BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

MODEL OF CYCLING TRAFFIC INTENSITY IN BRNO

MODEL INTENZIT CYKLISTICKÉ DOPRAVY V BRNĚ

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. RADOSLAV ELIÁŠ**
AUTOR PRÁCE

SUPERVISOR **Ing. JIŘÍ HYNEK, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2023

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

141152

Institut:	Department of Information Systems (UIFS)
Student:	**Eliáš Radoslav, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Information Systems and Databases
Title:	**Model of Cycling Traffic Intensity in Brno**
Category:	Information Systems
Academic year:	2022/23

Assignment:

1. Get acquainted with the data modeling of traffic intensities with a focus on bicycle traffic. Study ways of processing and evaluating such data.
2. Research the current approaches of cities to the creation of models of bicycle traffic intensities, the processing, and visualization of such data.
3. Perform an analysis of current data sources on cycling transport in the city of Brno ("Go to work by bike" campaign, Strava application, point sensors, traffic census, data from cameras, etc.).
4. Design a general data model for the analyzed data sources.
5. Implement the tool for processing the analyzed data sources and create a database based on the designed data model.
6. Visualize the processed cycling data in order to test the suitability of the data model and the created database.

Literature:

- Pucher, J., & Buehler, R. (Eds.). (2012). *City cycling*. MIT press.
- Oliveira, F., Nery, D., Costa, D. G., Silva, I., & Lima, L. (2021). A survey of technologies and recent developments for sustainable smart cycling. *Sustainability*, *13*(6), 3422.
- El Esawey, M., Lim, C., & Sayed, T. (2015). Development of a cycling data model: City of Vancouver case study. *Canadian journal of civil engineering*, *42*(12), 1000-1010.
- Francke, A., & Lißner, S. (2018). *Big Data in Bicycle Traffic: A User-oriented Guide to the Use of Smartphone-generated Bicycle Traffic Data*. Technische Universität Dresden, Chair of Transport Ecology and Chair of Traffic and Transportation Psychology.
- Tufte, E. (2001): *The visual display of quantitative information*. Cheshire, USA: Graphics Press.
- Strava. (2001). Better cities for cyclists and pedestrians. Online: https://metro.strava.com/.
- ArcGIS Developers. (2001). Documentation. Online: https://developers.arcgis.com/documentation/.

Requirements for the semestral defence:
Items 1 - 4.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Hynek Jiří, Ing., Ph.D.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	1.11.2022
Submission deadline:	24.5.2023
Approval date:	25.10.2022

Abstract

The Brno Data Department has access to multiple datasets regarding cycling traffic numbers. The goal of the thesis was developing a model integrating these sources for the Transportation Department of the city planning office to gain insights about how the infrastructure is used daily. Each dataset is aggregated to a different basemap with a slightly different street network. This thesis introduces an algorithmic approach to street matching based on similarity, overlap percentage and other parameters. Two algorithms for matching point-based and polyline-based geometries are presented. As well as a model mapping locations among different datasets and a dashboard visualizing values from them side-by-side. The robustness of the algorithms enables usage in any geographical application using spatial data. The dashboard provides useful information about cycling transport for both casual users and professionals designing the infrastructure of Brno.

Abstrakt

Oddelenie dát v Brne má prístup k viacerým dátovým sadám o počtoch cyklistov. Cieľom práce bolo vytvoriť model integrujúci tieto zdroje pre odbor dopravy magistrátu mesta, aby získali prehľad o tom, ako sa infraštruktúra denne využíva. Každý súbor údajov je agregovaný na inú základnú mapu s mierne odlišnou sieťou ulíc. Táto práca predstavuje algoritmický prístup k porovnávaniu ulíc na základe podobnosti, percentuálneho prekrytia a ďalších parametrov. Poskytnuté sú dva algoritmy na porovnávanie geometrie založenej na bodoch a úsečkách geometrie. Rovnako aj model priraďujúci lokácie medzi rôznymi dátovými sadami a informačný panel vizualizujúci hodnoty z nich vedľa seba. Robustnosť algoritmov umožňuje ich použitie v akejkoľvek geografickej aplikácii využívajúcej priestorové údaje. Informačný panel poskytuje užitočné informácie o cyklistickej doprave pre bežných používateľov aj odborníkov, ktorí navrhujú infraštruktúru mesta Brna.

Keywords

cycling, transport, traffic, infrastructure, data analysis, data warehouse, visualization, data modelling, datasets, geographical information systems

Klíčová slova

cyklistika, doprava, infraštruktúra, dátová analýza, dátový sklad, vizualizácia, modelovanie dát, dátové sady, geografické informačné systémy

Reference

ELIÁŠ, Radoslav. *Model of Cycling Traffic Intensity in Brno.* Brno, 2023. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.

Rozšířený abstrakt

Mestá orientované majú negatívny vplyv na životné prostredie a sú náročné na priestor. Bicykle sú skvelou alternatívou dopravy, ale ich integrácia do existujúcej infraštruktúry prináša rôzne problémy. Získavanie viac informácií pre dopravné oddelenie umožňuje vykonávať lepšie rozhodnutia a zabezpečiť, aby sa cyklistická doprava stávala atraktívnejšia a populárnejšia. Existuje viacero dátových sád, avšak každá z nich má len čiastočné časové alebo priestorové pokrytie mesta. Ich integrácia by mohla zlepšiť oboje.

Dostupné súbory údajov zahŕňajú automatické snímače na bicykle, ktoré detekujú cyklistov na rôznych miestach v meste, údaje z kampane „Do Práce Na Kole", ktorá podporuje udržateľné spôsoby dopravy vytvorením súťaže, v ktorej môžu skupiny ľudí alebo spoločnosti súťažiť, kto za mesiac najazdí najviac kilometrov. Ďalej údaje zo štatistík sčítania cyklistov, ktoré vykonáva Brnenská komunikačná spoločnosť a údaje z športovej aplikácie Strava, ktorá sa používa na meranie a zdieľanie GPS trás. Každý súbor údajov agreguje zozbieraný počet cyklistov do inej uličnej siete.

Nekonzistentnosť základných máp nedovoľuje možnosť jednoduchého prepojenia, takže je potrebný algoritmický prístup na porovnanie ulíc medzi rôznymi sieťami. Algoritmus na integráciu rôznych uličných sietí využíva zaokrúhľovanie súradníc na transformovanie pseudo-spojitého priestoru na diskrétny. Váhuje podobnosť línií podľa uhlov a používa na porovnávanie percentuálny prístup k prekrývaniu ohraničujúcich obálok. Konečný dátový model využíva nepriame kódovanie a mapovanie na koncové súbory údajov. Algoritmus párovania je unikátny prístup a je vytvorený robustne, takže ho možno použiť v množstve aplikácií, kedykoľvek sa vyskytnú nezrovnalosti v základnej mape. Cieľom algoritmu je nájsť rovnakú ulicu v rôznych sieťach základných máp a vytvoriť mapovanie, ktoré umožňuje porovnávať hodnoty medzi zdrojmi údajov. Optimalizácia algoritmu je vykonaná tak, že vybrané okno mapy je rozdelené na menšie, rovnako veľké segmenty. Potom sa ulice porovnávajú len s tými, ktoré sa nachádzajú v rovnakom segmente.

Navrhovaný dátový model využíva sieť ulíc z OpenStreetMap ako zdroj pravdy. Poskytuje užitočné informácie o type cesty, stave vozovky atď. Uvedený algoritmus k nej priradí koncové súbory údajov o cyklistickej doprave v Brne. Sú mapované podľa príslušných ID, takže ich vnútorné hodnoty nie sú duplikované, ale len nepriamo zakódované a vzájomne mapované. Týmto spôsobom sa o pridávanie nových údajov starajú ich vlastníci a nie je potrebné ich v tejto práci riešiť.

Výsledné dáta zobrazuje vytvorený informačný panel (*dashboard*). Je vytvorený v rámci portálu geografického informačného systému ArcGIS, ktorý využíva aj Dátove oddelenie mesta Brna a má v ňom uložené dátové sady využívané v tejto práci. To znamená, že nie sú treba prenášať alebo hosťovať samostatne, keďže ovládací panel ArcGIS sa môže natívne dotazovať na tieto vrstvy, ktoré sú verejné. Hlavné zobrazenie panela využíva integračný model. Výberom ulice v mapovom prehliadači sa zobrazí používateľovi výber tabuliek a grafov na pravej strane, zostavené z údajov týkajúcich sa vybranej ulice. V ľavom bočnom paneli sa nachádza výber dátumu, ktorý slúži len na vyhľadávanie údajov z automatických snímačov, pretože tieto sú generované na hodinovej báze. Prehliadač máp možno prepnúť na rôzne pentlogramy zobrazujúce niektoré bežné spôsoby použitia dátových sád. Údaje zo služby Strava nebolo možné pridať, pretože podliehajú veľmi prísnym licenčným pravidlám a nemôžu byť zdieľané s nikým bez výslovného prístupu od tímu Strava Metro. Jednoduchý nástroj na zobrazovanie dát zo Stravy v podobnom formáte ako zvyšok panelu je zahrnutý v práci a môže byť využitý ľudmi, ktorí majú prístup k týmto dátam. Táto práca bola zapojená do konferencie EXCEL@FIT 2023, kde vyhrala ocenenie odborným panelom, ako aj partnerom z praxe.

Model of Cycling Traffic Intensity in Brno

Declaration

I hereby declare that this Master's thesis was prepared as an original work by the author under the supervision of Mr. Jiří Hynek Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Radoslav Eliáš
May 15, 2023

Acknowledgements

I would like to sincerely thank my supervisor for his guidance and hands-on approach to counselling.

Contents

Chapter 1

Introduction

In today's more environmentally conscious culture, more and more people realize that it's better for society, to limit car dependency in big cities. Alternative ways of moving around town are greener and more inclusive to people without cars, or driving licenses, or those who are unable to operate a vehicle for a multitude of reasons. Other forms of transport create significantly less air pollution, noise pollution, and they decrease the risk of fatal traffic accidents. Designing our cities for people, not cars, creates a better and richer community.

That's where major cities started to implement **cycling** as an ideal form of transportation. Bicycles are cheap, sustainable, and often faster than cars in dense areas as they can easily skip through traffic. They also save space for all those overcrowded parking lots and most importantly — they save money. With this shift of movement, the **bike sharing services** became widely popular around larger urban hubs. Cycling infrastructure is still in the early stages in most cities. The major challenges are the funds required and the space limitations of already established streets, roads, and pathways. This leads to cyclists sharing their designated part of the street with either cars or pedestrians.

The only way to make cycling safe and attractive to the wide masses is by creating safer and dedicated infrastructure for bicycles. In order to do that, the city planning office needs to have a good understanding of how the current infrastructure is used, and where are the critical spots that need to be addressed.

This thesis attempts to solve these issues in the Czech city of Brno, by creating a data pipeline integrating multiple sources of cycling traffic data into a database, storing it, and preparing it for future use. An algorithm for matching different street networks is developed and used to incorporate said datasets. The final created model is then visualized by an intuitive dashboard. Chapter number 2 breaks down cartography basics and geographical applications. Ways of storing and processing such data are explained in Chapter 3. Cycling data and their nuances are addressed in Chapter 4, as well as approaches to documenting cycling traffic in different cities around the world. There are multiple complex data sets used in the presented application. These are analyzed in Chapter 5. Chapters 6 and 7 introduce the proposed architecture of the application and the data. As well as its final implementation, including the public dashboard created. Chapter 8 explains experiments conducted with the system and their findings.

Chapter 2

Geospatial Applications

This chapter introduces basic information and terminology concerning geospatial data, projecting them onto a 2D map, and transforming them into a digital space. Understanding the process of generating maps and how they relate to Earth, helps design more accurate and robust applications.

2.1 Map Projections

This section gathers information mostly from [20, 4, 8].

Displaying a spherical planet like the Earth in two dimensions requires a process of "flattening", called projection. There is no way to preserve all information when reducing dimensions. The surface of the sphere needs to be distorted to correctly draw it onto a flat canvas. Common properties that are distorted as an aftermath of projection are area, form, distance, and direction [22].

There are three commonly used types of projections classified according to the surface area from which they are derived [23]:

- Cylindrical
- Conical
- Azimuthal

Every projection has strengths and weaknesses. This is the reason for the need for multiple philosophies. A good cartographer needs to understand the differences and choose the correct model for his purpose 2.1.

Conical Projections

Conical projection places a cone around the earth and unwraps it into a canvas. It's mostly useful for mapping long east-to-west regions because the created distortion is constant along common parallels. This is why conical projections are very popular for regional maps in mid-latitude areas (approximately 20° to 60° North and South). Perhaps the most popular and widely used conical projection is **Lambert Conformal Conic**, released in 1772 and used to this day[1].

[1] https://en.wikipedia.org/wiki/Lambert_conformal_conic_projection

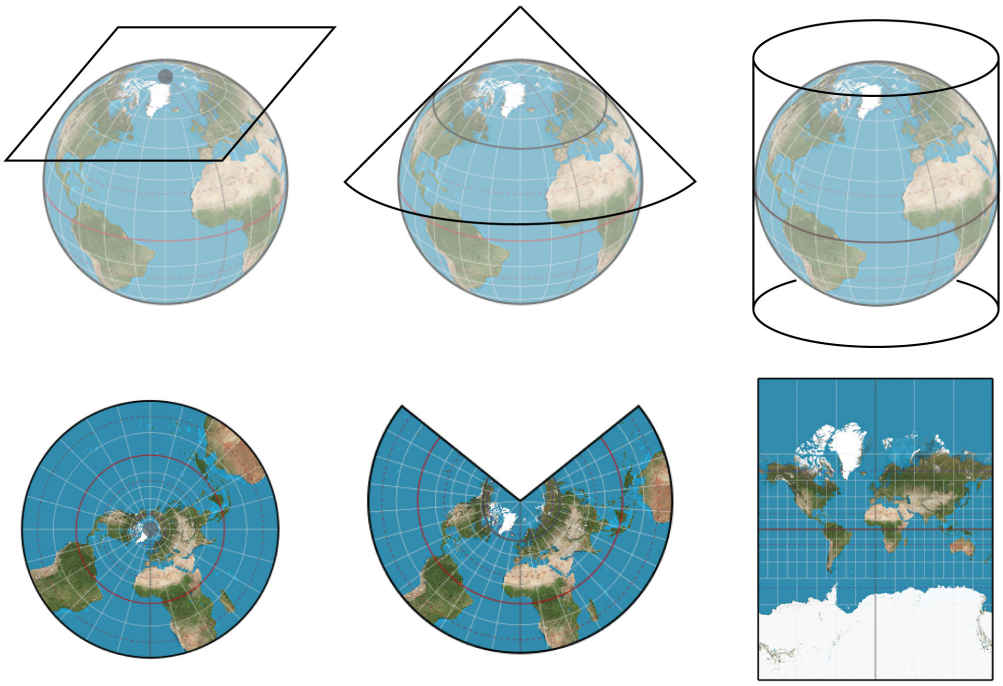


Figure 2.1: Types of mapping projections from the left: azimuthal, conical, cylindrical. Source: [8].

Cylindrical Projections

Placing a cylinder around a globe and unraveling it creates a projection considered a standard for nautical purposes and general navigation in the world. It keeps constant true direction, meaning a straight line on the map is the same direction as would show a compass. The most famous cylindrical projection is the **Mercator**, released in 1569[2]. While this projection is very close to the reality around the Equator, its distortion rapidly increases while moving away and closer to the Arctic. Many newer projections are still based on this first model.

Azimuthal Projection

These types of projections plot the surface of the Earth using a flat plane. They are most commonly used over Polar areas but can be used for small-scale maps of continents such as Australia. This is a conformal projection in that shapes are well preserved over the map, although extreme distortions do occur towards the edge of the map.

2.1.1 Coordinate Systems

A geodetic coordinate system is a system used to identify the location of points on the Earth's surface. It uses a set of mathematical formulas to describe the location of a point in relation to a reference ellipsoid, which is a mathematically defined shape that approximates the Earth's shape. This section gathers knowledge from [10].

[2] https://en.wikipedia.org/wiki/Mercator_projection

Latitude and Longitude

The location of a point on the surface of the Earth in reference to the equator and the prime meridian is indicated by these coordinates. Latitude, which is measured in degrees north or south of the equator, is the angle between a place on the surface of the Earth and the equator. Longitude, which is represented in degrees east or west of the prime meridian, is a measurement of the angle between a place on Earth's surface and the prime meridian. Geographic information systems (GIS) and other applications frequently identify locations on the Earth's surface using latitude and longitude, which are widely used in geodetic coordinate systems.

Universal Transverse Mercator (UTM) Coordinates

These coordinates are used to represent a point's distance from a central meridian in meters. Maps of smaller areas typically use UTM coordinates because they accurately depict the surface of the Earth, while greater scales can lead to distortion. In surveying, mapping, and other fields where an exact position is crucial, UTM coordinates are frequently utilized.

2.2 Geographical Information Systems

A geographical information system (GIS) is a tool used for capturing, storing, analyzing, and managing spatial and geographic data [21]. In order to collect, manage, analyze, and present all types of geographically linked data, it incorporates hardware, software, and data. GIS can be used to create maps and reports based on spatial data and to analyze and visualize spatial data, such as the locations of roads, buildings, and rivers.

Numerous fields, such as urban planning, resource management, transportation, and environmental evaluation, utilize this technology. To manage resources more effectively, it can be used to examine spatial linkages, patterns, and trends. Additionally, interactive maps that let people explore and engage with spatial data visually can be produced using GIS.

Some common components include data input and management tools, mapping and visualization tools, and analysis and modeling tools. Maps, satellite photos, and other types of spatial data are just a few of the different formats that GIS data can take. New techniques and tools are continually being created to expand this technology's capabilities and open the door to more complex types of analysis and visualization.

Geospatial File Types

There are many commonly used file types to store geospatial data [20].

- **Geojson** file format was adapted from JavaScript Object Notation (JSON) format. It stores coordinates as text including points, lines, and polygons. GeoJSON stores objects within curly braces and in general, has less markup overhead.
- **Shapefile** is the most commonly used geospatial format, often considered an industry standard. It consists of three different types of files
  - .shp — feature geometry
  - .shx — shape index

  - .dbf — attribute data

- **GPX** (GPS Exchange format) is an XML schema that describes waypoints, tracks, and routes captured from a GPS receiver. Because GPX is an exchange format, you can openly transfer GPS data from one program to another based on its description properties.

- **GML** allows for the use of geographic coordinates extension of XML. GML stores geographic entities (features) in the form of text. Each feature has a list of properties, geometry (points, lines, curves, surfaces, and polygons), and a spatial reference system.

- **OSM** files are the native XML-based file for the OpenStreetMap project (discussed later in this section). These files are a collection of vector features from crowd-sourced contributions from the open community. the more efficient, smaller PBF Format ("Protocolbuffer Binary Format") is an alternative to the XML-based format stored in binary.

OpenStreetMap

The OpenStreetMap project (OSM) is a free geographic database developed and maintained by the open-source community. It gathers data from volunteer contributors and other available sources like public government datasets. OSM is the golden standard for base maps and street networks in all GIS applications. Thanks to the open-source nature of the data, it's often more recent and up-to-date than the data of the private sector mapping vendors like Google Maps.

This rich and robust database of streets, objects, earth topology, and many other points of interest is used to make electronic maps, create turn-by-turn navigation systems, or visualize any kind of geospatial data [28]. OpenStreetMap Foundation also provides a public API useful for building custom solutions that require communication with the OSM database. Full snapshots of the database are also periodically released as downloadable files.

ArcGIS

ArcGIS is a software platform for creating, managing, analyzing, and sharing geographic and spatial data. It consists of a suite of products designed to work together to enable users to create, share, and use maps and other spatial data.

It provides a robust set of utilities and APIs in multiple programming languages, for creators of geospatial applications. ArcGIS uses a subscription payment model but offers a free base-level subscription for students and open-source developers.

The key feature of this geographic information system is cloud-based data hosting, the basic structure can be seen in Figure 2.2. It enables developers to store and retrieve data in multiple formats, version them, and process the data close to the final application according to the "data locality" theorem [9]. It states that heavy computations should be moved as close to the data location as possible, instead of moving the data into the computational cluster.

A standard file or a dataset is represented as a **feature layer**. ArcGIS provides rich querying capabilities with SQL and spatial queries and supports all commonly used data file formats from Section 2.2.

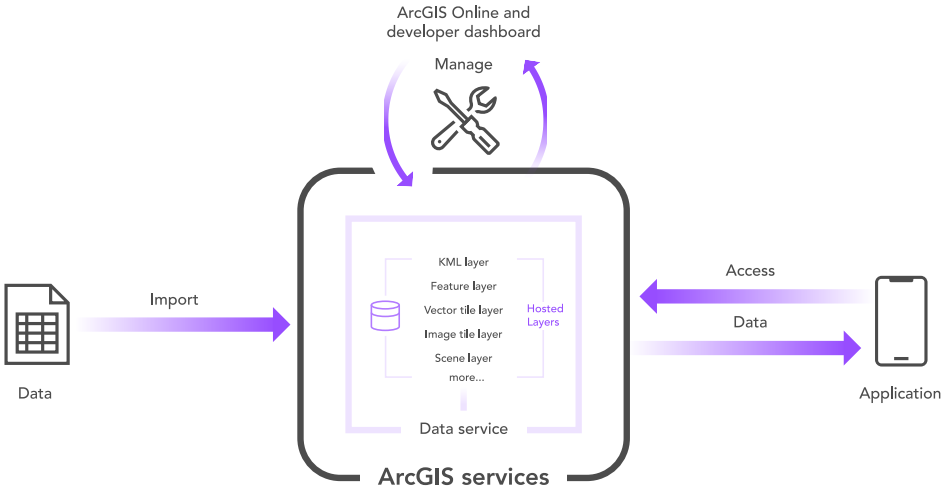


Figure 2.2: Basic structure of ArcGIS data hosting features. Source: [16].

The ArcGIS web interface has an easy-to-use tool to create maps and visualizations from the stored data. It features multiple base map options and an interactive user-friendly filtering system for ad-hoc visuals and analyses.

2.3 Personal Localization and Navigation

With the development of Global Navigation Satellite Systems (GNSS), it became possible to collect individual location information. They are satellite-based systems that locate an object on the ground by using a network of satellites orbiting the planet.

GNSS systems operate by sending satellite-based signals to a ground-based receiver. Based on how long it takes a message to get to the receiver, the receiver determines how far away each satellite is. The receiver can establish its location on the surface of the Earth by calculating the distance from at least three satellites. [27]

GNSS systems are widely used for navigation, location tracking, and timing applications. They are utilized in a variety of gadgets, such as drones, smartphones, and GPS units for automobiles. Surveying, mapping, and other geospatial applications all make use of GNSS systems.

The first GNSS to be put into use was the Global Positioning System from the United States of America. Subsequently, several nations adopted their own GNSS. Even today, the acronym GPS is frequently used as a term for a navigation system. It was initially introduced and exclusively intended for military use, but it soon became accessible to the general population. Thus, enabling the acquisition of personal location data.

Table 2.1 illustrates the differences among the most popular systems.

System	GPS	GLONASS	BeiDou
Owner	United States	Russian Federation	China
Orbital altitude	20,182 km	19,310 km	21,150 km
Period	11 h 58 min	11 h 16 min	12 h 38 min
Number of satellites	31 over 6 orbital planes	28	5 GEO satellites 30 MEO satellites
Frequency	1.57542 GHz (L1 signal) 1.2276 GHz (L2 signal)	1.602 GHz 1.246 GHz	1.561098 GHz (B1) 1.20714 GHz (B2)

System	Galileo	IRNSS
Owner	European Union	India
Orbital altitude	23,222 km	36,000 km
Period	14 h 5 min	23 h 56 min
Number of satellites	4 in-orbit validation satellites 8 operational satellites (22 satellites budgeted)	3 GEO satellites 4 GSO satellites
Frequency	1.559 GHz (E2) 1.260 GHz (E6)	1.17645 GHz (L5) 2.492028 Ghz (S1)

Table 2.1: Comparison of five largest Global Navigation Satellite Systems and their specifications. Source: [30].

Chapter 3

Data Storage Solutions

Large volumes of data are managed and stored effectively and efficiently using databases. They offer a mechanism to access, modify, and work with data. Databases can also guarantee data security, consistency, and integrity. They are also helpful in situations where multiple users need to obtain the data at the same time. Thus, some kind of a database is inevitable, when building a model from large quantities of data, that needs to be regularly queried and updated, like the cycling traffic model of Brno.

3.1 Data Warehousing

A data warehouse (DW) is an integrated repository of data put into a form that can be easily understood, interpreted, and analyzed by the people who need to use it to make decisions. It's a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions [31].

The subject-oriented property describes how a DW's data is arranged around the main subjects of an organization's interests. Customers, products, sales, and vendors are a few examples of subjects. The integrated property denotes that some meta-data and other pertinent external data are also incorporated into a DW's data, in addition to the data from all operational database systems. Because of their nonvolatile nature, DWs seldom have their data updated. DW typically contains data spanning several years. This enables DW users to examine historical trends, patterns, correlations, rules, and exceptions.

3.1.1 Online Transactional and Analytical Processing (OLTP, OLAP)

An OLTP system captures and maintains transaction data in a database. Individual database entries with many fields or columns are used in each transaction. Due to the constant reading, writing, and updating of OLTP databases, the focus is placed on quick processing. Built-in system logic assures data integrity even if a transaction fails. Relational databases are generally used for transactional processing.

OLAP applies complex queries to large amounts of historical data, aggregated from OLTP databases and other sources, for data mining, analytics, and business intelligence projects. The emphasis here is on response time to these complex queries. Each query involves one or more columns of data aggregated from many rows.

Table 3.1 breaks down the basic characteristics and aspects of these two approaches.

	OLTP	OLAP
Characteristics	Handles a large number of small transactions	Handles large volumes of data with complex queries
Query types	Simple standardized queries	Complex queries
Operations	Based on INSERT, UPDATE, DELETE commands	Based on SELECT commands to aggregate data for reporting
Response time	Milliseconds	Seconds up to hours depending on the amount of data to process
Design	Industry-specific, such as retail, manufacturing, or banking	Subject-specific, such as sales, inventory, or marketing
Source	Transactions	Aggregated data from transactions
Purpose	Control and run essential business operations in real-time	Plan, solve problems, support decisions, discover hidden insights
Data updates	Short, fast updates initiated by user	Data periodically refreshed with scheduled, long-running batch jobs
Space requirements	Generally small if historical data is archived	Generally large due to aggregating large datasets

Table 3.1: Comparison of OLAP and OLTP key properties and differences. Source: [33].

3.2 Extract-Transform-Load Process

This section is paraphrased from an article called "A proposed model for data warehouse ETL processes" [15].

Extraction–transformation–loading (ETL) tools are pieces of software responsible for the extraction of data from several sources, it's cleansing, customization, reformatting, integration, and insertion into a data warehouse. Building the ETL process is potentially one of the biggest tasks of building a warehouse; it is complex, time-consuming, and swallows most of the data warehouse project's implementation efforts, costs, and resources. Building a data warehouse requires focusing closely on understanding three main areas: the source area, the destination area, and the mapping area.

During the process, data are extracted from an OLTP database, modified to fit the data warehouse schema, and then fed into the data warehouse. A lot of data warehouses also include information from non-OLTP systems, like spreadsheets, legacy systems, and text files. ETL solution is often a complex problem that consumes a significant portion of the data warehouse development efforts and requires the skills of business analysts, database designers, and application developers. The ETL process is not a one-time event. As data sources change, the data warehouse will be periodically updated.

Here are the individual steps explained in more detail [26]:

- **Extraction**: The extraction step is responsible for extracting data from the source systems into the staging area. Each data source has its distinct set of characteristics that need to be managed in order to effectively extract data for the ETL process. The process needs to effectively integrate systems that have different platforms, such as different database management systems, different operating systems, and different communications protocols.

- **Transformation**: In the staging area, the raw data undergoes data processing. Here, the data is transformed and consolidated for its intended analytical use case. The first steps of transformation often are filtering, cleansing, de-duplicating, validating, and authenticating the data. Afterward, it's possible to perform calculations, translations, or summarizations based on the raw data. This can include changing row and column headers for consistency, converting currencies or other units of measurement, editing text strings, and more.

- **Loading**: In this last step, the transformed data is moved from the staging area into a target data warehouse. Typically, this involves an initial loading of all data, followed by periodic loading of incremental data changes and, less often, full refreshes to erase and replace data in the warehouse. For most organizations that use ETL, the process is automated, well-defined, continuous, and batch-driven. Typically, it takes place during off-hours when traffic on the source systems and the data warehouse is at its lowest.

3.3 Data Warehouse Schemas

When designing a data warehouse, there are two popular options: the star or the snowflake schema. They both utilize relational databases as the founding blocks, but differ in the final model. The common theme is that each model consists of a central **fact table** and multiple adjacent **dimension** tables [35]. A fact table is a table that stores facts that measure the subject of the warehouse, such as sales, cost of goods, or profit for a business. Fact tables also contain foreign keys to the dimension tables. These foreign keys relate each row of data in the fact table to its corresponding dimensions and levels. The dimension tables contain the attributes of the data, such as dates or addresses.

Star and Snowflake Schemas

In a star schema, all dimension tables are directly connected to the fact table. The fact table contains information about metrics or measures, while the dimension tables contain information about descriptive attributes. The star schema is very simple and easy to understand, making it ideal for cloud data warehousing and business intelligence applications.

The snowflake schema is an extension of the star model. By normalizing the dimension tables, we get a stretched version of the star schema, where each original dimension table is represented by a series of new, linked tables. Multiple advantages and disadvantages come with this step. For example, normalized data into a snowflake schema require less storage space. However, it's harder to implement, query, and update. Visual representation of these schemas can be seen in Figure 3.1.

3.3.1 Data Lake

A data lake is a centralized repository that allows the storage of raw, unstructured data in its native format. Unlike traditional data warehouses, which store data in a structured way, data lakes allow data to be stored in its original state.

Large volumes of data from many sources, including relational databases, social media, and sensors, as well as semi-structured data from log files and JSON, can be handled by data lakes. This information can be saved in its raw form and then later altered, enriched, and structured as necessary. Table 3.2 compares them to data warehouses.

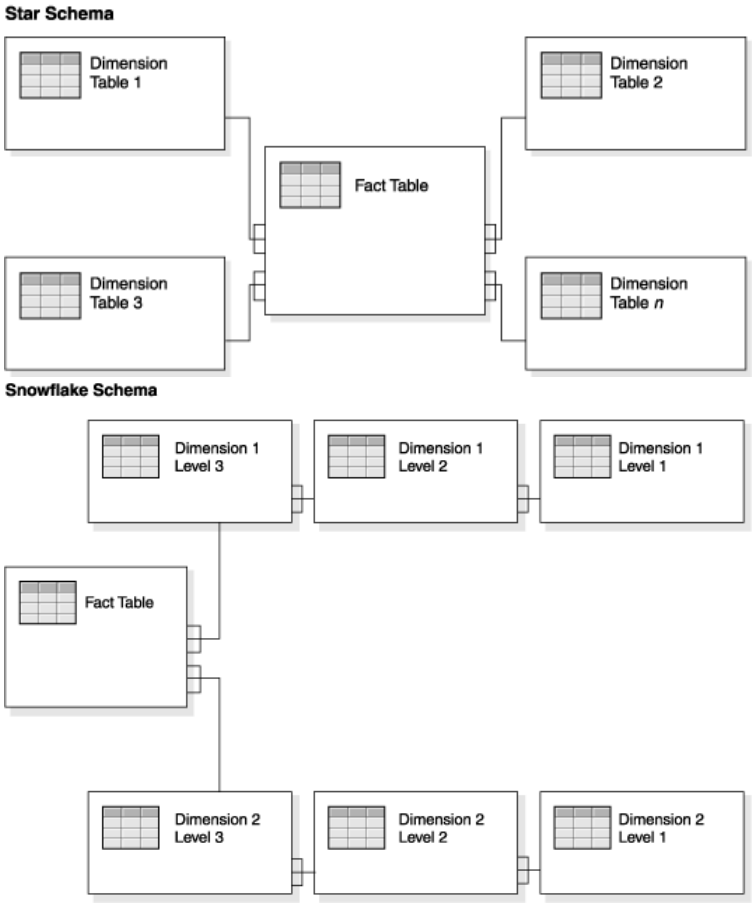


Figure 3.1: Star and snowflake schemas examples. Sources: [25, 24].

Characteristics	**Data Warehouse**	**Data Lake**
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, websites, mobile apps, and social media
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost storage
Data Quality	Highly curated data that serves as the central version of the truth	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling

Table 3.2: Key differences between data warehouses and data lakes. Source: [6]

Chapter 4

Cycling Data and Existing Solutions

Understanding the spatial, temporal, and social dimensions of cycling is becoming increasingly important for both urban planning practice and mobility research in order to adapt to the fast-changing urban environment and encourage more people to cycle. Cities must place more attention on understanding cycling and active travel than motorized transport in order to promote human-scale transportation. Types of information deducible from cycling data:

- spatial patterns (destinations and route selection)
- temporal patterns (specific times of the day, parts of the month, or entire seasons)
- user demographics (gender, age, personal background)
- travel types/reasons (commute, leisure, other)

4.1 Types of Cycling Data

Despite the growing demand, finding cycling data and getting access to it remain major obstacles. The lack of relevant data is often a limiting factor for transportation and urban planning. Authorities normally have well-established data-gathering procedures for motorized transportation, but cycling-related data is rarely included. These data are generally gathered on a project-by-project basis at isolated places, with little to no reuse potential.

Part of the difficulty stems from the typical methods of data collection, which include travel diaries, questionnaires, interviews, static bike counters, and various statistics. Although these materials are helpful, their accessibility and thoroughness are both constrained. Small sample sizes, inadequate spatio-temporal coverage, difficult route and travel chain detection, and reliance on self-reporting are some of the typical problems. The quick advancement of sensor technologies has made it possible to collect mobility data from a number of sources that reveals people's daily movements and mobility patterns.

Data sources mostly used in cycling analyses include three categories [37]:

1. automatic bike counters and manual observations
2. bike sharing services (BSS), GPS tracking, sports applications

3. surveys, interviews, and public participation events

Each category has its own benefits and downsides. Counters and manual consensus counting by observation have high accuracy and detection rate, but are very limited in the spatial department and generally capture only a small portion of the area of interest. Additionally, manual observations require human interaction and usually happen only monthly or even annually, decreasing the temporal knowledge gained from these data.

The second category (GPS data) introduces potentially the highest volume of recorded data by far. However, these are somewhat unreliable inputs caused by faulty sensors and other unpredictable impacts, that require much more cleaning before they can be used further. Another undesirable feature is the inherent need for user participation or willingness by a third-party company to provide these data sets.

Surveys usually rely on a good distribution among users, otherwise, the data can get highly biased towards specific groups. For example, if a survey is conducted on a social media platform, those are generally used by younger generations and the resulting demographic won't be uniformly distributed. Table 4.1 serves as a summary of common attributes and types of information possible to obtain from introduced data sources.

4.1.1 Bike Counters

Counters are an easy tool to gather information on cycling traffic levels in specific places. Automatic bike counts employ a range of detection methods, including inductive loops, magnetometers, thermal imaging, infrared and radar waves, pneumatic tubes, and piezoelectric detectors [37]. Counters are typically seen in areas with high bike traffic to maximize their potential because they are permanently fixed to one position. Counter data often include the counter's coordinates and are usually aggregated to multiple time intervals. An example of such a machine can be seen in Figure 4.1.

Figure 4.1: Automatic bicycle counter in Copenhagen, Denmark. Source: [11]

4.1.2 Manual Observations

Manual observations are any type of data that is gathered by an observer from one or more locations on a single occasion, on a regular basis, or in a variety of ways. Simple cyclist

counts, sub counts (by age, gender, etc.), bike type counts (such as regular, sport, or cargo bikes), identifying whether people cycle alone or in groups, or evaluating cycling behavior, such as at an intersection to identify potential hazard risk are some examples of this type of data. These are most often conducted by some sort of governing entity like a city municipality or a road and motorway directorate.

4.1.3 Bike Sharing Trip Data

Bike-sharing services are now typical in many places. These systems typically keep track of bike rental journeys with a separate record for each trip. These records normally include the user and bike identifications, departure and arrival times, and the origin and destination stations. Trip-level datasets may be linked to standard user characteristics like age, gender, postal code, and subscription type, depending on the data supplier and the system registry settings. Some BSS providers equip their bikes with GPS locators that collect specific journey routes, but those are often the subject of strict security protocols and are not made available to external entities.

4.1.4 GPS Tracking

Numerous sources of bicycle statistics are progressively incorporating GPS monitoring. This data type is recorded with a separate GPS device or a specific location-tracking application on a portable device (for example, a mobile phone or a smartwatch). It captures the coordinates of a device at an exact time. GPS units can be carried by cyclists as portable devices or mounted to their bicycle frames. Most modern devices have a recording frequency of one second. This sequence of points in space and time can then be used to generate an exact route of the cyclist, which enables many different analyses and visualizations. Sensitive personal information about the user can be gained from these data, so they should always be shared cautiously.

4.1.5 Sports Applications

Activity and health monitoring applications such as Strava[1] or Sports Tracker[2] are increasingly used as a source of cycling data. They enable users to track their movements, heart rate, altitude, traveling speed, and distance, establish performance goals, and share results with other users. Data from these solutions are often owned by the app vendors, but are under strict privacy policies and can't be shared in their raw form. The majority of data sets in this category are aggregated due to privacy concerns and stripped of any personal information about the end users. Using data like this eliminates the need for communication with individual users and streamlines the process of gathering sources.

4.1.6 Surveys

Survey information on cycling can range from brief, specialized questionnaires focusing on a single subject to extensive mobility studies at the national level focusing on the overall travel habits of the population, including cycling. Survey data are typically collected in a disaggregated form, however, to protect respondents' privacy, when disseminated

[1] https://strava.com
[2] https://www.sports-tracker.com

through intermediary sources, survey data are frequently aggregated. Online forms are being used more frequently in surveys, which has made it simpler to distribute them and gather information from larger audiences. In general, national travel surveys are gathered annually or every two to three years over the course of a few days of intensive data collection.

4.1.7 Public Participation GIS

Public participation GIS (PPGIS) mapping refers to techniques that allow the collection of both qualitative and quantitative data through the active involvement of individuals. Participants can map locations (points), routes (lines), areas (polygons), or combinations of these using a normal PPGIS program in addition to answering survey questions. It can be used in the context of cycling to gather data on things like trip origins and destinations, route preferences, or points of interest and link this data to specific place-related experiences and preferences associated with the mapped entities.

Data type	Accessing the data	Typical attributes	Spatial accuracy and coverage	Temporal information	User information
bike counters data	through data portals of public sector agencies or through measuring campaigns	counter ID, bike count, time	limited to the counter locations, accuracy is good when the exact locations of counters are known	year-round continuous data with good temporal resolution, aggregated counts by time intervals	no individual user data
manual observation data	through observation campaigns	observation ID, lat./long. coordinate, time, observation count/description	limited to places the observers are located or visit	limited to observation periods, repeated observations possible	user counts and basic description of sub counts in passive observations
bike sharing origin-destination trip data	through the system operator or from open data portals	trip ID, depart/return station, depart/return time, user ID	high as it is linked to the station locations, coverage bound to the station network	timestamps available for hires, collected throughout the system operation period	anonymized individual-level data, sometimes enriched with demographic variables
GPS-tracking data	through GPS-tracking campaigns or survey apps	user ID, track ID, lat./long. coordinates, timestamp	high and accurate, track records show exact routes; potential irregularities	high temporal resolution, typically from 1 second to 1 minute	detailed individualized data may be collected separately
sports application data	by purchasing or through collaborations with responsible companies	segment ID, time, cyclist count, cyclist count by trip, types	high accuracy with street segment-level data, coverage typically the best	high resolution and coverage (raw data can contain minute-by-minute counts of segments)	aggregated user counts or limited anonymized user information (age, gender)
survey data	through survey campaigns, open data portals, or public sector agencies	respondent ID, answers, comments	typically limited spatial coverage (main origin and destination location)	limited, based on a number of recurrent surveys	anonymous and detailed individual-level data
PPGIS data	through PPGIS-survey campaigns, open data portals	user ID, marker ID, answers, comments	varies by user, data quality is uncertain	limited, users may provide time values	anonymous individual-level data

Table 4.1: Comparison of commonly used cycling data sources, their attributes, and achievable knowledge from such data. Source: [37]

4.2 Existing Solutions by Other Cities

The goal of modeling cycling traffic in a city is not unique to Brno. Other municipalities have tackled this problem with various levels of success. Multiple studies by universities or private companies have been conducted to try to build this type of model. This Section introduces some of the notable ones, as well as their findings and obstacles.

4.2.1 Dresden

Possibly the most comprehensive study was published by the researchers at the Technical University of Dresden (TUD), called "Big Data in Bicycle Traffic" [18]. Their goal was to develop a practical introduction to the use of GPS data in bicycle traffic planning. First, the guide explains the basics of GPS data, where can they be acquired, how can they be used, and what can be learned from them. Afterward, a case study applying all the introduced principles is performed in the city of Dresden.

Data Sources

This study focuses on GPS data collected by the users of any location-tracking application. These are then provided by the app vendors, generally for a fee. Two examples given here are Strava[3] and BikeCitizen[4]. The latter is a local German navigation app for cyclists. According to the authors, it proved to be not robust enough for such an analysis, thus they decided to use Strava. The case study didn't attempt to integrate multiple sources, like the one proposed in this thesis. It turned out to be outside of the scope of the research. Instead, the resources were used to thoroughly analyze just one dataset and gain from it as much knowledge as possible.

For the basemap, TUD researchers went with the most popular OpenStreetMap (Section 2.2), which the Strava also uses for providing data aggregated by street sections. Additional information about the infrastructure was gained from the street network provided by the city administration, which is linked to Germany's Authorative-Topographic-Cartographic Information System.

Findings [18]

The study mostly used the visualizations available in the Strava Metro dashboard, which is provided with the data for free. A closer look at these is taken in the next chapter. The goal was to determine whether the dataset was satisfactory for this type of analysis.

According to multiple statistical tests conducted, these data seem representative of the actual traffic volume. However, it is recommended to calibrate the absolute values by some sort of manual census carried out by humans or mechanical devices. Although, the speed values found in the dataset are slightly higher, on average by about 5 km/h. Relative differences are still reasonably accurate. Strava, as an application mainly focused on recreational and fitness-based cycling, implements an algorithm to determine whether an activity is a commute or not. It identifies them mainly through a "point-to-point" matching method. Frequently cycled relations between origin and destination are classified as regular and therefore as a commuting or everyday trip [18]. This definition of a commute is slightly broader than the one used in day-to-day English, thus the numbers might be higher

[3] www.strava.com
[4] www.bikecitizens.net

than expected. Circular trips with the beginning and end at the same spot are usually cut out as those do not satisfy the generic understanding of a commute. The study found this algorithm to be adequate with high accuracy.

4.2.2 Vancouver

The next reference study was published in the Canadian Journal of Civil Engineering, with the name "Development of a cycling data model: City of Vancouver case study" [14]. These researchers attempted to develop a framework for building a model of cycling traffic from limited available data, using various mathematical methods and estimation techniques. The aforementioned invented procedures were then applied to specific datasets concerning the Vancouver case study. Available data sources included:

- automatic bicycle counts — by mechanical counters
- manual intersection counts — census counted by humans
- bicycle road network — basemap
- historical weather data

Therefore, this case study had very limited spatial coverage of the city, but fairly accurate counts on those observed sections and streets.

4.2.3 Methodology

First, the census and the counters data needed to be integrated into one model. This step is fairly straightforward, as both data points represent just a single location in space. The only required transformation was to match the temporal values. The census data were aggregated on an hourly basis, while the counters had 15-minute intervals.

Estimation of daily bicycle volumes (DBV) was the first metric calculated. Some of the available data had information for the whole 24-hour period, while some only captured a part of the day. Each option required a slightly different approach. With the full interval, the daily volume is just a simple summation of its parts. Peak hours are easily calculated here as well. The study introduces multiple search methods capable of matching the street to one with complete DBV data. Each approach uses a combination of these parameters: matching by the same road class (e.g. bike path, highway...), matching by using the same date, matching by using similar weekdays during the same months and years, matching by any weekday but in the same month and same year. After approximating the corresponding streets, it's possible to estimate DBVs for the incomplete data, using **periodic count factor** denoted PCF. PCF for link (street) j, time period t, and captured volume V can be calculated as shown in 4.1.

$$\mathrm{PCF}_{jt} = \frac{V_{jt_1} + V_{jt_2} + \ldots}{\mathrm{DBV}_j} \tag{4.1}$$

Then, each street has multiple periodic factors, which can be used to estimate minimal, maximal, and mean daily bicycle volume 4.2.

$$\mathrm{DBV}_{i_{\min}} = \frac{V_{it}}{\mathrm{PCF}_{jt_{\max}}}, \quad \mathrm{DBV}_{i_{\mathrm{mean}}} = \frac{V_{it}}{\mathrm{PCF}_{jt_{\mathrm{mean}}}}, \quad \mathrm{DBV}_{i_{\max}} = \frac{V_{it}}{\mathrm{PCF}_{jt_{\min}}} \tag{4.2}$$

Equally, the monthly and annual average daily bicycle traffic (denoted MADB and AADB respectively) can be estimated. First, for links with full month or year coverage as a simple mean, then, adjustment factor is calculated from that, which is then used to calculate MADB for partially recorded or unrecorded streets. These three steps for the monthly statistic can be seen in Equation 4.3.

$$\mathrm{MADB}_i = \frac{\sum \mathrm{DBV}_i}{n}, \qquad \mathrm{DF}_{ij} = \frac{\mathrm{MADB}_j}{\mathrm{DBV}_{ij}}, \qquad \mathrm{MADB}_{i_{\mathrm{mean}}} = \mathrm{DBV}_{\mathrm{day}_{i_{\mathrm{mean}}}} * \mathrm{DF} \tag{4.3}$$

This process enables a city administration to generate cycling volumes and intensities from limited resources and create a model for future planning and decision-making. The Vancouver case study also developed a simple web-based user interface to visualize these new data.

4.2.4 Salzburg

Department of Geoinformatics at the University of Salzburg wrote an article called "Agent-based Bicycle Traffic Model for Salzburg City" [36]. This approach greatly differs from others typically used, as well as those mentioned in this chapter. It doesn't use collected datasets to approximate bicycle volumes, instead, it uses an **agent**-based simulation, creating a stochastic model. Comprehensive domain knowledge is required to accurately design and parametrize the agents to reflect actual cyclists and their movement patterns.

This model was designed to simulate bicycle traffic during a weekday in late spring, carefully choosing an optimal time and date for the experiment. It considered the effects weather has on cycling traffic, thus a part of the year with limited precipitation and ideal temperatures. Summer months were undesirable because a significant portion of Salzburg's population are students, who have school break during that time. Thus, the date `6th of June 2013` was selected.

Three categories of agents were selected: working cyclists, student cyclists, and leisure cyclists. Each agent was assigned two trips during the day of testing, one from home to an individual destination and the other one back. Starting locations were generated randomly from a resident density grid. The type of agent determined destinations. Student cyclist destinations were generated from locations of universities and faculties. Destinations for working cyclists were rendered from the workplace density grid and leisure cyclists selected destinations within residential areas. The number of agents and their category affiliation were deducted from studies and surveys conducted in the city in the previous years. Based on this, 16% of the absolute population of Salzburg were cyclists, from which 44% were work-related and the rest divided between students and leisure cyclists. These parameters could be adjusted to simulate possible shifts in the cycling culture and future of the city. The exact times of the agent trips were stochastically chosen, following normal distribution around 7:00 in the morning and other normal distribution around 16:00 for working cyclists and 11:30 — 14:30 for students.

Results

The outputs of the simulation were validated by data from automatic bike counters in three separate locations. The comparison can be seen in Figure 4.2. This result suggests that the agent-based model could not intuitively estimate the complex behavior of a large group

of various people. However, validation data were very limited so a more comprehensive experiment could alter the outcomes.

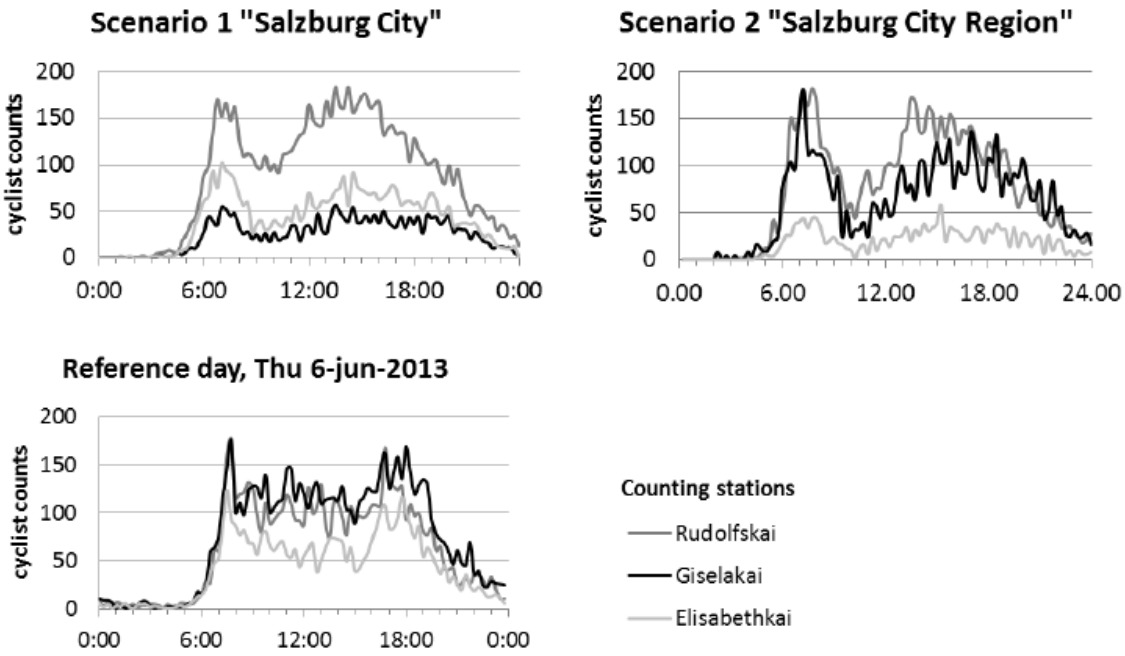


Figure 4.2: Cyclist counts for three counting stations for the simulated scenarios (top) and the validation reference day, 6th June 2013 (bottom). Source: [36].

4.2.5 České Budejovice

The Department of Transport and Logistics of the Institute of Technology and Businesses in České Budejovice conducted research [7] inspecting possible usages of available data in cycling infrastructure planning. It analyzed the current state of bicycle transport in the South Bohemian region and introduced possible solutions for collecting data to improve it. Figure 4.3 shows the impact the size of a city has on the role of cycling transport in the municipality. Bikes had the most popularity in mid-sized cities. In the case of larger municipalities, it would suggest unsatisfactory infrastructure.

The study then puts forward the types of data that are commonly possible to obtain about cycling traffic. These include national and city-wide traffic censuses, automatic counting devices, as well as phone applications that collect GPS locations. Based on a census conducted by the Czech Statistical Office, Figure 4.4 depicts the relationship between the length of a journey and the type of transportation used for it. Bikes seem to be used for journeys shorter than 7km, which would mean that most travel in Ceske Budejovice could be substituted for bicycles, as the area of the city is of similar size.

Automatic bike sensors are introduced as a good option for long-term data collection, as they provide information almost non-stop, besides outages and other errors. However, they lack spatial coverage of the city. Furthermore, the functionality of 5 different types of counters is explained. These include pressure sensors, pyroelectric sensors, infrared sensors, electric induction loops, and magnetic sensors. This section is useful for a city infrastructure planning office, to decide which is the best choice.

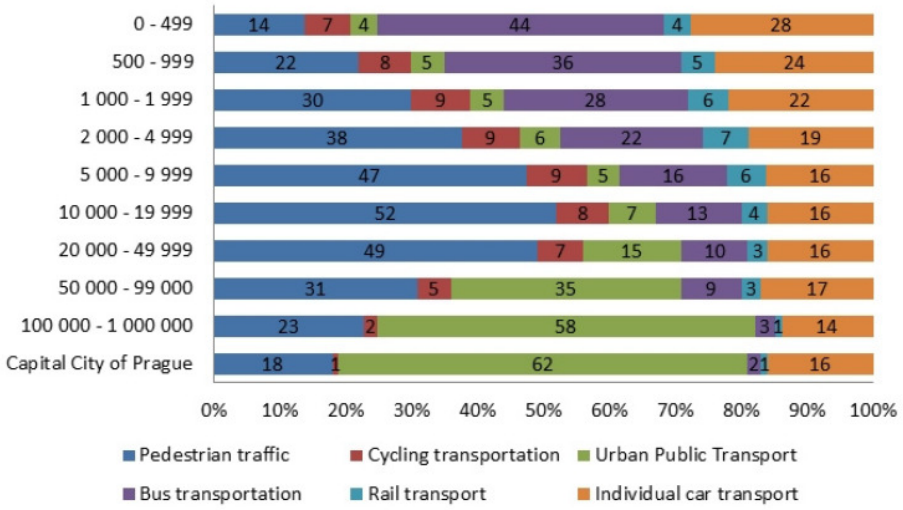
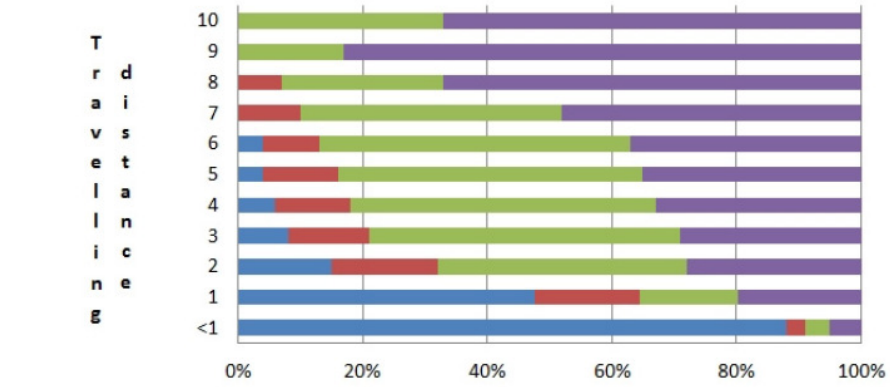


Figure 4.3: Portions of different transport methods depending on the size of a municipality. Source: [7].



	<1	1	2	3	4	5	6	7	8	9	10
Pedestrian traffic	88	48	15	8	6	4	4	0	0	0	0
Cycling transportation	3	17	17	13	12	12	9	10	7	0	0
Urban Public Transport	4	16	40	50	49	49	50	42	26	17	33
Individual car transport	5	20	28	29	33	35	37	48	67	83	67

Figure 4.4: Type of transportation used for a specific length of journey in Ceske Budejovice. Source: [7].

Data acquired from Strava company are mentioned as a source of big volume but questionable relevance caused by the focus of the application on leisure-oriented users.

Chapter 5

Analysis of Available Datasets

In the case study concerning the city of Brno, multiple datasets were available. The municipality gathers or acquires this data from third party vendors and sources. The goal of this proposed thesis was to integrate them and enable the city planning office to use the gained knowledge for better decision-making. The city is running an online hub with a large number of datasets freely available to the public called "Brno in Numbers"[1]. Open data in this catalog vary from economy and labor market to transport, public health, and housing. This chapter serves as an analysis of those datasets, providing useful information about their characteristics and traits.

5.1 Bike Counters

A brief introduction to automatic bike counters was introduced in Section 4.1.1. As of the beginning of the year 2023, Brno has 19 sensors installed around the city. Data from the counters are automatically stored and uploaded to the online Data Hub mentioned previously. The refresh happens daily and the counts are aggregated at one-hour intervals. Therefore, each data point represents one counter during one hour of a given day. New data are appended to the last dataset so a periodic update from the source just needs to take the new rows into account. Most of the Brno sensors also differentiate between cyclists in opposite directions and provide this information in separate columns. These specific counters also measure pedestrians, but that detail isn't relevant to this use case. The dataset can be downloaded in common file formats like CSV, Shapefile, or GeoJSON. Table 5.1 shows a few rows from the dataset.

UnitId	Latitude	Longitude	EndOfInterval	FirstDirection	SecondDirection
CAM1	49.183699	16.602423	2021-06-22 03:59:59	6.0	18.0
CAM8	49.226283	16.658782	2021-06-21 17:00:00	8.0	5.0
CAM2	49.183642	16.602337	2021-06-22 03:59:59	13.0	5.0
CAM9	49.218652	16.575043	2021-06-21 02:00:00	0.0	0.0

Table 5.1: Example data from the counters trimmed of irrelevant columns for simplicity.

[1] https://data.brno.cz/

5.2 Strava Application Metro-view

Strava is a company developing an equally-named popular application and a social platform used by the majority of sport-oriented people to share and compare activities, usually collected by GPS devices, heart rate monitors, altitude meters, power meters, and other sensors. Running and cycling are the most common types of exercise uploaded by the community. An example activity is shown in Figure 5.1.

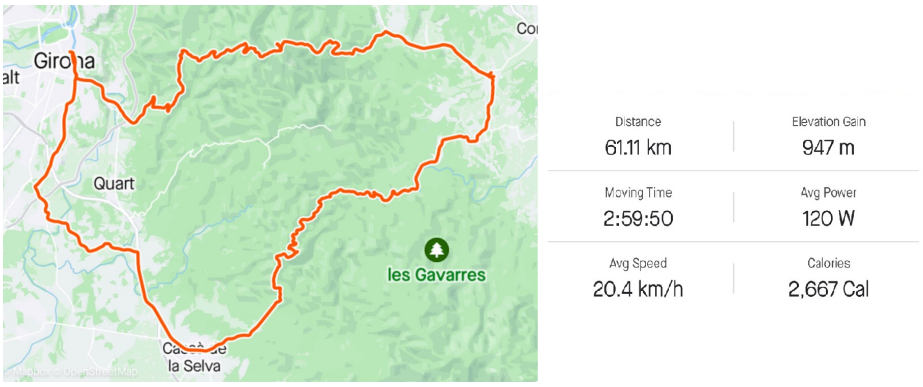


Figure 5.1: An example activity recorded in the Strava application, with tracked location, heart rate, and commonly calculated attributes like final distance and elevation covered, average speed, and others.

This firm launched a program called Strava Metro[2]. Its purpose is to provide the millions of data collected to urban planning offices and other governing bodies to help them better understand how the infrastructure is currently used and to determine the best possible changes for the future.

5.2.1 Data Collection and Methodology

All the activities are owned by the end-users, therefore, they are subject to strict licensing rules and restrictions. The company has an obligation to protect the privacy of its users. In order to do that, all the personal information connecting an activity to its user is stripped before shipping the data [32].

Additionally, to protect the user's mobility patterns, all the activities are aggregated by street segments, meaning there is no way to track a person's location from the Metro dataset. Strava uses the OpenStreetMap street network as a basemap for this aggregation process. Each small section from an activity is matched to the closest link from the network and it's assumed that is the road the cyclist took. The final model delivered to the Strava Metro subscriber is just a list of individual street segments and the absolute number of cyclists using that link in a given time frame. This parameter can be customized by the user, as the aggregation from that point is just a simple summation from the smallest possible time frame. During this process, the original street segments found in the OpenStreetMap network are broken down into smaller pieces, where each intersection is a splitting point. A short data sample with daily granularity is shown in Table 5.2. The dataset provides additional columns splitting the absolute count into intervals by user gender, age, time of day of the activity, and average speed.

[2] https://metro.strava.com/

As an application focused on leisure cycling, the representativeness of the data regarding the generic population is in question. Therefore, the company developed an algorithm used to identify commuting trips among the data. It was discussed in more detail in the Findings Section 4.2.1 of the Dresden case study. The non-commute trips can then be filtered out, to keep only the relevant information depending on the goal of the model.

5.2.2 Dataset Access and Refreshing

The MetroView is accessible through a dashboard provided by the developers. It supplies multiple visualizations, map views, and an option to generate a new dataset for a specified time interval. It's possible to set both the ultimate period from which data are included in the final file, as well as whether the included data are aggregated to hourly, daily, monthly, or yearly counts. New data are generally added to the dashboard periodically once a month. These datasets are generated on an on-demand basis, thus aren't accessible through an API, which makes it difficult to automate the process of extracting new data. They can also only be downloaded while logged in to a user account.

edge_uid	activity_type	date	forward_count	reverse_count	osm_reference_id
175537337	Ride	2021-11-09	5	0	4934858
175555034	Ride	2021-11-15	0	5	704390536
175555034	Ride	2021-11-24	0	5	704390536

Table 5.2: Sample simplified data from the Strava Metro export.

5.3 Bike Census BKOM

"Brněnské komunikace, a.s." (BKOM) or **Brno road network institution** is responsible for doing traffic census in the city of Brno. Bike census happens independently from the generic motor-vehicle count, therefore the numbers should be valid. It transpires biannually on the even years [13]. However, the data for the year 2022 are missing, so available counts are from 2016, 2018, and 2020. As of early 2023, there are 628 different street segments in this dataset. The count depicts one regular work day and one day of the weekend, to capture differences in commuter behavior. These are denoted **prac_YYYY** and **vik_YYYY** respectively. The supplier doesn't provide any documentation explaining the methodology of the data collection. Street links in the dataset are defined by their coordinates, but it isn't clear if they are collected from a public road network database like OpenStreetMap, or if the location is determined by a private basemap owned by the company. Table 5.3 shows representative data.

ObjectId	prac_2016	vik_2016	prac_2018	vik_2018	prac_2020	vik_2020	coordinates
2	30	60	45	80	40	70	16.5176,49.2298,...
3	15	20	15	20	10	15	16.4630,49.2558,...
4	0	0	0	0	15	0	16.4657,49.2561,...

Table 5.3: Sample data from the BKOM bike census, irrelevant columns were discarded.

5.4 Road Census ŘSD

Road and Motorway Directorate of the Czech Republic (ŘSD) also conducts a vehicle and traffic census of the whole country. Data regarding bicycle counts can be extracted and used in the proposed model of bike traffic intensity. The census happens every 5 years and the last was in 2020 [39]. It focuses on multiple types of roads, including both highways and local inner-city streets. The 2020 dataset has 108 streets marked as Brno-related. Again, a portion of the data can be found in Table 5.4.

PČ	SIL	ÚSEK	ZAČÁTEK ÚSEKU	KONEC ÚSEKU	C	Skupina komunikací
6562	23	6-7540	mimoúr.x s D1	mimoúr.x s 602	0	II
6563	23	6-7541	mimoúr.x s 602	ul.Petra Křivky	1	II
6564	23	6-7542	ul.Petra Křivky	mimoúr.x s 42	5	II
6565	41	6-6091	x s 42	zaús.374 od Černovic	35	II
6566	41	6-6090	zaús.374 od Černovic	zaús.15278	81	II

Table 5.4: State-wide traffic census data sample. Information regarding fuel-powered vehicles was removed. The column headers' meanings are as follows: PČ — road ID, SIL — international road number, ÚSEK — international road section ID, ZAČÁTEK ÚSEKU — the beginning of the segment in plain words, KONEC ÚSEKU — the ending of the segment in plain words, C — number of cyclists in a 24 hour period, Skupina komunikací: road classification.

These data have no location information, so the only way to integrate them with other sources is to manually match and assign them to a generally available basemap. The source does not specify an exact date of the calculation, instead, it's presented as an average of multiple days during the period from June to August. It doesn't take into account the effect of work day versus weekend on cycling commuters.

5.5 Bike to Work Campaign

AutoMat[3] is a Czech organization focused on improving the environment and being sustainable. One of its most popular campaigns is called "Bike to Work". People build teams with friends or coworkers and register for the competition. The purpose is to use green forms of transport to work or school, like walks, bikes, or scooters, as much as possible. Users record their commutes with GPS sensors and upload them to an application. The most successful teams then win prizes provided by the sponsors of the campaign, while simultaneously lowering CO2 emissions and all kinds of pollution. One run of the campaign is usually for a month. The collected data are then aggregated and anonymized by the firm, similarly to Strava, in Section 5.2. They provide these datasets to municipalities and other companies, in order to analyze commuter patterns and improve the current infrastructure [5].

The available dataset has information for the month of May each year. Raw data from the users were aggregated to a road network basemap. The authors decided not to use the standard OpenStreetMap, instead, a dataset by the name "StreetNet", developed by Central European Data Agency[4]. This dataset is a paid service and is not available to researchers. Thus, the road IDs present can't be used to integrate with other data

[3] https://auto-mat.cz
[4] https://www.ceda.cz/

sources and matching must happen by location coordinates. A fragment from the data is demonstrated in Table 5.5.

GID_ROAD	data_2018	data_2019	data_2020	data_2021	geometry
3515.0	3	5	3	34	(16.6631 49.2968, 16.6632, ...
3516.0	10	17	10	49	(16.6623 49.2970, 16.6631, ...
4168.0	2	4	2	7	(16.6411 49.3932, 16.6413, ...
4169.0	1	1	1	18	(16.6411 49.3930, 16.6417, ...

Table 5.5: Data sample from the "Bike to work" campaign. Unnecessary columns were abandoned for simplicity.

Chapter 6

Proposed Algorithmic Approach

In order to integrate all the aforementioned data, one important problem needed to be solved, and that is dataset incompatibility. Each dataset is aggregated to a different street network, meaning that a simple join operation on the street ID or geometry columns isn't possible. Many basemap differences found include different street borders like beginning and endings, different street granularity where one street from dataset a equaled to multiple ways joined together from dataset B, or even occurrences of streets running parallel a few meters next to the same one from the other network. Figure 6.1 should help visualize these different possibilities.

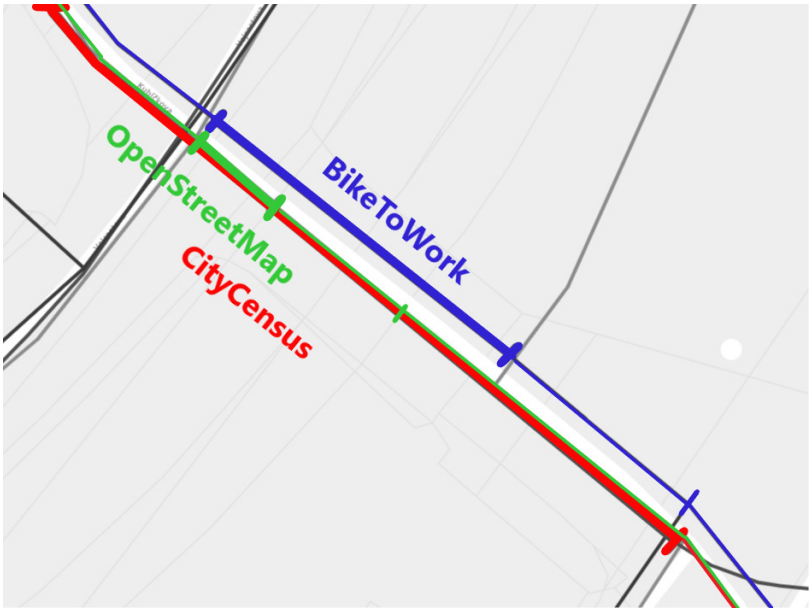


Figure 6.1: Demonstration of basemap inconsistency among datasets. Seemingly, the same street has a different granularity, length, and even location coordinates.

The number of roads in a single city can reach tens or hundreds of thousands, so this problem needs to be solved by an algorithmic approach that can be automatized, as doing it manually would be massively time-consuming. Datasets based on a street network model in the Brno case study include Strava data, "BikeToWork" campaign data, and the city census data. Another type of geometry is for example in the Automatic Sensors dataset and it is on a single-point location basis. Basically, each record is spatially represented by a point,

instead of a line like in the matter of streets. Another algorithm to assign point locations to corresponding roads needs to be implemented. Both solutions should ideally be generic enough so they don't depend on a specific dataset, rather just require one type of geometry included.

6.1 Point Matching Algorithm

Finding the best match between a street network and a set of points is a fairly straightforward task. In terms of cycling traffic analysis, these can represent automatic counters, locations of accidents, points of interest, or even bike-sharing spots. The most obvious solution is to assign the point to the nearest street, which is represented by a polyline. Only a few problems can occur during this process. If the real-life street where the point is located is missing from the dataset, it will be assigned to the nearest existing one. However, this possibility can't be solved besides a manual creation of the street in the dataset, so the only option is to assume the completeness of the network. Another potential issue is a case where multiple lines have the same distance to the point, but two adjacent streets shouldn't differ in the number of cyclists greatly, as they usually have to pass through both of them. Figure 6.2 demonstrates how the algorithm will calculate distance and choose the best match.

Figure 6.2: A simple visualization of the point matching algorithm. The point in blue represents an automatic counter and distances to the closest streets are displayed as dashed lines. It should choose the red street based on its proximity.

The algorithm should have these attributes. Input:

- Dataset with basemap streets defined by their geometries
- Set of points with their coordinates

Output:

- the same basemap dataset with a new column consisting of point IDs

Using this type of indirect encoding eliminates duplicity that would occur if the datasets would be merged by a regular JOIN operation, in the case of multiple matches for one street.

6.2 Line Matching Algorithm

As mentioned at the beginning of this chapter, there are multiple complications needed to solve in order to find the best match between two different street networks. The final algorithm should take them into consideration. These are broken down below. The inputs of the algorithm should be a single street with its geometry and a set of streets serving as candidates for matching. The output would be the street found as the best match, referenced either by its ID or geometry.

6.2.1 Offset Tolerance

For example, calculating overlap between two lines and choosing the highest match isn't enough, because the same actual street can be represented as two parallel lines in different datasets, as is the case in Figure 6.1 for the OpenStreetMap and BikeToWork datasets. These lines never actually touch, even though they're supposed to be the same object, therefore this approach won't suffice. A possible solution is to introduce **deviation tolerance**. All the spatial information in the geometry column of the datasets is in the standard coordinate system defined by latitude and longitude. These numbers are in the range $< -90, 90 >$ and $< -180, 180 >$ respectively. So, one point is a combination of two `real` numbers. Therefore, the number of decimal points determines the accuracy of the precise location and thus it's possible to round to a specific point and omit an offset error.

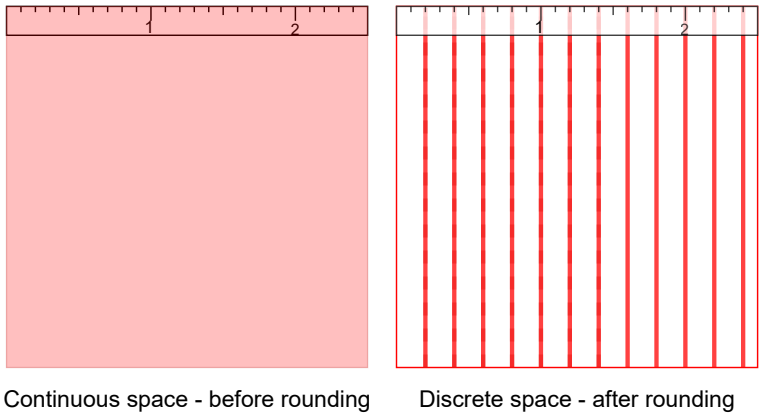


Figure 6.3: Rounding up the coordinates transforms a pseudo-continuous space into a discrete one.

A practical example of this theory is as follows (calculated by [12]):

```
Point A: lat=16.00001, long=49.00001
Point B: lat=16.00002, long=49.00001
distance(A, B) == 1.84 meters (rounded)
```

Rounding all coordinates to 5 decimal points would then mean that objects with that much span actually touch and overlap, so it's easier to calculate similarity. The number of rounding digits should be a hyperparameter of the algorithm. It's important to note that the rounding up will raise the chance of false positive matches, by grouping streets in close proximity together, as in Figure 6.3.

6.2.2 Maximal Accepted Angle

If the algorithm tries to find candidate matches by a simple boolean overlap attribute, it's possible to eliminate fake positive matches by taking the angle between the lines into account. The working assumption is that the same line in different datasets should go mostly parallel. Automatically disqualifying lines that have a bigger angle than at least 45° will eliminate perpendicular lines and others that have a low level of similarity. However, the optimal angle limit can't be found and does not really exist, therefore the actual number is the next hyperparameter of the algorithm. An important note is that most streets in all the datasets are actually polylines and not simple lines. It's not possible to calculate the angle between them as each consists of multiple line segments, thus a more complicated method (visualized in Figure 6.4) is required.

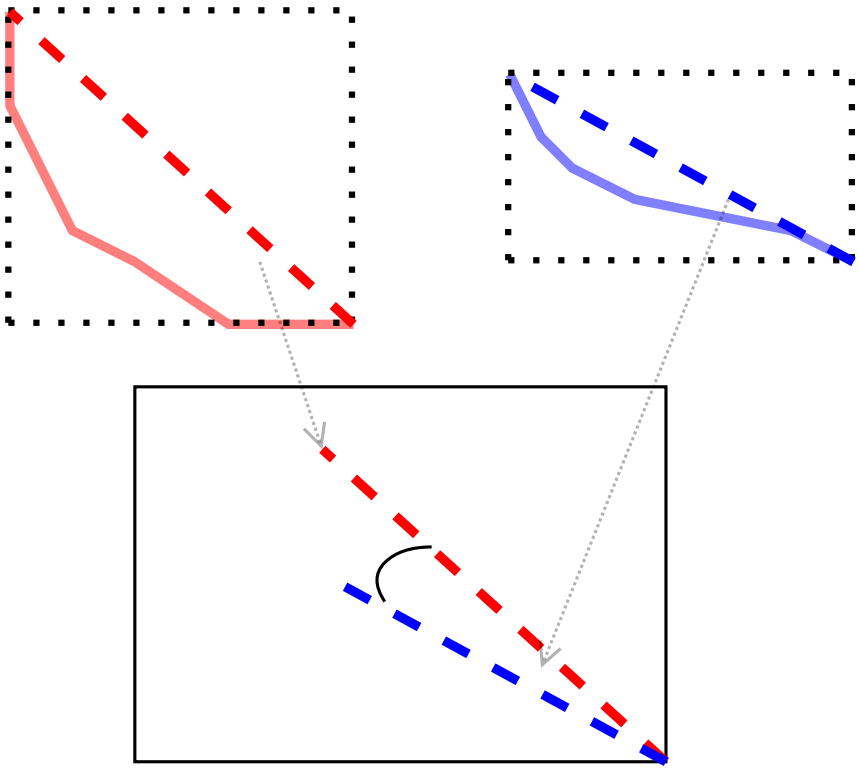


Figure 6.4: The only way to calculate the angle between two polylines is to get their bounding box coordinates and evaluate the angle between their diagonals.

6.2.3 Best Match Score

While the first two attributes mentioned before can serve as a sort of preprocessing or cutoff limit, the calculation of the final score or similarity will determine the street that's picked

at the end. Even after introducing the offset tolerance, choosing the match with the highest overlap percentage isn't enough, because the line will definitely touch intersecting streets, but the actual target line the algorithm is trying to find can run perfectly parallel just outside the scope of the rounding buffer. It's also really difficult to calculate overlap in the space of longitude and latitude coordinates, as it's continuous and not a discrete space, therefore even a small deviation would mean two points don't technically lie perfectly on one another. Another thing to consider is the time complexity of the final solution. The OpenStreetMap dataset for the greater Brno area has around 65 000 streets, each one defined by multiple lines. Comparing all the points of each street with the others would require hours if not days of computation each time.

The most promising option seems to be adding the bounding box or the buffer around the line to the equation. Evaluating the overlap between two buffers instead of the exact lines eliminates those subtle differences and deviations. Note that this will not help in all cases, namely with perfectly horizontal and vertical streets, as their bounding boxes are the same as the original lines. This principle is shown in Figure 6.5.

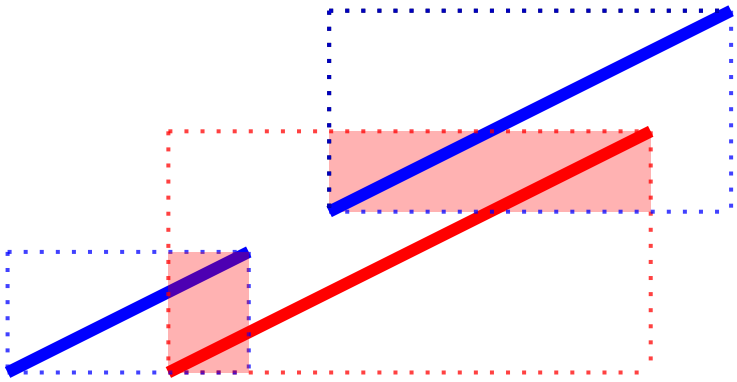


Figure 6.5: The figure shows how calculating overlap between line buffers finds matches even when lines don't intersect. The upper right blue line would be picked as the best match because the overlap percentage of the other one is much lower.

6.3 Optimalization

When dealing with big data, it's important to think about the time and space complexity of the algorithm. The volume of the data will only increase in the feature so it's important to future-proof. In the case study of Brno, the selected area has $\simeq 65,000$ streets in the OpenStreetMap basemap. If the matching algorithm would be run on it and some other set with similar street coverage *naively*, comparing each street from one to each street from the other (N:N) would mean square complexity of $\mathcal{O}(n^2)$. That would final in $65,000^2 = 4,225,000,000$ comparisons. This number needs to be addressed and cut down to a more manageable count.

6.3.1 Window Segmentation

The idea behind this optimization technique is that distant streets can't possibly match, therefore it's not necessary to compare one street with others that are far away. It's possible

to split both datasets into segments and compare a street with just the ones in the same segment (Figure 6.6).

The first step is segment generation. The selected area of streets has a bounding box defined by its limits. Splitting this region into equally-sized rectangles will generate a list of their corresponding coordinates: [min_x, min_y, max_x, max_y]. After that, each dataset can be processed by adding a new column with segment IDs, determined by the streets segment membership. There are multiple options on how to decide that. Full line coverage by the segment isn't ideal, because streets will often cross the segment borders. One way is to assign the street to both segments, for the cost of some duplicity. The other is to decide by a single limit point of the line, for example, the leftmost one. This could introduce a small false negative rate, if those points of corresponding streets would lay on opposite sides of the border.

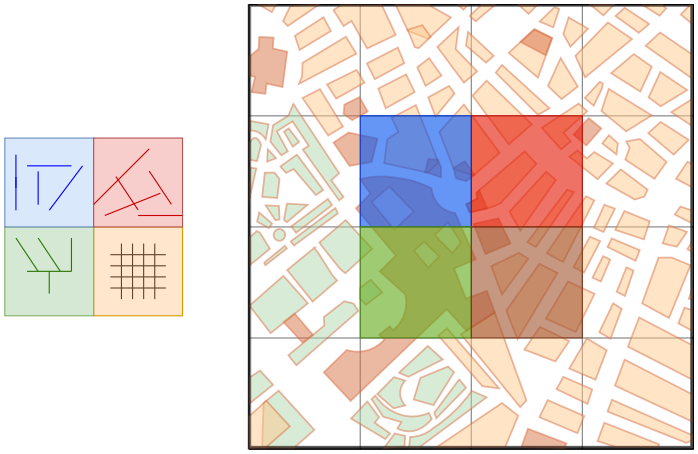


Figure 6.6: Visual representation of the segmentation optimization.

The number of these segments can be variable and changed from run to run. The only limitation is that both datasets need to use the same ones. More, smaller segments would mean more overhead while segment generating and assigning, but fewer comparisons per one street. The borderline false negative rate would also increase slightly. After this one-time setup, the algorithm can be run per segment. These can also be run independently and the results concatenated in the future, meaning that a user can get the values only for a smaller area he or she is interested in.

6.4 Proposed Data Model for Brno

With these two algorithms implemented, it's possible to integrate all datasets from the Brno cycling traffic case study and create a single model with information about the number of cyclists collected by different means. When designing an ETL pipeline to combine multiple datasets, it's important to note how updates will be handled. For example, the data from automatic counters are updated every hour, so the final analytical warehouse should also acquire these new values often. The end-user might want to look at the most recent numbers. Data gathered from the Brno Data Portal webpage[1] are already stored in a cloud storage solution under the hood of ArcGIS. It uses a standard relational database model,

[1] https://data.brno.cz/

with additional extensions for geospatial processing. These are publicly available and can be queried on demand. Thus, the updating phase is being handled on their side of things. This enables the final model to use indirect mapping by IDs to these respective datasets. It's possible because the network of spatial objects doesn't change much often in the datasets, just new numbers of cyclists are appended with newer timestamps. The proposed data model architecture of location mapping among datasets can be seen in Figure 6.7. OpenStreetMap basemap network of streets was chosen as the main source of spatial information. Of all the basemap networks used it's the most robust and reliable one, plus the largest dataset—Strava, uses it by default. It holds valuable information about streets and roads like the type of surface, number of lanes, or vehicle suitability. These will be extended by columns with IDs mapping the original streets to foreign networks.

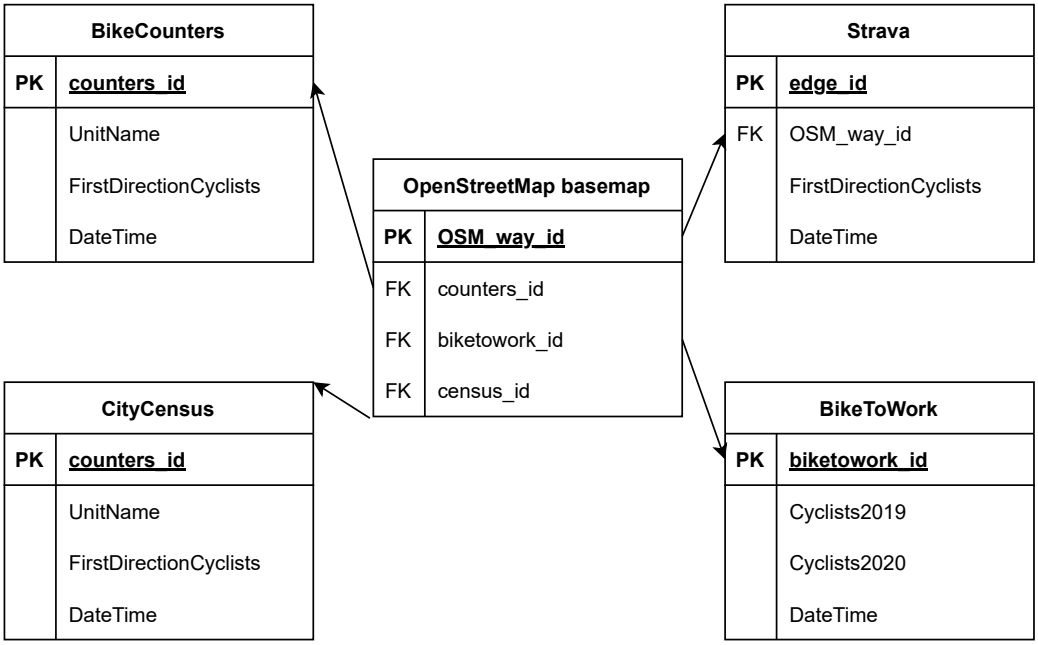


Figure 6.7: Proposed data model for the integration of datasets. Some attributes are omitted for simplicity.

The nationwide road census dataset was excluded, due to missing spatial information like GPS coordinates. Integrating it would require manual street searching and assignment. Datasets from bike counters, city census, and the bike-to-work campaign are all hosted publicly by the city, however, the Strava data are locked away under a user account and there's no public API to access them. This means that querying them ad-hoc isn't possible and the dataset needs to be saved in a local storage and accessed that way. Also, these data can't be shared with a third party outside of the Strava Metro portal, therefore the final application must not be available to the public.

6.5 Updating to a New Street Network

Thanks to the proposed model (6.7), the end datasets don't need to be updated as they are queried from the source. However, the centerpiece table mapping their basemap networks needs to be updated when they change. This includes additions of new streets and new

points (e.g. bike counters), reworks of existing objects (e.g. road maintenance that alters its definition), and others. In order to eliminate the need for a full model rebuild from scratch, another method should be implemented that finds differences between the old and the new network and extends the existing model with new values. It should take the already existing model and a new version of a single dataset and output the model with updated values.

6.6 Dashboard Mockup

Visualizing the model generated by previous algorithms can be done by creating a dashboard 6.8. The map should be interactive and selecting a street should update the graphs and tables on the right-side panel, to display data related to it.

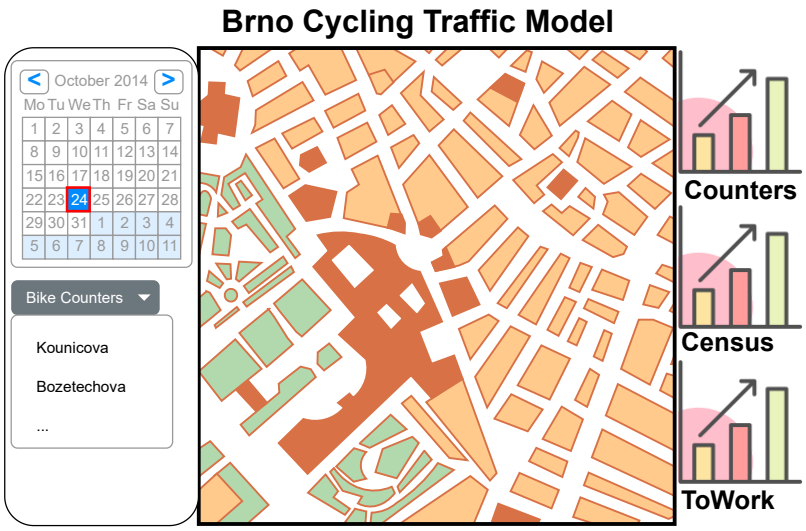


Figure 6.8: Wireframe of the final dashboard visualizing the integration model.

Some of the datasets have a timestamp column, determining the time interval of the recorded data. Choosing a specific date or a range of days in the left-side calendar should also filter and aggregate values in the charts. Additionally, it would be beneficial to include a dropdown menu of the bike counters, as there are just a few of them in the area and they are sparsely distributed, therefore they might be challenging to locate. Selection should move the map view to its spot. This dashboard will work with the publicly available datasets hosted in the ArcGIS workspace, so it can be fully automated and published to the general population. However, to include the Strava dataset, a second dashboard private to just the city employees and people with Strava Metro access would have to be created. Additional views can be added, like heatmaps showing cyclist numbers trends or comparisons between different datasets.

Chapter 7

Implementation Details

The implementation of the data processing and algorithmic part was done using the Python programming language. It has a rich domain of data science libraries and frameworks, as well as good integration of geospatial information and processing of geometries and coordinates. Geopandas[1] library was used in this thesis, with its underlying parts like NumPy for mathematical operations and Pandas for data handling. Jupyter Notebooks[2] were utilized for preliminary experimentation and prototyping. All the important functions that could be used in a future expansion or other development are commented by the PEP8 standard and take advantage of the Python type-hinting mechanism to provide a better experience for newly adopting developers.

7.1 Dataset Exploration

The first step to implement was the dataset exploration. All datasets were downloaded and loaded into Jupyter Notebooks to inspect them and learn about included attributes, their semantics, rules, and common values. These were mostly created and used during the dataset analysis process broken down in Chapter 5. One Notebook explores one dataset and all of them can be found in the `src/dataset_exploration/` directory of the project.

The individual steps of different explorations are usually similar. First, the dataset is loaded into a Geopandas Dataframe (internal data structure representation in memory) and then, some data cleaning occurs like type casting or formatting. Afterward, a few rows are displayed to inspect their format and values. It's good practice to inspect the number of columns and rows to gain a sense of the robustness of the dataset, as some might be too large to process at once due to the limitations of the workstation and its memory capacity. Pandas also provides a useful method called `describe()` that shows common statistics of individual columns, like minimal and maximal values or mean. This is useful for improving domain knowledge and its limits. At last, a correlation matrix (Figure 7.1) of the most important attributes was generated, just as an interesting analysis of possible relationships among the data. These Notebooks also include a text field explaining how are spatial objects identified and denoted in the dataset, and which columns together form a sort of primary key. A unique identificator for the bike counters dataset is a combination of the `LocationId`, `datum(date)`, and the `FirstDirection_Name` or `SecondDirection_Name` values.

[1] https://geopandas.org
[2] https://jupyter.org/

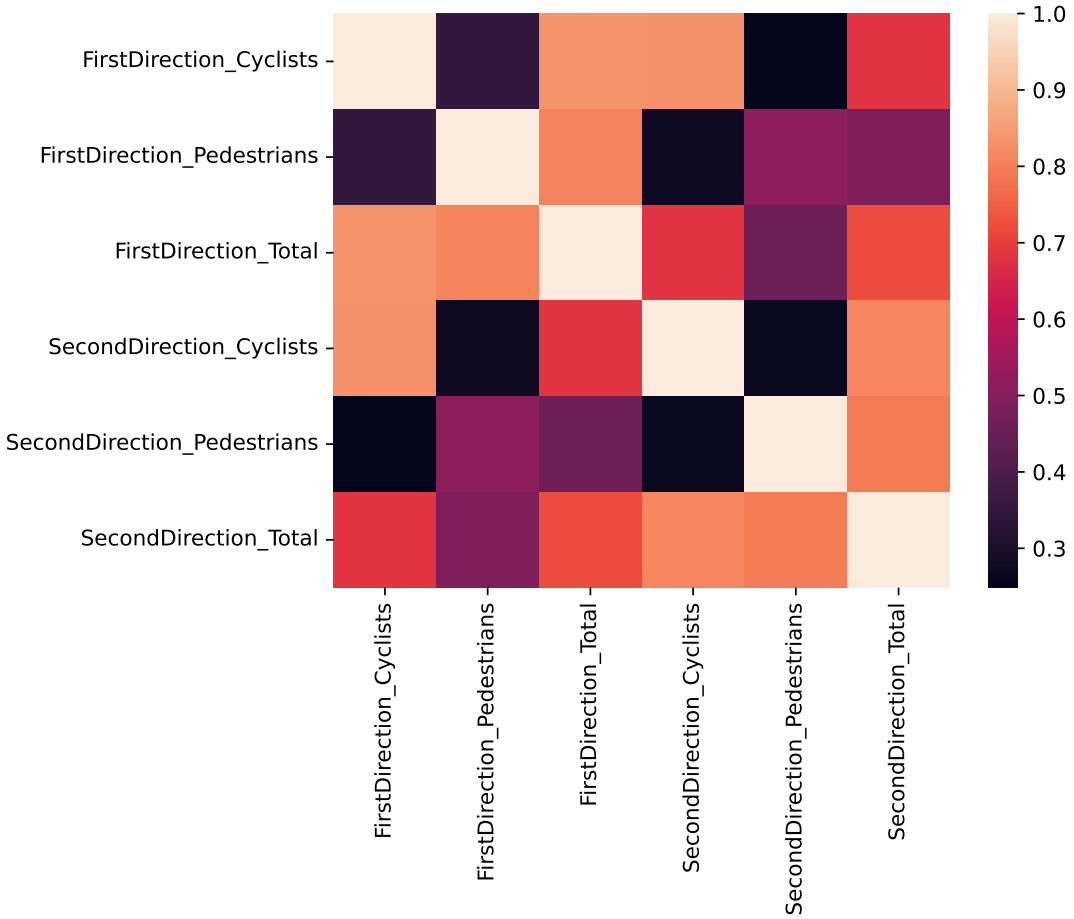


Figure 7.1: Correlation matrix of the bike counters dataset attributes. Data seem to be fairly uncorrelated, besides the link between the numbers of cyclists in one direction and the other. This could imply either that most cycleways are bidirectional, or that people choose bike-friendly streets over a variety of routes. Note that these are just speculations.

7.2 Geometry Utilities

All methods used for geometry handling are encapsulated in the `src/geometry_utils.py` source file. These are mainly implemented leveraging the Shapely[3] package, designed for manipulation of geometric objects in the cartesian plane [19].

This whole section uses the term **polyline** or **multiline**, let it be defined as an ordered collection of simple lines, each defined by two points, where the beginning is the same as the ending of the previous line, with the exception of the first one. The terms **buffer**, **bounding box**, **bounds** can be assumed to mean the same thing; the smallest possible square envelope of the object (e.g. line).

[3] https://shapely.readthedocs.io/

7.2.1 Angles Evaluation

The first method is a computation of an angle between two polylines. Here's the step-by-step breakdown of this approach:

1. Transform multilines into their bounding boxes.
2. Take diagonals of these bounds and convert them into mathematical vectors.
3. Normalize those vectors into unit ones (length of 1).
4. Calculate dot ($a \cdot b = \sum_{i=1}^{n} = a_i b_i$) product between them.
5. Clip the resulting value to the $< -1, 1 >$ interval.
6. Compute `arccos` from the previous number and the output is an angle in radians.
7. These can be optionally transformed to degrees.

All these mathematical operators are calculated using their numpy equivalents, and the solution is based on a post [38] from the StackOverflow forum. The final method takes two instances of either Shapely `MultiLineString` or `LineString` and outputs a single floating point number.

7.2.2 First Line Matching Algorithm

This section discusses the first prototype implementation of the line similarity detection algorithm. It proved to be inefficient and crashed against the limitations of the Shapely library. The method called `lines_overlap()` takes two line arguments, again, either as shapely `MultiLineString` or `LineString` and returns a boolean whether the lines match or not.

The Shapely API doesn't implement a method for calculating lines overlap percentage, instead only a simple boolean condition of overlap is present. However, it's still not available for the `MultiLineString` object, but only for the regular `LineString` one. This means that each underlying line of the polyline collection must be compared for overlap with each line from the other collection. This is implemented using the private helper method called `_lines_overlap()`. Besides the high inefficiency caused by the N-to-N comparisons, it's definitely not very accurate, as slightly different segmentation of the polyline into lines means full overlap will never happen, because the lines must match in the endpoints exactly to equal.

The method does a simple pre-processing at the beginning. It rounds up all coordinates of the input lines to a specific number of decimal points, based on the `round_digits` parameter. Default value specified by the global variable `NDIGITS` is 5, which evaluates as a roundup of 2 meters in real distance. If any combination of two line segments overlaps perfectly, it's assumed that the full polylines are the same, and the match was found.

7.3 Bounding-box Approach to Line Matching

The second attempt at line matching algorithm was designed with findings from the first prototype. Counting an exact overlap between two lines is not the best option, because even a small deviation disrupts the computation. This new and improved algorithm takes

advantage of all three principles introduced in Chapter 6, which are offset tolerance by coordinate rounding, maximal angle limitation, and the best score metric. The last one was missing completely in the line overlapping algorithm from the previous version. Besides that, the first two principles were implemented using a hard cut-off, meaning that no line outside of the accepted scope would ever be selected. A better approach would be to allow bigger and bigger offsets in each iteration if the previous one didn't find anything.

The method `match_lines_by_bbox_overlap()` takes one Shapely `MultiLineString` argument as the input line against which the match will be searched. The second argument is an instance of Geopandas `GeoSeries`, a collection of lines from the other dataset in which the search will happen. Passing all lines to the algorithm allows more iterations of the line set and finding multiple candidates. It returns one Shapely `MultiLineString` from the set as the best match, or None if no candidates were found. A simplified pseudocode of the final algorithm is shown in Listing 7.1.

```
while not found_match:
    max_angle = DEFAULT_MAX_ANGLE + STEP
    round_digits = DEFAULT_ROUND_DIGITS
    if max_angle >= 45: # nothing was found -> bigger allowed offset
            round_digits -= 1

    for~other_line in~other_lines:
        angle = calculate_angle(line, other_line)
        round(line, round_digits)
        round(other_line, round_digits)
        bbox_overlap = overlap_percentage(line.bounds, other_line.bounds)
        if (angle < max_angle)~and~(bbox_overlap > found_match.overlap):
            found_match = other_line
    # closing condition
    if (max_accepted_angle >= 45)~and~(not found_match):
            return None
return found_match
```

Listing 7.1: Pseudocode of the line matching algorithm based on a bounding-box overlap.

It starts with a maximal accepted angle of 15 degrees and rounds up all line coordinates to 5 decimal places (these default values can be easily changed as read-only global variables). The `found_match` variable stores the current best candidate. In each iteration, all lines from the set are traversed. If the angle between the original and the challenger lines is lower than the limit, and the overlap is higher than the current best, the line under examination gains the position of a leading candidate. After the whole set is processed, if any match was found, the algorithm exits and returns it. If not, the maximal accepted angle is raised by a fixed step, which is 5 degrees by default. Then the algorithm repeats the same cycle again. When the maximally allowed angle reaches 45 degrees, it's assumed that no quality matches will be found beyond that point. Therefore, the algorithm updates the number of decimal places to round down to 4 and runs one last iteration. This allows a bigger offset and tries to find slightly worse matches from the nearby area.

All computationally heavy operations in the algorithm like angle evaluations and bounding box overlap ratios are done using the numpy library, which is optimized for high performance. Thanks to that, this version of the algorithm is much faster than the first prototype,

even though it runs multiple cycles over the same set. The bounding box overlap ratio formula can be seen in Equation 7.1.

$$\text{overlap} = \frac{\text{buffer1} \bigcap \text{buffer2}}{\text{buffer1} \bigcup \text{buffer2}} \tag{7.1}$$

7.4 Segmentation Tools

The window segmentation principle introduced in the previous chapter is implemented in the `src/segmentation_utils.py` source code. Generation of the segments and their coordinates is the first step. The method takes the bounds of the desired window of a map and the number of segments (denoted **N**) to be generated in one dimension. Therefore, the resulting number of segments is actually N^2. The width of a single segment is calculated as `window_width / num_of_segments` and identically for height. After that, the min coordinates of segments in the x and y dimensions are generated by cumulative addition of the segment width and height respectively. From these two collections of coordinates, pairs of left-right and top-bottom borders are built by a sliding window technique, grouping two values at a time and moving by one value to the next. After reordering these pairs accordingly, the output is a list of tuples, each tuple in the format of `[min_x, min_y, max_x, max_y]` representing one segment of the map. This collection is then returned and serves as a read-only structure for accessing segments by their IDs for the rest of the application.

There are multiple options on how to define a segment membership of a line. Full encapsulation isn't a good metric, as many lines will cross the segment borders and therefore wouldn't be assigned to either one. The simplest and most universal alternative is appointing based on a single point of the line borders, for example, the bottom-left end of the bounds. All the lines will be distributed among segments this way, however, it's possible that two lines that should be matched can end up in different segments if their bottom-left points are slightly distant and they get split by the segment border. Another option is to assign the line to multiple segments and increase the likelihood of ideal matches being in the same one, for the cost of more comparisons and overhead. The method `is_in_segment()` is implemented using the "bottom-left" approach, so it checks if the min_x and min_y values of the street borders are in between the respective min and max coordinates of the segment.

With generated segments and a way to decide in which an object lies, it's possible to annotate any dataset with segment information. The method `assign_segments_to_dataset()` takes a GeoDataFrame object and the collection of segments generated in the first step. Objects in the geometry column (usually polylines, however, it works for other types too) are iterated over and each is tested for a segment membership. A dictionary mapping row IDs of the original dataset with IDs of segments in the collection is created and afterward merged into a copy of the dataframe, which is then returned from the function. The result is the input dataset with a new column of segment IDs for each row. After annoying both datasets about to be integrated, by a simple filter, lines can be compared with only those in the same section.

7.5 Location Matching

The location-matching library encapsulates the implemented algorithms. First, the OpenStreetMap dataset is loaded as the source of the truth. It uses its custom `Protocolbuffer`

`Binary Format` file format that's optimized for the full planet coverage that OpenStreetMap has to support [29]. It's read using OSM's own Python library called Pyrosm [34]. The `load_osm_basemap()` function takes a path to the OSM file and optionally a bounding box coordinates of the desired section of the map in the final model. The default value is a compact area around the Brno municipality. Basemap is loaded into a GeoPandas DataFrame object and stored in memory for further processing.

The rest of the package is structured as two pairs of methods used to match and update either point-based geometries or polyline-based ones.

7.5.1 Matching Points to a Basemap

Starting with the point matching, the method `match_points_to_osm()` takes the previously built basemap Dataframe and a path to any dataset file compatible with GeoPandas. Additionally, an exact name of the attribute with unique IDs needs to be specified, to determine how it will be mapped, as well as how the column should be named in the resulting dataset (rename might be desirable for clarity or in case of clashing ID column names among different datasets). Absolute distances to basemap streets are then calculated for each unique point in the dataset and a simple map of a *point — closest street* is created. A new column named by the method param is appended to the basemap with unmatched streets evaluating to None.

The updating method takes identical parameters, finds points from the new version of the dataset that are not already present in the model and applies the same logic as before, calculating distances and rewriting those new points into the model. Note that both the IDs' names arguments must be the same as used previously as they are used to address objects in both datasets.

7.5.2 Matching Streets to a Basemap

Matching street networks requires a generated collection of the map segments and annotated basemap by them. Besides the segment array, the `match_street_network_to_osm()` method follows a similar signature as the point-matching one from the previous section. After some preliminary processing like dropping duplicate values, renaming ID columns by a specified parameter, and others, segments are assigned to the input datasets that are about to be matched. Then, both the basemap and the processed dataset are filtered by the last method parameter `segment_ids`, which enables assigning only a subset of the map segments and therefore is useful for prototyping. The rest of the algorithm is functionally identical to the code in Listing 7.2.

```
final_model = GeoDataFrame()
for segment in~segments:
    basemap_segment = basemap[segment]
    foreign_segment = foreign_dataset[segment]
    matched_lines = []
    for basemap_line in~basemap_segment[geometry]:
        new_match = match_lines(basemap_line, foreign_segment)
        matched_lines.append(new_match)
    basemap_segment[foreign_id_column_name] = matched_lines
    final_model = final_model.append(basemap_segments)
```

Listing 7.2: Pseudocode of the line matching process done per segment.

Equally, updating a street network follows the same logic as with points. Streets that were not present in the current version of the model are parsed and attempted to append to the same basemap. The update method, however, doesn't allow specifying a subset of segments to be processed. This is caused by the fact that the set of newly added streets is mostly shallowly sparse, so the segmentation process would add unnecessary overhead. Also, it's not possible to assign the matched streets in bulk as before, because they are not processed in the same order as they are stored in the model and must be written into a specific row of the DataFrame, which is a slow and expensive operation.

7.6 Visualization

For the visualization of the integration model, the ArcGIS platform was chosen due to the fact that the city Data Department already uses it, so it's easier to incorporate it into their existing solutions. ArcGIS uses objects called **Feature layers** to represent spatial datasets that group geographical features like points, lines, and polygons [3]. These are typically built from common file types like geojson or shapely. They are stored in an integrated ArcGIS storage that uses standard SQL query language with additional extensions for spatial data. They can be easily queried through an API, visualized on basemaps, or integrated into all kinds of applications.

The first attempt at visualization implementation was using the basic map viewer of the platform. It offers an option to put multiple Feature layers onto one map and interact with them, by clicking on a certain object that triggers a pop-up menu with other attributes and values of that entity. However, the integration model built in previous work maps the end datasets by indirect encoding, and therefore a custom pop-up needs to be created that queries the cycling values by object IDs from the model. ArcGIS developed a custom scripting language called Arcade [1] specific for this purpose. It provides easy querying for any feature layers imported into the map and the dynamic pop-up that displays values regarding the selected street from all datasets can be seen in Figure 7.2.

This approach proved to be undesirable as the Arcade language quickly hit its limitations. It wasn't designed to process and aggregate high volumes of data and the temporal processing that's required for datasets like the bike counters, which have an hourly granularity and need to be digested for better understanding.

7.6.1 Dashboard

The next approach used the dashboard builder from the ArcGIS suite of applications [2]. It loads the same Feature layers as the map viewer before, but the default pop-ups are disabled altogether, and instead, custom elements like tables (Figure 7.3) are present and rendered out according to the selected streets on the map component. If a single street is selected, the tables can be transformed into charts (Figure 7.4) for a better visual representation of relationships in the data.

The bike sensors dataset is more tricky, as the values have an **hourly** granularity, therefore a desired level of aggregation must be specified by the user. For this specific use case, the left side of the dashboard provides a date picker element. After picking a street with aligning automatic sensor and selecting an interval of one or more days, the counters table is rendered with corresponding data. The number of cyclists is displayed as a sum during the time frame. This dataset differentiates between the directions of the cyclists, so these are displayed separately. Identically, two charts (Figure 7.5) are generated, each

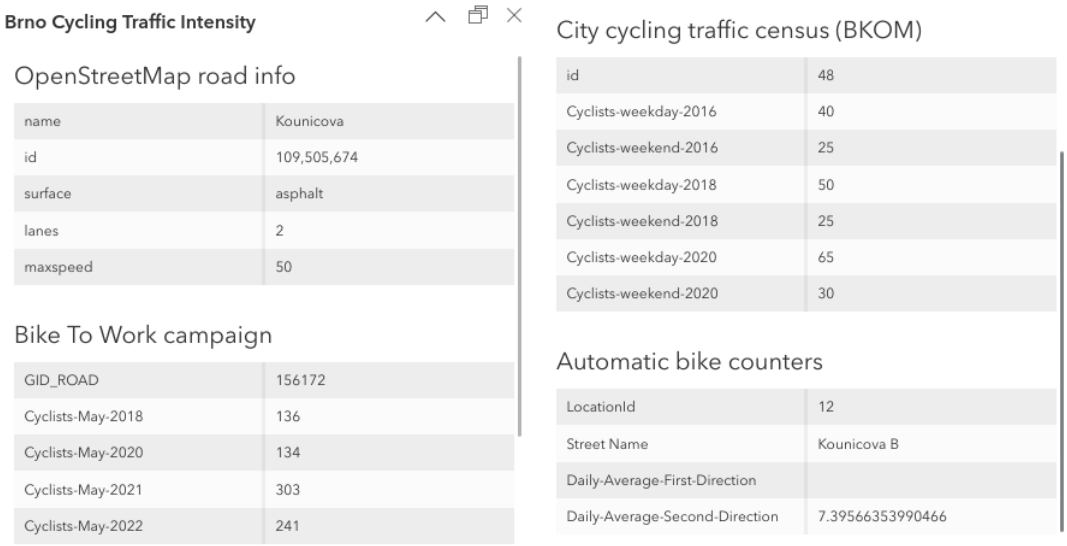


Figure 7.2: Custom pop-up for the integration model in ArcGIS map viewer built using the Arcade language. The code is saved in the `src/arcgis_map_popups.js` source file for future reference.

in one direction. The left sidebar also holds a counter picker, that moves the map viewer to the selected bike sensor.

All these elements are created using the ArcGIS Dashboard Builder by filtering the values passed from the Feature layers, as well as hooks updating them dynamically. Visual design follows the red and white theme of the Brno Data Department. English as the main language of the UI was chosen for the added inclusivity towards people of different nationalities living in Brno. The dashboard enables a user to change the basemap drawn in the map element, as well as an option to display the original datasets with their geometries, which might be useful in cases where the matching algorithm fails. The selection of streets is done by the top-left button on the map viewer element. It's possible to choose a single point or a rectangle that gathers all streets partially covered by it.

7.6.2 Strava Parser

The Strava data couldn't be uploaded to the ArcGIS storage and used in the aforementioned dashboard due to the licensing requirements. Therefore, a simple tool for parsing them was implemented in the `generate_strava_report()` method of the `dataset_query.py` file. Necessary inputs are:

- CSV file exported from the Strava Metro page
- ID of the OpenStreetMap way (displayed in the bottom section of the ArcGIS dashboard)
- start and end date of the analyzed interval (same as the date-picker element in the dashboard [YYYY-MM-DD])

The file export must be on a **daily** basis (selectable in the export section) and must cover the requested time interval. The output `html` file follows a similar format as the dashboard.

BikeToWork Table

GID_ROAD	Cyclists2018	Cyclists2019	Cyclists2020	Cyclists2021	Cyclists2022
149401.000000	87	142	85	214	276
156172.000000	136	222	134	303	241
221039.000000	338	553	333	634	682
221040.000000	89	145	87	263	318

Number of cyclists in **May** of the respective year, collected in the "Bike to work" campaign.

Table Chart

City Census Table

id	week2018	weekend2018	week2020	weekend2020	week2022	weekend2022
1	40	0	50	0	50	0
16	25	0	20	0	20	0
21	20	0	15	0	15	0
48	50	25	65	30	65	30

Number of cyclists in one unspecified **day**, counted by the Brno Communications company.

Table Chart

Figure 7.3: Tables displaying BikeToWork and city census data for 4 selected streets.

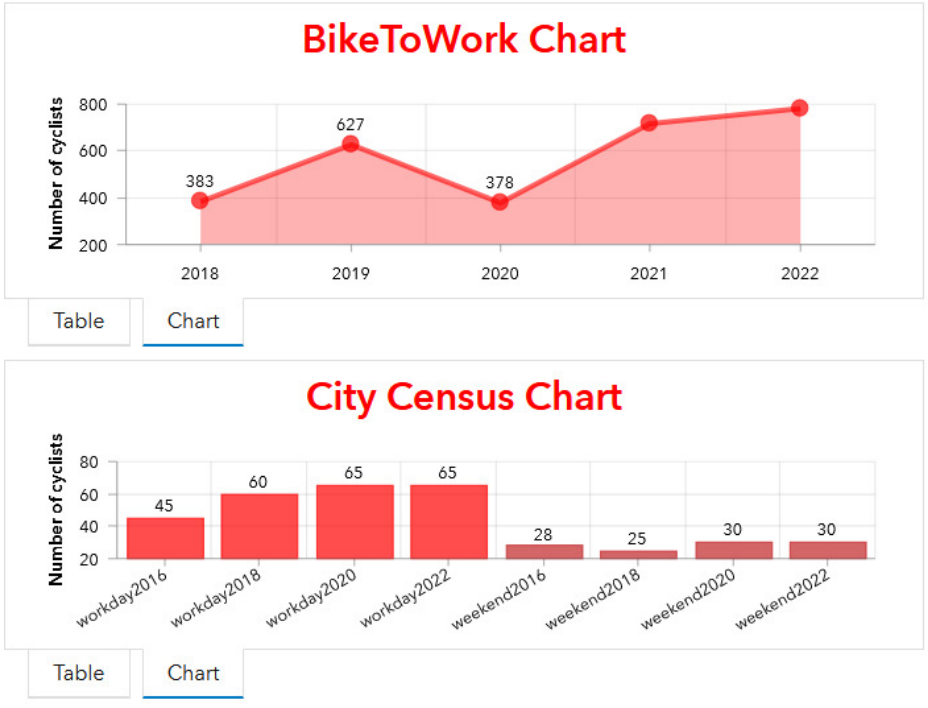


Figure 7.4: Charts illustrating the data for a single selected street. These views are valuable for understanding long-term trends in cycling traffic.

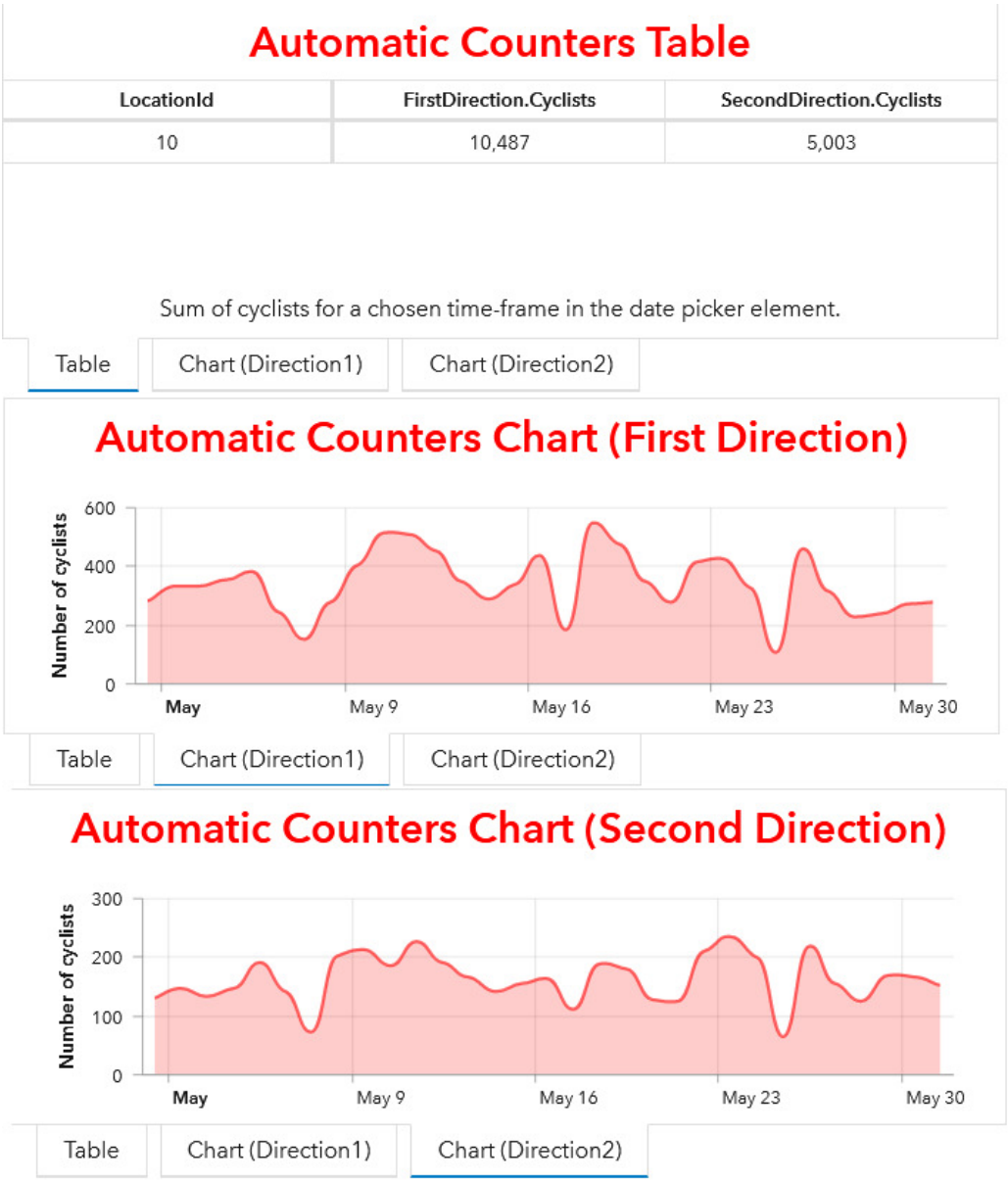


Figure 7.5: A table and two charts from the dashboard displaying numbers of cyclists detected on Královopolská street in the May of 2022.

It provides a table with the absolute values of cyclists and a simple line chart showing daily trends.

Chapter 8

Experiments

In this chapter, experiments are conducted to evaluate the implemented algorithm. After that, a few visualizations are shown and analyzed to find insights about the final model, as well as the end datasets and their values.

8.1 Algorithm Evaluation

The only way to quantify the accuracy and performance of the algorithm is to have annotated data. However, these do not exist, therefore a manual annotation was required to create a reproducible way of evaluating different algorithm versions during the prototyping and development phase. This is a very time-consuming task, as for each street there might be multiple correct matches from other basemaps. Also, annotating the whole OpenStreetMap dataset of $\simeq 65,000$ streets is basically impossible, therefore a small sample of 21 data points was chosen and processed.

These were picked to cover as much variety as possible. Streets from the dense city center with multiple parallel lines are represented, as well as country roads on the outskirts of Brno. Each one is located in a different part of the city with varying lengths, angles, trajectories, and types of intersections in close proximity. They were rendered over the other datasets on the same basemap and visually inspected to choose the best corresponding counterparts. Some OSM streets represent multiple inter-connected sections of a street in the BikeToWork dataset therefore all of these had to be assigned, as the algorithm can pick any one of them. A small sample dataset mapping IDs from different basemaps shown in Table 8.1 was created.

OSM	BikeToWork	Census
5606150	221028, 221027	12
31642926	202970	224
113965475	243069, 221076, 243079, 243080	10
...	...	...
25769383	224883	107

Table 8.1: Sample rows from the algorithm evaluation dataset.

After the evaluation dataset was finished, a simple script was made that parses it, as well as the final model built by the algorithm. It's located in the `src/experiments.py` file and the `eval_street_algorithm()` function. For each street, it takes these two val-

ues and compares them. Final accuracy percentages are generated independently for the BikeToWork and Census datasets and are as follows:

- BikeToWork matches: 16/21 [76.19 %]
- Census matches: 17/21 [80.95 %]

Some of the incorrectly evaluated streets were inspected to gain better knowledge about which cases are problematic for the algorithm. Three found cases are displayed in Figure 8.1.

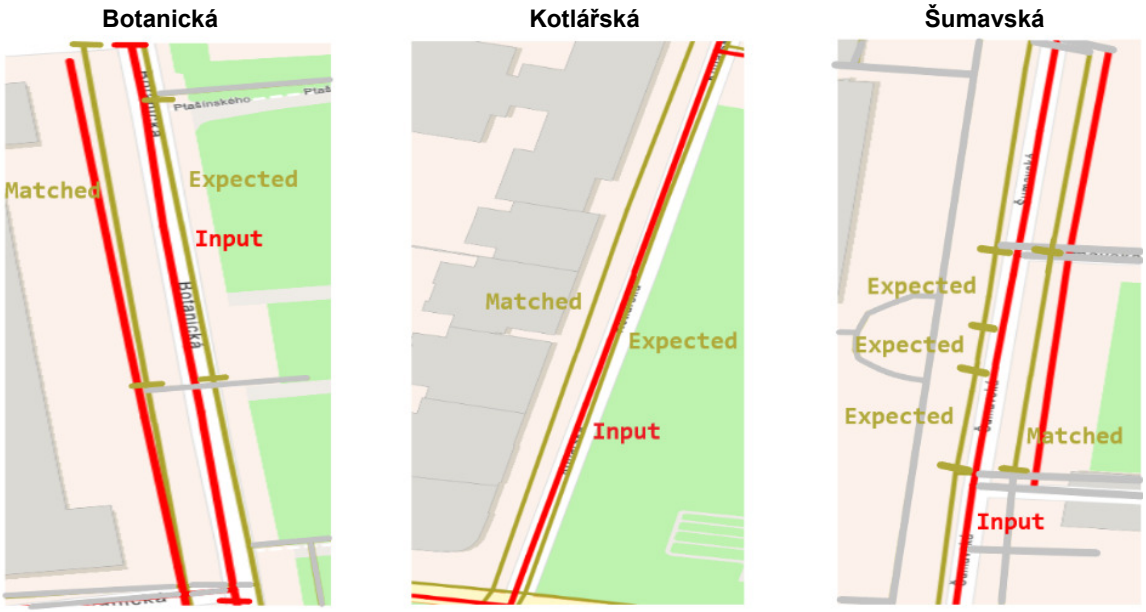


Figure 8.1: Three cases where the street matching algorithm failed. As can be seen, all of them stem from the fact that datasets also contain sidewalks that run in close proximity and parallel to the inspected street. It's best demonstrated at Šumavská Street, where the road is actually segmented into smaller pieces, so each one has only a small partial overlap with the OSM street in question. However, the sidewalk is defined as a long single line, therefore the overlap is much higher even though it's slightly distant from the original street.

8.2 Intensity Heatmaps

The final dashboard introduced earlier also provides a number of heatmaps, visualizing the cycling trends around the city. This analysis takes a look at two of them and compares their values to determine if there are any common similarities. The first one (Figure 8.2) shows the absolute number of cyclists in the BikeToWork dataset for May of 2022 as a direct proportion of the line width. Simply, the wider the line on the map, the more cyclists were recorded. Identically, the second Figure 8.3 shows numbers of cyclists in the City Census dataset, registered in one workday of 2022. Both focus on commuting to work, therefore common trends should correspond to one another.

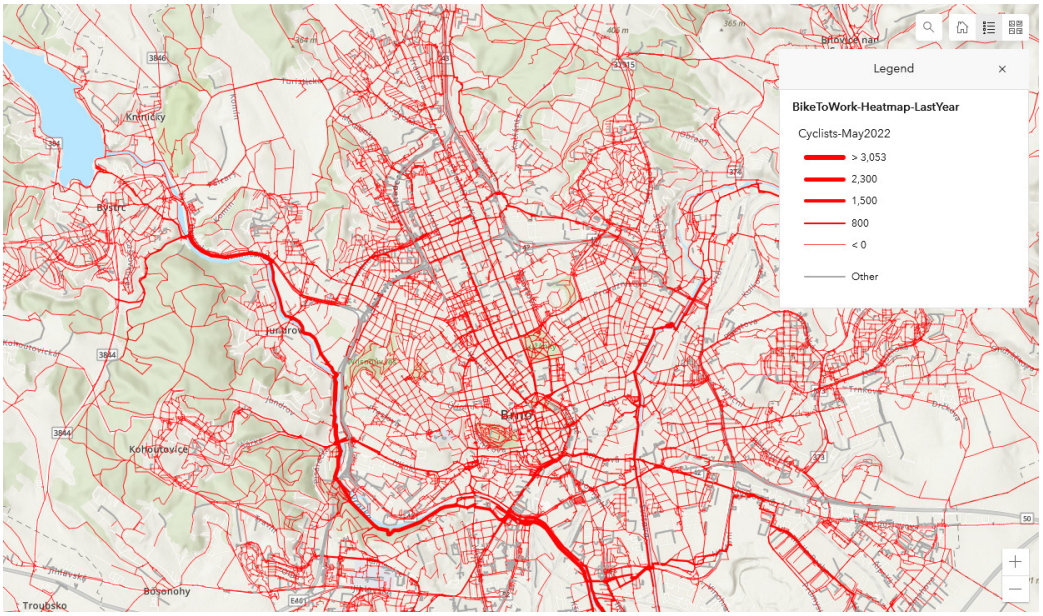


Figure 8.2: Heatmap visualizing the number of cyclists from the BikeToWork dataset in May of 2022.

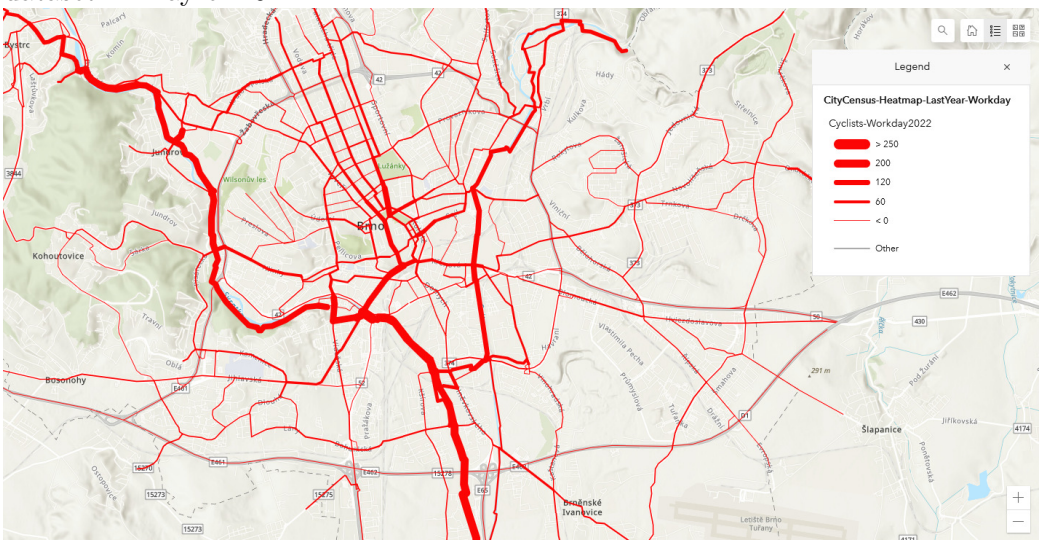


Figure 8.3: Heatmap visualizing the number of cyclists from the City Census dataset in one workday of 2022.

The highest intensity can be seen around the river Svratka, that's present in both cases. However, the Census data show significant numbers around the river Svitava on the right side of the city as well. In the case of the town's center, both datasets record big density around the main train station, but contrast can be seen in the Census dataset where a high intensity happened through the middle of the old city center, whereas BikeToWork doesn't show this occurrence.

There are definitely similarities, but the city census dataset has very limited both spatial ($\simeq$ 620 streets) and temporal (one day a year) coverage, therefore the values might not correspond to the real trends happening in the city.

8.3 Comparison Heatmap

This section focuses on an examination of cyclists' behavior differences during a workday and the weekend. The City Census dataset collects data each year for one unspecified workday and one weekend day. The heatmap in Figure 8.4 shows the comparison. It's clear that cyclists prefer the tight city center on regular days. They use it to commute to work, run errands, or just go shopping. However, during the weekends the majority can be found around the Svratka River and the Brno Dam. These parts of the city have many kilometers of cycling paths separated from the car infrastructure, which leisure cyclists prefer for safety and enjoyment reasons.

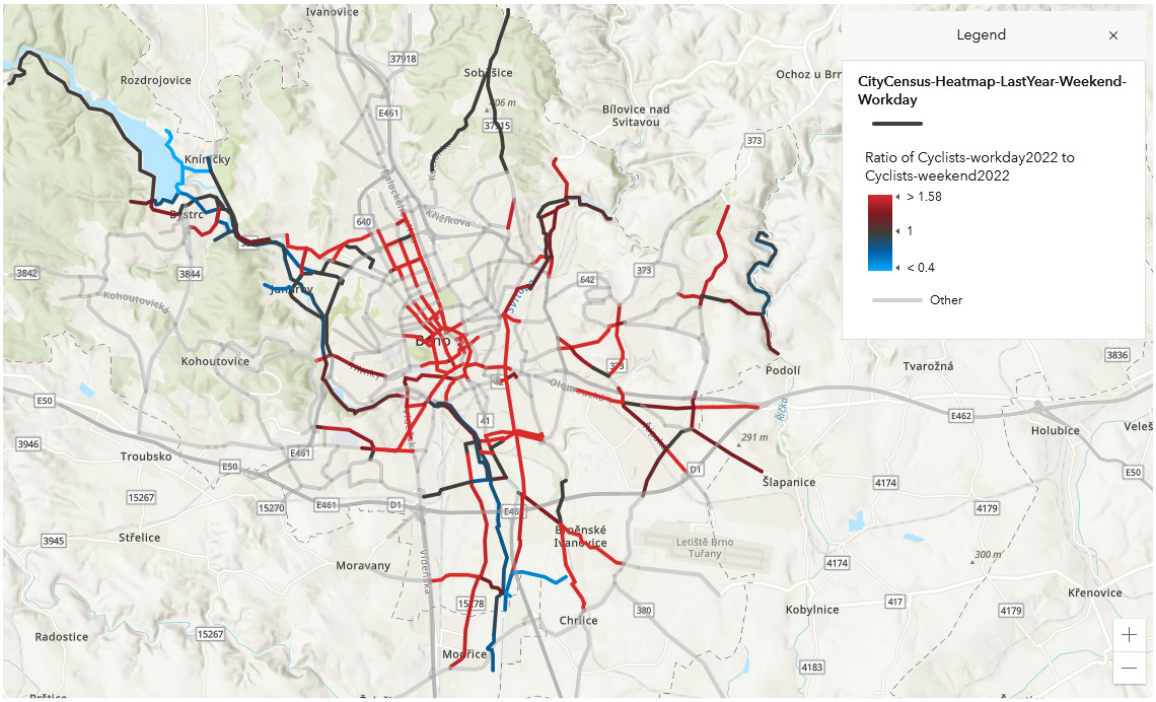


Figure 8.4: Comparison of cycling numbers during one workday and one weekend day in 2022 from the City Census datasets. The red indicator represents that workday numbers have the majority while blue is the other way around. The rest is defined as a spectrum between these two colors shown in the legend.

8.4 Cycling Numbers

The final analysis aims to utilize the side-by-side visualization of the different datasets and compare the numbers of cyclists recorded to find similarities among them. Eight streets that are present in all of them were picked, these were mostly limited by the bike counters dataset. Next, a time interval for the analysis was selected, filtering all the numbers for May of 2022, as it was the most recent month with data for all datasets during the work on this thesis. This was influenced primarily by the BikeToWork dataset which provides values aggregated only for this month of each year. Data from counters and Strava are generated on an hourly basis, therefore these could be transformed by a simple summation of this interval. Census data are provided for one workday and one weekend day. To represent the best estimation

of the whole month, the final number was calculated as $(23 * \text{workday\_cyclists}) + (8 * \text{weekend\_cyclists})$. These values correspond to the number of days in May of 2022. The final results can be seen in Table 8.2.

Location	Counters	BikeToWork	Census	Strava
Královopolská	15,000	900	1,500	1,900
Černovice	22,000	1,100	None	3,100
Ikea	60,000	2,300	None	8,300
Jundrov	50,000	2,300	5,500	6,600
Komín	46,000	1,800	700	6,200
Kounicova a	10,000	240	1,700	890
Nové sady B	12,000	80	3,800	1,900
Renneská 1	6,000	2,800	7,600	2,600

Table 8.2: Comparison of the absolute numbers of cyclists among datasets on the same streets.

It's apparent from Figure 8.5 that the numbers vary greatly. Bike counters that record each bicycle passing them without the need for cooperation show much higher values. These 8 comparisons don't indicate any connection between pairs of datasets. The exception happening between the counters and the Strava dataset, where 6 out of 8 streets have a ratio of cyclists around 7:1. The sample is too small to call this statistically significant, however, it indicates that the Strava dataset provides a permissible representation of the cycling traffic intensity. One of the streets that don't follow this relationship is Renneská, which shows high irregularity in the counters numbers, implying the unit could have been malfunctioning during this time.

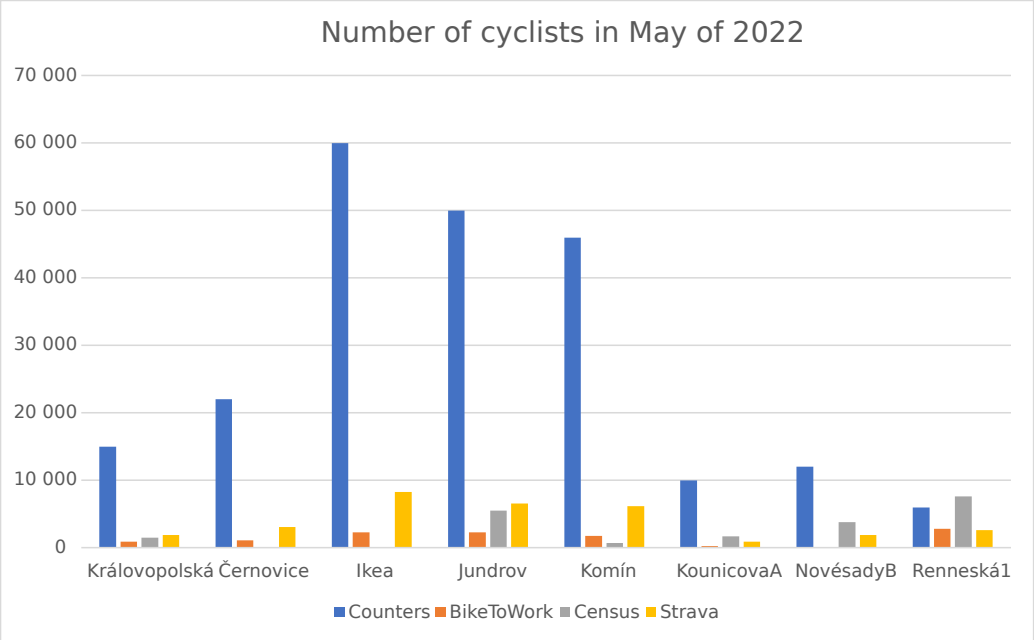


Figure 8.5: Visualized cycling numbers comparison among datasets.

Chapter 9

Conclusion

This thesis breaks down the topic of modeling cycling traffic intensity from multiple datasets. Theoretical principles relevant to the subject are explained, including the required geographical knowledge base, as well as ways of acquiring, storing, and managing spatial data. Next, a run-down of approaches to this problem by different cities from around the world is presented. Existing datasets regarding cycling intensity in Brno are analyzed. These include numbers from bike counters, a city-wide cycling census, or values from the "Bike to Work" campaign. Furthermore, the issue of basemap inconsistencies is defined.

The thesis introduced a new algorithmic approach to matching different street networks. It works by transforming a GPS-defined pseudo-continuous space to a discrete one, then estimating the angle between two candidate polylines, and finally by calculating the similarity score by the overlap percentage of bounding boxes of the streets. This correctly solves most cases of inconsistencies in the datasets, which estimated an accuracy of 80% in a small test sample of the data. The algorithm is optimized for large amounts of data by segmenting the space into smaller chunks and finding candidate street matches inside them. The approximate time for model building is around 3 hours, for the greater Brno area, consisting of roughly 65,000 streets. This generic algorithm can be used on any datasets with polyline-based geometries and therefore could be valuable in a multitude of other applications. The most notable limitation is definitely the score calculation, which returns poor results for perfectly horizontal or vertical lines due to the bounding box evaluation. This could be improved in future work by switching to a circle-like area, which might yield more accurate values, for the cost of higher computational requirements. A second algorithm for matching point-defined datasets to a street network was also implemented.

A data model of cycling traffic intensity specifically for Brno was designed, using indirect mapping to the end datasets by their respective identificators. This way the model doesn't need to handle updates that happen on the vendor's side. It could be expanded later with additional datasets like cycling accidents or data from bike-sharing companies, with the same set of algorithms. Finally, it's visualized in a public dashboard which digests values from all the datasets next to one another. The dashboard should be used by the Brno Transportation Department to improve decision-making regarding cycling infrastructure. It is affected mostly by the limited functionality provided by the ArcGIS dashboard platform. Switching to a more modern framework could be beneficial in the future.

This project was submitted to the EXCEL@FIT 2023 conference [17] at the Brno University of Technology, where it was awarded by an expert panel, as well as by industry partners for an original approach to data integration from different sources and the potential of practical usage of the solution.

Bibliography

[1] ArcGIS. *Arcade* [online]. 2023 [cit. 2023-04-27]. Available at: https://developers.arcgis.com/arcade/.

[2] ArcGIS. *ArcGIS Dashboards* [online]. 2023 [cit. 2023-04-27]. Available at: https://www.esri.com/en-us/arcgis/products/arcgis-dashboards/overview.

[3] ArcGIS. *Feature layers* [online]. 2023 [cit. 2023-04-27]. Available at: https://doc.arcgis.com/en/arcgis-online/reference/feature-layers.htm.

[4] Australia. *The Intergovernmental Committee on Surveying and Mapping* [online]. 2022 [cit. 2022-12-12]. Available at: https://www.icsm.gov.au/.

[5] Auto*Mat. *Do práce na kole* [online]. 2023 [cit. 2023-02-26]. Available at: https://dopracenakole.cz/data.

[6] AWS. *What is a data lake?* [online]. 2023 [cit. 2023-01-13]. Available at: https://aws.amazon.com/big-data/datalakes-and-analytics/what-is-a-data-lake/.

[7] Bartuska, L., Hanzl, J. and Lizbetinova, L. Possibilities of Using the Data for Planning the Cycling Infrastructure. *Procedia Engineering*. 2016, vol. 161, p. 282–289. DOI: https://doi.org/10.1016/j.proeng.2016.08.555. ISSN 1877-7058. Available at: https://www.sciencedirect.com/science/article/pii/S1877705816327631.

[8] Battersby, S. *Map Projections. The Geographic Information Science & Technology Body of Knowledge*. 2nd ed. 2017. ISBN 978-80-244-5827-4.

[9] Bonner, S., Kureshi, I., Brennan, J. and Theodoropoulos, G. *Exploring the Evolution of Big Data Technologies*. 1st ed. Boston: Morgan Kaufmann, 2017. ISBN 978-0-12-805467-3. Available at: https://www.sciencedirect.com/science/article/pii/B9780128054673000144.

[10] Chang, K.-T. *Introduction to Geographic Information Systems*. 3rdth ed. McGraw Hill, 2006. ISBN 0070658986.

[11] Cridland, J. *Bicycle counter in Copenhagen* [online]. 2009 [cit. 2023-01-23]. Available at: https://www.flickr.com/photos/jamescridland/3959398194/.

[12] CSGNetwork. *GPS Latitude and Longitude Distance Calculator* [online]. 2023 [cit. 2023-04-11]. Available at: http://www.csgnetwork.com/gpsdistcalc.html.

[13] Data Department, B. *Bike-traffic-intensity* [online]. 2023 [cit. 2023-02-26]. Available at: https://data.brno.cz/datasets/mestobrno::intenzita-dopravy-intenzita-cyklist%C5%AF-bike-traffic-intensity/about.

[14] El Esawey, M., Lim, C. and Sayed, T. Development of a cycling data model: City of Vancouver case study. *Canadian Journal of Civil Engineering*. 1st ed. 2015, vol. 42, no. 12, p. 1000–1010. DOI: 10.1139/cjce-2015-0065. Available at: https://doi.org/10.1139/cjce-2015-0065.

[15] El Sappagh, S. H. A., Hendawi, A. M. A. and El Bastawissy, A. H. A proposed model for data warehouse ETL processes. *Journal of King Saud University - Computer and Information Sciences*. 1st ed. 2011, vol. 23, no. 2, p. 91–104. DOI: https://doi.org/10.1016/j.jksuci.2011.05.005. ISSN 1319-1578. Available at: https://www.sciencedirect.com/science/article/pii/S131915781100019X.

[16] Esri. *ArcGIS developer documentation* [online]. 2022 [cit. 2022-12-29]. Available at: https://developers.arcgis.com/documentation/.

[17] FIT. *Student conference of innovation, technology and science in IT* [online]. 2023 [cit. 2023-05-13]. Available at: https://excel.fit.vutbr.cz/.

[18] Francke, A. and Lissner, S. *Big Data in Bicycle Traffic*. March 2018. DOI: 10.13140/RG.2.2.27176.62726.

[19] Gillies, S. *Shapely* [online]. 2013 [cit. 2023-04-18]. Available at: https://shapely.readthedocs.io/.

[20] GisGeography. *GIS Geography* [online]. 2022 [cit. 2022-12-12]. Available at: https://gisgeography.com/.

[21] Harder, C. and Brown, C. *The ArcGIS Book: 10 Big Ideas about Applying the Science of where*. 1st ed. Esri Press, 2017. Arcgis Books. ISBN 9781589484870.

[22] Heyman, D., Sheesley, B. and Woodruff, A. *Axis maps* [online]. 2020 [cit. 2022-12-12]. Available at: https://www.axismaps.com/.

[23] Huisman, O. and By, R. de. *Principles of Geographic Information Systems (GIS): an Introductory Textbook*. 4th ed. ITC Educational Textbook Series, 2009. ISBN 978-90-6164-269-5.

[24] IBM. *Snowflake schemas* [online]. 2021 [cit. 2023-01-13]. Available at: https://www.ibm.com/docs/en/db2/9.7?topic=sss-star-schemas.

[25] IBM. *Star schemas* [online]. 2021 [cit. 2023-01-13]. Available at: https://www.ibm.com/docs/en/db2/9.7?topic=sss-star-schemas.

[26] IBM. *What is ETL?* [online]. 2023 [cit. 2023-01-13]. Available at: https://www.ibm.com/topics/etl.

[27] Jeffrey, C. *An Introduction to GNSS: GPS, GLONASS, Galileo and Other Global Navigation Satellite Systems*. 1st ed. NovAtel, 2010. ISBN 9780981375403.

[28] OpenStreetMap. *About OpenStreetMap — OpenStreetMap Wiki*. 2022. [Online; accessed 27-December-2022]. Available at: https://wiki.openstreetmap.org/wiki/About_OpenStreetMap.

[29] OpenStreetMap. *PBF Format — OpenStreetMap Wiki,*. 2022. [Online; accessed 25-April-2023]. Available at: https://wiki.openstreetmap.org/w/index.php?title=PBF_Format&oldid=2266330.

[30] Romani, M. *GNSS constellations: differences, what have in common and vulnerabilities* [online]. 2019 [cit. 2022-12-27]. Available at: https://www.linkedin.com/pulse/gnss-constellations-differences-what-have-common-marco-romani.

[31] Song, I.-Y. *Data Warehouse*. Boston, MA: Springer US, 2009. 657–658 p. ISBN 978-0-387-39940-9. Available at: https://doi.org/10.1007/978-0-387-39940-9_882.

[32] Strava, I. *Strava Metro FAQ* [online]. 2023 [cit. 2023-02-25]. Available at: https://metro.strava.com/faq.

[33] Talend, I. *OLTP vs OLAP* [online]. 2023 [cit. 2023-01-12]. Available at: https://www.stitchdata.com/resources/oltp-vs-olap/.

[34] Tenkanen, H. *Pyrosm, OpenStreetMap PBF data parser for Python* [online]. 2023 [cit. 2023-04-25]. Available at: https://pyrosm.readthedocs.io.

[35] ThoughtSpot. *Star Schema vs Snowflake Schema: 6 Key differences* [online]. 2022 [cit. 2023-01-13]. Available at: https://www.thoughtspot.com/data-trends/data-modeling/star-schema-vs-snowflake-schema.

[36] Wallentin, G. and Loidl, M. Agent-based Bicycle Traffic Model for Salzburg City. *GI-Forum – Journal for Geographic Information Science*. 1st ed. july 2015, vol. 2015, no. 1, p. 558–566. DOI: 10.1553/giscience2015s558.

[37] Willberg, E., Tenkanen, H., Poom, A., Salonen, M. and Toivonen, T. Comparing spatial data sources for cycling studies: A review. *Transport in Human Scale Cities*. 1st ed. Edward Elgar Publishing. 2021, no. 1.

[38] Wolever, D. *Angles between two n-dimensional vectors in Python*. 2012 [cit. 2023-05-14]. Available at: https://stackoverflow.com/a/13849249.

[39] ŘSD. *Výsledky celostátního sčítání dopravy 2020* [online]. 2023 [cit. 2023-02-26]. Available at: https://www.rsd.cz/silnice-a-dalnice/scitani-dopravy#zalozka-celostatni-scitani-dopravy-2020.

Appendix A

Contents of the Included Storage Media

The included storage media holds these files:

- **README.md**: readme file explaining the installation steps and usage
- **src/**: directory with all the source files
- **thesis.pdf**: thesis paper for this project
- **datasets/**: directory with datasets used to build the model
- **requirements.txt**: list of Python packages required
- **venv/**: python virtual environment with packages required by the app
- **data_urls.conf**: configuration file defining API URLs to Brno datasets

Appendix B

Manual

The project can also be downloaded from the public Github repository[1]. Both the installation steps and the usage is defined in the readme file present in the repository and the storage media included.

B.1 Installation

The installation process follows the standard Python procedure. All the necessary packages are defined in the `requirements.txt` file. To build a closed, working environment, you need to:

- `python -m venv venv` (Create empty Python env in the root folder)
- `source venv/bin/activate` (Enable it)
- `python -m pip install -r requirements.txt` (Install required packages)

B.2 Usage

The first step is always to enable the Python virtual environment encapsulating all the required packages and libraries: `source venv/bin/activate`

All the methods for dataset matching are implemented in the `src/location_matching.py` source file, it can be imported to any Python application and used as a library. Specifically for the Brno list of datasets, the whole process is implemented as the main method of this script. The directory structure of the storage media should be set up correctly to start the model build by simply running: `python location_matching.py` with enabled virtual environment from the `src` directory.

Brno datasets can be downloaded from the ArcGIS-hosted storage using the `query_arcgis_layer()` method from the `src/dateset_query.py` source file (examples are in the main function, but the documentation explains all the required parameters).

[1] https://github.com/Radluy/Cycling-traffic-intensity-Brno