aoi%ROWTYPE;
    node2 node_ogd_aoi%ROWTYPE;
    height2 real;
    pos_height_diff real;
    height_diff real;
    gradient real;

   BEGIN
    -- get the nodes which height is need
    SELECT INTO node1 n.* FROM node_ogd_aoi n, edge_ogd_aoi e WHERE e.nodefromid = n.objectid
       AND e.objectid = edge_id;
```

```
    SELECT INTO node2 n.* FROM node_ogd_aoi n, edge_ogd_aoi e WHERE e.nodetoid = n.objectid AND e.objectid = edge_id;

    -- calculate distance between the two nodes
    distance := ST_Distance(ST_Transform(node1.geom, 31256), ST_Transform(node2.geom, 31256));

    -- get the heights of the two nodes
    height1 := (SELECT ST_Value(r.rast, ST_Transform(n.geom, 31256), true)
              FROM node_ogd_aoi n, raster_dem_aoi r
              WHERE n.objectid = node1.objectid ORDER BY 1 asc LIMIT 1);
    RAISE NOTICE 'Craziness?';
    height2 := (SELECT ST_Value(r.rast, ST_Transform(n.geom, 31256), true)
              FROM node_ogd_aoi n, raster_dem_aoi r
              WHERE n.objectid = node2.objectid ORDER BY 1 asc LIMIT 1);
    RAISE NOTICE 'craziness2?';

    pos_height_diff := (SELECT ABS(height1 - height2));

    -- calculate the gradient/slope in percents
    gradient := (pos_height_diff / distance);

    return gradient;

  END;
$$ LANGUAGE 'plpgsql';




-- 5.4 - Street Types - goes through each of the edges that lead to the node.
-- if one of them is counted as major street (hoeherrangig),
-- the function returns immediately 1 - otherwise it returns 0.
DROP FUNCTION IF EXISTS check_streettypes(bigint);
CREATE OR REPLACE FUNCTION check_streettypes(bigint)
RETURNS boolean AS $$
  DECLARE
    node_id ALIAS FOR $1;
    no_edges integer;
    edge edge_ogd_aoi%ROWTYPE;

  BEGIN
    -- get number of edges connected to the node
    no_edges := (SELECT COUNT(e.*) FROM edge_ogd_aoi e WHERE e.nodefromid = node_id OR e.nodetoid = node_id);

    FOR q IN 1..no_edges LOOP
       SELECT INTO edge e.* FROM edge_ogd_aoi e
       WHERE e.nodefromid = node_id OR e.nodetoid = node_id
       ORDER BY e.objectid
       LIMIT 1
       OFFSET q-1;

       -- the major roads are those where the edgecategory is A, S, B or L
       -- (GIP Documentation, p. 47)
       IF edge.edgecat IN ('A', 'S', 'L', 'B') THEN
          RAISE NOTICE '1 is returned -> major road';
          return true;
       END IF;

    END LOOP;

    RAISE NOTICE '0 is returned -> just minor roads';
    return false; -- return whether the intersection belongs to a round about or not.

  END;
$$ LANGUAGE 'plpgsql';


-- 5.5 - Street Condition




------------ ROUND ABOUT ----------------------------------

-- 6.1 - calculates the inner circle radius of the round about
DROP FUNCTION IF EXISTS check_inner_circle_ra(integer);
CREATE OR REPLACE FUNCTION check_inner_circle_ra(integer)
RETURNS real AS $$
  DECLARE
    v_node_id ALIAS FOR $1;
    v_edge_id bigint;
```

```
    edge edge_ogd_aoi%ROWTYPE;
    middle geometry;
    dist_to_middle double precision;
    link_width double precision;
    innenkreis_radius double precision;

  BEGIN
    SELECT INTO edge e.* FROM edge_ogd_aoi e
    WHERE (e.nodefromid = v_edge_id OR e.nodefromid = v_edge_id) AND e.fow = 4;
    v_edge_id := edge.objectid;

    WITH
       edge_geometry AS ( -- copy edge geom into variable
          SELECT ST_Transform(geom, 31256) as geom FROM edge_ogd_aoi
          WHERE objectid = v_edge_id),
        -- store three significant points in variables which are needed
        -- to find out about the middle point
       points AS(
          SELECT ST_StartPoint(geom) as p1,
          ST_LineInterpolatePoints(geom, 0.50, false) as p2,
          ST_EndPoint(geom) as p3
          FROM edge_geometry),
       radius AS ( -- get the distance between the points
          SELECT p1, p2, p3, (ST_Distance(p1, p2)*1.1) as circle_radius FROM points),
        -- this number is now used to serve as the radius so circles can be drawn
       circles AS (
          SELECT (ST_ExteriorRing(ST_Buffer(p1, circle_radius))) as circle1,
                (ST_ExteriorRing(ST_Buffer(p2, circle_radius))) as circle2,
                (ST_ExteriorRing(ST_Buffer(p3, circle_radius))) as circle3
          FROM radius),
       -- get the intersection points of the circles and make them lines
       intersection_points AS (
          SELECT ST_MakeLine(ST_Intersection(circle1, circle2)) as set1,
          ST_MakeLine(ST_Intersection(circle2, circle3)) as set2
          FROM circles)

     -- where these lines meet, there is the middle point of the round about
    SELECT INTO middle ST_Intersection(set1, set2) FROM intersection_points;

    dist_to_middle := (ST_Distance(ST_Transform(ST_StartPoint(edge.geom), 31256), middle));
    SELECT INTO link_width l.width FROM edge_ogd_aoi e, gip_linknetz_ogd_aoi l
    WHERE e.objectid = v_edge_id AND e.objectid = l.edge_id LIMIT 1;
    innenkreis_radius := dist_to_middle - (link_width/2);

    return innenkreis_radius;
  END;
$$ LANGUAGE 'plpgsql';




-- 6.2 - finds out, if one of the links that lead to the intersection belongs
-- to a round about. the according attribute would be fow = 4
DROP FUNCTION IF EXISTS check_roundabout(integer);
CREATE OR REPLACE FUNCTION check_roundabout(integer)
RETURNS boolean AS $$
  DECLARE
    node_id ALIAS FOR $1;
    ra boolean;
    no_ra_links integer;

  BEGIN
    -- get number of links that are part of a round about (formofway = 4)
    no_ra_links := (SELECT COUNT(l.*) FROM gip_linknetz_ogd_aoi l
                WHERE (l.from_node = node_id OR l.to_node = node_id) AND l.formofway = 4);

    IF (no_ra_links > 0) THEN
     -- if one or more links with fow = 4 were found and counted in no_ra_links,
     -- the function returns true
       ra = TRUE;
    ELSE
       ra = FALSE; -- else, false is returned
    END IF;

    return ra; -- return whether the intersection belongs to a round about or not.

  END;
$$ LANGUAGE 'plpgsql';



-- 6.3 - finds out, whether there is cycle infrastructure within the roundabout
```

```
DROP FUNCTION IF EXISTS check_cyclinfra_in_ra(integer);
CREATE OR REPLACE FUNCTION check_cyclinfra_in_ra(integer)
RETURNS boolean AS $$
  DECLARE
   node_id ALIAS FOR $1;
   ci_exists boolean;
   no_edges integer;
   v_edge edge_ogd_aoi%ROWTYPE;
   no_lu integer;
   v_linearuse linearuse_ogd_aoi%ROWTYPE;
   counter_cycl_inf integer;

  BEGIN
  --going with edges as they also hold the necessary information
  -- (the links also do), but are better connected to the linear uses
   no_edges := (SELECT COUNT(*) FROM edge_ogd_aoi e
             WHERE (e.nodefromid = node_id OR e.nodetoid = node_id));
    -- first, the boolean saying whether cycling infrastructure is available,
    --  is set to zero
   ci_exists := FALSE;

   FOR q IN 1..no_edges LOOP
      -- this function is only called in case the intersections was classified as
      -- part of a roundabout before.
      -- it is checked whether there is a separated linearuse for bicyclists - this
      -- would mean that there is some kind of separated infrastructure for cyclists
      SELECT INTO v_edge e.* FROM edge_ogd_aoi e
      WHERE (e.nodefromid = node_id OR e.nodetoid = node_id)
      ORDER BY e.objectid LIMIT 1 OFFSET q-1;

      counter_cycl_inf := 0;

      IF v_edge.fow = 4 THEN
          -- get number of linearuses connected to the current edge
         no_lu := (SELECT COUNT(*) FROM linearuse_ogd_aoi
                 WHERE edge_id = v_edge.objectid
                 AND basetype IN (2, 22, 23, 31, 33, 35, 36));
         counter_cycl_inf := counter_cycl_inf + 1;


         IF counter_cycl_inf > 0 THEN
            ci_exists := TRUE;
            return ci_exists;
         END IF;
      END IF;

   END LOOP;

    -- return whether the intersection belongs to a round about or not.
   return ci_exists;

  END;
$$ LANGUAGE 'plpgsql';



------------- CYCLING INFRASTRUCTURE ------------------------

-- 7.1 - is there mixed traffic - if yes, which kind?
DROP FUNCTION IF EXISTS check_mixed_traffic(bigint, bigint);
CREATE OR REPLACE FUNCTION check_mixed_traffic(bigint, bigint)
RETURNS varchar(45) AS $$
  DECLARE
    -- from the linearuse, I need to get to the gip_linknetz.
    -- therefore, i have to go through edge
   linearuse_lanes_fid ALIAS FOR $1;
   v_node_id ALIAS FOR $2;
   linearuse_id bigint;
   v_edge_id bigint;
   v_link_id bigint;
   v_use_id bigint;
   my_bikehike bikehike%ROWTYPE;
   bikefeature varchar(10);

  BEGIN

   linearuse_id := (SELECT ll.lu_objectid FROM linearuse_lanes ll
                WHERE ll.fid = linearuse_lanes_fid);
   RAISE NOTICE 'LInearuse FID: %', linearuse_id;
   v_edge_id := (SELECT l.edge_id FROM linearuse_ogd_aoi l
              WHERE l.objectid = linearuse_id);
   RAISE NOTICE 'Edge ID: %', v_edge_id;
```

```
    v_link_id := (SELECT l.link_id FROM gip_linknetz_ogd_aoi l
               WHERE l.edge_id = v_edge_id LIMIT 1);
    RAISE NOTICE 'Link ID: %', v_link_id;

    -- get the use_id of the layer bikehike that is referring to the linear use
     -- then get the linkuse's use_id. As there can be so many as 25 use_ids for
     -- one link, we have to check to get the one which belongs to the
     -- correct linearuse (found out via the offset to the link)
    v_use_id := (SELECT lku.use_id
              FROM linkuse_aoi lku
              WHERE lku.link_id = v_link_id
                AND lku.offsett = (SELECT ROUND(offsetavg::decimal, 1)
                               FROM linearuse_ogd_aoi
                               WHERE objectid = (SELECT ll.lu_objectid
                                           FROM linearuse_lanes ll
                                           WHERE ll.fid = linearuse_lanes_fid
                                           )
                              )
             );

    RAISE NOTICE 'Use ID: %', v_use_id;

    SELECT INTO my_bikehike bh.* FROM bikehike bh WHERE bh.use_id = v_use_id;

    -- if the lane is not a car lane, it does not matter which side we look at -
    -- because bike lanes are not split in my model
    IF ((SELECT basetype FROM linkuse_aoi WHERE use_id = v_use_id) <> 1) THEN
       IF (my_bikehike.bikefeaturetow IS NOT NULL) THEN
          bikefeature := my_bikehike.bikefeaturetow;
       ELSIF (my_bikehike.bikefeaturebkw IS NOT NULL) THEN
          bikefeature := my_bikehike.bikefeaturebkw;
       ELSE
          bikefeature := NULL;
       END IF;

    -- in case of the lane being for cars, we need to have a look, if it is in
    -- the same digitisation direction as the link or against the dig. direction.
    -- This needs to be done because the values differ depending on the direction.
    ELSE
       IF ((SELECT to_node FROM gip_linknetz_ogd_aoi WHERE link_id = v_link_id) =
          (SELECT leads_to FROM linearuse_lanes WHERE fid = linearuse_lanes_fid)) THEN
          --get the linearuses which are directed in the same direction as the
          -- link they are derived from
          bikefeature := my_bikehike.bikefeaturetow;
       ELSIF ((SELECT to_node FROM gip_linknetz_ogd_aoi WHERE link_id = v_link_id) =
             (SELECT comes_from FROM linearuse_lanes WHERE fid = linearuse_lanes_fid)) THEN
          -- get the backwards attribute as link and linearuse_lane do not have
          -- the same direction
          bikefeature := my_bikehike.bikefeaturebkw;
       END IF;

    END IF;

    return bikefeature;

   END;
$$ LANGUAGE 'plpgsql';


-- 7.2 - is there any cycling infrastructure? and what kind is it?
DROP FUNCTION IF EXISTS check_cycl_infra(bigint, bigint);
CREATE OR REPLACE FUNCTION check_cycl_infra(bigint, bigint)
RETURNS varchar(40) AS $$
  DECLARE
    v_linearuse_lane_fid ALIAS FOR $1;
    v_node_id ALIAS FOR $2;
    v_linearuse linearuse_lanes%ROWTYPE;
    v_cycl_infr varchar(100);
    v_linearuse_id bigint;
    v_edge_id bigint;
    v_link_id bigint;
    v_use_id bigint;

  BEGIN
    v_cycl_infr := NULL;
    SELECT INTO v_linearuse ll.* FROM linearuse_lanes ll WHERE ll.fid = v_linearuse_lane_fid;
    RAISE NOTICE 'in check_cycl_infr. v_linearuse.fid = %', v_linearuse.fid;

    -- Option 1: check basetype
    --check whether the basetype of the linearuse already means,
    -- that it is a bicycle infrastructure
    IF (v_linearuse.basetype IN (2, 22, 23, 31, 33, 35, 36)) THEN
```

```
      v_cycl_infr := (SELECT lb.name FROM lut_basetype lb
                  WHERE lb.id = v_linearuse.basetype);
      return v_cycl_infr;
   END IF;

   -- Option 2: check Bikehike
   v_linearuse_id := (SELECT ll.lu_objectid FROM linearuse_lanes ll
                  WHERE ll.fid = v_linearuse_lane_fid);
   v_edge_id := (SELECT l.edge_id FROM linearuse_ogd_aoi l
              WHERE l.objectid = v_linearuse_id);
   v_link_id := (SELECT l.link_id FROM gip_linknetz_ogd_aoi l
              WHERE l.edge_id = v_edge_id LIMIT 1);

      -- get the use_id of the layer bikehike that is referring to the linear use
       -- then get the linkuse's use_id. As there can be so many as 25 use_ids for
       -- one link, we have to check to get the one which belongs to the correct
       -- linearuse (found out via the offset to the link)
   v_use_id := (SELECT lku.use_id
             FROM linkuse_aoi lku
             WHERE lku.link_id = v_link_id
               AND lku.offsett = (SELECT ROUND(offsetavg::decimal, 1)
                              FROM linearuse_ogd_aoi
                              WHERE objectid = (SELECT ll.lu_objectid
                                          FROM linearuse_lanes ll
                                          WHERE ll.fid = v_linearuse_lane_fid
                                          )
                             )
            );

   -- is the linearuse_lane directed inthe same direction as the link
   -- --> is the lane tow?
   IF (v_linearuse.leads_to = (SELECT gl.to_node FROM gip_linknetz_ogd_aoi gl
                        WHERE gl.link_id = v_link_id)) THEN
      v_cycl_infr := (SELECT bh.bikefeaturetow FROM bikehike bh WHERE bh.use_id = v_use_id);

   -- the lane is bkw compared to its parent link
   ELSIF (v_linearuse.comes_from = (SELECT gl.to_node FROM gip_linknetz_ogd_aoi gl
                            WHERE gl.link_id = v_link_id)) THEN
      v_cycl_infr := (SELECT bh.bikefeaturebkw FROM bikehike bh WHERE bh.use_id = v_use_id);
   END IF;

   IF (v_cycl_infr IS NULL) THEN
   ELSE
      return v_cycl_infr;
   END IF;

   IF v_cycl_infr IS NULL THEN
      return NULL;
   END IF;


  END;
$$ LANGUAGE 'plpgsql';





------------ MORE GENERAL HELPER FUNCTIONS -----------------

--- read bit-mask
DROP FUNCTION IF EXISTS read_bitmask(integer, varchar(4));
CREATE OR REPLACE FUNCTION read_bitmask(integer, varchar(4))
RETURNS boolean[] AS $$
  DECLARE
   linkid ALIAS FOR $1;
   variable ALIAS FOR $2;
   bit_len integer;
   v_access bit(7);
   accesses boolean [7];
   -- the rows are for the 7 access that are documented in the bit mask:
   -- 0 - pedestrian, 1 - bike, 2 - private car, 3 - public bus, 4 - railway,
   -- 5 - tram, 6 - subway, 7 - ferry boat. for more information see GIP Documentation
   -- helpful resource for handling bit strings:
   -- https://www.postgresql.org/docs/current/functions-bitstring.html

  BEGIN
   bit_len := 7;

   CASE variable
       -- variable gets filled with value from the link whose id was given (toward value)
```

```
      WHEN 'l_t'  THEN v_access := (SELECT access_tow FROM gip_linknetz_ogd_aoi WHERE linkid
      = link_id)::bit(7);
      -- variable gets filled with value from the link whose id was given (backward value)
      WHEN 'l_b'  THEN v_access := (SELECT access_bkw FROM gip_linknetz_ogd_aoi WHERE linkid
      = link_id)::bit(7);
      -- table: bikehike, direction: towards
      WHEN 'bh_t' THEN v_access := (SELECT use_access_tow FROM bikehike WHERE linkdid =
      link_id)::bit(7);
       -- table: bikehike, direction: backwards
      WHEN 'bh_b' THEN v_access := (SELECT use_access_bkw FROM bikehike WHERE linkdid =
      link_id)::bit(7);
   END CASE;

   FOR q IN 1..bit_len LOOP
      -- go the the last value of the bit-mask (pedestrians). check if it is one (=1)
      -- if yes: set accesses[0] = true, else: set accesses[0] = false
      IF get_bit(v_access, (bit_len - q)) = 1 THEN
         accesses[q-1] := TRUE;
      ELSE
         accesses[q-1] := FALSE;
      END IF;

   END LOOP;

   RAISE NOTICE 'ACCESS ARRAY: %', accesses;
    -- 1st value: Pedestrian, 2: Bike, 3: Private Car,
    -- 4: Public Bus, 5: Railway, 6: Tram, 7: Subway
   return accesses;

  END;
$$ LANGUAGE 'plpgsql';
```

Here, the information collected before are being normalized. So if we know, that the intersection is part of a roundabout, we would give it the value 0 since roundabouts are hazardous for bicyclists.

Listing A.13: [The retrieved data is being normalized and evaluated. (`normalization.sql`)

```
DROP FUNCTION IF EXISTS normalize_factors();
CREATE FUNCTION normalize_factors()
RETURNS void AS $$
DECLARE
   no_lanes integer;
   current_lane risk_value_table%ROWTYPE;
   v_fid integer;
   v_indicator numeric;
   v_indicator_help numeric;

BEGIN
   no_lanes := (SELECT COUNT(*) FROM risk_value_table);

   DELETE FROM normalized_risk_values;


   FOR q IN 1..no_lanes LOOP

      SELECT INTO  current_lane rvt.* FROM risk_value_table rvt
      ORDER BY fid LIMIT 1 OFFSET (q-1);

      v_fid := current_lane.fid;

      INSERT INTO normalized_risk_values (fid, nodeid)
      VALUES (v_fid, current_lane.nodeid);



      -------------INDICATORS-------------------------------------
      -- 0 -> unbikeable
      -- 1 -> very bikeable

      -- 1.1 urban
       v_indicator :=
         CASE
            WHEN current_lane.urban = TRUE THEN 1.0
            WHEN current_lane.urban = FALSE THEN 0.0
            WHEN current_lane.urban IS NULL THEN NULL
         END;
      UPDATE normalized_risk_values SET urban = v_indicator WHERE fid = v_fid;

      -- 1.2 rails
       v_indicator :=
         CASE
            WHEN current_lane.rails = TRUE THEN 0.0
            WHEN current_lane.rails = FALSE THEN 1
            WHEN current_lane.rails IS NULL THEN NULL
         END;
      UPDATE normalized_risk_values SET rails = v_indicator WHERE fid = v_fid;

      -- 2.1 traffic_lights
      v_indicator :=
         CASE
            WHEN current_lane.traffic_lights = TRUE THEN 0
            WHEN current_lane.traffic_lights = FALSE THEN 1
            WHEN current_lane.traffic_lights IS NULL THEN NULL
         END;
      UPDATE normalized_risk_values SET traffic_lights = v_indicator WHERE fid = v_fid;

      -- 3.1 intersection_legs
      v_indicator :=
         CASE
            WHEN current_lane.intersection_legs <= 2 THEN 1
            WHEN current_lane.intersection_legs = 3 THEN 0.7
            WHEN current_lane.intersection_legs = 4 THEN 0.5
            WHEN current_lane.intersection_legs = 5 THEN 0.1
            WHEN current_lane.intersection_legs >= 6 THEN 0.0
            WHEN current_lane.intersection_legs IS NULL THEN NULL
         END;
      UPDATE normalized_risk_values SET intersection_legs = v_indicator WHERE fid = v_fid;

      -- 4.1 speed
      v_indicator :=
         CASE
            WHEN current_lane.speed >= 100 THEN 0
            WHEN current_lane.speed >= 80 THEN 0.2
            WHEN current_lane.speed >= 70 THEN 0.3
```

```
              WHEN current_lane.speed >= 60 THEN 0.4
              WHEN current_lane.speed >= 50 THEN 0.6
              WHEN current_lane.speed >= 30 THEN 0.85
              WHEN current_lane.speed > 0 THEN 0.9
              WHEN current_lane.speed = 0 THEN 1
              WHEN current_lane.speed IS NULL THEN NULL
           END;
        UPDATE normalized_risk_values SET speed = v_indicator WHERE fid = v_fid;

        -- 5.1 number of lanes that are being crossed
         -- cars
        v_indicator :=
           CASE
              WHEN current_lane.lane_number_cars >= 7 THEN 0
              WHEN current_lane.lane_number_cars = 6 THEN 0.05
              WHEN current_lane.lane_number_cars = 5 THEN 0.1
              WHEN current_lane.lane_number_cars = 4 THEN 0.25
              WHEN current_lane.lane_number_cars = 3 THEN 0.3
              WHEN current_lane.lane_number_cars = 2 THEN 0.5
              WHEN current_lane.lane_number_cars = 1 THEN 0.6
              WHEN current_lane.lane_number_cars = 0 THEN 1
              WHEN current_lane.lane_number_cars IS NULL THEN NULL
           END;
        UPDATE normalized_risk_values SET lane_number_cars = v_indicator WHERE fid = v_fid;
         -- bikes
        v_indicator :=
           CASE
              WHEN current_lane.lane_number_bikes >= 10 THEN 0
              WHEN current_lane.lane_number_bikes BETWEEN 7 AND 9 THEN 0.25
              WHEN current_lane.lane_number_bikes BETWEEN 4 AND 6 THEN 0.5
              WHEN current_lane.lane_number_bikes BETWEEN 1 AND 3 THEN 0.75
              WHEN current_lane.lane_number_bikes <= 0 THEN 1
              WHEN current_lane.lane_number_bikes IS NULL THEN NULL
           END;
        UPDATE normalized_risk_values SET lane_number_bikes = v_indicator WHERE fid = v_fid;

        -- 5.2 gradient
        v_indicator :=
           CASE
              WHEN current_lane.gradient >= 0.2 THEN 0
              WHEN current_lane.gradient >= 0.1 THEN 0.15
              WHEN current_lane.gradient >= 0.07 THEN 0.2
              WHEN current_lane.gradient >= 0.05 THEN 0.3
              WHEN current_lane.gradient >= 0.03 THEN 0.55
              WHEN current_lane.gradient >= 0.01 THEN 0.8
              WHEN current_lane.gradient < 0.01 THEN 1
              WHEN current_lane.gradient IS NULL THEN NULL
           END;
        UPDATE normalized_risk_values SET gradient = v_indicator WHERE fid = v_fid;

        -- 5.3 physical barriers

        -- 5.4 Street types
        v_indicator :=
           CASE
              WHEN current_lane.street_type = TRUE  THEN 0.0-- major road
              WHEN current_lane.street_type = FALSE  THEN 1.0 -- minor road
              WHEN current_lane.street_type IS NULL   THEN NULL
           END;
        UPDATE normalized_risk_values SET street_type = v_indicator WHERE fid = v_fid;

        -- 6.1 inner circle of round about
        v_indicator :=
           CASE
              WHEN current_lane.round_about_inner_circle >= 20 THEN 0
              WHEN current_lane.round_about_inner_circle <= 20 THEN 0.2
              WHEN current_lane.round_about_inner_circle <= 15 THEN 0.4
              WHEN current_lane.round_about_inner_circle <= 10 THEN 0.6
              WHEN current_lane.round_about_inner_circle <= 7 THEN 0.8
              WHEN current_lane.round_about_inner_circle <= 3 THEN 1
              WHEN current_lane.round_about_inner_circle IS NULL   THEN NULL
           END;
        UPDATE normalized_risk_values SET round_about_inner_circle = v_indicator WHERE fid =
        v_fid;

        -- 6.2 existence round about
         v_indicator :=
           CASE
              WHEN current_lane.round_about = TRUE THEN 0.0
              WHEN current_lane.round_about = FALSE THEN 1.0
              WHEN current_lane.round_about IS NULL THEN NULL
           END;
```

```
    UPDATE normalized_risk_values SET round_about = v_indicator WHERE fid = v_fid;

    -- 6.3 cycling lane in round about
    v_indicator :=
       CASE
          WHEN current_lane.round_about_cycle_infstr = TRUE THEN 1.0
          WHEN current_lane.round_about_cycle_infstr = FALSE THEN 0.0
          WHEN current_lane.round_about_cycle_infstr IS NULL THEN NULL
       END;
    UPDATE normalized_risk_values SET round_about_cycle_infstr = v_indicator WHERE fid =
v_fid;



    -- 7.1 mixed traffic
    -- the from-lanes
    v_indicator :=
       CASE
          WHEN current_lane.mixed_traffic_f LIKE 'TRF' THEN  0.0 -- Trasse nur fuer
Fussgaenger
          WHEN current_lane.mixed_traffic_f LIKE 'MZSTR' THEN  0.0 -- Mehrzweckstreifen
          WHEN current_lane.mixed_traffic_f LIKE 'BGZ'  THEN 0.1 -- Begegnungszone
          WHEN current_lane.mixed_traffic_f LIKE 'RFUE' THEN 0.1 -- Radfahrerueberfahrt
          WHEN current_lane.mixed_traffic_f LIKE 'WSTR' THEN 0.25-- Radfahren in
Wohnstrassen
          WHEN current_lane.mixed_traffic_f LIKE 'BS'  THEN 0.25 -- Radfahren auf
Busspuren
          WHEN current_lane.mixed_traffic_f LIKE 'GRW_M' THEN 0.3 -- Gemischter Geh- und
Radweg
          WHEN current_lane.mixed_traffic_f LIKE 'FUZO' THEN 0.3 -- Radfahren in
Fussgaengerzonen
          WHEN current_lane.mixed_traffic_f LIKE 'GRW_MO' THEN 0.35 -- Gemischter Geh-
und Radweg ohne Benuetzungspficht
          WHEN current_lane.mixed_traffic_f LIKE 'ABBK'  THEN 0.4 -- Anrainerstr.
Radverkehr
          WHEN current_lane.mixed_traffic_f LIKE 'RFGE' THEN  0.4 -- Radfahren gegen die
Einbahn
          WHEN current_lane.mixed_traffic_f LIKE 'RFGE_N' THEN  0.45 -- Radfahren gegen
die Einbahn (Nebenfahrbahn)
          WHEN current_lane.mixed_traffic_f LIKE 'RWO' THEN  0.55 -- Radweg ohne
Benuetzungspflicht
          WHEN current_lane.mixed_traffic_f LIKE 'RR' THEN  0.55 -- Radroute
(beschilderte Route, Radverkehr wird im Mischverkehr gefuehrt)
          WHEN current_lane.mixed_traffic_f LIKE 'RVW' THEN  0.6 -- Radfahren auf
verkehrsarmen Wegen
          WHEN current_lane.mixed_traffic_f LIKE 'VK_BE' THEN  0.7 -- Verkehrsberuhigte
Bereiche
          WHEN current_lane.mixed_traffic_f LIKE 'FUZO_N' THEN 0.75 -- Radfahren in
Fussgaengerzonen (Nebenfahrbahn)
          WHEN current_lane.mixed_traffic_f LIKE 'GRW_TO' THEN  0.75 -- Getrennter Geh-
und Radweg ohne Benuetzungspficht
          WHEN current_lane.mixed_traffic_f LIKE 'RW' THEN  0.8 -- Baulicher Radweg
          WHEN current_lane.mixed_traffic_f LIKE 'GRW_T' THEN  0.8 -- Getrennter Geh-
und Radweg
          WHEN current_lane.mixed_traffic_f LIKE 'TRR' THEN  0.8 -- Treppe auch fuer
Radfahrer geeignet
          WHEN current_lane.mixed_traffic_f LIKE 'RF' THEN  0.85 -- Radfahrstreifen
          WHEN current_lane.mixed_traffic_f LIKE 'WSTR_N' THEN  0.85 -- Radfahren in
Wohnstrassen (Nebenfahrbahn)
          WHEN current_lane.mixed_traffic_f LIKE 'RRN' THEN  0.9 -- Hauptradroute
          WHEN current_lane.mixed_traffic_f LIKE 'FRS'  THEN 1 -- Fahrradstrasse

          WHEN (current_lane.basetype = 1 AND current_lane.mixed_traffic_f IS NULL) THEN
0.0 -- car lane without bicycle infrastructure
          WHEN (current_lane.basetype IN (7, 21, 37, 41) AND current_lane.mixed_traffic_f
IS NULL) THEN 0.15 -- pedestrian lane without bicycle infrastructure

          WHEN current_lane.mixed_traffic_f LIKE 'MTB' THEN NULL -- Mountainbikestrecke
(im Wald). Null da es sich nicht auf mein Thema bezieht und auch nicht auf meine AOI.
          WHEN current_lane.mixed_traffic_f LIKE 'SGT' THEN NULL -- Singletrail - keine
Ahnung.
          WHEN current_lane.mixed_traffic_f LIKE 'FE' THEN NULL -- Faehre
          WHEN current_lane.mixed_traffic_f LIKE 'HI_IV' THEN NULL
          WHEN current_lane.mixed_traffic_f LIKE '-1'  THEN NULL
          WHEN current_lane.mixed_traffic_f IS NULL THEN NULL
       END;

    -- the to-lanes
    v_indicator_help :=
       CASE
          WHEN current_lane.mixed_traffic_t LIKE 'TRF' THEN  0.0 -- Trasse nur fuer
Fussgaenger
```

```sql
            WHEN current_lane.mixed_traffic_t LIKE 'MZSTR' THEN  0.0 -- Mehrzweckstreifen
            WHEN current_lane.mixed_traffic_t LIKE 'BGZ'  THEN 0.1 -- Begegnungszone
            WHEN current_lane.mixed_traffic_t LIKE 'RFUE' THEN 0.1 -- Radfahrerueberfahrt
            WHEN current_lane.mixed_traffic_t LIKE 'WSTR' THEN 0.25-- Radfahren in Wohnstrassen
            WHEN current_lane.mixed_traffic_t LIKE 'BS'  THEN 0.25 -- Radfahren auf Busspuren
            WHEN current_lane.mixed_traffic_t LIKE 'GRW_M' THEN 0.3 -- Gemischter Geh- und Radweg
            WHEN current_lane.mixed_traffic_t LIKE 'FUZO' THEN 0.3 -- Radfahren in Fussgaengerzonen
            WHEN current_lane.mixed_traffic_t LIKE 'GRW_MO' THEN 0.35 -- Gemischter Geh- und Radweg ohne Benuetzungspficht
            WHEN current_lane.mixed_traffic_t LIKE 'ABBK'  THEN 0.4 -- Anrainerstr. Radverkehr
            WHEN current_lane.mixed_traffic_t LIKE 'RFGE' THEN  0.4 -- Radfahren gegen die Einbahn
            WHEN current_lane.mixed_traffic_t LIKE 'RFGE_N' THEN  0.45 -- Radfahren gegen die Einbahn (Nebenfahrbahn)
            WHEN current_lane.mixed_traffic_t LIKE 'RWO' THEN  0.55 -- Radweg ohne Benuetzungspflicht
            WHEN current_lane.mixed_traffic_t LIKE 'RR' THEN  0.55 -- Radroute (beschilderte Route, Radverkehr wird im Mischverkehr gefuehrt)
            WHEN current_lane.mixed_traffic_t LIKE 'RVW' THEN  0.6 -- Radfahren auf verkehrsarmen Wegen
            WHEN current_lane.mixed_traffic_t LIKE 'VK_BE' THEN  0.7 -- Verkehrsberuhigte Bereiche
            WHEN current_lane.mixed_traffic_t LIKE 'FUZO_N' THEN 0.75 -- Radfahren in Fussgaengerzonen (Nebenfahrbahn)
            WHEN current_lane.mixed_traffic_t LIKE 'GRW_TO' THEN  0.75 -- Getrennter Geh- und Radweg ohne Benuetzungspficht
            WHEN current_lane.mixed_traffic_t LIKE 'RW' THEN  0.8 -- Baulicher Radweg
            WHEN current_lane.mixed_traffic_t LIKE 'GRW_T' THEN  0.8 -- Getrennter Geh- und Radweg
            WHEN current_lane.mixed_traffic_t LIKE 'TRR' THEN  0.8 -- Treppe auch fuer Radfahrer geeignet
            WHEN current_lane.mixed_traffic_t LIKE 'RF' THEN  0.85 -- Radfahrstreifen
            WHEN current_lane.mixed_traffic_t LIKE 'WSTR_N' THEN  0.85 -- Radfahren in Wohnstrassen (Nebenfahrbahn)
            WHEN current_lane.mixed_traffic_t LIKE 'RRN' THEN  0.9 -- Hauptradroute
            WHEN current_lane.mixed_traffic_t LIKE 'FRS'  THEN 1 -- Fahrradstrasse

            WHEN (current_lane.basetype = 1 AND current_lane.mixed_traffic_t IS NULL) THEN 0.0 -- car lane without bicycle infrastructure
            WHEN (current_lane.basetype IN (7, 21, 37, 41) AND current_lane.mixed_traffic_t IS NULL) THEN 0.15 -- pedestrian lane without bicycle infrastructure

            WHEN current_lane.mixed_traffic_t LIKE 'MTB' THEN NULL -- Mountainbikestrecke (im Wald). Null da es sich nicht auf mein Thema bezieht und auch nicht auf meine AOI.
            WHEN current_lane.mixed_traffic_t LIKE 'SGT' THEN NULL -- Singletrail - keine Ahnung.
            WHEN current_lane.mixed_traffic_t LIKE 'FE' THEN NULL -- Faehre
            WHEN current_lane.mixed_traffic_t LIKE 'HI_IV' THEN NULL
            WHEN current_lane.mixed_traffic_t LIKE '-1'  THEN NULL
            WHEN current_lane.mixed_traffic_t IS NULL THEN NULL
        END;
    -- store the worse value (compare from-lane and to-lane) in the normalized table. Reason: The two will meet each other anyway, so the worse option is going to happen anyways.
    IF (v_indicator <= v_indicator_help AND v_indicator IS NOT NULL) THEN
        UPDATE normalized_risk_values SET mixed_traffic = v_indicator WHERE fid = v_fid;
    ELSE
        UPDATE normalized_risk_values SET mixed_traffic = v_indicator_help WHERE fid = v_fid;
    END IF;



    -- 7.2 cycling infrastructure. gleiche Dtenquelle wie oben, sollte aber anders gewertet weden
    -- the from-lane (the lane the turnuse comes from)
    v_indicator :=
        CASE
            -- option 1 & 2: basetype is cycling infrastructure
            WHEN current_lane.cycle_infrastructure_f LIKE 'Radfahrerueberfahrt' THEN 0.5
            WHEN current_lane.cycle_infrastructure_f LIKE 'Schutzweg und Radfahrerueberfahrt' THEN 0.5
            WHEN current_lane.cycle_infrastructure_f LIKE 'Radfahrstreifen' THEN 0.6
            WHEN current_lane.cycle_infrastructure_f LIKE 'Radfahrstreifen gegen die Einbahn' THEN 0.6
            WHEN current_lane.cycle_infrastructure_f LIKE 'Geh- und Radweg' THEN 0.65
            WHEN current_lane.cycle_infrastructure_f LIKE 'Radweg mit angrenzendem Gehweg' THEN 0.95
```

```
            WHEN current_lane.cycle_infrastructure_f LIKE 'Radweg' THEN 1
            -- option 3: look it up in bikehike
            WHEN current_lane.cycle_infrastructure_f LIKE 'BGZ'  THEN 0.0 -- Begegnungszone
            WHEN current_lane.cycle_infrastructure_f LIKE 'TRF' THEN  0.0 -- Trasse nur fuer Fussgaenger
            WHEN current_lane.cycle_infrastructure_f LIKE 'RFGE' THEN  0.1 -- Radfahren gegen die Einbahn
            WHEN current_lane.cycle_infrastructure_f LIKE 'ABBK'  THEN 0.1 -- Anrainerstr. Radverkehr
            WHEN current_lane.cycle_infrastructure_f LIKE 'FUZO' THEN 0.2 -- Radfahren in Fussgaengerzonen
            WHEN current_lane.cycle_infrastructure_f LIKE 'MZSTR' THEN  0.2 -- Mehrzweckstreifen
            WHEN current_lane.cycle_infrastructure_f LIKE 'RVW' THEN  0.3 -- Radfahren auf verkehrsarmen Wegen
            WHEN current_lane.cycle_infrastructure_f LIKE 'WSTR' THEN 0.3-- Radfahren in Wohnstrassen
            WHEN current_lane.cycle_infrastructure_f LIKE 'RFGE_N' THEN  0.4 -- Radfahren gegen die Einbahn (Nebenfahrbahn)
            WHEN current_lane.cycle_infrastructure_f LIKE 'TRR' THEN  0.4 -- Treppe auch fuer Radfahrer geeignet
            WHEN current_lane.cycle_infrastructure_f LIKE 'VK_BE' THEN  0.4 -- Verkehrsberuhigte Bereiche
            WHEN current_lane.cycle_infrastructure_f LIKE 'RR' THEN  0.45 -- Radroute (beschilderte Route, Radverkehr wird im Mischverkehr gefuehrt)
            WHEN current_lane.cycle_infrastructure_f LIKE 'GRW_MO' THEN 0.45 -- Gemischter Geh- und Radweg ohne Benuetzungspflicht
            WHEN current_lane.cycle_infrastructure_f LIKE 'FUZO_N' THEN 0.5 -- Radfahren in Fussgaengerzonen (Nebenfahrbahn)
            WHEN current_lane.cycle_infrastructure_f LIKE 'GRW_M' THEN 0.5 -- Gemischter Geh- und Radweg
            WHEN current_lane.cycle_infrastructure_f LIKE 'BS'  THEN 0.6 -- Radfahren auf Busspuren
            WHEN current_lane.cycle_infrastructure_f LIKE 'RFUE' THEN 0.7 -- Radfahrerueberfahrt -- weil keine Ahnung...
            WHEN current_lane.cycle_infrastructure_f LIKE 'WSTR_N' THEN  0.7 -- Radfahren in Wohnstrassen (Nebenfahrbahn)
            WHEN current_lane.cycle_infrastructure_f LIKE 'RWO' THEN  0.8 -- Radweg ohne Benuetzungspflicht
            WHEN current_lane.cycle_infrastructure_f LIKE 'SGT' THEN 0.9 -- Singletrail
            WHEN current_lane.cycle_infrastructure_f LIKE 'GRW_TO' THEN  0.9 -- Getrennter Geh- und Radweg ohne Benuetzungspflicht
            WHEN current_lane.cycle_infrastructure_f LIKE 'GRW_T' THEN  1 -- Getrennter Geh- und Radweg
            WHEN current_lane.cycle_infrastructure_f LIKE 'RRN' THEN  1 -- Hauptradroute
            WHEN current_lane.cycle_infrastructure_f LIKE 'RW' THEN  1 -- Baulicher Radweg
            WHEN current_lane.cycle_infrastructure_f LIKE 'RF' THEN  1 -- Radfahrstreifen
            WHEN current_lane.cycle_infrastructure_f LIKE 'MTB' THEN 1 -- Mountainbikestrecke (im Wald).
            WHEN current_lane.cycle_infrastructure_f LIKE 'FRS'  THEN 1 -- Fahrradstrasse
            WHEN current_lane.cycle_infrastructure_f LIKE 'FE' THEN NULL
            WHEN current_lane.cycle_infrastructure_f LIKE 'HI_IV' THEN NULL
            WHEN current_lane.cycle_infrastructure_f LIKE '-1'  THEN 0
            WHEN current_lane.cycle_infrastructure_f IS NULL THEN 0
         END;

      -- the to-lane (the lane the turnuse leads to)
      v_indicator_help :=
         CASE
            -- option 1: basetype is cycling infrastructure
            WHEN current_lane.cycle_infrastructure_t LIKE 'Radfahrerueberfahrt' THEN 0.5
            WHEN current_lane.cycle_infrastructure_t LIKE 'Schutzweg und Radfahrerueberfahrt' THEN 0.5
            WHEN current_lane.cycle_infrastructure_t LIKE 'Radfahrstreifen' THEN 0.6
            WHEN current_lane.cycle_infrastructure_t LIKE 'Radfahrstreifen gegen die Einbahn' THEN 0.6
            WHEN current_lane.cycle_infrastructure_t LIKE 'Geh- und Radweg' THEN 0.65
            WHEN current_lane.cycle_infrastructure_t LIKE 'Radweg mit angrenzendem Gehweg' THEN 0.95
            WHEN current_lane.cycle_infrastructure_t LIKE 'Radweg' THEN 1
            -- option 2: look it up in bikehike
            WHEN current_lane.cycle_infrastructure_t LIKE 'BGZ'  THEN 0.0 -- Begegnungszone
            WHEN current_lane.cycle_infrastructure_t LIKE 'TRF' THEN  0.0 -- Trasse nur fuer Fussgaenger
            WHEN current_lane.cycle_infrastructure_t LIKE 'RFGE' THEN  0.1 -- Radfahren gegen die Einbahn
            WHEN current_lane.cycle_infrastructure_t LIKE 'ABBK'  THEN 0.1 -- Anrainerstr. Radverkehr
            WHEN current_lane.cycle_infrastructure_t LIKE 'FUZO' THEN 0.2 -- Radfahren in Fussgaengerzonen
            WHEN current_lane.cycle_infrastructure_t LIKE 'MZSTR' THEN  0.2 -- Mehrzweckstreifen
```

```
        WHEN current_lane.cycle_infrastructure_t LIKE 'RVW' THEN  0.3 -- Radfahren auf verkehrsarmen Wegen
        WHEN current_lane.cycle_infrastructure_t LIKE 'WSTR' THEN 0.3-- Radfahren in Wohnstrassen
        WHEN current_lane.cycle_infrastructure_t LIKE 'RFGE_N' THEN  0.4 -- Radfahren gegen die Einbahn (Nebenfahrbahn)
        WHEN current_lane.cycle_infrastructure_t LIKE 'TRR' THEN  0.4 -- Treppe auch fuer Radfahrer geeignet
        WHEN current_lane.cycle_infrastructure_t LIKE 'VK_BE' THEN  0.4 -- Verkehrsberuhigte Bereiche
        WHEN current_lane.cycle_infrastructure_t LIKE 'RR' THEN  0.45 -- Radroute (beschilderte Route, Radverkehr wird im Mischverkehr gefuehrt)
        WHEN current_lane.cycle_infrastructure_t LIKE 'GRW_MO' THEN 0.45 -- Gemischter Geh- und Radweg ohne Benuetzungspflicht
        WHEN current_lane.cycle_infrastructure_t LIKE 'FUZO_N' THEN 0.5 -- Radfahren in Fussgaengerzonen (Nebenfahrbahn)
        WHEN current_lane.cycle_infrastructure_t LIKE 'GRW_M' THEN 0.5 -- Gemischter Geh- und Radweg
        WHEN current_lane.cycle_infrastructure_t LIKE 'BS'  THEN 0.6 -- Radfahren auf Busspuren
        WHEN current_lane.cycle_infrastructure_t LIKE 'RFUE' THEN 0.7 -- Radfahrerueberfahrt -- weil keine Ahnung...
        WHEN current_lane.cycle_infrastructure_t LIKE 'WSTR_N' THEN  0.7 -- Radfahren in Wohnstrassen (Nebenfahrbahn)
        WHEN current_lane.cycle_infrastructure_t LIKE 'RWO' THEN  0.8 -- Radweg ohne Benuetzungspflicht
        WHEN current_lane.cycle_infrastructure_t LIKE 'SGT' THEN 0.9 -- Singletrail
        WHEN current_lane.cycle_infrastructure_t LIKE 'GRW_TO' THEN  0.9 -- Getrennter Geh- und Radweg ohne Benuetzungspflicht
        WHEN current_lane.cycle_infrastructure_t LIKE 'GRW_T' THEN  1 -- Getrennter Geh- und Radweg
        WHEN current_lane.cycle_infrastructure_t LIKE 'RRN' THEN  1 -- Hauptradroute
        WHEN current_lane.cycle_infrastructure_t LIKE 'RW' THEN  1 -- Baulicher Radweg
        WHEN current_lane.cycle_infrastructure_t LIKE 'RF' THEN  1 -- Radfahrstreifen
        WHEN current_lane.cycle_infrastructure_t LIKE 'MTB' THEN 1 -- Mountainbikestrecke (im Wald).
        WHEN current_lane.cycle_infrastructure_t LIKE 'FRS'  THEN 1 -- Fahrradstrasse
        WHEN current_lane.cycle_infrastructure_t LIKE 'FE' THEN NULL
        WHEN current_lane.cycle_infrastructure_t LIKE 'HI_IV' THEN NULL
        WHEN current_lane.cycle_infrastructure_t LIKE '-1'  THEN 0
        WHEN current_lane.cycle_infrastructure_t IS NULL THEN 0
      END;
    IF (v_indicator <= v_indicator_help AND v_indicator IS NOT NULL) THEN
      UPDATE normalized_risk_values SET cycle_infrastructure = v_indicator WHERE fid = v_fid;
    ELSE
      UPDATE normalized_risk_values SET cycle_infrastructure = v_indicator_help WHERE fid = v_fid;
    END IF;



  END LOOP;

END;
$$ LANGUAGE 'plpgsql'
```

The function takes weights for each factor as input. Then, the weights and the information about the intersection are being combined. The output is a value for each turning relation in an intersection that makes statements regarding the risk for cyclists.

Listing A.14: Calculation of the risk index for each turning relation (`calculate_risk_factor.sql`)

```
DROP FUNCTION IF EXISTS weighting(numeric, numeric, numeric, numeric, numeric, numeric,
      numeric, numeric, numeric, numeric, numeric, numeric, numeric, numeric);
CREATE OR REPLACE FUNCTION weighting(numeric, numeric, numeric, numeric, numeric, numeric,
      numeric, numeric, numeric, numeric, numeric, numeric, numeric, numeric)
RETURNS void AS $$
  DECLARE
   weight_urban ALIAS FOR $1;
   weight_rails ALIAS FOR $2;
   weight_traffic_lights ALIAS FOR $3;
   weight_intersection_legs ALIAS FOR $4;
   weight_speed ALIAS FOR $5;
   weight_lane_number_cars ALIAS FOR $6;
   weight_lane_number_bikes ALIAS FOR $7;
   weight_gradient ALIAS FOR $8;
   weight_street_type ALIAS FOR $9;
   weight_round_about ALIAS FOR $10;
   weight_round_about_inner_circle ALIAS FOR $11;
   weight_round_about_cycle_infstr ALIAS FOR $12;
   weight_cycle_infrastructure ALIAS FOR $13;
   weight_mixed_traffic ALIAS FOR $14;

   no_lanes integer;
   nrv_lane normalized_risk_values%ROWTYPE;
   weight_counter numeric; -- in this variable, indicator*weight is summed up
   sum_weights numeric;
   v_risk_factor numeric;

  BEGIN
   no_lanes := (SELECT COUNT(*) FROM normalized_risk_values);

   DELETE FROM weighted_turnuses;

   FOR q IN 1..no_lanes LOOP
      -- loop through each turning relation
      SELECT INTO nrv_lane nrv.* FROM normalized_risk_values nrv ORDER BY fid LIMIT 1 OFFSET
      (q-1);
      weight_counter := 0;
      sum_weights := 0.0;

      -- in each of these if-statements it is checked, whether information
      -- concerning the variable is available, and whether is is weighted.
      -- if so, the product of factor's weight * factor's normalized risk value
      -- is added to the variable weight_counter.
      -- finally, the weight is added to the variable sum_weights

      -- urban
      IF (nrv_lane.urban IS NOT NULL AND weight_urban IS NOT NULL AND weight_urban > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.urban * weight_urban));
         sum_weights := sum_weights + weight_urban;
      END IF;

      -- rails
      -- Harris et al. (2013) detected that rails solely do play a role when the
      -- intersection is not regulated by traffic lights
      IF (nrv_lane.rails IS NOT NULL AND weight_rails IS NOT NULL AND weight_rails > 0
         AND (nrv_lane.traffic_lights IS NULL OR weight_traffic_lights IS NULL
             OR weight_traffic_lights <= 0)) THEN
         weight_counter := (weight_counter + (nrv_lane.rails * weight_rails));
         sum_weights := sum_weights + weight_rails;
      END IF;

      -- traffic lights
      IF (nrv_lane.traffic_lights IS NOT NULL AND weight_traffic_lights IS NOT NULL
         AND weight_traffic_lights > 0) THEN
         weight_counter := (weight_counter
                     + (nrv_lane.traffic_lights * weight_traffic_lights));
         sum_weights := sum_weights + weight_traffic_lights;
      END IF;

      -- edge degree
      IF (nrv_lane.intersection_legs IS NOT NULL AND weight_intersection_legs IS NOT NULL AND
      weight_intersection_legs > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.intersection_legs *
      weight_intersection_legs));
         sum_weights := sum_weights + weight_intersection_legs;
      END IF;
```

```
        -- speed
        IF (nrv_lane.speed IS NOT NULL AND weight_speed IS NOT NULL AND weight_speed > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.speed * weight_speed));
           sum_weights := sum_weights + weight_speed;
        END IF;

        -- number of crossed turning relations (cars)
        IF (nrv_lane.lane_number_cars IS NOT NULL AND weight_lane_number_cars IS NOT NULL AND weight_lane_number_cars > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.lane_number_cars * weight_lane_number_cars));
           sum_weights := sum_weights + weight_lane_number_cars;
        END IF;

        -- number of crossed turning relations (VRUs)
        IF (nrv_lane.lane_number_bikes IS NOT NULL AND weight_lane_number_bikes IS NOT NULL AND weight_lane_number_bikes > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.lane_number_bikes * weight_lane_number_bikes));
           sum_weights := sum_weights + weight_lane_number_bikes;
        END IF;

        -- gradient
        IF (nrv_lane.gradient IS NOT NULL AND weight_gradient IS NOT NULL AND weight_gradient > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.gradient * weight_gradient));
           sum_weights := sum_weights + weight_gradient;
        END IF;

        -- road type
        IF (nrv_lane.street_type IS NOT NULL AND weight_street_type IS NOT NULL AND weight_street_type > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.street_type * weight_street_type));
           sum_weights := sum_weights + weight_street_type;
        END IF;

        -- roundabout inner circle
        IF (nrv_lane.round_about_inner_circle IS NOT NULL AND weight_round_about_inner_circle IS NOT NULL AND weight_round_about_inner_circle > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.round_about_inner_circle * weight_round_about_inner_circle));
           sum_weights := sum_weights + weight_round_about_inner_circle;
        END IF;

        -- roundabout present?
        IF (nrv_lane.round_about IS NOT NULL AND weight_round_about IS NOT NULL AND weight_round_about > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.round_about * weight_round_about));
           sum_weights := sum_weights + weight_round_about;
        END IF;

        -- roundabout cycle infrastructure
        IF (nrv_lane.round_about_cycle_infstr IS NOT NULL AND weight_round_about_cycle_infstr IS NOT NULL AND weight_round_about_cycle_infstr > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.round_about_cycle_infstr * weight_round_about_cycle_infstr));
           sum_weights := sum_weights + weight_round_about_cycle_infstr;
        END IF;

        -- mixed traffic
        IF (nrv_lane.mixed_traffic IS NOT NULL AND weight_mixed_traffic IS NOT NULL AND weight_mixed_traffic > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.mixed_traffic * weight_mixed_traffic));
           sum_weights := sum_weights + weight_mixed_traffic;
        END IF;

        -- cycling infrastructure
        IF (nrv_lane.cycle_infrastructure IS NOT NULL AND weight_cycle_infrastructure IS NOT NULL AND weight_cycle_infrastructure > 0) THEN
           weight_counter := (weight_counter + (nrv_lane.cycle_infrastructure * weight_cycle_infrastructure));
           sum_weights := sum_weights + weight_cycle_infrastructure;
        END IF;



        RAISE NOTICE 'Fid: %', nrv_lane.fid;
        RAISE NOTICE 'Weight Counter: %', weight_counter;
        RAISE NOTICE 'Sum of weights: %', sum_weights;
        RAISE NOTICE '_______________________________________________________________';

```

```
      -- the risk factor of a turning relation is calculated by dividing the sum of
      -- the weighted normalized factors by the sum of the weights
      v_risk_factor := (weight_counter / sum_weights);

      INSERT INTO weighted_turnuses (fid, geom, risk_factor)
      VALUES (nrv_lane.fid, (SELECT ST_Transform(i.curved_geom, 3857)
                       FROM iplateau i WHERE i.fid = nrv_lane.fid), v_risk_factor);


    END LOOP;



  END;
$$ LANGUAGE 'plpgsql';
```

This is the default-version of the weighting function. Therefore, the weights are already known, meaning that weights and the information about the intersection can directly be combined. The output is a value for each turning relation in an intersection that makes statements regarding the risk for cyclists.

Listing A.15: Calculation of the risk index for each turning relation with the default values (`calculate_risk_factor_def_values.sql`)

```sql
-- this function uses the default values for the weighting  process
-- therefore, no user inputs are required

DROP FUNCTION IF EXISTS weighting_def();
CREATE OR REPLACE FUNCTION weighting_def()
RETURNS void AS $$
  DECLARE
   weight_speed numeric;
   weight_lane_number_cars numeric;
   weight_lane_number_bikes numeric;
   weight_gradient numeric;
   weight_round_about numeric;
   weight_cycle_infrastructure numeric;

   no_lanes integer;
   nrv_lane normalized_risk_values%ROWTYPE;
   weight_counter numeric; -- in this variable, indicator*weight is summed up
   sum_weights numeric;
   v_risk_factor numeric;

  BEGIN
   no_lanes := (SELECT COUNT(*) FROM normalized_risk_values);

   DELETE FROM weighted_turnuses;

   weight_speed := 0.3;
   weight_lane_number_cars := 0.15;
   weight_lane_number_bikes := 0.05;
   weight_gradient := 0.1;
   weight_round_about := 0.2;
   weight_cycle_infrastructure := 0.2;

   FOR q IN 1..no_lanes LOOP
      -- loop through each turning relation
      SELECT INTO nrv_lane nrv.* FROM normalized_risk_values nrv ORDER BY fid LIMIT 1 OFFSET
      (q-1);
      weight_counter := 0;
      sum_weights := 0.0;

      -- speed
      IF (nrv_lane.speed IS NOT NULL AND weight_speed IS NOT NULL AND weight_speed > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.speed * weight_speed));
         sum_weights := sum_weights + weight_speed;
      END IF;

      -- number of crossed turning relations (cars)
      IF (nrv_lane.lane_number_cars IS NOT NULL AND weight_lane_number_cars IS NOT NULL AND
      weight_lane_number_cars > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.lane_number_cars *
      weight_lane_number_cars));
         sum_weights := sum_weights + weight_lane_number_cars;
      END IF;

      -- number of crossed turning relations (VRUs)
      IF (nrv_lane.lane_number_bikes IS NOT NULL AND weight_lane_number_bikes IS NOT NULL AND
      weight_lane_number_bikes > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.lane_number_bikes *
      weight_lane_number_bikes));
         sum_weights := sum_weights + weight_lane_number_bikes;
      END IF;

      -- gradient
      IF (nrv_lane.gradient IS NOT NULL AND weight_gradient IS NOT NULL AND weight_gradient >
      0) THEN
         weight_counter := (weight_counter + (nrv_lane.gradient * weight_gradient));
         sum_weights := sum_weights + weight_gradient;
      END IF;

      -- roundabout
      IF (nrv_lane.round_about IS NOT NULL AND weight_round_about IS NOT NULL AND
      weight_round_about > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.round_about * weight_round_about));
         sum_weights := sum_weights + weight_round_about;
      END IF;

      -- cycling infrastructure
```

```
      IF (nrv_lane.cycle_infrastructure IS NOT NULL AND weight_cycle_infrastructure IS NOT
      NULL AND weight_cycle_infrastructure > 0) THEN
         weight_counter := (weight_counter + (nrv_lane.cycle_infrastructure *
      weight_cycle_infrastructure));
         sum_weights := sum_weights + weight_cycle_infrastructure;
      END IF;




      RAISE NOTICE 'Fid: %', nrv_lane.fid;
      RAISE NOTICE 'Weight Counter: %', weight_counter;
      RAISE NOTICE 'Sum of weights: %', sum_weights;
      RAISE NOTICE '_______________________________________________________________';

      v_risk_factor := (weight_counter / sum_weights);

      INSERT INTO weighted_turnuses (fid, geom, risk_factor)
      VALUES (nrv_lane.fid, (SELECT ST_Transform(i.curved_geom, 3857) FROM iplateau i
                        WHERE i.fid = nrv_lane.fid), v_risk_factor);


   END LOOP;



  END;
$$ LANGUAGE 'plpgsql';
```

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić with minor modifications by Hannah Augustin, who also kindly provided a set-up to write the body of the text in Markdown, leaving compilation and reference management to pandoc. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of May 1, 2023 (version 1.0).