VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2021 Bc. Jaroslav Pizur

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KLASIFIKACE LINEK MHD Z GNSS DAT
PUBLIC TRANSPORTATION LINES CLASSIFICATION BY GNSS DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Jaroslav Pizur**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Jílek, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Jaroslav Pizur ***ID:*** 186164
Ročník: 2 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Klasifikace linek MHD z GNSS dat

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a realizovat algoritmus pro zpracování a klasifikaci GNSS dat zaznamenaných během jízdy vozidel hromadné dopravy ve městě. Vozidlo během absolvování své pravidelné linky zaznamenává svou aktuální polohu. Takto je realizován sběr dat na více linkách hromadné dopravy. Práce ve výsledku provádí klastrování jednotlivých průjezdů do tříd, které odpovídají konkrétní lince veřejné městské dopravy.

1. Seznamte se s formátem poskytnutých dat, proveďte rešerši dalších datových formátů vhodných pro práci s záznamy GNSS dat.
2. Proveďte obecnou rešerši metod pro reprezentaci a komparaci trajektorií.
3. Navrhněte a realizujte způsob předzpracování a datové reprezentace GNSS záznamů jízd vozidel hromadné dopravy.
4. Vytvořte datový model města, pro které bude probíhat zpracování dat. Totéž realizujte pro vybranou množinu zaznamenaných průjezdů městem.
5. Implementujte metodu pro zpracování množiny GNSS záznamů a jejich klastrování.
6. Podrobně otestujte výkon Vámi navrženého algoritmu. Diskutujte také případy, pro které Váš algoritmus selhává.

DOPORUČENÁ LITERATURA:

Everett, H.R., 1995. Sensors for mobile robots. CRC Press

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 17.5.2021

Vedoucí práce: Ing. Tomáš Jílek, Ph.D.
Konzultant: Ing. Adam Ligocki

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Práca sa venuje digitalizácii autobusovej dopravy. Na vstupe pracuje so sekvenciou GNSS dát, ktoré pretransformuje do formátu OpenStreetMap. Tým sa obohatí o jeho informácie a zvýši sa jej pozičná presnosť. Následne sa venuje spôsobom, ktorými možno získať autobusové linky z tejto všeobecnej postupnosti GNSS súradníc. Pod autobusovou linkou sa rozumie opakovaná trajektória, ktorá spĺňa kritériá vychádzajúce z jej očakávateľných, prípadne daných, vlastností. Navrhne sa niekoľko klastrovacích riešení a otestuje sa ich výkon. Na základe vykonaného testovania sa určí najvhodnejšie navrhnuté riešenie pre identifikáciu liniek. Výstupom práce je teda automatické namapovanie modelov autobusových liniek v rámci ľubovoľne veľkej oblasti do mapového podkladu s minimálnou nutnosťou manuálnych zásahov. Otvára dvere mnohým inteligentným procesom v reálnom čase, umožňuje tvoriť štatistiku nad infraštruktúrou a prehľadnejšie mestské plánovanie.

KĽÚČOVÉ SLOVÁ

OpenStreetMap, OSM, GNSS, geojson, GIS, Python, Folium

ABSTRACT

The subject of this thesis is digitalization of bus transportation. The input is represented by a sequence of GNSS data which are transformed to the OpenStreetMap format. Doing so, it is enriched by the information OpenStreetMap format provides and it gains its positional advantages as well. Then this thesis deals with ways by which one can detect bus lines from this general sequence of GNSS coordinates. A bus line is recognized as a repeating trajectory, which satisfies criteria derived from its expectable or defined characteristics. A few clustering solutions are proposed and tested for their performance. On the basis of this testing, there is one solution chosen as the best performing one, to be the proposed solution of this thesis. The overall output will therefore be formed by automatic mapping of bus lines with no theoretical area limit and with minimum manual intervention needed. It lays the foundations for various intelligent real-time processes to be implemented as well as allowing for infrastructure to be processed for the statistics purposes or urban planning.

KEYWORDS

OpenStreetMap, OSM, GNSS, geojson, GIS, Python, Folium

PIZUR, Jaroslav. *Klasifikace linek MHD z GNSS dat*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2021, 90 s. Diplomová práca. Vedúci práce: Ing. Tomáš Jílek, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Bc. Jaroslav Pizur
VUT ID autora:	186164
Typ práce:	Diplomová práca
Akademický rok:	2020/21
Téma závěrečnej práce:	Klasifikace linek MHD z GNSS dat

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Ďakujem vedúcemu diplomovej práce pánovi Ing. Tomášovi Jílkovi, Ph.D. za odborné vedenie, konzultácie a podnetné návrhy k práci v znamení aktívnej ochoty a priateľského prístupu.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Na úvod je vhodné čitateľovi vysvetliť kvôli čomu práca vznikla a aký problém v skutočnosti rieši, prípadne aký proces urýchľuje alebo vylepšuje. Návrh zadania pramení z myšlienok pre digitalizáciu dopravy, umožňuje ľahší monitoring infraštruktúr, ktorého model je snaha vytvoriť v tejto práci. Povaha témy vhodne zapadá aj do myšlienok inteligentného mesta (smart city), ktoré možno veľmi všeobecne vnímať ako myšlienky komplexnej virtualizácie a mapovania verejného priestranstva. Je žiadúce aby v digitálnom modele mesta bola možnosť vizualizácie verejnej dopravy v zmysle napríklad real-time pozície konkrétneho dopravného prostriedku. Tá by sa nevedela zaobísť bez modelu do ktorého patrí. Je možnosť so zmenovými požiadavkami robiť neprestajne nové modely manuálne alebo vytvoriť nástroj, ktorý tieto modely nájde a namapuje automaticky.

Ďalšou aplikáciou je vizualizácia dopravného meškania. Aktuálne je reprezentovaná informáciou o x-minútovom meškaní. Je ale reálne a uskutočniteľné na vstupných dátach a modeloch vytvorených touto prácou vytvoriť real-time mapu v ktorej je zvizualizovaná aktuálna pozícia autobusu.

Vyvíjaný nástroj automaticky namodeluje žiadúce linky s minimálnou vstupnou investíciou úsilia. Pri hlbšom zamyslení je ale jasné, že je to sekundárny krok, ktorému predchádza samotný plán tejto linky. Prídavnou hodnotou nad týmto manuálne vykresleným plánom sú informácie formátu do ktorého sa vstupné GNSS dráhy mapujú. Vybraný OpenStreetMap formát teda vykreslenému plánu dodá digitálnu formu v podobe súradníc jednotlivých nódov ktoré ho tvoria a daným nódom zároveň dodá správne umiestnenie na mape. Druhou záležitosťou sú spomínané informácie alebo dátový balíček, ktorý poskytuje OSM formát. To znamená, že danému modelu sa priradia povolené maximálne rýchlosti jednotlivých úsekov, typ danej infraštruktúry (cesta prvej/druhej triedy ...), prípadne dokonca aj materiál danej vozovky, príznak o tom, či je to jednosmerná cesta, pozície konkrétnych dopravných značiek, semafórov, informácie o tom, či má cesta veľa seba aj chodník pre chodcov, pozície autobusových zastávok a podobne. Viac k datovému balíčku viď [31].

Okrem spomenutého, samotná normalizácia (prevod GNSS dát na OSM dátový formát) umožňuje obrovskú škálu výpočtov nad získanými dátami. S asistenciou podkladovej mapy možno pracovať na optimalizácií danej dráhy, využívaní ciest vyšších tried ak vyhovujú podmienkam a vytvárať ako dráhy čo možno najkratšie, tak čo možno najviac efektívne z hľadiska životného prostredia a teda očakávaného množstva zbytočných zastávok. Normalizácia je súčasťou tejto práce a nie jej koncovým cieľom, každopádne len ona sama otvára dvere množstvu nových riešení.

Prídavná hodnota vychádzajúca z premisy nad výstupom tejto práce je zrejmá a či už aktuálne aplikovaných myšlienok na podobných projektoch, alebo potenciál-

nych myšlienok, ktoré čakajú na svoju realizáciu, je značný počet. Preto je spomenutá len veľmi malá vzorka týchto možností pre predstavu, práca sa bude ďalej venovať navrhnutému riešeniu.

Na vstupe úlohy je dráha autobusu v časovom rámci napr. jedného dňa vyjadrená v GNSS záznamoch. Očakávaný výstup bude súbor liniek, ktoré autobus obsluhoval počas tejto doby. Pod linkou si teda možno predstaviť dráhu, ktorou prešiel autobus s jeho konkrétnym číselným označením. Prakticky sa vloží akvizičná jednotka GNSS dát do ľubovoľného autobusu na isté obdobie. Dáta zozbierané touto jednotkou sa nechajú prejsť navrhnutým skriptom – výsledkom práce. Na výstupe sa očakáva súbor trás, ktoré spĺňajú požiadavky na to, aby boli označené ako linka.

V počiatočných štádiách práce bude pozornosť venovaná teoretickým rozborom, rešeršiam a hľadaniu tej správnej cesty v riešení konkrétnych problémov úlohy. Prejde sa formát vstupných dát, s ktorými sa bude pracovať. Rozoberie sa ktoré časti dát sa dajú využiť akým konkrétnym spôsobom. Ďalej sa práca sústredí na dátové formáty, ktoré partikulárne sa v tomto priemysle využívajú, povedie sa rozprava ohľadom ich využiteľnosti a ich kvalitách.

Ďalšia časť práce sa bude venovať hľadaniu vhodného algoritmu, ktorým sa dajú porovnávať úseky. Bude sa to hodiť do neskorších štádií projektu, kedy dôjde k avizovanému určeniu linky. Rozoberie sa viacero možností, opäť sa povedie rozprava nad tým aké výhody a nevýhody by mohla mať daná metóda alebo algoritmus pre porovnanie trajektórií.

Vychádzajúc z tretieho bodu zadania sa preberie problematika predspracovania dát. Prakticky sa bude jednať o metodiku ako previesť GNSS záznamy do vhodného uceleného, už existujúceho formátu. Bude vysvetlené prečo je vôbec tento bod v projekte nutný a aké benefity prináša. Popíšu sa zdroje pre extrakciu dát a aké sú medzi nimi rozdiely. Na záver sa vysvetlí akým spôsobom sa dajú pomocou vybraného nástroja dáta z geodatabázy filtrovať.

Od bodu 4 dochádza k implementáciám diskutovaných témat prvých troch kapitol. Vytvorí sa ucelená vstupná vzorka a pripravia sa potrebné náležitosti vo veci dátového modelu mesta. Diskutované metódy porovnávania dráh sa algoritmicky vhodne implementujú a použijú na pripravenú vstupnú vzorku. Na výstupe sa očakávajú modely liniek vyskytujúce sa v GNSS dátach. Na záver je treba navrhnúť spôsob ktorým sa vhodne otestuje a posúdi výkon navrhnutého riešenia.

Práca vznikla náväznosťou na semestrálnu prácu [36].

1 Formát poskytnutých dát a rešerš dátových formátov vhodných pre prácu s GNSS dátami

Je vhodné tematicky uceliť tieto dva body v názve sekcie, pretože prvý bod značne naväzuje na druhý. Naštudovanie vstupných dát môže značne ovplyvniť ktorý dátový formát pre GNSS manipuláciu bude potenciálne najvhodnejší.

GNSS dáta V tomto paragrafe bude vysvetlené, čo sa myslí pod pojmom GNSS dáta. Tieto dáta sú zozbierané metódou, kedy je v autobuse zavedený tracker polohy, ktorý pravidelne zaznamenáva súbor dát - ich formát a obsah sa ujasní v ďalšej sekcii. V autobuse tento tracker ostáva zaimplementovaný na dlhú dobu a to v rade dní. Zapne sa so štartom motoru autobusu a vypne sa taktiež spolu s motorom autobusu. Autobus v dobe zapnutého trackeru obsluhuje viacero liniek spolu s obmenenými trasami súvisiacimi s nočnými spojmi. Vodič s ním nekomunikuje ani pri štarte autobusu ani pri zmene linky. Informácia o určitom identifikačnom čísle konkrétnej linky nebude k dispozícii a identifikácia linky (podľa podmienok opakujúcej sa trajektórie) bude jedným z cieľov tejto práce. Platí, že o trajektóriách liniek sa vopred nevie.

1.1 Formát poskytnutých dát

Zberná jednotka, ktorá bola použitá pre vstupné dáta, zaznamenáva každú sekundu 19 vlastností o danom stave. V tejto sekcii bude vyjadrené, ktoré pre prácu môžu byť v rámci riešenia úlohy podstatné a k vybraným bude uvedený príklad.

- **Longitute, Latitude**
  - jedná sa o zemepisnú dĺžku a zemepisnú šírku
  - sú zdrojom hlavnej informácie, ktorá nás zaujíma a vyjadrujú pozíciu vozidla
  - ak autobus stojí, tieto dva parametre sa vôbec nemenia, čo vychádza z istej implicitnej doprogramovanej funkcionality v zariadení, pretože sú v tom prípade ignorované aj šumy

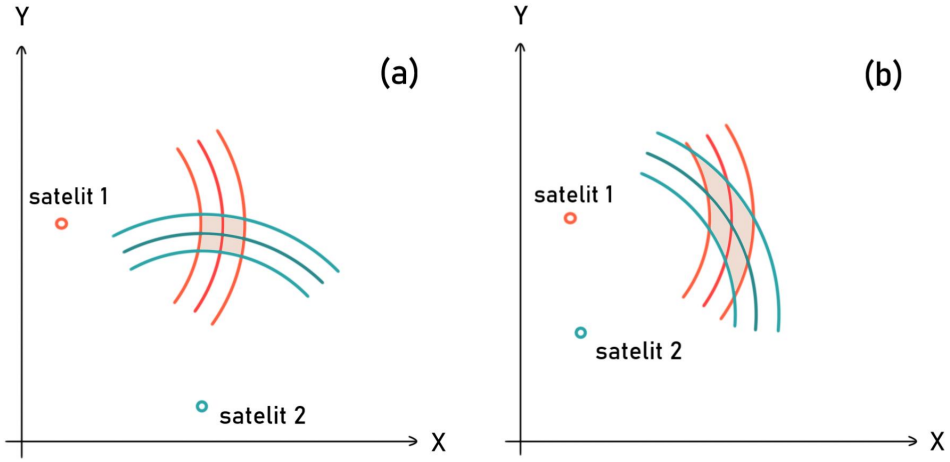
- **Speed**
  - táto vlastnosť je doprogramovateľná, stačí vychádzať zo vzdialenosti medzi pozíciami autobusu s jedno-sekundovým odstupom

  - zaznamenávanie rýchlosti autobusu a následná práca s dátami, prípadne štatistika nad takýmito dátami má nespochybniteľne množstvo aplikácií; príklady aplikácií, ktoré súvisia s konkrétnym vypracovaním tejto práce:
    * komparácia trajektórií na základe údaju o rýchlosti na princípe korelácie
    * očakávanie prechodu križovatkou na základe zmeny rýchlosti – možnosť oddeliť správanie sa ľubovoľného algoritmu v takýchto úsekoch na základe podmienky o rýchlosti

- **Course**
  - ďalšia doprogramovateľná vlastnosť ktorou sa prakticky určuje smer jazdy autobusu vo forme uhla od magnetického alebo skutočného severu [1]
  - použiteľné pri tvorbe podmienok napríklad v kombinácii s rýchlosťou, kedy sa neočakáva, že sa po krátkom čase s vysokou rýchlosťou výrazne zmení kurz; ako v prípade rýchlosti, znova je to použiteľné v normalizácii

- **DOP - Dilution of Precision (parameter presnosti) [2]**
  - jedná sa o vyjadrenie neistoty viazané ku konkrétnemu GNSS záznamu
  - nižšia hodnota DOP reflektuje presnejší výstup, a teda menšiu chybu
  - viď obr.č.1.1, v ktorom možno vidieť dva prípady (a) a (b); výsledné súradnice sú dané prienikom ôs pásov neistôt týchto dvoch satelitov – DOP bude v teórii vychádzať z veľkosti tohto pásu, kedy braním v dotaz chyby jednotlivých satelitov sa bude skutočný výsledok nachádzať v ľubovoľnom bode tohto pásma – záleží na tom, ako je definovaná chyba výsledku pre satelit; z obrázku teda možno odvodiť, že vhodným geometrickým usporiadaním satelitov dôjde k nižšiemu DOP, a teda k presnejšiemu záznamu

- **DGPS [3]**
  - princíp a technológia je založená na rozmiestnení základní, ktorých poloha je určená s dôrazom na to, že to bude referenčný bod pre budúce záznamy a merania; keď táto referenčná sieť získava GPS záznamy o polohe (jej známych) bodov, dokáže určiť/vypočítať potrebný set korekcií, ktoré treba aplikovať na daný odhad finálnej lokácie
  - táto referenčná sieť a záznamy o korekciách sa typicky prenášajú oddelene, len v rámci staníc v tejto referenčnej sieti
  - vo vstupnom súbore sa získava informácia o tom, či pri výpočte súradníc bodu došlo k využitiu DGPS alebo nie; nadobúda hodnoty True/False



Obr. 1.1: DOP – Príklad. Pri meraní doby letu vzniká pri identifikácii polohy bodu určité pásmo v ktorom sa nachádza výsledok. Podobne to je pri meraní polohy druhým satelitom. Výsledná oblasť v ktorej sa očakáva poloha hľadaného bodu je ich prienikom. Veľkosť tejto plochy je závislá ako od veľkostí neurčitostí daných meraní vzdialeností, tak ale aj od polôh satelitov, čoho efekt ilustruje obrázok. [2]

1.2 Dátové formáty vhodné pre prácu s GNSS dátami

Problematika výberu vhodného dátového formátu je veľmi široká a silne sa odvíja od požiadavok projektu. Konkrétne požiadavky budú prebrané v nasledujúcich kapitolách. Ich opodstatnenie môže byť na prvý pohľad zjavné, každopádne ide o to, že čím sú požiadavky vyššie, tým viac financií, prostriedkov a ľudských kapacít bude potrebných pre vyriešenie tejto časti projektu. Riešiť cestnú infraštruktúru v rámci Brna možno za pomoci dát, ktoré zozbierala miestna firma, mesto, alebo externá entita, ktorá garantuje potrebnú presnosť. Ak chce mať projekt pokrytie v rámci napríklad celej strednej Európy, v závislosti na požadovanej presnosti je nutné zanalyzovať riešenia od globálnejších a celistvejších až po služby menších miestnych firiem, ktoré garantujú presnosti a relevantnosť dát na menších úsekoch. Ľahko si to možno predstaviť napríklad na známom portále mapy.cz pri nahliadnutí na licenčné podmienky [4]. Každý jeden mapový podklad vznikol fúziou produktov viacerých spoločností. Či už zmluvne alebo použitím v súlade s existujúcou licenciou vidíme, že je v konkrétnych prípadoch k dispozícii len určitý level detailu, určitá úroveň zoomu. Spoločnosť ESRI, ako ľahko dohľadateľný najväčší hráč vo svete GIS providerov [7] je sama o sebe taktiež len poskytovateľ hostingu pre dáta skutočných vlastníkov, ktorými sú napríklad mestá alebo štáty. Dáta získané napríklad manuálnym zberom môžu alebo nemusia byť poskytnuté pre verejnosť a rozhodovať o tom bude

vlastník dát a teda najčastejšie mesto alebo štát ktorého územie je zmapované [5]. Manuálny zber v súčasnosti bol spomenutý v zmysle preskenovania ciest kamerami na účelných automobiloch. Presnejšie povedané sú na týchto skenovacích vozidlách napríklad aj LiDAR senzory, rôzne zoskúpenia kamier a množina iných senzorov pre dôveryhodné nasnímanie prostredia a sebalokalizáciu pre rôzne projekty spoločností, ako Here alebo Google [6] v oblasti GIS.

1.2.1 Presnosť a rozmanitosť geodatabáz

Set metód, ktorý sa používa na tvorbu datovej reprezentácie miest je obsažný, tak isto je následná licenčná povaha tejto tematiky na úrovni rôznych štátov znova ďalšou kapitolou. Budú spomenuté faktory a najčastejšie skloňované myšlienky témy tvorby GIS dát.

V tematike presnosti, bude často krát delená na úseky, ktorých sa určitá presnosť týka. Žiaden provider týchto dát doposiaľ nedošiel do bodu v ktorom si jednotne, systematicky a len jednou metódou svojpomocne namapoval celý svet. Vždy, ak sa bavíme aspoň o hranične použiteľnom dátovom zdroji na globálnej úrovni, ide o spojenie veľkého množstva dát a snahu o ich homogenizáciu. Príkladom je napríklad projekt Global Roads from GRIP pod ESRI [8], ktorý zaintegroval dáta o cestách zo 60 rôznych geodatabáz. Takýmito metódami možno dosiahnuť zaujímavých výsledkov, zaujímavých presností a hlavne hodnoverných dát s istou predpokladanou presnosťou. Problémom je ale ich hodnovernosť s odstupom času. Výsledky takýchto projektov a snáh potrebujú formu aktualizácie aby reagovali na zmeny – nové cesty a zmeny v aktuálnych usporiadaniach. To, ako riešiť túto problematiku je opäť obšírna téma o ktorej bude základ vysvetlený v ďalšej kapitole.

1.2.2 Konzistentnosť, praktická hodnovernosť v reálnom čase

Svet sa mení, menia sa cesty, stavajú sa parcely, vznikajú a zanikajú podniky. Preberanou témou sú databázy, ktoré majú zachytávať všetky, alebo vybraný prvok z tejto množiny. Či už sú to cesty, entity menej fluktuačné čo do existencie, alebo podniky, ktorých životnosť je omnoho viac premenlivá – majú spoločný prvok, ktorým je táto popisovaná variantnosť, ktorú je nutné zaznamenať. Fixná databáza takže môže a nemusí vyhovovať účelom daného projektu.

V dôsledku spomenutého, rozhodnutie vo veci zdroja dát vedie k zakúpeniu týchto dát ako služby, pričom ich správca má potenciálne čo najvhodnejšiu metódu na udržiavanie aktuálnosti daných dát. To, ktorého providera si na projekt vybrať, závisí od toho, na čo sa sústreďuje. Jedna z metód je napríklad dovoliť užívateľom robiť zmeny. Ak entita/človek založí podnik, môže túto skutočnosť zaznamenať do Google Maps a nemusí na nič čakať – takto to teda od roku 2008 robí samotný

Google [10]. Spoľahlivosť takýchto dát je založená na tom, že si Google uchováva históriu úprav, užívatelia ju majú sprístupnenú a teda aj v prípade nežiadúcich chýb dokáže z konečného výsledku užívateľ spoľahlivo ťažiť (žiadna strata dát, overovanie zadávateľa - výsledná konvergencia k správnemu finálnemu údaju).

Here, spoločnosť historicky známa skôr pod názvom Navteq je v súčasnosti vlastnená aj konzorciom nemeckých automobiliek (Audi, BMW, a Daimler) [12], ktorí okrem série iných spôsobov využívajú fakt, že sú integrovaní do navigačných systémov 150 miliónov vozidiel[11]. Jeden zo spôsobov aktualizácií máp, je využitie vbudovaných senzorov užívateľských vozidiel ktoré na týchto cestách jazdia. Napojením sa napríklad na cloud a umožnenie promptného prístupu k týmto informáciám umožňuje prakticky meniť mapy za behu priebežne spracúvajúc obrovské množstvá dát z množstva senzorov bežne prítomných v autách dnešnej doby.[13] Zároveň je aj verejný výčet ich áut, ktoré, ako už bolo spomenuté, brázdia ulice so sériou kamier, LiDAR a rôznych iných senzorov.[15] To je ďalší spôsob ktorým držia mapy aktuálne, prípadne ktorým aktívne dopĺňajú množstvá informácií. Pre náhľad o množstve úprav a významnosti takejto služby je zaujímavé spomenúť, že čo do množstva úprav ich denne vykonajú okolo 2.7 milióna [14]. Tieto spoločnosti v rámci snáh o implementáciu a živú aktualizáciu dát samozrejme vytvárajú neustále prieniky viacerých služieb. Najrýchlejšie a najznateľnejšie to možno vidieť na kópii licenčných podmienok pre Apple Maps [16], kde vidno, že čo do mapových podkladov a dát z tejto všeobecnej sféry možno vidieť OpenStreetMaps a Waze, prípadne sú Apple mapy doplnené informáciami z webov ako Booking.com, Yelp alebo Trip Advisor. Mená ďalších hráčov sú napríklad Tele Atlas – TomTom, NASA, Navteq – Here, OpenStreetMaps (vo forme kontribútorov – keďže je to opensource a užívateľmi budovaný obsah, bez proprietárnych licencií), AutoNavi – dodáva ako pre Apple Maps, tak aj pre Google pre oblasť Číny [17], Zenrin a ďalší. Posledné dve spomenuté sú spoločnosti operujúce o čosi viac lokálne, AutoNavi pre oblasť Číny a Zenrin v Japonsku.

V tejto sekcii sa teda zhrnulo ako fungujú spoločnosti ktoré poskytujú produkt, ohľadom výberu ktorého je potrebné sa zorientovať. Záleží v akej mierke, sú to často obrovské spoločnosti s obrovským obratom, pracujúce na produkte o ktorý majú záujem stovky tisíc klientov. Konkrétne mal už viac krát spomínaný ESRI ešte v roku 2009 obrat 1,2 miliardy dolárov a 300 000 klientov. Ich snahy sú teda inteligentne dáta získavať a na základe potrieb zákazníckej klientely doplňovať produkty dátami z iných dátových zberieň oblastí, ktoré by mohli byť pre užívateľa koncového produktu zaujímavé – ako už spomenutý napríklad Booking.com alebo Yelp.

1.2.3 Pokrytie

Ako bolo spomenuté v úvode pre sekciu o dátových formátoch, vychádza sa z požiadavok na základe ktorých možno ďalej usudzovať možnosti. Je otvorené na akom území má v závere fungovať výstup tejto práce, a preto je treba poňať túto tému zo všeobecného hľadiska. Z už spomenutých licenčných podmienok od Google Maps [9], Mapy.cz [4] alebo Apple Maps [16] zjavne vidno, ako je ich služba pozlepovaná či už z dôvodu doplnenia miestnych alebo globálnych dát. Vo výbere dátovej bázy je teda kľúčové začať pokrytím a zhodnotiť služby poskytované na územiach v ktorých sa uvažuje implementácia.

1.2.4 Odôvodnenie výberu dátového formátu OSM

Na čo je dobré poukázať, sú často vyskytujúce sa chyby v porovnaniach k OSM mapám. Google Maps, Apple Maps, Bing Maps a mnoho iných nie sú služby porovnateľné s OpenStreetMaps - nie úplne. V jednom hľadisku všetky poskytujú bežnému používateľovi určitý priestor a konzumpciu istých dát, na čo sa ale celý čas sústreďuje táto práca sú práve dáta **pod** službou, ktorú extrémna majorita používateľov využíva. Tieto dáta sú takzvané "raw" vektorové dáta alebo GIS datové zdroje – určite vhodnejšie využívať anglické termíny Raw Vector Data, Raw Map Data, GIS Data Sources a pod. Je vhodné si uvedomiť, že hoci užívatelia majú možnosť prispievať do Google Maps, ich kontribúcia priamo napomáha súkromnej spoločnosti a napomáha to jej profitu. Naopak prispievaním do OSM benefituje spoločnosť, benefituje akýkoľvek užívateľ OSM na celej zemi, ktorý môže dáta využívať a vytvárať nové riešenia či aplikácie z dôvodu vhodného nastavenia ich licenčných podmienok. [23]

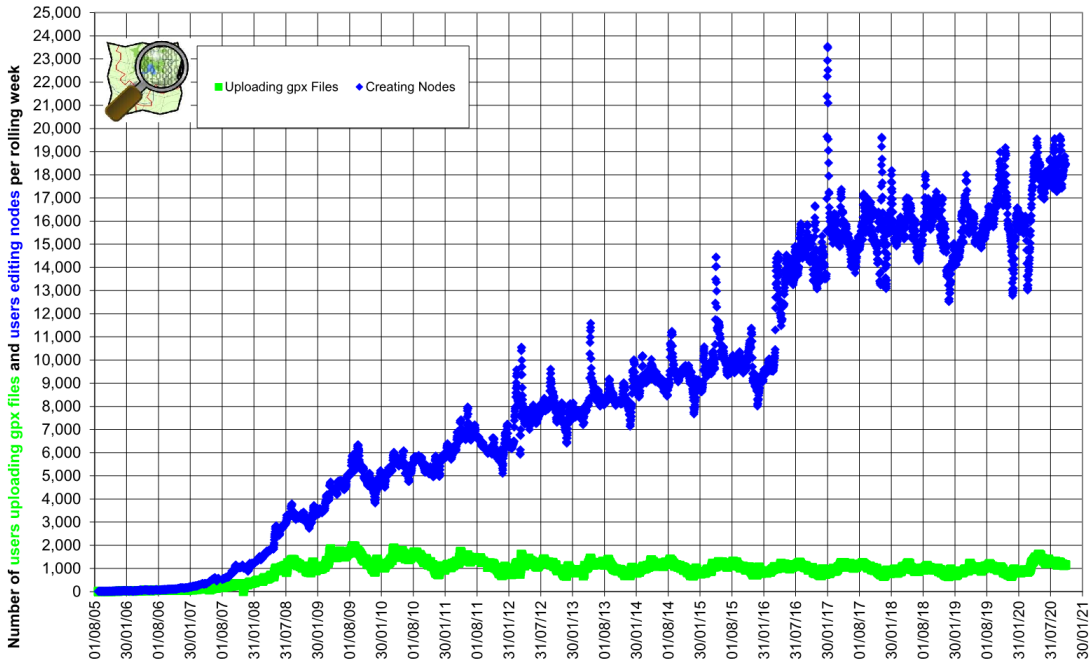
Licencia, formát, pokrytie OSM reprezentuje projekt, ktorý je zaviazaný tomu byť opensource. Má potenciál do budúcnosti systematicky rásť. Premisa je veľmi sľubná hlavne z nedostatku faktorov, ktoré vytvárajú negatívne črty - respektíve ktoré kazia jeho potenciál. Ide o model podobný Wikipedii, každopádne autorizáciou prispievateľov OSM eliminuje parazitných prispievateľov ešte o čosi viac – je to podobné ako v prípade Google Maps, kde je taktiež vytvorený takpovediac inteligentný systém na zachytenie cielene parazitných počinov[20].
OSM uchováva dáta vo formáte XML, pričom z dostupných rozhraní na stiahnutie konkrétnych oblastí je k dispozícii napríklad GeoJSON, GPX alebo KML. Na prácu z hľadiska kompatibility sú to postačujúce možnosti, ku ktorým sa ľahko nachádzajú knižnice pre programové spracovanie.

Pokrytie je a bude otázka, rovnako ako presnosť dát konkrétnych úsekov. V rešerši som sa dopracoval k viacerým štúdiám, ktoré sa tejto problematike venovali – nepovažujem ale za úplne relevantné z ich výsledkov dedukovať závery. Dôvody sú dva. OSM projekt zaznamenáva od skorých mesiacov jeho existencie kontinuálny nárast ako užívateľov, tak pridávaných bodov [20]. Viď tak isto obr.č.1.2, prípadne jeho zdroj v popise obrázku - je tam množstvo štatistík nad kontribúciami, počtom nových užívateľov, či kontribútorov. Niekoľko rokov staré štúdie zaoberajúce sa či už dátami na globálnej úrovni alebo len konkrétnou oblasťou môžu byť považované v tomto bode z časového hľadiska za zastaralé. Druhý dôvod je fakt, že bez ohľadu na to, v akom stave je globálny priemer, môžu byť relevantné úseky pre potenciálny projekt zmapované veľmi dôkladne. Oplatí sa každopádne naskenovať najnovšie štúdie, ako napríklad nasledujúci zdroj [25], ktorý sa zaoberá konkrétnou Indickou oblasťou a za rok 2018 a 2019 by dávalo zmysel túto referenciu považovať za hodnovernú vo veci stavu prejednávaných máp. Zároveň prejednávajú možnosti autonómnych procesov vylepšenia, ktoré budú pripomenuté v paragrafe o potenciále OSM 1.2.4. V prípade nutnosti všeobecného zorientovania sa, je priložená štúdia z roku 2008 [20], ktorá poskytuje zaujímavé náhľady, každopádne je v tento moment nie najrelevantnejšia práve z dôvodu už spomenutého kontinuálneho vývoja tejto služby a nie je zmysluplné zdôrazňovať konkrétne čísla. Štúdia je zaujímavou z pohľadu rozpravy nad VGI (Volunteered geographic information) [19] na čom je založený projekt OSM.

Zdrojmi pre projekt môžu byť aj dáta z nasledovného zdroja [27], primárne zostavený pre kontribútorov OSM máp. Jeho základnou charakteristikou pod ktorou bol zostavený je potenciál, že licenčne súhlasí s OSM licenčným modelom [26].

Diffs Boli spomenuté nevýhody statických dát. OSM ich rieši pravidelnými vydaniami aktuálnych zmien, ktoré sa dajú získať v najkratšom možnom horizonte len jednej minúty.[24] Vývojári teda môžu vopred vyriešiť problém relevantnosti dát implementáciou týchto pravidelných aktualizácií.

Potenciál OSM OSM vytvára objemný manipulačný priestor pre nové projekty. Tak ako si možno predstaviť obrovskú obrázkovú databázu spoločnosti Google na ktorých trénuje vlastné produkty v oblasti image processingu, tak je vo svete mapovacích snáh tvorená báza OSM, pričom je ale verejne dostupná, a teda jej potenciál ponúka širokej verejnosti. Týmto spôsobom získava pozornosť rôznych inštitúcií a na jej báze vzniká množstvo projektov z oblasti machine learningu, ktoré majú potenciál byť zaimplementované a túto službu ďalej zlepšovať. Vzniklo množstvo štúdií, ktoré sa venujú eliminácií chýb, trénovaniu inteligentných algoritmov na dopĺňanie chýbajúcich dát, prípadne aj inteligentnému zisťovaniu ich spoľahlivosti [25][21][22]. Následkom implementácie takýchto riešení dôjde napríklad k potenciálnej eliminácii



Obr. 1.2: Štatistika nad OSM kontribúciami v lineárnej mierke. Ilustruje kontinuálny nárast kontribúcií v časovom slede cca každých 6 mesiacov, zobrazujúc dáta jedného týždňa k danému dátumu. Relevantný je trend zvýraznený modrou farbou symbolizujúci pridané nódy do OSM databázy. [34]

chýb, ktorých sa dopúšťajú neskúsení kontribútori. Posledná citovaná štúdia [22] zároveň poskytuje kvalitnú referenčnú databázu v oblasti OSM, ktorá sama o sebe hovorí o relevancii tejto služby. Následne je samozrejme otázne či sa tieto algoritmy implementujú do svojej alma mater na ktorej vznikli, alebo sa predajú do privátneho sektora. S mentalitou práce na opensource projekte ale treba podporovať následné obohatenie služby, ktorá umožnila vznik týchto algoritmov, a teda obohatenie OSM algoritmov a služieb. Koniec koncov je takýmto štýlom postavená aj samotná licencia OSM v rámci ktorej má byť výsledok práce, v rámci ktorej sú použité tieto dáta, šírený pod rovnakou licenciou [26]. Dodržiavaním stanoveného je potenciál pre OSM projekty vyšší hlavne z hľadiska priamej tvorby riešení na relevantných dátach, na rozdiel od potenciálneho predaja privátnemu sektoru, kde nastávajú minimálne potenciálne nezrovnalosti v kompatibilite.

2 Metódy pre reprezentáciu a komparáciu trajektórií

Všeobecný postup na projekte naznačuje zorientovanie sa vo formátoch a dátovej reprezentácii, nalezenie vhodného dátového zdroja (kapitola 1), normalizácia dát (kapitola 3) a záverečné klastrovanie dát – klasifikácia liniek (kapitola 5), v ktorej sa využijú algoritmy, ktorých rozbor je v kapitole 2. Postup reálnej práce na projekte prebiehal týmto sledom. Z dôvodu zachovania štruktúry a čitateľnosti práce, kapitoly rešpektujú sled zadania. V prípade, že chce čitateľ viac nasledovať sled praktickej práce, môže kapitolu 2 preskočiť a vrátiť sa k nej po kapitole 3 alebo po kapitole 4. Je to viacmenej samostatná teoretická časť práce, ktorá je prakticky zaimplementovaná v piatej kapitole. Všeobecne popisujú prvé tri kapitoly teoretický základ práce, v štvrtej kapitole sa predstaví vybraná množina vstupných dependencií (mapy, dráhy, GNSS dáta a pod.), v piatej kapitole sa aplikujú všetky poznatky z teórie na danej vybranej množine a v šiestej kapitole sa riešenie pretestuje.

Táto sekcia práce je venovaná druhému bodu zadania o znení: “Proveďte obecnou rešerši metod pro reprezentaci a komparaci trajektorií”. Výstup z tejto teoretickej rešerše bude implementovaný v piatom bode zadania, kedy bude potreba klastrovať úseky. Prakticky ide o problém nájdenia opakujúcich sa úsekov vo veľkej nadradenej množine. Pri aplikácií teda dôjde k žiadanej klasifikácii MHD liniek.

Jednotlivé algoritmy (všetky tri budú v závere zameniteľné a bude testovaná ich úžitkovosť) budú využité na zistenie podobnosti dvoch konkrétnych normalizovaných úsekov. Výstupom každého algoritmu nech je koeficient ich podobnosti. Na základe veľkosti koeficientu sa určí platná autobusová linka.

Voľba rozsahu linky

Vo všetkých nájdených alternatívach pre komparáciu trajektórií sa vychádza z predpokladu znalosti toho, aký dlhý má byť hľadaný opakujúci sa úsek. Z dôvodu normalizácie dát a teda podradenosti dátovému formátu nebude tento rozsah vyjadrený v štandardnej jednotke, respektíve ak to bude žiadúce, bude potreba vykonať istý spôsob normalizácie pre túto neexistujúcu mierku.

Príklad – vopred sa stanoví, že linka môže byť dlhá 3 až 15 kilometrov. Každopádne je otázne čo toto vyjadrenie znamená na normalizovanej mape, keďže vo formáte s ktorým sa pracuje môže byť 1 kilometer zložený zo 100 normalizovaných bodov, ale možných je aj 200, záleží o ako zložité úseky ide a ako sú reprezentované (koľko nódov bolo použitých na popis cestného úseku). Algoritmy budú minimálne spočiatku testované na základe zhodnosti v počte normalizovaných bodov. Riešením

tohto problému bude paralelný výpočet vzdialeností zhodujúcich sa bodov, a teda paralelná kontrola skutočnej dĺžky daných trajektórií napríklad v metroch. Očakávateľne týmto spôsobom dôjde k vysokej chybe, pretože budú zatáčky nedokonale oblé (budú vyslovene hranaté) - na rozdiel od oblej skutočnej cesty. Následne je možnosť identifikovať ktoré pôvodné GNSS body patrili danému potvrdenému normalizovanému bodu a vzdialenosť sa bude počítať zo sekvencie týchto pôvodných GNSS dát. Chyba sa takto do veľkej miery eliminuje. To, aký dodatočný kompenzačný algoritmus bude nasadený ešte nad to, bude predmetom v poslednom bode zadania – načrtnutý postup by mal byť dostatočne presný. Konkrétne čísla o presnosti a nutnosti vytvorenia spomínaného dodatočného algoritmu sa budú diskutovať priebežne pri praktickom riešení úlohy a testovaní. Navrhovaný nadradený algoritmus by sa mohol snažiť sledovať zatáčkovosť úsekov na základe čoho potenciálne dráhe nadsadí veľkosť. Ideálnym riešením by bolo zavedenie algoritmu pre preloženie bodov krivkou, každopádne môže dôjsť k významnému spomaleniu z dôvodu jeho výpočtovej náročnosti.

Predpoklady problémov sú teda jasné a v tomto bode je navrhnuté riešenie pre každý z nich. Bude záležať na parametroch reálnej implementácie a jej štatistiky, či bude nutné siahnuť po všetkých navrhovaných riešeniach, alebo iba po niektorých.

2.1 Levenshteinova vzdialenosť

Výstupom aplikácie algoritmu je číslo, ktoré značí celkový minimálny počet úprav nutných k trasformácii jednej množiny na druhú.[28] Tieto úpravy môžu byť troch typov:

- Vymazanie prvku
- Nahradenie prvku iným prvkom
- Pridanie prvku

Predstavme si teda dve množiny čísel A a B (čísla budú v práci zamenené za dvojice súradníc):

- A = 10,1,2,3,11,4,5
- B = 1,2,3,18,4,5,6

V tomto prípade by sa odobralo z množiny A číslo 10 na začiatku, zmenilo číslo 11 na 18 a na záver pridalo číslo 6, a teda Levenshteinova vzdialenosť je 3 (v zmysle celkového minimálneho súčtu, a teda troch úprav). Bola predstavená značne jednoduchá možnosť, každopádne pri komplexných radoch čísel je určenie minimálneho počtu úprav o čosi zložitejšie, hlavne z dôvodu posunu radu čísel pri vymazávaní

či pridávaní prvkov. Princíp v zovšeobecnenej algoritmickej podobe spočíva v postupnej iterácii od nultého po posledný prvok pridávaním vždy nového prvku do reťazca a rozpravou nad počtom operácií, ktoré sú nutné pre transformáciu jedného čiastkového reťazca do druhého čiastkového reťazca. Takými postupnými iteráciami, kde má každá jedna v skutočnosti opodstatnenie, lebo sa od nej odvíja každý ďalší výpočet, sa v závere skonverguje do jediného čísla o celkovom počte operácií nutných na vykonanie nad jedným reťazcom pre jeho premenu v druhý reťazec (znakový reťazec, prípadne postupnosť čísel sa rozumie ako rovnaký prípad).

Algoritmus je vhodný pre účel tejto úlohy hlavne z dôvodu toho, ako sa vo výpočte chová a aké chyby v nezrovnalostiach sekvencií zarátava do koeficientu (v tomto prípade výslednej vzdialenosti). Napríklad, ak by bol tento algoritmus striktne viazaný na pozíciu konkrétneho prvku, vhodný by jednoznačne nebol. Z jeho podstaty teda vyplýva žiadaná účelnosť, a preto bol zvolený pre implementáciu a následné pretestovanie v neskorších štádiách práce.

2.2 Gestalt Pattern Matching

Algoritmus slúži na porovnávanie dvoch jedno-dimenzionálnych polí.[29] Identifikovať ho možno pod názvom *Ratcliff/Obershelp pattern-matching algorithm*. Postup výpočtu prebieha následovne:

- Medzi dvoma celkami sa identifikuje najväčšia možná totožná neprerušená sekvencia.
- Následne sa postupnosť rozdelí na pravý a ľavý zostatok v ktorom nastáva totožný proces ako v prvom bode, a teda sa rekurzívne hľadá najdlhší reťazec na základe ktorého sa znova daný celok rozdelí na pravú a ľavú časť (ak je to možné).
- Výsledný koeficient sa počíta ako dvojnásobok súčtu dĺžok týchto čiastočných zhodových úsekov, ktorý sa predelí súčtom prvkov dvoch porovnávaných sekvencií. Výstup je koeficient v rozmedzí 0 až 1 ktorý vyjadruje podobnosť postupností vychádzajúcu z tohto algoritmu.

Vhodnosť tohto prístupu pre riešenie zadanej úlohy spočíva v tom, že ponúka prístup na inej báze ako Levenshtein, zatiaľ čo rovnako berie do úvahy postupnosť celej vzorky.

Jeho implementácie v Python difflib knižnici [30] navyše poskytujú komputačne menej náročné alternatívy. Konkrétne sa jedná o funkcie quick_ratio() a real_quick_ratio() z ktorých od ich komputačnej podstaty druhá nepripadá v úvahu a teda sa bude ignorovať. Prvá spomenutá ale ponúka prístup, ktorý bude stáť za to – otestovať ho. Konkrétne padne potenciálne vhod vo vývojových fázach, kedy bude

schodnejšie pracovať s rýchlymi algoritmami pre testovanie konkrétnych úsekov pri ladení vyvíjaného skriptu.

Funkcia quick_ratio() vynásobí dvomi všeobecný počet zhodujúcich sa prvkov medzi dvoma porovnávanými množinami a opäť ich vydelí súčtom ich dĺžok. Nódy reprezentujúce trajektórie na mape sú všetky unikátne pre konkrétne úseky, a teda ich samotná zhodnosť v oboch množinách bude znamenať prejazd. Tento algoritmus sa bude vo veci prístupu k problému lepšie porovnávať s mnou vytvoreným jednoduchým algoritmom vychádzajúcim zo vzdialeností jednotlivých nódov v porovnávaných sekvenciách. Toto riešenie je predmetom nasledujúcej sekcie.

2.3 Vlastný komparačný algoritmus

Riešenie prakticky pracuje s rovnakými vstupmi a na výstupe pracuje s číslom, ktoré rozhoduje o zhodnosti úsekov. Popis algoritmu:

- Rovnako ako v predchádzajúcich možnostiach sa preiteruje celou hlavnou množinou a dospeje v jednotlivých iteráciách k dvom úsekom pre porovnanie.
- Prvý nód prvej množiny sa preiteruje celou druhou množinou a zaznamená jednotlivé vzdialenosti medzi týmito nódmi. Preiterujú sa všetky možnosti a zaznamená sa najkratšia možná.
- Postup sa opakuje pre všetky body prvej množiny a tieto najkratšie vzdialenosti sa sčítajú.
- V závere sa predelia celkovým počtom nódov prvej množiny a tým sa získa priemerná najmenšia možná vzdialenosť nódov medzi množinami.

Predpoklad je, že toto výstupné číslo bude menšie pre linky, ktoré sú si podobné. Pre jednoduchosť tohto algoritmu si ho možno ukázať, viď výpis č.2.1. V prípade záujmu o nahliadnutie na exemplár funkcionality, je treba nahliadnúť na unit testy, kde je otestovaný na dvoch príkladných vstupných dátach.

Výpis 2.1: Programová implementácia vlastnej komparačnej metódy. Viď príloha A adresár **functions**/pattern_finding, modul patternfinding_functions.py

```
1    def myratio_calc(list1, list2):
2        suma = 0
3        lengthb = len(list1)
4        for i in range(0, lengthb):
5            fdist = 100
6            for j in range(0, lengthb):
7                a = list1[i][0] - list2[j][0]
8                b = list1[i][1] - list2[j][1]
9                dist = math.sqrt(a**2 + b**2)
10               if fdist > dist:
11                   fdist = dist
12           suma += fdist
13       return suma / lengthb
```

3 Návrh a realizácia spôsobu predspracovania a dátovej reprezentácie GNSS záznamov jázd vozidiel hromadnej dopravy

V tejto sekcii práce bude prejednávaný tretí bod zadania. Predspracovanie je v tejto práci vykonané formou normalizácie. Je to predspracovanie do formy, ktorá dáva zmysel širšiemu publiku a je implementovaná v špecifickej dátovej reprezentácii.

3.1 Normalizácia

Pod normalizáciou sa rozumie prevod vstupných GNSS dát, ktoré boli popísané v kapitole 1.1, do formátu OpenStreetMap, ktorý bol vybraný ako najvhodnejší na základe kladených požiadavok. Proces výberu bol dedukciou z teoretického rozboru kapitoly 1.2 a následne popis a odôvodnenie výberu OSM formátu v kapitole 1.2.4.

3.1.1 Nevyhnutnosť normalizácie

Nevyhnutnosť zmeny formátu z čisto GNSS vstupu nastala z viacerých dôvodov:

- **Veľkosť dát** – normalizáciou došlo k redukcii dát a uchovaniu len zložiek, ktoré sú pre projekt podstatné. Došlo k použitiu dodatočných informácií 1.1 zo vstupných GNSS dát a k následnej absencii nutnosti ich mať uchované. Pochopiť to možno na jednoduchom príklade, kedy zber GNSS dát nastáva každú sekundu a celý dátový balíček teda konzistentne narastá, hoci autobus napríklad stojí. V normalizovanej reprezentácii bude všetkým týmto dátam prisúdený jeden nód. Typická ukážka napríklad na obr.č.3.1. Kedy periodickým zberom GNSS môže dôjsť k nadbytočnému počtu týchto bodov a navyše konštantne podliehajú nepresnostiam a vplyvom prostredia. Vedieť ich pretransformovať na dáta zvýraznené červenou jasne evokuje vyššiu úžitkovosť a presnosť.
- **Zmena formátu** – detaily o formáte v kapitole 3.1.2. Hmlisto navigovateľný prístup k vlastnostiam daného nódu z GNSS formy vstupných dát pomocou napríklad vnorených listov nahradzuje zrozumiteľný systém. V novom formáte sa konverguje k možnostiam ľahšie programovo pristupovať k dátam a upravovať ich. Tým, že sa jedná o zaužívaný formát, je k dispozícii viacero nástrojov pre jednoduchšiu manipuláciu. Kontribútorským a vizualizačným príkladom v jednom je JOSM editor - viac k tomuto editoru v paragrafe 3.2.1.

Obr. 3.1: Ukážka zmyslu transformácie GNSS dát do formátu OSM. Obrázok je iba názorný a vykreslenie GNSS dát môže byť presnejšie alebo menej presné v závislosti na použitých trackeroch. Pri vzorkách reálnych dát tohto projektu obrázok ilustruje realitu.

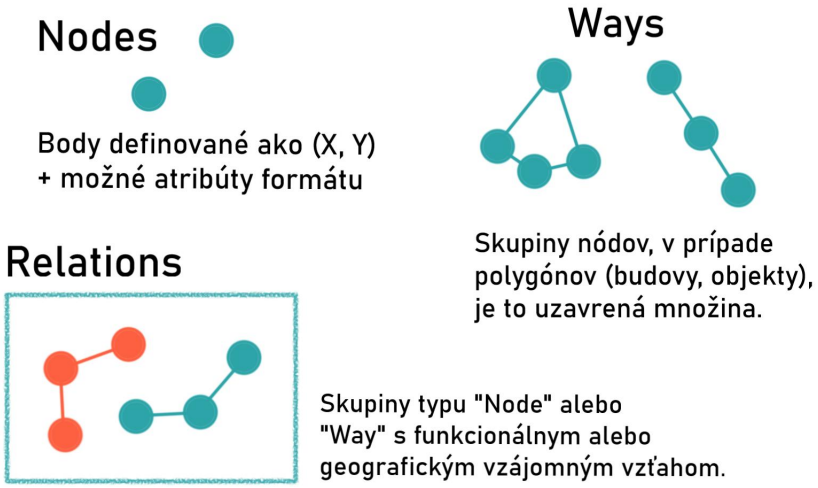
- **Nadobudnutie informácie o ľudsky zrozumiteľnej geografickej polohe** – pochopiteľne je zrozumiteľnejšie počuť, že sa autobus nachádza na ulici Skácelova, ako získať jeho geografickú polohu v súradniciach. Normalizáciou, spolu s polohou prichádza aj informácia o mieste na ktorom sa entita nachádza v podobe názvu ulice, prípadne sú daným nódom pridelené ďalšie mnohé informácie o povolených rýchlostiach a podobne. V tomto bode záleží aký konkrétny výstupný formát z OSM dát sa zvolí. Niekedy sú formáty definované pomocou väčších celkov - cesta a následne nódy, ktoré pod ňu patria, k dispozícii je ale aj formát, kde sú nódy vyjadrené ako hlavné jednotky a ako prívlastok majú číslo cesty pod ktorú patria.

3.1.2 Porozumenie OSM formátu

OSM formát, podobne ako iné príbuzné dátové zdroje, popisujú jednotlivé objekty na zemskom povrchu a ich vlastnosti. Pre predstavu konkrétnej implementácie tohto systému v OSM – využíva sa systém párov kľúč = hodnota.[31] Následne môže ale byť hodnota aj komplexnejšou dátovou štruktúrou. Objekty sú v tejto dátovej reprezentácii popísané vždy troma základnými stavebnými prvkami viď obr.č.3.2:

- Node
- Way
- Relation

Použitím nódu sa popisujú objekty ako prechody pre chodcov, autobusové zastávky, menšie objekty alebo budovy, dopravné značky atď. Očakávateľne teda objekty pomerne nerozľahlé a s centrovanou povahou.
Cesta sa môže skladať z istého množstva bodov, kde je každý definovaný súradnicami a k tomu môže držať vlastnosti o povolenej rýchlosti, prípadne o jej type ako



Obr. 3.2: Principiálna ilustrácia prvkov typu *node*,*way* a *relation* vo formáte OSM [35]

cesta prvej, alebo druhej triedy. Cesta je pochopiteľne entita, ktorá je predmetom záujmu tejto práce a extrakcie dát z OSM boli vedené práve filtráciou dát od nódov a prvkov typu *relation* pre zachovanie iba entít pod typom *way*.
Relation je komplexný typ, ktorý je zvyčajne poskladaný z viacerých prvkov typu *way* alebo *node*. Príkladom je verejná zastávka, kde sú zastávky električiek, autobusov, prípadne aj taxíkov alebo metra na jednom mieste. Takáto entita potrebuje mať priestor na popis vlastností všetkých týchto druhov, a to jej umožňuje definícia *relation*.

3.1.3 Programové spracovanie

OSM dáta ľubovoľnej oblasti možno stiahnuť vo viacerých formátoch, kde každý má svoje nástroje na spracovanie, nástroje na vizualizáciu atď. Ide o následovné možnosti:

- .gpx
- .kml
- .geojson
- .osm
- .json

Pomocou jazyku Python možno .geojson formát čítať ako json file a pristupovať pomocou kľúčov k hodnotám ako v dátovom type *dictionary*. Na výpise č.3.1 možno vidieť danú dátovú štruktúru. Pod kľúčom “features” sa v liste nachádzajú všetky

prvky danej nadradenej entity. V tomto prípade je tam len jedna pre zjednodušenie. V tomto prvku sa nachádzajú 4 kľúče:

- "type"
- "properties"
- "geometry"
- "id"

V základe bude dôležitá hodnota pod kľúčom "geometry", ktorá vyjadruje typ daného prvku a jeho definíciu v podobe konkrétnych súradníc, ktoré daný prvok tvoria. Všeobecne pôjde v práci často o to vyfiltrovať prvky typu "LineString" a o odčítavanie hodnôt – súradníc, pod kľúčom "coordinates". Na základe nich sa navrhuje následná algoritmická práca. Bol prednesený spôsob ich extrakcie a teda prístupom k .geojson súboru, následnom načítaní napríklad pomocou knižnice json v Pythone a prístupom k hodnotám spôsobom akým sa v Pythone pracuje s dátovým typom dictionary. Pre tento moment nebude kľúčové daný obsah dát rozoberať viac dopodrobna.

Výpis 3.1: Ukážka OSM formátu v .geojson súbore. Pre širšiu vzorku k preskúmaniu formátu v tejto podobe viď príloha A adresár **Basemaps** - ľubovoľná podkladová mapa

```
{
    "timestamp": "2021−03−30T11:41:43Z",
    "features": [
    {
        "type": "Feature",
        "properties": {
            "highway": "secondary",
            "lanes": "3",
            "@user": "mmoessner",
        },
        "geometry": {
            "type": "LineString",
            "coordinates": [
            [
              9.1167114,
              48.9554555
            ],
            [
              9.1169333,
              48.9554569
            ]
          ]
        },
        "id": "way/4757910"
    }
  ]
}
```

3.2 Extrakcia dát

V tejto sekcii budú predstavené zdroje pre získanie dát z OSM databázy. Preberie sa aké sú medzi nimi rozdiely a akú funkcionalitu z nich možno vyťažiť. Všeobecne sa vždy čerpá z rovnakej databázy, tieto rozdielne cesty ale majú svoje limitácie alebo prídavné funkcie o ktorých je dobre vedieť.

3.2.1 Zdroje získania dát

OpenStreetMap.org Jedná sa o hlavný zdroj, kde možno nájsť aj prepojenia na ďalšie – iné cesty k získaniu dát z OSM geodatabázy. Zdroj limituje užívateľa na 50 000 nódov (zdroj je chybové hlásenie po skúške väčšej oblasti). Navyše neponúka žiadnu prídavnú funkcionalitu, skôr je tento zdroj použiteľný ako dostupný prehliadač mapy na OSM podklade.

overpass-turbo.eu Táto alternatíva je prakticky najvyhovujúcejšou, hlavne z dôvodu toho, že je najkomplexnejšia. Množstvo sťahovaných dát nie je obmedzené a poskytuje možnosť práce s dedikovaným jazykom pre definíciu toho aké prvky z OSM dátovej sady treba vyextrahovať. Viac o tomto jazyku v kapitole 3.2.2.

JOSM editor JOSM editor je v prvom rade nástrojom pre prispievateľov obsahu OSM máp. V porovnaní s jeho konkurentmi je veľmi detailný a poskytuje čo možno najviac nástrojov pre precíznu a profesionálnu prácu – mapovania dát do tejto otvorenej platformy [33]. V práci sa naň bude odkazovať, pretože je rýchlym a kvalitným vizualizačným nástrojom pre dáta, ktoré sa extrahujú z OSM databázy. Vie rýchlo a jednoducho aktivovať a deaktivovať konkrétne dátové typy na mape a poskytuje všetky detailné informácie o dátach, ktoré sú v ňom vizualizované.
Pridelený do tejto kapitoly je z dôvodu, že je taktiež metódou, ktorou možno stiahnuť ľubovoľné množstvo dát – bez limitu. Funguje v ňom tak isto dotazovací jazyk ako v overpass API a zároveň je schopný tieto stiahnuté dáta rovno zvizualizovať a teda poskytnúť všetky jeho funkcie rovno po stiahnutí. Vie to v mnohých prípadoch, napríklad extrahovania viacerých odlišných oblastí, ušetriť čas. Negatívom oproti overpass API je fakt, že tvorbu skriptu tohto dotazovacieho jazyka nesprevádza žiaden pomocník. Typy ktoré sa žiada odfiltrovať, prípadne zložky na pridanie bude treba manuálne vyhľadať napríklad z tejto databázy zložiek formátu OSM [31] a potom ich náležite zakomponovať do filtrovacieho skriptu.

3.2.2 Použitie dotazovacieho jazyka na predspracovanie dát

Na výpise č.3.2 je ukážka databázového dotazu pre overpass-turbo web, prípadne pre sťahovacieho klienta v JOSM editore. Je vytvorený na základe preskenovania možností OSM formátu [31]. Táto práca sa sústreďuje na prvky infraštruktúry po ktorých jazdí autobus. Ostatné nie sú primárne zaujímavé a teda sú zaradené medzi výnimkami. Tento skript teda na začiatku pridá všetky prvky pod kľúčom highway a následne sa zadávajú výnimky, ktoré v exporte nie sú žiadúce. Pod kľúčom highway sa nachádza množina minimálne toho čo je relevantné, a teda z nej sa ďalej filtruje. Ako možno vidieť, všetky filtrovacie hodnoty ako aj kľúč *highway* sú pod typom *way*. Ak v neskorších štádiách práce dôjde k opodstatneniu zaradenia napríklad autobusových zastávok, prechodov pre chodcov a podobne, nebude ich problém neskôr dodatočne zaradiť zmenou tohto skriptu. Konkrétne sa pridá aj typ "node" a následne sa vyfiltrujú opäť všetky nepotrebné hodnoty pre vybrané kľúče. Znova ale pôjde pravdepodobne o kľúč "highway" a výnimky v hodnotách v rámci tohto kľúča. Zároveň vo všetkých uvažovaných využitiach zatiaľ nedošlo na potrebu pracovať s prvkami typu node, pretože vo veci riešenia zadania práce nateraz nevidno hmatateľný úžitok. Potenciál môžu predstavovať v určitých nanovo vyšpecifikovaných nadstavbách nad rámec zadania.

Výpis 3.2: Ukážka použitého databázového dotazu v Overpass-XML. Pre textovú formu viď príloha A adresár **basemap_dep**, súbor query_example.txt

```
<osm-script output="json" timeout="25">
    <!-- gather results -->
        <union>
            <query type="way">
                <has-kv k="highway"/>
                <has-kv k="highway" modv="not" v="footway"/>
                <has-kv k="highway" modv="not" v="pedestrian"/>
                <has-kv k="highway" modv="not" v="steps"/>
                <has-kv k="highway" modv="not" v="path"/>
                <has-kv k="highway" modv="not" v="track"/>
                <has-kv k="highway" modv="not" v="cycleway"/>
                <has-kv k="highway" modv="not" v="raceway"/>
                <has-kv k="highway" modv="not" v="bridleway"/>
                <has-kv k="highway" modv="not" v="bus_stop"/>
                <has-kv k="highway" modv="not" v="construction"/>
                <has-kv k="highway" modv="not" v="corridor"/>
                <has-kv k="highway" modv="not" v="elevator"/>
                <has-kv k="highway" modv="not" v="living_street"/>
                <has-kv k="highway" modv="not" v="platform"/>
                <has-kv k="highway" modv="not" v="proposed"/>
                <has-kv k="highway" modv="not" v="service"/>
                <has-kv k="highway" modv="not" v="toll_gantry"/>
                <has-kv k="highway" modv="not" v="traffic_island"/>
                <has-kv k="highway" modv="not" v="virtual"/>
                <has-kv k="highway" modv="not" v="yes"/>
                <bbox-query {{bbox}}/>
            </query>
        </union>
    <!-- print results -->
    <print mode="meta"/>
    <recurse type="down"/>
    <print mode="meta" order="quadtile"/>
</osm-script>
```

4 Vybraná množina prejazdov mestom a datový model mesta na ktorom prebehne spracovanie tejto množiny

4.1 Dátový model mesta (podkladová mapa / basemap)

Pod dátovým modelom mesta sa bude rozumieť množina potrebných geografických prvkov prislúchajúcich určitej geografickej oblasti. Veľkosť je rozhodujúca, pretože bude ovplyvňovať dĺžku a pamäťovú náročnosť ako samotného jej spracovania, tak algoritmov, ktoré nad ňou budú vykonávať operácie. Pri extrakcií dát je teda vhodné zvoliť oblasť, ktorá bude akurát postačovať na spracovanie potrebných úsekov. Veľkosť tohto okna a teda veľkosť extrahovaného úseku sa volí za pomoci nástrojov, ktoré boli popísané v sekcii 3.2. Výstup popisovanej extrakcie si možno predstaviť ako dátovú vizualizáciu na obr.č. 4.1 s tým, že je to vizuál dátovej vzorky a nie obyčajný obrazový materiál. Každý zvizualizovaný bod na obrázku je reprezentovaný svojimi súradnicami a zhlukuje sa v dátových objektoch, ktoré boli popísané v sekcii 3.1.2.

Programový postprocessing podkladovej mapy

Postprocessingu podkladovej mapy sa venuje modul **basemap_creator.py**. Funkcionalita je oddelená v samostatnom module, pretože spracovať podkladovú mapu stačí jeden krát a následne ju možno používať a náležite pracovať s jej výstupmi bez nutnosti opakovania tohto iniciačného kroku. Vykoná teda súhrn potrebných operácií a výstupy uloží v **.p** a **.geojson** súboroch, ktoré budú k dispozícii napríklad aj hlavnému skriptu, ktorý bude pracovať už len s týmto výstupom. Operácie, ktoré teda nebudú musieť byť neustále opakované sú:

- zbavenie mapy prvkov typu “Point”, ktoré nie sú dôležité pre účely tejto práce a pomocou Overpass-XML skriptu v už spomínanom Overpass API nebolo jednoduchou cestou možné tieto prvky vyfiltrovať, preto k tomu došlo v tomto bode
- pridanie vlastného ID každému prvku na mape - nahliadnutím na výpis č.3.1 sa priradí ID každému slovníku v liste pod kľúčom s názvom “features”

Obr. 4.1: Ukážka podkladovej mapy. Jednotlivé vizuály (cesty) sú číselne vyjadrené prvky databázy - nódy pospájané v jednotlivých infraštruktúrnych prvkoch a pod. Zdroj: vyextrahované dáta pomocou overpass-turbo.eu oblasti Bietigheim-Bissingenu a jeho okolia

- získanie potrebných výstupov:
  1. všetky nódy podkladovej mapy
  2. iba nódy, ktoré sú súčasťou križovatiek
  3. doplnok ku križovatkovým nódom a teda nódy, ktoré budú nazývané ako unikátne
  4. vyfiltrovaná podkladová mapa
  5. podkladová mapa tvorená iba križovatkami

Body 1-3 reprezentujú skupiny nódov. V jazyku Python sú implementované v objekte o dátovom type list, v ktorom sú jednotlivé prvky taktiež objekty typu list o vzorci [zemepisná šírka, zemepisná dĺžka]. Body 4-5 reprezentujú dátové objekty vo forme OSM formátu, typovo totožné s pôvodnou extrahovanou mapou. Výsledné udržanie kompatibility tohto formátu je kľúčovou oblasťou tejto práce, pretože množstvo zmien sa vykonáva práve priamo na objektoch OSM dátového formátu.

OSM dátový formát je v jazyku Python, ako bolo spomenuté v sekcii 3.1.3, vyjadrený ako datový typ dictionary a je uložený ako .geojson file. Tieto datové objekty sa dajú načítať pomocou json enkódera a dekódera zo štandardnej knižnice jazyka Python.

Konkrétna vzorka podkladových máp pre prácu s pripravenými vstupnými súbormi

V zložke **Basemaps** sú pripravené vyexportované 3 podkladové mapy, ktoré oblasťou pokrývajú vybrané vstupné vzorky GNSS dát. V súbore **constants.py**, ktorý definuje potrebné vstupy do programu, je určené pre ktoré vstupné vzorky platia ktoré podkladové mapy. Podľa už spomínaného odporúčania je časovo výhodnejšie použiť menšiu ako väčšiu mapu, záleží okolo akých vstupných vzoriek sa bude pohybovať práca s programom. V dokumentácii v priečinku docs je odkaz na videotutoriál v ktorom je predstavené ako manipulovať so súborom constants.py. Doporučený postup je ten, že si užívateľ zvolí vstupnú vzorku a ozrejmí sa vo veci, ktorú podkladovú mapu potrebuje, následne spustí skript **basemap_creator.py** s relevantnou vstupnou podkladovou mapou. Tento skript poskytne vo výstupe potrebné náležitosti pre to, aby mohol program ďalej pokračovať a pracovať s touto podkladovou mapou v ďalších funkčných častiach programu viacmenej prostredníctvom funkcie **main.py**. Na všetky predpripravené vstupné GNSS dáta, ktoré budú rozobrané v ďalšej sekcii budú stačiť 3 podkladové mapy ako vybraná vzorka dátového modelu mesta zo zadania:

- Sachsenheim.geojson
- Klein_Bietig_Unterr.geojson
- Erlig_Tamm_Inger_Unterr.geojson

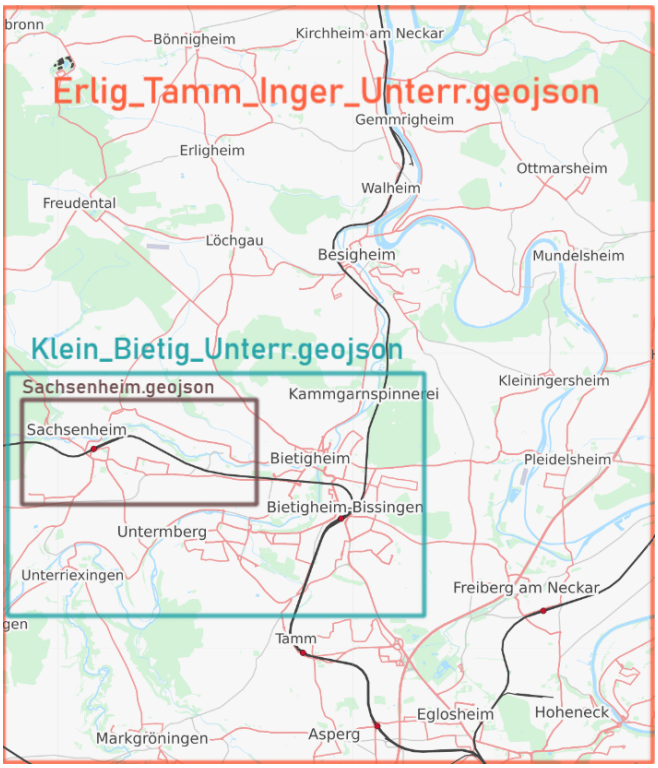
Obr.č. 4.2 tieto možnosti vizualizuje na reálnej mape pre pochopenie rozsahu a umiestnenia týchto dátových modelov. Zároveň pri porovnaní obrázkov č. 4.1 a č. 4.2 možno vidieť, že na prvom spomenutom je vizualizovaný dátový model mapy Klein_Bietig_Unterr.geojson.

4.2 Množina prejazdov mestom - dátový vstup

Formát a použité atribúty

V riešení navrhovanom touto prácou bol použitý pre všetky výpočty, normalizáciu a záverečné klastrovanie výčet zemepisných šírok a dĺžok. Riešenie sa tým pádom neobmedzuje na konkrétnu vstupnú vzorku zaznamenanú určitým konkrétnym zariadením. Prakticky sa očakáva, že sa bude môcť použiť na dáta získané akýmkoľvek zariadením, ktoré sníma výlučne polohu.

Jednoduchosťou tohto faktu je následne možné vytvárať si vlastné vstupné vzorky, pretože je jednoduché si takúto vstupnú sadu vzoriek aj nasimulovať-vygenerovať.



Obr. 4.2: Predpripravené dátové modely mesta - pre ilustráciu oblastí, ktorých sa jednotlivé súbory týkajú.

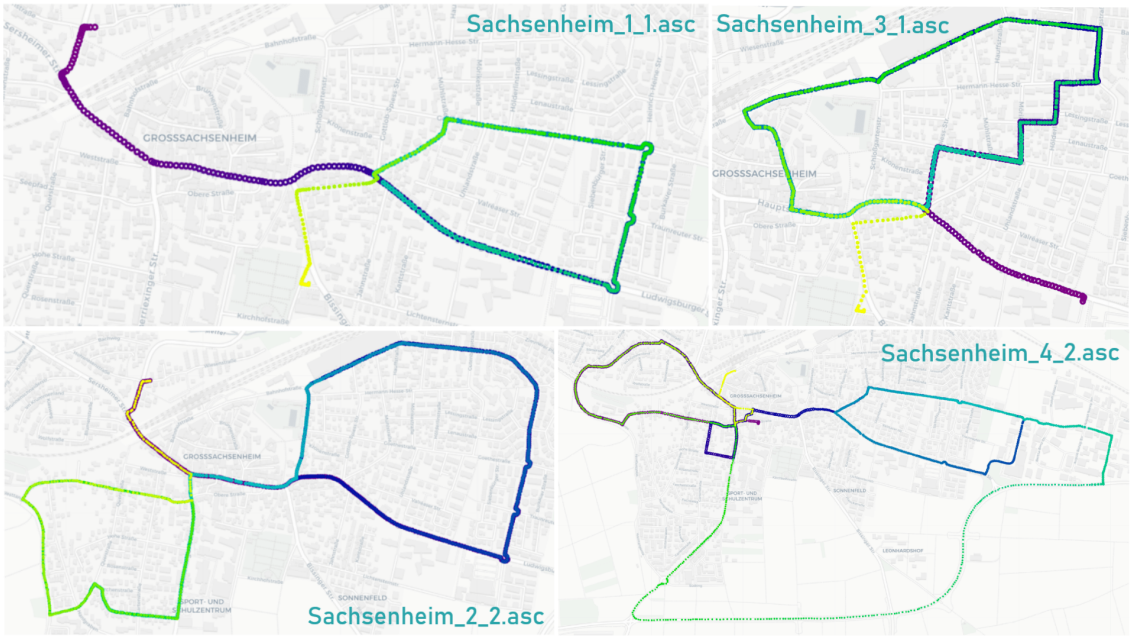
Ak si niekto bude chcieť generovať takéto vstupné dáta, najskôr v oblastiach riešenia v ktorých sa pohybuje táto práca narazí na funkcionalitu knižnice folium folium.LatLngPopup(), ktorá vráti súradnice kliknutého bodu (o tejto knižnici viac v sekcii 7.3). Každopádne nie je až také ľahké a priamočiare ich z tadiaľto ale získať a tak pre potenciálne ušetrenie času odporúčam využiť ucelenejší produkt tretej strany pre vygenerovanie vlastnej vstupnej vzorky GNSS dát (viacero možností - nie je nutné špecifikovať z dôvodu, že je to ľahko dohľadateľné).

Konkrétne vzorky vstupných dát

Brno Urban dataset Vstupné dáta na ktorých bol stavaný tento projekt boli spočiatku dáta z Brno Urban Datasetu [32]. Začiatky a veľká časť normalizačných tendencií vznikli práve za pomoci týchto dát na dátovom modele Brna. Využitá bola ich zložka s GNSS dátami, ktoré obsahujú nasnímané dráhy idúceho osobného vozidla, čo vyhovuje povahe trajektórií, ktoré požaduje tento projekt.

Data poskytnuté spoločnosťou Siemens Následne bolo riešenie budované na reálnych dátach poskytnutých spol. Siemens, ktoré boli nasnímané autobusmi, ktoré v danom čase akvizície dát obsluhovali skutočné linky v širšej oblasti Bietigheim-Bissingenu na sever od Stuttgartu.

Vlastné dáta simulujúce autobusové linky V snahe vyvíjané algoritmy a metódy čo najviac pozorovať a kontrolovať, som vytvoril aj vlastnú vzorku vstupných dát, ktorá simuluje správanie autobusu obsluhujúceho jednu alebo viacero autobusových liniek. Výhodou tohto súboru vstupných dráh je okrem očakávateľnej definície vstupnej dráhy aj presná definícia očakávaných autobusových liniek vyplývajúcich z konkrétnych vstupných setov. Na základe tohto faktu som v závere bol schopný automatizovať overovanie kvality nájdenia autobusovej linky na vstupných dátach. Malú ukážku vstupných dát v podobe mnou vygenerovaných vstupných GNSS dát možno vidieť na obr.č. 4.3, kde druhé číslo v názve indikuje počet vyskytujúcich sa autobusových liniek. Ako je možné si povšimnúť, všetky vzorky boli cielene vytvorené v Sachsenheime a teda všetky možno použiť pod rovnomennou podkladovou mapou **Sachsenheim.geojson**.



Obr. 4.3: Ukážka vlastných vstupných dát. Ilustrácia dráh, ktoré majú simulovať jazdu autobusom. Farba a veľkosť nódu sa mení s poradovým číslom nódu a teda s ubiehajúcim časom. Pomáha to hlavne vizuálnemu zorientovaniu sa v oblastiach prekrývajúcich sa nódov.

Overovanie a analýza vstupu

Všetky vzorky reálnych dát, ktoré boli pri riešení úlohy k dispozícii neobsahovali referenčnú množinu pomocou ktorej bolo možné overiť autenticitu výsledku. Bolo nutné si ju v počiatočných štádiách urobiť svojpomocne, prípadne vo veci dát od Siemensu bolo možné dohľadať skutočné linky a hľadať prieniky aspoň vizuálne. Relevantným bude testovanie kvality na vlastných dátach, kde som vopred definoval linky a bolo možné vecne pretestovať kvalitu výstupu.

Pre oddelené vzhliadnutie vstupných dát bol vytvorený skript **input_checker.py**. Súbor generuje iba vizualizáciu vstupných dát, spolu so súradnicami a časom, kedy boli nasnímané. Pre zobrazenie týchto dát je nutné prejsť myšou nad dané nódy. Je nastavený len na dáta od Siemensu, pretože sa jedná o najobsažnejšie dáta spomedzi všetkých zo vstupnej množiny. V prípade potreby vizualizácie ostatných súborov je vhodné spustiť priamo skript **main.py**, pretože jeho výstupná vizualizácia obsahuje aj samostatné zobrazenie vstupných dát. Neočakáva sa, že by tento skript bežal s ostatnými vstupnými súbormi tak dlho, aby pre nich bolo nutné vizualizáciu vstupných dát osamostatniť.

V snahe mať možnosť funkčne prefiltrovať vstupné dáta bol vytvorený skript **input_data_preprocessor.py**, ktorý prefiltruje vstupný súbor na základe daného časového úseku, s ktorým je potreba pracovať. Pre potreby tohto projektu bol spolu s už spomenutými skriptami a prístupmi preprocesing tejto úrovne postačujúci na to, aby bolo následne pohodlné ďalej pracovať so vstupnými súbormi.

5 Implementácia metód pre spracovanie množiny GNSS záznamov a ich klastrovania

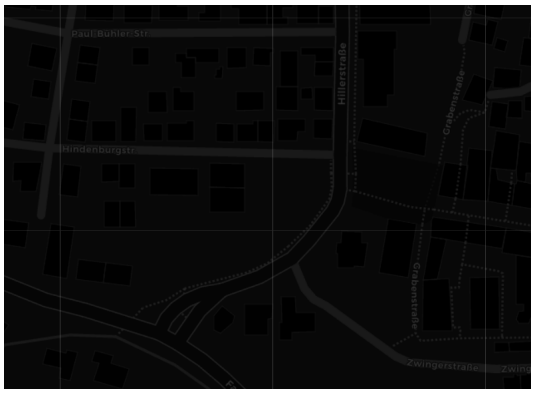
Táto kapitola prakticky popisuje hlavný pilier práce. Normalizácia a následné klastrovanie autobusových liniek sú dva hlavné ciele ktorých sa snaží projekt docieliť. V dvoch hlavných podkapitolách tejto kapitoly bude vysvetlené na akých princípoch stojí navrhnuté riešenie. Nadchádzajúce vysvetlenie bude miernym zovšeobecnením a zjednodušením krokov, ktoré boli skutočne implementované a teda pre dôkladnejšie porozumenie jednotlivých metód a prístupov je možnosť nahliadnutia na priložený kód riešenia. Bola vynaložená špecifická snaha o to, aby kód nasledoval štylistické konvencie PEP8 spolu s rozumným zreteľom na čitateľnosť. Kód sprevádza dokumentácia, ktorá taktiež dopomáha čitateľovi hlbšie a jasnejšie pochopiť jednotlivé funkcie, moduly v podobe docstringov a programový beh pomocou všadeprítomných komentárov.

Ďalšou potenciálnou pomôckou pre čitateľa kódu môžu byť aj unit testy v zložke **tests**, ktoré testujú funkcionalitu 14 základných funkcií tejto práce. Poskytujú možnosť na príklade pochopiť zámer danej testovanej funkcie. Každá funkcia ako aj modul, v rámci ktorého boli pôvodne definované, sú v tejto zložke pomenované pôvodným názvom s prídavkom predpony **test_** pred ich názvy - čo je koniec koncov v určitom bode implementácie nutnosť z dôvodu použitia balíka **unittest** zo štandardnej knižnice Pythonu.

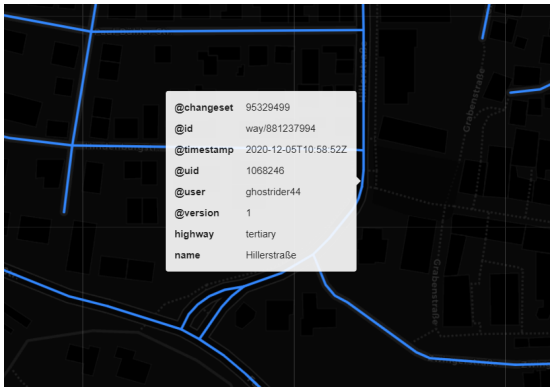
5.1 Implementácia metódy pre spracovanie GNSS záznamov - normalizácia

To, čo sa bude rozumieť pod normalizáciou je vysvetlené v sekcii 3.1. Pre zopakovanie - sú k dispozícii na vstupe GNSS dáta a na výstupe je treba získať túto dráhu reprezentovanú nódmi ktoré patria do OSM formátu a nesú so sebou taktiež relevantné dáta. Sekcia normalizácie, vysvetlená z programového hľadiska, bude v tejto podkapitole teda vysvetlená od bodu akvizície GNSS dát až po získanie normalizovanej formy vstupných GNSS dát.

V prvom kroku sa načítajú vstupné GNSS súradnice autobusovej jazdy. Ako bolo spomenuté v sekcii 1.1, pre dáta poskytnuté spol. Siemens to bude súbor údajov snímaných v jedno-sekundových intervaloch. Tie sa cez stĺpce zemepisnej šírky a dĺžky načítajú do jednotlivých dvojíc a zaradia na určenú pozíciu do listu so súradnicami



Obr. 5.1: Mapový raster - je dôležité si uvedomiť, že dáta sú ilustrované na obyčajnom rastri, ktorý hoci má svoju dynamiku na jednotlivých úrovniach zoomu, tak je to vždy obyčajný raster a s vizualizovanými dátami komunikuje len svoju polohu, aby na ňom boli dáta správne zvizualizované.



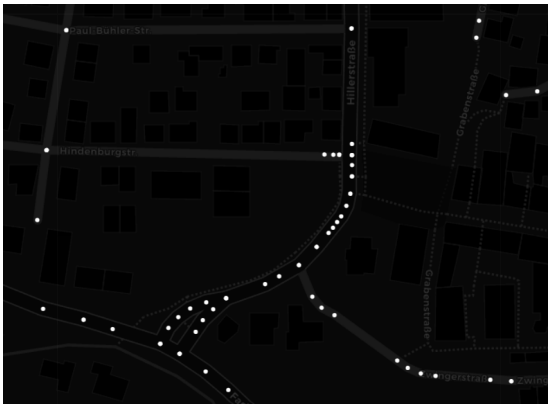
Obr. 5.2: Vizualizácia databázy OSM - reprezentovaná ako pozičným vyjadrením, tak datovým balíčkom, ktorý ho sprevádza. Predstavuje dáta nad ktorými možno algoritmizovať.

s ktorým sa bude narábať v priebehu celej práce. V aktuálnom stave už sú v procese main funkcie načítané náležitosti podkladovej mapy ako bolo popísané v sekcii 4.1.

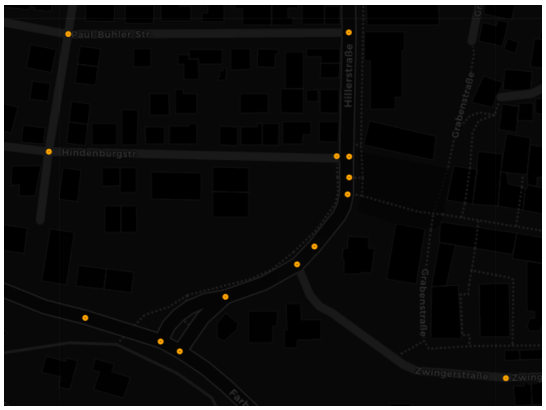
5.1.1 Vizualizácia princípu normalizácie a pochopenie použitých dátových objektov

Je dôležité pochopiť rozdiel medzi datovou reprezentáciou geografických prvkov a mapovým obrazovým podkladom. Na obr.č.5.1 sa nachádza kus mapy, každopádne z programátorského hľadiska ju možno vnímať iba ako obyčajný vizuál, z ktorého nemožno čítať názov ulice a ani jej polohu. Je to v úvodzovkách odroda obyčajného rastru, kde je poznať napríklad hodnotu RGB zložiek jednotlivých pixelov - komplexnejšia abstrakcia nie je k dispozícii.

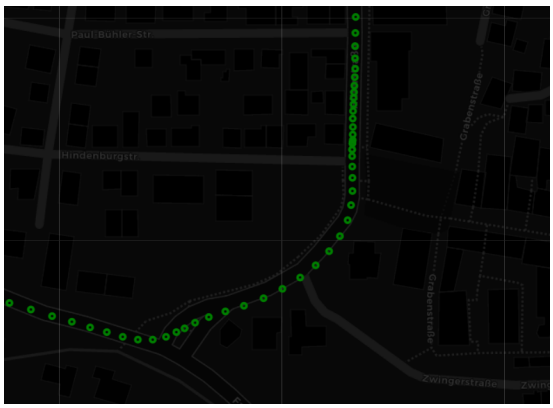
Na to, aby sa nad mapou a jej obsahom dali vykonávať operácie, sú nutné geografické dáta reprezentujúce daný vizuál. Ako je už v tomto bode jasné, projekt využíva databázu OSM. Tieto dáta zvizualizované na totožnom mapovom úseku možno vidieť na obr.č.5.2. Okrem vizuálu databázy je v strede obrázku zobrazený dátový balíček úseku na ktorý ukazuje. Takáto originálna inštancia dátového balíčku náleží ľubovoľnému úseku tejto dráhy.



Obr. 5.3: Podkladová datová mapa vyjadrená vo všetkých jej nódoch.



Obr. 5.4: Podkladová datová mapa vyjadrená iba v jej križovatkových nódoch.



Obr. 5.5: Vizualizácia vstupných GNSS dát.

Nódy, ktoré sa spájajú do celkov, tvoria cesty a iné inštancie OSM formátu. Pri nahliadnutí na obr.č.5.3 možno vidieť podkladovú mapu tentokrát reprezentovanú v nódoch. Pre objasnenie výstupu z **basemap_creator.py** a teda vstupného setu súbor pre **main.py** je nutné spracovať a vytriediť nódy, ktoré patria križovatkám. Podmienka ich detekcie je fakt, že sú súčasťou viacerých odlišných ciest. Ich vizualizáciu možno vidieť na obr.č.5.4

Vizuály do tohto bodu popisovali výstup z **basemap_creator.py**. Pre doplnenie vstupnej vzorky s ktorou sa pracuje a nad ktorou začne prebiehať výpočet je nutné objasniť samotné vstupné GNSS dáta. Tie sú na diskutovanom mapovom úseku zvizualizované na obr.č.5.5. Zvizualizované dáta sú reálnou vzorkou prechádzajúceho autobusu. Očakávateľne vzorka dokonalo nepodlieha dráhe po ktorej autobus išiel z dôvodu nepresností snímajúceho zariadenia a z dôvodu princípu podľa ktorého táto akvizícia nastáva - dochádza k variantnému DOP, prípadne nedokonalo

skorelovaným odrazom signálov od budov, nestabilnému počtu zachytených satelitov atď. Pre viac informácii viď sekcia 1.1.

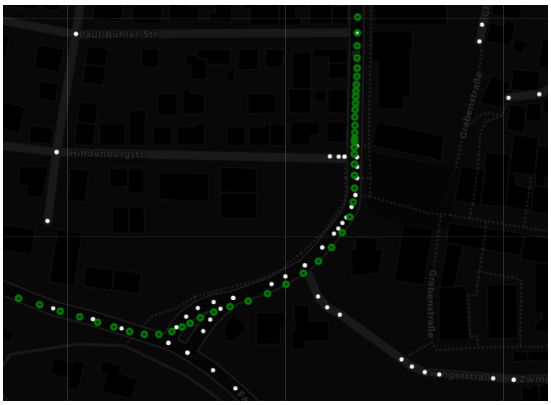
Pre zjednodušenie stavu v ktorom sa skript v počiatku po načítaní GNSS dát nachádza, je vhodné nahliadnuť na obr.č.5.6. Vo všeobecnosti je k dispozícii podkladová mapa a vstupné GNSS dáta. Stav do ktorého je potrebné sa dostať popisuje obr.č.5.8 - najmä časť modelu vizualizovaná modrou farbou. Sú to nódy popisujúce vstupnú GNSS dráhu, obsahovo tvorené nódmi podkladovej mapy, čím naberajú všetky výhody ktoré prináša OSM formát - **ako pozičné, tak dátové.** Pozičné výhody - normalizovaných nódov je menej a dokonalo opisujú cestu po ktorej autobus prechádzal. Dátové výhody - k normalizovaným dátam sa pripája celý OSM dátový balíček a prakticky nedochádza ku žiadnej strate dát zo strany GNSS dát. Ideálny výstup normalizácie vidieť na obr.č.5.9.

5.1.2 Programová implementácia normalizácie

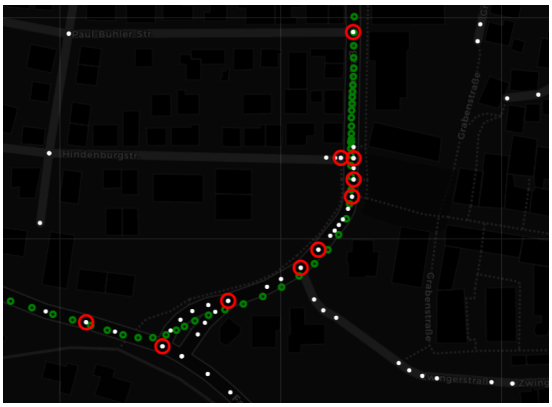
Získanie križovatkovej reprezentácie vstupu

Funkcionalita vychádza z funkcie **get_input_crossroad_representation** modulu refactor_functions.py. Výstup o ktorý sa snaží táto funkcia je zobrazený na obr.č.5.7 červenou farbou. Snaha je teda získať OSM formát vstupu zahŕňajúci iba križovatky. Otázkou je prečo sa vôbec zvolil taký prístup a načo je dobré začať snahou o získanie práve tejto skupiny nódov. Je to napríklad z dôvodu rýchlosti algoritmu (dôvodov je viac, rozoberú sa postupne). Prvotný spôsob, ktorý bol naimplementovaný, bol založený na princípe, kedy sa priradí každému vstupnému GNSS bodu ten najbližší z podkladovej mapy. Inými slovami - vyráta sa vzdialenosť vstupnej GNSS súradnice od každého nódu podkladovej mapy a vstupný GNSS bod sa priradí tomu nódu z OSM formátu, ktorý zaznamenal najmenšiu vzdialenosť. Tým sa získa postupnosť ktorá prakticky reprezentuje celkový výstup normalizácie.

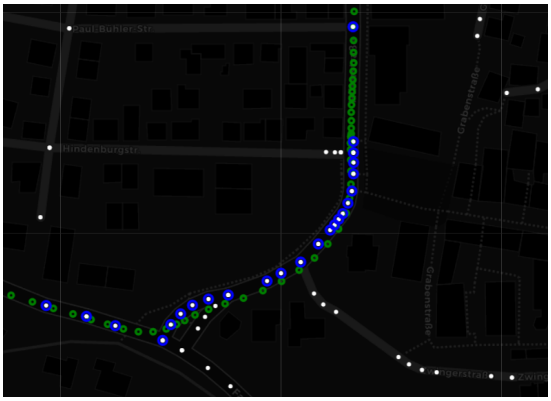
Čo sa týka už spomínanej rýchlosti - je to komputačne veľmi náročné a teda algoritmus je pomalý, v Landauovej notácii je tento úkon rovný $O(n^2)$, celkovým minimom sú 2 násobenia a jedna odmocnina v každej iterácii, pričom či už násobenie alebo **sqrt** funkcia z knižnice **math** by z ich programovej implementácie mali mať notáciu $O(1)$, takže samé o sebe neovplyvnia pôvodne stanovenú notáciu komplexnosti (suma jednotlivých komplexností sa prakticky rovná najväčšej z nich na rozdiel od násobenia, kedy sa komplexnosti vo výsledku násobia) [37]. Je vhodné v tomto momente pripomenúť poznámku ktorá bola uvedená k veľkosti podkladovej mapy - jej veľkosť priamou úmerou ovplyvňuje počet nódov a tento počet exponenciálne o miere druhej mocniny ovplyvňuje dĺžku behu normalizačného algoritmu.



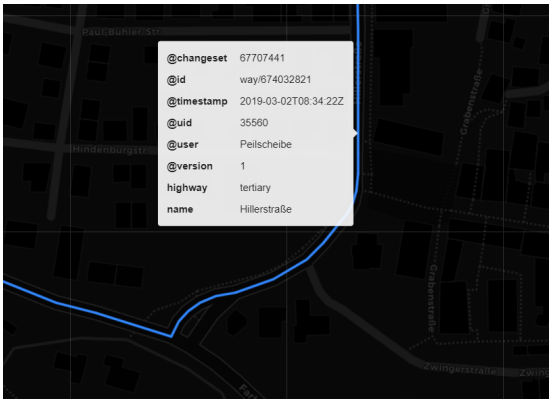
Obr. 5.6: Stav po načítaní GNSS dát vo funkcii main.py. Ilustrované sú vstupné GNSS dáta zelenou a všetky nódy podkladovej mapy bielou.



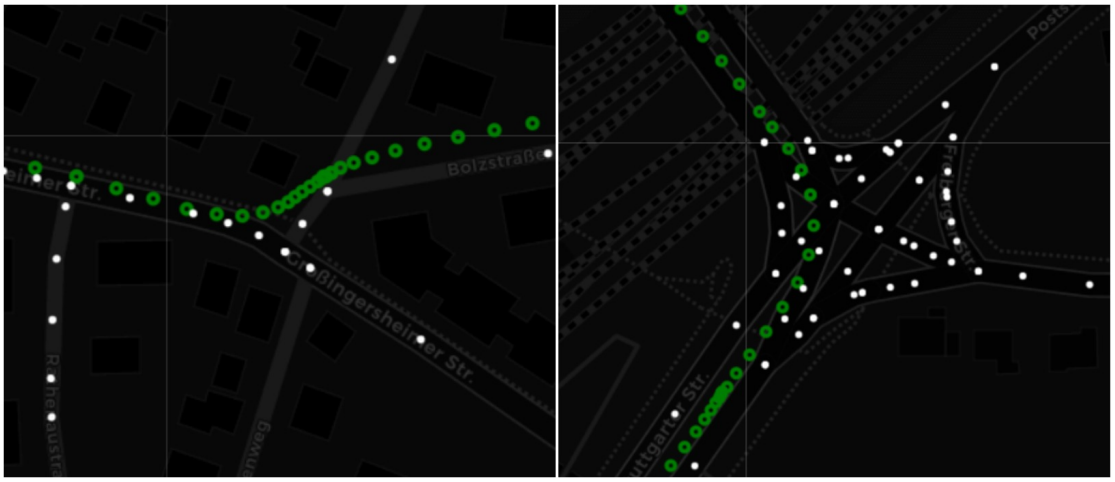
Obr. 5.7: V procese získania ideálneho výstupu je priebežným krokom získanie križovatkovej reprezentácie vstupu. Tá je zvýraznená červenou farbou - sú to vybrané nódy podkladovej mapy, ktoré všetky patria minimálne dvom rôznym infraštruktúrnym prvkom.



Obr. 5.8: Dáta ku ktorým je potreba sa dostať vo veci normalizácie - modrou farbou. Sú to nódy podkladovej mapy, ktoré vyjadrujú celú informáciu vstupných GNSS dát (zelenou). Teda na základe vstupných GNSS dát treba vybrať správne nódy podkladovej mapy.



Obr. 5.9: Ideálny výstup normalizácie - plný OSM formát v tvare vstupných GNSS dát, resp. vychádzajúci z nich. Možno ho docieliť najprv získaním vhodných nódov podkladovej mapy a následným vyvodením uceleného formátu danej formy.



Obr. 5.10: Chyba prístupu hľadania najbližších nódov - ilustrácia potenciálne veľkej nekonzistencie vstupných dát. GNSS dáta (zelenou) nepopisujú dostatočne jasne infraštruktúry, ktoré reprezentujú. Na ľavom obrázku je GNSS dráha značne vzdialená od križovatky ktorou objekt prechádzal. Na obrázku napravo je miestami ťažšie identifikovať po ktorej ceste objekt skutočne išiel.

Ďalšia verzia snahy o implementáciu normalizácie viedla k uskutočneniu rovnakého algoritmu iba nad nódmi, ktoré sú križovatkové. Stálo to na premise, že odstráni problém nízkej rýchlosti - čo sa do značnej miery podarilo. Následne bolo treba otestovať výslednú kvalitu, ktorá nespĺňala očakávania. Schyľuje sa ku problému, ktorý súvisel už s predchádzajúcim návrhom - kde okrem problému rýchlosti taktiež figuroval aj tento problém, ktorý vidieť na obr.č.5.10. Pri určitej nadmernejšej nepresnosti vstupných GNSS dát následne nebude možné dosiahnuť na určité potenciálne veľmi kruciálne nódy - či už križovatkové alebo nekrižovatkové. To môže znamenať veľký problém, lebo toto riešenie je založené na predpoklade, že sa využije pozičná výhoda OSM formátu a teda práve tieto hraničné križovatkové nódy je potreba zachytiť aby bol výsledok zaujímavý.

Tieto uvedomenia nakoniec vyústili do implementácie ďalšieho riešenia, ktoré stanovuje pevnú hranicu vzdialenosti v rámci ktorej sa zoberú všetky nódy daného okruhu v úvahu. Každý vstupný GNSS nód teda zahrnie všetky OSM nódy napríklad v okruhu 10 metrov. Zároveň bude toto riešenie v tomto kroku aplikované iba na križovatkové nódy a takto v zásade vznikne križovatková reprezentácia vstupu v OSM formáte. Tým, že je potreba zachytiť križovatky, bolo vhodné brať v úvahu fakt, že autobus ide okolo križovatiek pri zníženej rýchlosti. Táto potenciálna citlivosť na okruh 10 metrov by mohla byť vyššia pri nižšej rýchlosti a nižšia pri vyššej rýchlosti. Algoritmus je teda v závere aj dvojstupňovo naprahovaný podľa zvoleného prahu v

súbore konštánt **constants.py** pod názvom LAST_COORDINATE_DISTANCE. Pre predstavu bude stanovená hodnota o veľkosti 0.00005 predstavovať to, že ak posledný zaznamenaný GNSS bod je vo vzdialenosti viac ako 5.5 metrov od aktuálneho, znamená to, že rýchlosť je vyššia ako 5.5 metrov za sekundu (20 km/h) a nastav hranicu citlivosti na 10 metrov. Ak bude vzdialenosť medzi dvoma po sebe idúcimi GNSS bodmi menšia ako 5.5 metrov, znamená to, že autobus ide rýchlosťou menšou ako 5.5 m/s (zvolená rozumná hodnota) a teda prepni citlivosť okruhu na 15 metrov. Obe tieto hodnoty sú taktiež v súbore **constants.py** pod názvami SLOWSPEED_XROAD_DISTANCE a NORMALSPEED_XROAD_DISTANCE. Samozrejme sa ráta s tým, že vstupné súradnice sú zaznamenávané v 1-sekundových intervaloch, čo pre reálne vstupné vzorky projektu platí.

Po získaní výstupu z doteraz popísaného procesu je nutné preiterovať celú množinu a zbaviť sa akýchkoľvek zbytočných duplikátov. Vyfiltrovaný zoznam nódov je treba následne použiť pre vhodnú filtráciu podkladovej mapy, aby zo zoznamu (listu) nódov vznikol opäť OSM formát, pretože bez toho výstup nebude obsahovať dátovú zložku OSM formátu - využil by iba pozičný benefit, bez dátového benefitu. Týmto sa docieli popisovaný očakávaný výstup ako na obr.č.5.7 samozrejme aj s OSM formátom, ktorý už bol viac krát vizualizovaný, akurát by obsahoval formu definovanú križovatkovými nódmi.

Získanie hladkej normalizácie pomocou križovatkovej reprezentácie vstupu

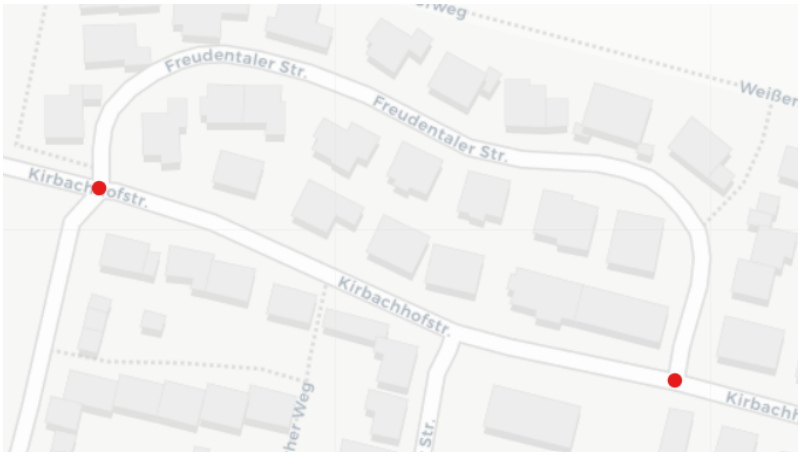
Funkcionalita vychádza z funkcie **get_input_fine_representation** modulu refactor_functions.py. Princíp získania čo najhladšej finálnej reprezentácie (v skriptoch označovaná pod prívlastkom **fine**), spočíva v získaní hladkej reprezentácie dvoma spôsobmi a v následnom porovnaní týchto dvoch vytvorených reprezentácií. Pôjde o to, že princíp nájdenia najbližších možných nódov, ktorý už bol popisovaný aj s jeho nedostatkami v minulej sekcii, je v skutočnosti v istom ohľade veľmi presný a vykazoval čo do jednotlivých objemov daných štruktúr dobré pokrytie. Zároveň v tomto bode vývoja bola k dispozícii križovatková reprezentácia vstupu. Otázkou bolo, ako ich vhodne spojiť a získať benefity oboch reprezentácií a zároveň udržať svižný chod algoritmu a nepridávať parazitnú komplexnosť.

Prvou otázkou teda je, ako vôbec z križovatkovej reprezentácie získať hladkú reprezentáciu pre dopracovanie sa k prvej spomínanej alternatíve z dvoch. Ideálne by bolo do tejto reprezentácie pridať celú cestu v rámci ktorej boli identifikované 2 križovatky. Následovalo teda vytvorenie algoritmu, ktorý pridá všetky cesty s 2 a viac identifikovanými križovatkovými nódmi a ostatné z reprezentácie vyhodí. Oča-

kávateľne sa po implementácii zistilo, že to zachytáva celé cesty ale potrebné sú len úseky medzi týmto križovatkovými nódmi. Po pozornejšom zamyslení je zrejmé, že identifikovaných križovatiek mohlo byť v celkovom počte na jednej ceste aj 5. Spôsob ďalšej filtrácie teda musí ísť z jedného kraja cesty, vyhodiť všetky nódy až do najbližšieho križovatkového, následne prejsť na druhý koniec tohto zoznamu nódov a mazať týmto spôsobom nódy z druhej strany. Túto funkcionalitu v rámci funkcie **get_input_fine_representation** vykonáva funkcia **get_fine_indata**. Odporúčam si prezrieť jej testovací príklad v prípade, že funkcionalita nie je jasná. Nachádza sa v zložke **tests** pod modulom **test_postx_functions.py**, očakávateľne názov testovacieho prípadu je rovnomenný s názvom funkcie s predponou **test_**.

Príklad chyby ktorá nastane v tejto reprezentácii je načrtnutá na obr.č. 5.11. Je jedno po ktorej z ulíc - Freudentaler Str. alebo Kirbachhofstr. autobus prešiel, tento algoritmus vždy započíta obe alternatívy. V tomto bode je nutné implementovať spôsob, ktorým sa odlíšia skutočne prejdené medzi-križovatkové úseky od neprejdených. Tu prichádza obrovská prídavná hodnota už spomínaného prístupu, kedy ku každej vstupnej GNSS polohe možno priradiť najbližší bod určitého listu nódov. Z prvej diskutovanej reprezentácie sa teda extrahujú nódy a vytvorí sa nový zoznam nódov, ktorý každej GNSS polohe priradí jeden nód z výstupu reprezentácie, ktorá má chybu diskutovanú v úvode tohto odstavca. Tento prístup sa všeobecne v tomto projekte označuje pod prívlastkom **each**. Po získaní nového zoznamu ho treba zbaviť duplikátov a pomocou týchto nódov vytvoriť mapu OSM formátu na báze novo-získaných nódov. Týmto sa vytvorila druhá alternatíva hladkej reprezentácie. Aktuálne sú teda implementované 2 reprezentácie v ktorých má každá svoju špecifickú hlavnú chybu, ale vzájomne sa vhodne doplňujú. V ďalšom odstavci sa bude diskutovať spôsob, ktorým sa docieli extrakcia benefitu oboch týchto reprezentácií.

K dispozícii sú v tomto bode dve reprezentácie - prvá vychádzajúca z križovatkových nódov a doplnení všetkých nekrižovatkových nódov v najširšom možnom rozpätí týchto platných križovatiek. A druhá, ktorá poňala výstup prvej spomenutej ako pomyselnú podkladovú mapu nad ktorou každému vstupnému GNSS bodu priradila ekvivalent z tejto pomyselnej podkladovej mapy. Prvá reprezentácia obsahuje úseky ciest, ktoré tam nemusia patriť a druhá reprezentácia má problém s detekciou odľahlejších nódov - akými sú napríklad nódy v ostrých križovatkách. Výsledným riešením na konci ktorého sa docieli výsledná hladká reprezentácia vstupu a teda normalizácia vstupných dát, je vhodné porovnanie týchto dvoch reprezentácií za účelom eliminácie chýb.



Obr. 5.11: Príklad kedy zlyhá jedna z implementovaných alternatív pre hladkú normalizáciu - medzi dvomi križovatkovými nódmi sú 2 rôzne cesty. Z toho vyplýva, že bez ďalšej dodatočnej informácie nie je jasné, ktorou autobus prechádzal. Tieto dodatočné informácie môžu byť informácie o ďalších polohách, jednosmernosti cesty a pod. V riešení tohto projektu sa ďalej pracuje s informáciou o ďalších konkrétnych polohách autobusu.

Od druhej reprezentácie sa čaká, že jej nódy chýbajú len výnimočne ale prakticky objemovo pojme signifikantné percento relevantných nódov. Prvá reprezentácia spĺňa do kvality to, čo má, potrebuje iba eliminovať svoju hlavnú chybu. Riešenie je zaimplementované vo funkcii **feature_deletion_matchwise**. Ich mapy - ich stelesnenia v OSM formáte sa budú porovnávať po ich jednotlivých prvkoch. Ekvivalencia prvkov sa testuje na základe priradených ID čísel zo sekcie 4.1. Funkcia na vstupe očakáva percentuálne vyjadrenie pod ktoré užívateľ nechce zájsť čo sa týka nezhody daných cestných entít. Prakticky je to len vyjadrenie miery, podľa ktorej sa musia dané dvojice prvkov podobať, aby bol daný prvok, ktorý oba reprezentujú, vo výslednej mape zachovaný.
Implementované je to na rozdiele celkového počtu nódov daných prvkov. Očakáva sa takmer úplné prekrytie alebo takmer žiadne. Zjednodušením tohto princípu (konkrétne prvky verzus porovnanie jednotlivých súm) sa zlepšila rýchlosť algoritmu.

Výstupom po aplikácií spomenutých metód je výsledná normalizácia vstupu. Do ďalších častí projektu poskytuje formy tohto výsledku ako v OSM formáte, tak vo forme listu všetkých nódov tejto mapy spolu s listom križovatkových nódov. Kvalita implementácie normalizácie bude otestovaná v sekcii 6.1.

5.2 Implementácia metód pre klastrovanie záznamov

5.2.1 Získanie potrebného vstupu

Prvorado je nutné zaistiť vstupné dependencie pre proces klastrovania. Menovite ide o získanie normalizovanej sekvencie ako ekvivalentu vstupných dát a druhá vec je výstup transformácie očakávaného rozpätia autobusových liniek. O tento proces sa stará funkcia **track_identification_dependencies** modulu refactor_functions.py.

Získanie normalizovanej sekvencie

V rámci normalizácie sa dbá o priebežné vymazanie duplikátov. Proces je schválne nastavený tak, že sa stráca informácia o opakovanom prejdení určitého úseku. A je to pochopiteľné, normalizácia má odfiltrovať podkladovú mapu na základe GNSS inputu, poskytnúť ideálne dáta na vizualizácie a podklad pre ďalšie výpočty nad cieľovou množinou. Pochopiteľne je pre identifikáciu autobusových liniek potreba pracovať so skutočnou sekvenciou a opakované úseky sú kľúčové. O to sa opäť postará už raz použitá funkcia **get_nodes_duplicates**. V jej prvom volaní sa vytvorila križovatková reprezentácia inputu a funkcia použila križovatkovú podkladovú mapu. Tentokrát bude podkladovou mapou hladký výstup normalizácie. Ten už je obohatený procesmi predchádzajúcich sekcií a znovu-spustením tohto algoritmu za stanovených podmienok skutočne nedochádza ku strate žiadneho nadobudnutého benefitu a na výstupe úraduje žiadaná sekvencia.

Táto sekvencia má rôzne typy duplikátov. V predchádzajúcich častiach sa vymazali vždy úplne všetky. Tentokrát je potreba vymazať všetky až na jeden typ duplikátu. Je potreba ponechať typ duplikátu, ktorý prináša informáciu o opakujúcom sa prejazde úsekov. Následne je potreba sa zbaviť duplikátov, ktoré sú po sebe idúce a duplikátov, ktoré sú fluktujúce a premenné, ale iba v určitej definovanej skupine prvkov. Tieto dve skupiny parazitných duplikátov rieši funkcia **consequtive_repetitions** z rovnomenného modulu. Pripomínam možnosť nahliadnutia na unit testy, kde je na príklade znázornená jej funkcionalita.

Nutnosť implementácie tohto riešenia vyplýva z povahy reálnych dát. Jednoznačne by na generovaných dátach nebolo nutné riešiť problémy tohto typu. Ide o prípady kedy autobus stojí na červenej alebo stojí zaparkovaný na stanici a má zapnutý snímač polohy. Pri státí mal údajne generovať absolútne totožné súradnice polohy, každopádne vyskytovali sa prípady kedy prepínal medzi X súradnicami, čo často krát absolútne rozhodilo všetky následovné algoritmy, ktoré sledujú repetitívne tendencie vstupu. Práve kvôli tomu je parameter množstva potenciálnych opakujúcich

sa súradníc otvorený a zadaný ako vstupný argument tejto súvisiacej funkcie. Z pochopiteľných dôvodov je dobré držať toto číslo čím menšie, pretože značí okno v ktorom nebude snímať akékoľvek opakujúce sa súradnice. Príliš nízke číslo môže otvoriť dvere popisovanému problému - je dobre vnímať, že táto možnosť existuje a poznať naimplementované riešenie.
Po tejto filtrácii je k dispozícii normalizovaná sekvencia vo finálnej forme, pripravená na ďalšie spracovanie.

Transformácia očakávaného rozpätia autobusových liniek

Nadchádzajúce hľadanie autobusových liniek funguje na princípe v rámci ktorého sa vopred stanoví rozsah autobusových liniek, ktoré sa majú nájsť. Jednoznačne je ideálne, ak je tento rozsah určený čo najpresnejšie, pretože to v mnohých prípadoch zrýchli beh navrhnutých metód, tak isto možno potenciálne očakávať lepšie výsledky, záleží ako veľmi široký interval sa zvolí. Definuje sa v súbore **constants.py** pomocou konštánt TRACK_LENGTH_START a TRACK_LENGTH_FINISH. Samozrejme je v poriadku a riešenie bude fungovať aj na extrémne široký interval, ktorý by mohol fungovať zovšeobecnene na akékoľvek vstupné dáta. Myšlienka za touto implementáciou je, že ak očakávané linky majú stanovené trvanie v určitom rozmedzí, je treba toto rozmedzie zapísať do spomínaných konštánt. V rámci reálnych vstupných dát na ktorých bolo testované riešenie sa do nich zapísali hodnoty 1000 a 3600 $[s]$. Čo znamená, že hľadané linky mali od cca 16 do 60 minút. V rámci procesu získania potrebného vstupu sa koná aj transformácia zadaného času na relevantný počet nódov ktorých sa to bude týkať. Prepočet vychádza z pomeru celkového počtu GNSS súradníc a celkového počtu normalizovaných nódov v sekvencii.

V tomto bode skript disponuje sekvenciou normalizovaných vstupných GNSS dát a prepočítaným rozsahom veľkostí prípustných liniek. To sú zároveň všetky vstupy ktoré bude potrebovať funkcia, ktorá vyhľadá konkrétne autobusové linky. Táto funkcia, prípadne jej alternatívy, budú popísané v ďalšej sekcii.

5.2.2 Konkrétne metódy a prístup ku klastrovaniu liniek

Výčet a definícia jednotlivých algoritmov pre hľadanie liniek sa nachádza v súbore **patternfinding_functions.py**. Volané sú z **main.py** pomocou referencie určenej v **constants.py**. V skratke sa prakticky vytvoril slovník referencií na jednotlivé funkcie v main.py a v súbore konštánt sa definuje hodnota kľúča pomocou ktorého sa zvolí referencia, ktorá sa na inkriminovanom mieste spustí so zadanými vstupnými parametrami. Prístup bol možný z dôvodu zachovania rovnakých argumentov pre všetky hlavné funkcie z modulu **patternfinding_functions.py**.

Otázku toho, ako porovnať dva úseky a ohodnotiť ich podobnosť riešia konkrétne komparačné algoritmy, ktoré sú vysvetlené v sekcii 2. Dôležité sú metódy a prístupy vďaka ktorým sa ku daným dvom úsekom možno dostať. Tieto prístupy ktoré vedú ku klastrovaniu liniek budú otestované v sekcii 6.2 a v krátkosti budú vysvetlené v tejto sekcii.

Vo funkciách definovaných ako názov_komparačnej_metódy**1** ide o preiterovanie celej vstupnej vzorky so všetkými možnými úsekmi o definovanej dĺžke a výber jedinej dvojice, ktorá v danej dĺžke zaznamenala najlepší koeficient daného komparačného algoritmu. Tá sa posiela vo výstupe na ďalšie spracovanie a vizualizáciu. Hľadaná dĺžka je definovaná ako spodná hranica intervalu v ktorom sa majú hľadať linky. Dôvodom je izolácia vplyvov pre testovanie jednotlivých čiastkových funkčných častí. Ak je nutnosť otestovať vplyv komparačnej metódy, je potreba aby bola čo najviac odizolovaná a aby okolité pridružené algoritmy trvali čo najkratšie a všeobecne mali čo najmenší vplyv. Podobne ak je nutnosť testovať prístup k nájdeniu dvoch vhodných úsekov pre porovnanie, je ideálne izolovať vplyv komparačnej metódy, aby do procesu nezavádzala svoj v tomto prípade parazitný vplyv. Viac k tejto téme v testovacej sekcii 6.2.

Vo funkciách definovaných ako názov_komparačnej_metódy**2** ide o zmenu v ktorej sa rieši rýchlostný benefit a sekundárna reakcia na kvalitu pri danej komparačnej metóde. Komparačné metódy sú volané až v momente, kedy sa v sekvencii narazí na totožný nód.

Funkcia **gestalt4** sa snaží reprezentovať prístup v ktorom budú mať prioritu dlhšie dráhy v rozpätí povolených koeficientov od dráh kratších v stanovenom rozsahu.

Funkcia **gestalt5** reprezentuje prístup v ktorom sa testuje minimálny možný úsek po spomínanej zhode nódov. Pri dosiahnutí minimálnej stanovenej zhody, ide testovať úsek počínajúc daným nódom. Iteruje od minimálneho úseku až po najdlhší možný a proces preruší v prípade, že v testovanej dvojici nastane narušenie minimálneho prípustného koeficientu. Premisa nad týmto prístupom je tá, že k prerušeniu dôjde až v bode, v ktorom bude skutočne obsiahnutá celá autobusová linka.

Funkcia **gestalt6** stanovuje minimálnu hranicu koeficientu, iteruje od najväčšej možnej veľkosti linky a využíva nájdenie dráh pre budúce skipovanie ich okolitých nódov. Okno ktoré preskakuje musí byť vhodne zvolené v závislosti od veľkosti danej nájdenej dráhy okolo ktorej sa okno vytvára, zároveň ale nemôže zamedziť nájdeniu novej dráhy pred touto linkou. Rezerva pred nájdenou linkou musí byť kompromisom medzi veľkosťou tejto linky a nesmie prekročiť veľkosť minimálnej možnej dĺžky

linky. V tejto testovacej verzii princípu je zvolená tretina veľkosti nájdenej linky, pretože je to pomerovo veľmi blízko reálnemu predpokladu toho, kde sa hranične táto rezerva bude pohybovať. Predstavuje to teda zvolený kompromis pre testovanie popisovaného princípu.

Tieto princípy boli vybrané k pretestovaniu. Na základe kvality ich výstupu na konkrétnych testovacích mapách sa im pridelili body a z jednotlivých princípov sa vytvorili finálne funkcie, ktoré boli komplexné a každá z nich mala potenciál vykazovať zmysluplný výstup. Týmito funkciami sú **gestalt7** a **gestalt8** - obe s verziou kedy volajú svoj komparačný algoritmus **ratio()** a **quick_ratio()**, **levenshtein3**, **levenshtein5** a **manual_trackfinder3**. Celkovo je to 7 finálnych prístupov, ktorých funkcionalita je poskladaná z princípov, ktoré boli popísané v predchádzajúcich paragrafoch. Výsledky tohto testovania a pridružené slovo k tejto téme je v sekcii 6.2.

6 Test výkonu navrhnutého riešenia a nadväzujúca diskusia nad výsledkom

Kapitola sa venuje výstupom jednotlivých častí práce. Bude prednesený výsledok snahy o to výstup kvantifikovať, posúdiť jeho kvalitu a nedostatky. Zdôvodnia sa zvolené vyhodnocovacie prístupy. Na záver dvoch nadchádzajúcich celkov nastane zhrnutie s celkovými výstupmi vyplývajúcimi z navrhnutého testovacieho procesu.

6.1 Otestovanie výkonu normalizácie

V sekcii 3.1 sa rozobrala normalizácia v teoretickej rovine a následne v sekcii 5.1 jej bol venovaný priestor z praktického hľadiska - bol popísaný spôsob jej implementácie. V tejto sekcii budú vecne popísané výstupy tohto procesu - ako je postavený proces testovania kvality, čo z výsledku vyplýva a aké možné ďalšie kroky možno určiť pre potenciálne napredovanie a odstránenie nedostatkov.

6.1.1 Navrhnutý princíp

Tým, že neexistuje dátová referencia, ktorou by sa dalo programovo porovnať tieto prvky (normalizované dáta), bolo nutné výsledok overovať a jeho kvalitu vyhodnocovať manuálne. Tým sa na druhú stranu otvárajú dvere komplexnejšiemu pohľadu na vec a roztriedeniu chýb do viacerých kategórií pre potenciálne pochopenie nedostatkov a zmysluplnejšie navrhnutie ďalších krokov.

Navrhnutý princíp spočíval v definícii posudzovaných kategórií chýb a následnom manuálnom hľadaní ich inštancií na mape. Z princípu normalizácie je ľahké okom posúdiť, ktoré nódy podkladovej mapy mali patriť do normalizovanej dráhy a ktoré nie. Týmto štýlom bolo preskenovaných celkovo 7 súborov a výsledky testovania možno vidieť v tabuľke č.6.1. Všetky testované súbory sú v priečinkoch vstupných dát, konkrétne 7 diskutovaných inštancií je v priečinku **normal_test**. V tabuľke reprezentujú stĺpec Testy vstupné súbory normalisation_track_1 až 4. V danom stĺpci figurujú výsledky ich celkového súčtu. Ďalšími troma súbormi sú 3 identifikované reálne linky, respektíve GNSS úseky, ktoré boli v rámci reálnych dát v závere identifikované ako autobusové linky. Vstupné parametre tohto merania sú NORMALSPEED_XROAD_DISTANCE rovný 0.0001 (predstavuje cca 11 metrový okruh) a možno nečakane SLOWSPEED_XROAD_DISTANCE rovnako rovný 0.0001, vyraďujúc efekt parametru LAST_COORDINATE_DISTANCE. Vložiť náhodné 3 premenné do vstupných podmienok znamená, že pri budúcich pokusoch o kompletné pretestovanie nebude aktuálne meranie až tak smerodajné. Na druhú stranu takto

navrhnuté testovanie overí základnú funkcionalitu všeobecného okruhu v ktorom sa detekujú križovatky. Zámer teda je vyhnúť sa komplexite, kým nie je otestovaný základ.

6.1.2 Výsledky

Zaujímavým poznatkom bude súčet, prípadne relevantná miera chýb jednej a druhej kategórie. Všeobecný celkový počet nódov chybne nájdených je 74 a nódov nesprávne nenájdených je 58. Pri výsledku značne nevybalancovanom sa dajú zmeniť parametre v súbore konštánt ako NORMALSPEED_XROAD_DISTANCE - parameter značí veľkosť okruhu okolitých križovatkových nódov ktoré berie v úvahu pri hodnote rýchlosti nad nastavenú mieru. Znížením tohto parametru by sa mal znížiť počet chýb typu chybne nájdené, následne bude dôležité sledovať ako by zareagovali chyby druhého typu. Je zaujímavé zmeniť aj hraničnú rýchlosť a teda množstvo identifikovaných križovatiek pri pomalej rýchlosti - ktoré reprezentujú parametre LAST_COORDINATE_DISTANCE a SLOWSPEED_XROAD_DISTANCE. Všetky 3 parametre majú priamy vplyv na výsledok.

Pri pohľade na posúdenie kritickosti chyby, je výsledok ešte jasnejší - 47 chybne nájdených kritických nódov (2,4%) a len 16 nódov kriticky nesprávne nenájdených (cca 0,8%) viď tab.6.1. Informácia jasne podporuje myšlienku meniť konštanty v diskutovanom smere. Každopádne, je to otázka. Je nutné zachytiť určité križovatkové nódy, inak sa stráca príležitosť vôbec zachytiť určitý cestný úsek. Vplyv na výsledok majú v skutočnosti chyby druhej kategórie omnoho znateľnejší, môžu sa vo výstupe prejavovať chybným cestným úsekom. Zatiaľ čo chybne nájdené boli najmä nódy kruhových prejazdov a blízkych križovatiek, ktoré nevytvárajú až taký negatívny funkcionálny dopad.

Celková výsledná chybovosť zameraná na kritické chyby je 3,2% na vzorkách o celkovej veľkosti 1968 normalizovaných nódov, ktoré reprezentovali vstupné GNSS dáta o 6942 nasnímaných polohách. Keďže sa jedná o konkrétne reálne dáta známeho formátu, je známe, že ide o sekundovú akvizíciu a teda vstupný úsek reprezentuje trajektóriu, ktorú autobus prešiel za 1 hod. a 55 min. K výslednej presnosti je vhodné dodať, že chyby ako na obr. č.6.1, boli započítané samostatne, teda v danom prípade ako 3 nesprávne nenájdené kritické chyby. Na tejto malej vzorke je možné vidieť takmer pätinu všetkých nájdených chýb tohto typu z celej vstupnej vzorky.

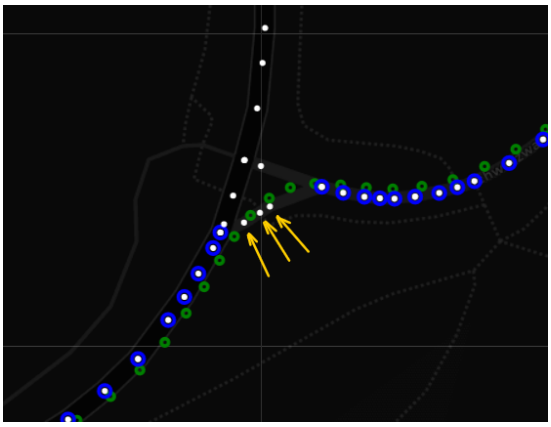
Tab. 6.1: Výstup normalizácie v číslach - vyhodnotenie presnosti. Smerodajný výstupný údaj je celková chybovosť, ktorá predstavuje za určených vstupných podmienok v texte hodnotu 3,2%.

Vstupný súbor	Testy	Linka 551	Linka 558	Linka 556	Celkovo
Počet vstupných GNSS nódov	1676	2888	651	1727	6942
Počet normalizovaných nódov	424	755	246	543	1968
Chybne nájdené - nekritické	11	10	0	6	27
Chybne nájdené - kritické	1	24	12	10	47
Nenájdené - nekritické	12	8	0	22	42
Nenájdené - kritické	7	7	2	0	16
Súčet kritických chýb	8	31	14	10	63
Výsledná chybovosť	1,89%	4,11%	5,69%	1,84%	**3,20%**

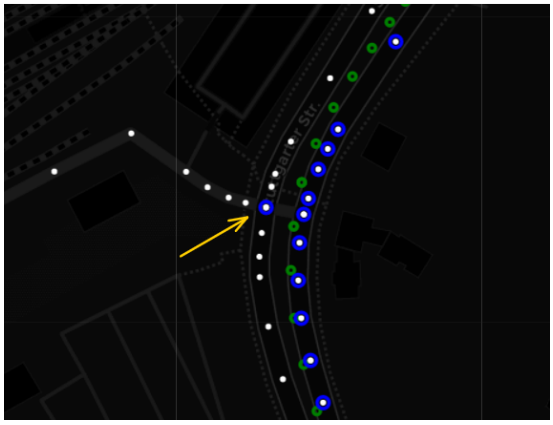
6.1.3 Chyby

Pre definíciu chýb boli navrhnuté 4 kategórie. Nódy, ktoré sa mali nájsť a nenašli sa (viď obr.č.6.1), a nódy, ktoré sa identifikovali, ale identifikovať sa nemali (viď obr.č.6.2). Očakávateľne prídavnú hodnotu možno očakávať vo vyjadrení miery v ktorej je dané vyjadrenie akútne. Preto sú obe kategórie vyjadrené prívlastkom, vyjadrujúcim či je chyba kritická, alebo druhoradá, prípadne z istého dôvodu očakávateľná. Prvé dva ponúknuté príklady na obrázkoch boli inštancie kritických chýb. Príklad nekritickej chyby typu kedy nód bolo treba nájsť a nenašiel sa, je na obr.č.6.3 a typ chyby kedy sa chybne našla na obr.č.6.4. V ďalšom odseku sa odôvodnia voľby kritérií, ktoré chyby zadelili do kategórie kritická/nekritická.

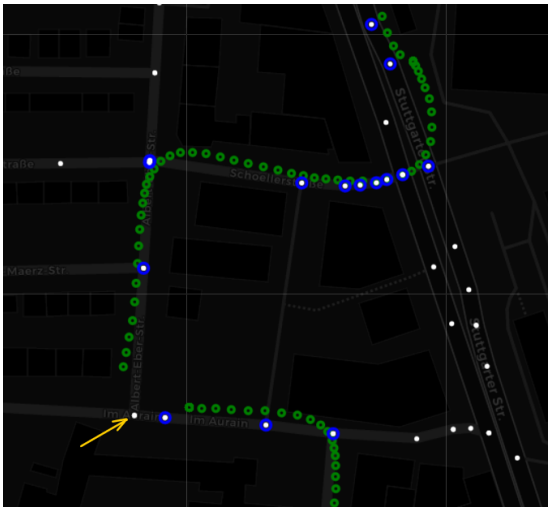
Interpretácia kritickej chyby je, že ju treba riešiť akútnejšie ako chybu nekritickú a všeobecne predstavuje väčší problém pre projekt. Nekritické chyby bolo potreba nejak definovať a rozmýšľať nad tým aký majú skutočne efekt na výsledok. Ideálnym príkladom je kruhový objazd. Ten sa takmer spravidla v OSM reprezentácii nachádza pod jedným prvkom, pod ktorým je X nódov. Problém je ten, že pri zahnutí na skorej odbočke autobus neprejde dostatočným množstvom nódov tohto prvku, aby to aktuálne navrhnuté riešenie zaregistrovalo ako validný prejazd. Na druhú stranu vždy je zachytený minimálne nód naviazaný na kruhový objazd, prípadne ho normalizačný algoritmus zaráta absolútne celý, čo v oboch prípadoch v skutočnosti funkčne vyhovuje, lebo sa nijak nestráca informácia - tá, že autobus bol v istý moment na kruhovom objazde (kruhový objazd je z tohto uhla pohľadu niečo ako križovatka a teda len jeden bod). Navyše možno očakávať, že pri opakovanom prejazde kruhovým objazdom sa zaregistruje rovnaký počet nódov a tým pádom to nebude mätúce pre ďalej navrhnuté alogritmy. Z vizuálneho hľadiska je ale vhodné



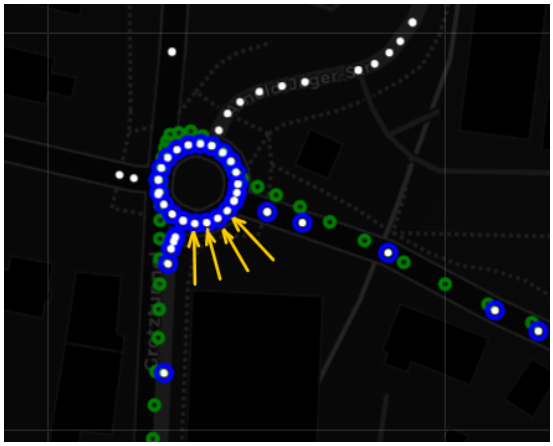
Obr. 6.1: Príklad chyby typu: Nenájdené - kritické. Žltými šípkami sú zvýraznené nódy, ktoré mali byť označené modrou farbou a teda mali byť patriť do normalizovanej dráhy.



Obr. 6.2: Príklad chyby typu: Chybne nájdené - kritické. V rámci normalizácie bol chybne identifikovaný nód zvýraznený žltou šípkou ako výstupný. Aktuálna rýchlosť autobusu teda detekovala danú križovatku - v rámci vyriešenia problému ide o doladenie koeficientov vo veci vzdialenosti právoplatných križovatiek pri špecifických rýchlostiach.



Obr. 6.3: Príklad chyby typu: Nenájdené - nekritické (výpadok vstupného signálu). Nód zvýraznený žltou šípkou mal byť detekovaný, ale nebol. Chyba nie je zarátaná ako kritická, lebo ide o silný vplyv nekvality vstupných dát.



Obr. 6.4: Príklad chyby typu: Chybne nájdené - nekritické. Na základe GNSS dráhy je jasné (smer jazdy na kruhovom objazde), že autobus naň vošiel z pravej strany a odišiel na ľavej. Normalizáciou sa označili za prejdené nódy zvýraznené modrou farbou a teda nódy označené žltými šípkami boli klasifikované nesprávne.

takéto inštancie chýb eventuálne vyriešiť, na čo sa v tomto prípade potenciálne využije tag **junction=roundabout**, ktorý identifikuje kruhový objazd a možno na tento úsek aplikovať špeciálne podmienky pre normalizáciu. Tým, že je to chyba bez známeho funkcionálneho nedostatku, je označená ako nekritická. Nekritické chyby ďalej reprezentujú zjavné chyby výpadku vstupného GNSS signálu, ktorých následkom došlo k nezachyteniu potrebných nódov. S extrémnymi výpadkami sa dá rátať a zaradiť pre tento prípad špeciálne podmienky. Problematika tejto témy spadá pod možné rozšírenia práce.

6.1.4 Zhrnutie

Vo veci testovania normalizácie bol navrhnutý spôsob testovania, boli navrhnuté javy, ktoré sa budú snímať a vyhodnocovať. Tieto prístupy boli zdôvodnené a výsledky priebežne vyhodnocované. Stanovili sa vstupné podmienky za platnosti ktorých bola zistená výsledná chybovosť o hodnote 3,2%.

6.1.5 Možné rozšírenia navrhnutého testovania

Komplexnosť rozšíreného testovania

V následujúcom zozname budú vymenované témy, ktoré je potrebné brať v dotaz pri testovaní tohto rázu s cieľom dosiahnutia opäť vyššej kvality výstupu testovacieho procesu:

1. zistenie ideálnej hodnoty argumentu **matchfactor** pre funkciu feature_deletion_matchwise (modul postx_functions.py)
2. partikulárna vzorka vstupných dát, ktorá relevantne zahrnie čo možno najviac očakávaných geografických lokácií a anomálií (téma prepojená s konkrétnym očakávaným celkom všetkých vstupných dát)
3. automatizovanie overovania výstupu normalizácie

1. Vplyv argumentu matchfactor - percentuálne vyjadrenie v sekcii 5.1.2 pri spomenutí feature_deletion_matchwise referuje práve na argument matchfactor. To, do akej miery sa musia podobať dve navrhované hladké reprezentácie vstupu na to, aby bol úsek označený ako platný, je tiež volatilný parameter. Vie byť rozumne stanovený a konzistentne nevytvárať chybné výstupy, každopádne platnosť tohto výroku je len priamo úmerná s konzistentnosťou vstupných dát. Tým, že sa pracuje s reálnymi dátami sa nájdu špecifické prípady kedy sa do otázky dostane hodnota tohto parametru a bude mať vplyv na výstup normalizácie. Je ho teda nutné ďalej zaradiť medzi vstupné parametre na základe ktorých sa naškáluje testovanie, hoci má sekundárnu prioritu v porovnaní s aktuálne zaradenými parametrami.

2. Výber vstupnej vzorky - tým, že kontrola normalizácie prebehla manuálne, nebolo možné ísť do vzoriek väčších veľkostí. V ďalšom paragrafe bude navrhnutý spôsob automatickej kontroly normalizácie, čím sa nadobudne možnosť testovať na väčšej vzorke. Výstupom by boli dve veľké výhody:

1. princíp automatického testovania ďalšieho paragrafu prinúti testovací proces manuálne namapovať tvar linky - v tomto prípade by boli veľmi precízne namapované všetky očakávané linky, čo poskytuje zaujímavý výstup aj mimo použitia v testingu
2. výsledná chybovosť a jej vyjadrenie je takpovediac absolútne - tým, že bude normalizácia pretestovaná na všetkých trasách liniek, bude možné pracovať s číslom, ktoré nie je len očakávaným vyjadrením s určitou volatilitou resp. neistotou, ale konečným vyjadrením skutočnosti

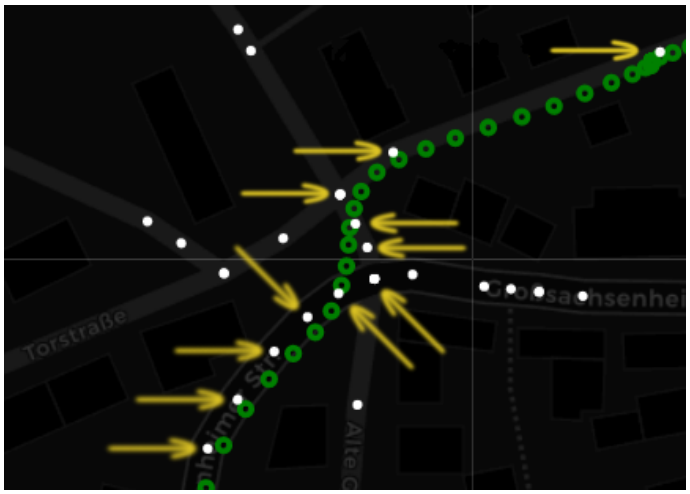
3. Automatizácia testovania - manuálne testovanie spočíva v procese kedy tester poskytuje prídavnú hodnotu na báze toho, že vie, ktoré nódy z podkladovej mapy mali byť normalizované. Tento model očakávaných normalizovaných nódov na vstupných úsekoch sa dá určiť vopred. Následne, ak sa určí pre všetky vstupné linky, nebude ho treba opakovať pre ďalšie meranie so zmenenými parametrami.
Vytvorenie modelu pre automatické testovanie normalizácie teda bude manuálne a spočívalo by v jednoduchom vytvorení listu platných normalizačných nódov. Tie sa následne budú porovnávať s výstupom algoritmickej normalizácie. Spôsob vytvárania referencie by sa teda spoľahol na ľudský faktor a uváženie, ktoré nódy podkladovej mapy majú patriť do normalizovanej dráhy. Pre príklad viď obr. č.6.5. Zelenou je označený GNSS vstup, bielou sú označené nódy podkladovej mapy a žltou farbou je zvýraznená ľudská úvaha nad vhodným výstupom.

6.2 Otestovanie výkonu komparačných algoritmov a pridružených metód

6.2.1 Navrhnutý princíp

Komparačné algoritmy a metódy k nim pridružené vo forme celkov boli popísané v sekcii 5.2.2. Je popísané, aký cieľ majú jednotlivé funkcie, ktoré sa v tejto testovacej sekcii porovnávajú. Pre nahliadnutie na výpis prvej sorty funkcií, ktorých sa to týka, viď prvý stĺpec tab.č. 6.2.

Navrhnutý princíp spočíva v sledovaní rýchlosti a presnosti jednotlivých navrhnutých algoritmov na 4 rôznych dráhach, pre náhľad viď obr.č. 4.3. V tab.č. 6.2 sa teda



Obr. 6.5: Princíp vytvorenia modelu pre automatické testovanie normalizácie - spočíva v manuálnom označení nódov podkladovej mapy (biela farba), o ktorých je rozumné usúdiť, že majú byť vo výstupnej normalizácii. V príklade na obrázku sú tieto nódy označené žltými šípkami. Vybrané nódy sa následne budú porovnávať s nódmi normalizovanými navrhnutým riešením/algoritmom.

sleduje doba behu, presnosť dvoch alternatívnych úsekov a výstup z komparačného algoritmu, ktorý využíva daná funkcia. To, aký komparačný algoritmus funkcia využíva, spravidla vychádza aj z jeho názvu. Číslo pridružené k názvu algoritmu značí štýl, ktorým sa komparačný algoritmus volá - popisuje súvisiacu pridruženú metódu. Pre pripomenutie, vždy pôjde o porovnanie dvoch úsekov, pretože sa hľadá opakovaný úsek a teda na dvoch pozíciách v postupnosti normalizovaných nódov sa porovnávajú vždy 2 úseky o rovnakej dĺžke. Je snaha aby boli tieto dva úseky vždy čo najviac podobné. Vmiešavajú sa tam ale chyby normalizácie a chyby metód klastrovania.

Význam jednotlivých stĺpcov tabuliek

Pred pokračovaním je vhodné nahliadnuť na tabuľku č.6.2, pretože sa v tejto sekcii budú popisovať jej jednotlivé stĺpce - čo vyjadrujú a prečo sú zahrnuté v testovaní.

Funkcia Funkcie z modulu patternfinding_functions.py; je v nich obsiahnutá funkcionalita klastrovacích snáh od čiastkových až po ucelené riešenia.

Doba behu [s] V hlavnom skripte sa časujú všetky potrebné úseky kódu a zapisujú do logfile **ccd.log** v zložke **tmp**. Ukážka údajov logu súvisiacich s časovaním sa nachádza pod týmto paragrafom. Do tabuliek sa zapisuje len doba súvisiaca s klastrovaním a teda **Pattern finding time**. Viac informácií k logfile v sekcii 7.4.

Script execution start: Fri Apr 2 19:39:04 2021
Normalisation time: 2.012278 seconds
Pattern finding time: 0.550498 seconds
Execution time: 5.552484 seconds

Presnosť alt.1 a alt.2 Tomuto výstupu sa venuje funkcia **verification_nodes** modulu refactor_functions.py. Tým, že sa testing klastrovacích metód vykonáva na vlastných vygenerovaných dátach, bolo zrejmé, ktoré dráhy sa majú nájsť. O čosi viac ako fakt, že boli vygenerované, pomáha ich menšia dĺžka s ktorou sa dá z mnohých hľadísk lepšie pracovať - overovanie konkrétnych situácií, rýchlosť normalizácie a klastrovania, absencia výpadkov vstupných dát a podobne. V rámci prístupu k testovaniu klastrovania na vlastných dátach bolo teda omnoho viac dosiahnuteľnejšie testing zautomatizovať a tým signifikantne navýšiť jeho objem a kvalitu výstupu.

Pre vstupné dráhy boli vytvorené dodatočné modely len liniek, ktoré sa mali v daných dráhach nájsť. Tie v závere poslúžili ako referencia s ktorou sa porovnávali nájdené linky navrhovanými algoritmami (stĺpec tabuľky - Funkcia). Rýchly sprievodca tohto procesu:

- spustí sa funkcia gestalt1 - quick_ratio pre vstupný súbor Dráhy 1
- algoritmus detekuje pozíciu dvoch úsekov na základe podmienok, ktoré definuje on sám - tieto úseky by na mape mali reprezentovať jednu oblasť, jednu autobusovú linku
- nastáva porovnanie hodnôt jednotlivých nódov referencie a oboch nájdených výsledkov
- výsledky tohto porovnania sa zapíšu do relevantných stĺpcov

Otázkou je, ako sa vyráta táto hodnota. Tým, že je to založené na jednoduchom princípe, je tu možnosť si to ukázať priamo, viď výpis č.6.1. Metóda intersection vykoná prienik medzi jej argumentom a setom nad ktorým je volaná. Množstvo prvkov spadajúcich pod tento prienik sa dá do pomeru k väčšej z dĺžok referenčnej alebo nájdenej linky. To znamená, že pri všetkých možnostiach, ktoré pripadajú v úvahu, bude výsledná presnosť vždy vykazovať najnepriaznivejšie číslo. Čo znamená, že ak bude výsledná presnosť v tabuľkách vyjadrená kladne, jednoznačne to znamená, že výstup je očakávaný a správny do vyjadrenej percentuálnej miery. Pre načrtnutie situácií ktoré sú aktuálne diskutované, ide o:

- prípad, kedy je nájdený dlhší úsek ako očakávaná linka a zhoduje sa pokojne vo všetkých nódoch linky - sa dá do pomeru k jeho veľkosti (väčšej veľkosti) a výsledok nebude 100% (počet zvalidovaných nódov môže mať maximálne veľkosť referenčnej linky)

- nájdený úsek je menší ako referenčná linka, všetkými svojimi nódmi sa zhoduje s očakávanou linkou - množstvo zvalidovaných nódov sa dá do pomeru veľkosti referenčnej linky a výsledok opäť kvantitatívne vyjadrí diskrepanciu v zníženom percentuálnom čísle
- nájdený úsek je menší ako referenčná linka a dokonca sa ani nezhoduje vo všetkých nódoch s referenčnou linkou; pomer zvalidovaných nódov s celkovým počtom referenčných bude vyjadrovať ešte vyšší nepomer a znova sa miera nezhody prejaví na výslednom percentuálnom čísle

Výpis 6.1: Princíp získania presnosti výstupu klastrovania - časť funkcie verification_nodes modulu refactor_functions.py v zložke **functions**/refactoring, viď A

```
1     valid_nodes = set(tupled_nodes_input_fine_nodup).intersection(script_output_nodes_nodup)
2     if len(script_output_nodes_nodup) > len(tupled_nodes_input_fine_nodup):
3         bigger_array = len(script_output_nodes_nodup)
4     else:
5         bigger_array = len(tupled_nodes_input_fine_nodup)
6     valid_ratio = len(valid_nodes) / bigger_array
7     return valid_ratio*100
```

Alg. koef Vyjadruje podobnosť dvoch nájdených alternatív medzi sebou. V každom riadku je tento koeficient vyjadrený súvisiacim komparačným algoritmom (gestalt - ratio(), levenshtein - fuzz.ratio(), manual_trackfinder - myratio_calc()). Pôvodne mal tento koeficient slúžiť na kvalitatívne ohodnotenie výstupu, každopádne jeho prídavná hodnota navrhnutým riešením presnosti v predchádzajúcom paragrafe, zaniká. Je úplne jedno, či sa dve alternatívy navzájom dokonalo podobajú, ak obe nevykazujú dobré výsledky čo sa týka relevancie voči referenčnej dráhe. Táto podobnosť je samozrejme relevantná a s jeho hodnotou sa v rámci riešenia manipuluje a je to kľúčový prvok výpočtov, každopádne pre prezentovanie výsledkov a pri prítomnosti navrhnutého vyjadrenia presnosti je zobrazenie tohto koeficientu nadbytočné - preto tento stĺpec nebude zaradený medzi ďalšie tabuľky. Pre ukážku vzorky je zobrazený v tab.č. 6.2.

Koeficienty sú v rozsahu (0-1), kde 1 je najlepší výsledok. Pri pohľade na prvý riadok tab.č. 6.2 sa dokazujú predchádzajúce tvrdenia, kedy sú obe dráhy na základe gestalt kritéria rovnaké, ale v skutočnosti to po validácii na referenčnej linke vykazuje nedokonalú zhodu. Popisuje to napríklad situáciu v ktorej sa našli dve alternatívy linky, ktoré sú dokonalo zhodné medzi sebou, ale nepopisujú to, čo zadávala referencia. Koeficient 4. a 8. riadku je vynechaný, lebo nespadá do vzorca 0-1. Tým, že vyjadruje absolútnu hodnotu spriemerovanú na jeden prvok, je ťažké ju zmysluplne zaradiť do mierky. Dá sa s ňou ale dobre pracovať a porovnávať ju medzi inými hodnotami jej rázu. Z dôvodu nezaradenia týchto koeficientov do tabuľkových vizualizácií prestalo byť relevantné koeficient napasovať na interval 0-1 a

bol vynechaný aj z ukážkovej vizualizácie. Užívateľ sa s ním stretne pri akomkoľvek behu skriptu s menom manual_trackfinder, koeficient sa zapíše do logfile, v ktorom si to samozrejme vie zobraziť a ďalej s ním pracovať.

6.2.2 Výsledky

Prikladám opäť referenciu na sekciu 5.2.2, kde je popísané, aký cieľ majú jednotlivé funkcie, ktoré sú testované. Princíp na základe ktorého sa získa prídavná hodnota tohto testovania spočíva v označení zaujímavých výsledkov v tabuľkách a na základe nich sa postavili výsledné funkcie. Tie sa následne samostatne opäť pretestujú a tá ktorá dosiahla najlepšie výsledky bude vybraným výsledným riešením úlohy. Tento spôsob bol zvolený preto, lebo úloha je pomerne obsiahla a bolo naimplementovaných viacero unikátnych prístupov. Myšlienka teda bola čo najviac oddeliť jednotlivé unikátne prístupy a každý jeden spustiť na potenciálne až 4 vstupných dráhach a sledovať ich výkon. Ďalej, ako bolo popísané, vyberú sa najviac výkonné črty a postavia sa z toho o takpovediac stupeň komplexnejšie algoritmy. Následne sa znova otestujú na daných dráhach a vyberie sa návrh ktorý vykazoval najlepší celkový výsledok.

Počiatočná testovacia séria pre vyskladanie komplexných verzií funkcií

V tabuľkách č.6.2 a č.6.3 sa použili dráhy kde sa mala identifikovať iba jedna linka. V rámci prvých štyroch riadkov na oboch dráhach vykazovali komparačné algoritmy veľmi dobré výsledky a teda prídavná hodnota vzišla skôr z porovnania ich rýchlostí, kde dominuje gestalt, najmä vo verzii quick_ratio(). Druhá štvorica riadkov pretestovala pokus o zvýšenie rýchlosti naimplementovaním jednoduchého princípu skipovania nezaujímavých nódov a zjavne priniesla zaujímavý časový benefit bez toho aby sa zhoršila kvalita nájdených liniek. V ostatných prípadoch je to sporadické a jediný výstup vyzerá tak, že pridružená metóda v rámci gestalt5 na komparačnom algoritme ratio() nevykazuje kvalitný výstup. Kvalita výstupu všetkých navrhnutých prístupov vyzerá byť zaujímavá, vyplývajúc z testov na dráhach 1 a 3. Domnievam sa, že v mnohých prípadoch metódy vykázali úplne totožné a bezchybné výsledky a daný offset od plnohodnotného výsledku vytvorila normalizačná chyba.

Finálna testovacia séria na komplexných riešeniach

Následovná vzorka funkcií je poskladaná z princípov predchádzajúcej sekcie, kde boli zaujímavé výsledky zvýraznené zelenou farbou. Zároveň boli výstupy viacmenej konzistentné a preto je aj táto finálna vzorka značne variabilná a nereprezentuje iba jeden prístup a jeden komparačný algoritmus.

Tab. 6.2: Počiatočná testovacia séria - Dráha 1. Červenou zvýraznené nevýkonné výsledky ktorých navrhnuté prístupy treba zvážiť nezahrnúť do ďalších testovacích kôl. Zelenou zvýraznené dobré výsledky, na ktoré sa je dobré špecificky pozrieť a snažiť zahrnúť do ďalších riešení.

Dráha 1 - Sachsenheim_1_1.asc				
Funkcia	Doba behu [s]	Presnosť alt.1	Presnosť alt.2	Alg. koef
gestalt1 - quick_ratio	0.206	96.20%	96.20%	1.000
gestalt1 - ratio	0.600	96.20%	96.20%	0.962
leveshtein1	14.464	93.67%	96.20%	0.980
manual_trackfinder1	13.676	96.20%	96.20%	x
gestalt2 - quick_ratio	0.011	96.20%	96.20%	1.000
gestalt2 - ratio	0.018	96.20%	96.20%	0.962
leveshtein2	0.091	96.20%	96.20%	0.980
manual_trackfinder2	0.107	96.20%	96.20%	x
gestalt4 - quick_ratio	7.745	96.34%	100%	0.965
gestalt4 - ratio	15.078	100%	100%	0.963
gestalt5 - quick_ratio	0.013	100%	98.75%	0.944
gestalt5 - ratio	0.010	60.75%	59.49%	0.940
gestalt6 - quick_ratio	3.604	95.18%	98.75%	0.953
gestalt6 - ratio	5.987	98.75%	100%	0.952

V tejto sekcii sa opäť začína na dráhach 1 a 3, ktoré obsahujú jednu linku. Výsledky vidno v tab.č.6.4 a tab.č.6.5. Následne tým, že sa sledujú komplexné verzie klastrovacích algoritmov, nastáva testovanie na dráhach 2 a 4, ktorých súčasťou sú 2 linky. V tabuľkách č.6.6 a 6.7 teda možno sledovať porovnanie s referenciou na oboch linkách v ramci danej dráhy. Účel tejto testovacej sekcie spočíva aj v tom, že sú finálne algoritmy testované na omnoho komplexnejších dráhach. Odráža sa to na vyššej pestrosti výsledkov a následnej možnosti lepšie vyselektovať kvalitu testovaných algoritmov.

Vychádzajúc zo spomínanej komplexnosti dráh teda nebolo problémom vyselektovať algoritmy s najlepším výkonom. Kritéria boli stanovené následovne:

- ak funkcia detekovala na ľubovoľnej linke ľubovoľnej dráhy výsledok s presnosťou pod 80%, zvýrazní sa červenou farbou a ďalej sa neberie v dotaz
- ak funkcia detekovala výsledok s presnosťou 80-90%, označí sa žltou farbou a berie sa v dotaz
- nezvýraznené hodnoty vykazujú kvalitu výsledku nad 90% a pochopiteľne sa ďalej berú v dotaz

Tab. 6.3: Počiatočná testovacia séria - Dráha 3. Účel je rovnaký ako v tab.č.6.2 s tým, že je zmenená testovacia dráha. Tabuľky majú pretestovať výkonnosť jednotlivých komparačných algoritmov a funkcií na klastrovanie ako takých, viď 5.2.2.

Dráha 3 - Sachsenheim_3_1.asc			
Funkcia	Doba behu [s]	Presnosť alt.1	Presnosť alt.2
gestalt1 - quick_ratio	0.018951	100%	100%
gestalt1 - ratio	0.022940	100%	100%
leveshtein1	0.511144	98.41%	100%
manual_trackfinder1	0.467358	100%	100%
gestalt2 - quick_ratio	0.005960	100%	100%
gestalt2 - ratio	0.006982	100%	100%
leveshtein2	0.025931	100%	100%
manual_trackfinder2	0.049873	100%	100%
gestalt4 - quick_ratio	0.595163	96.92%	98.44%
gestalt4 - ratio	2.651058	87.30%	87.30%
gestalt5 - quick_ratio	0.003990	100%	98.44%
gestalt5 - ratio	0.001995	55.55%	55.55%
gestalt6 - quick_ratio	0.354054	96.92%	96.92%
gestalt6 - ratio	0.397938	100%	100%

Vyhodením funkcií, ktoré vykázali kvalitu výsledku <80%, prišlo na finálnu trojicu **gestalt8 - quick_ratio**, **gestalt8 - ratio** a **leveshtein3**. Ostáva vysieť zohľadnenie rýchlosti danej implementácie a porovnať kvalitu výsledku aj kvantitatívne, respektíve porovnaním s ostatnými. Na to sa aplikoval jednoduchý scoringový systém, kedy ak v danej vlastnosti algoritmus predčil obe zbytné, získa 2 body, druhé miesto získa 1 bod a tretie získa 0 bodov. Týmto štýlom sa prejde cez trvanie na každej dráhe a cez kvalitu na jednotlivých linkách. Výsledok možno vidieť v tab.č.6.8 pre vyhodnotenie rýchlosti a v tabuľke č.6.9 pre vyhodnotenie presnosti. Záver a výsledok finálneho testovania je v tab.č.6.10.

6.2.3 Chyby

Normalizovaná sekvencia mohla v procese nanormalizovať dáta, ktoré nanormalizovať nemala, preto treba rátať s implicitnou potenciálnou chybou cca 3,2%, ktorá bola dokázaná normalizačným testovaním. Z tabuľky č. 6.1 možno posúdiť, že chybovosť nie je najpravidelnejšia. Tým, že v nachystaných dátach možno očakávať linky o veľkosti 20 až 100 normalizovaných nódov, je možné, že sa nájdu linky so 100% presnosťou. Z rozloženia 3,2% ich možno očakávať na cca každom 31. nóde. Zároveň boli vlastné vygenerované dáta potenciálne o niečo konzistentnejšie, hoci bola snaha ich generovať čo najviac realisticky.

Tab. 6.4: Finálna testovacia séria - Dráha 1. Vylúčenie riešení manual_trackfinder3 a gestalt7 - ratio, pre ich zlyhanie vo výsledku na danej dráhe dosiahnutím presnosti pod 80% - zvýraznené červenou. Žltou sú zvýraznené výsledky na hrane s presnosťou 80 až 90%.

Dráha 1 - Sachsenheim_1_1.asc			
Funkcia	Doba behu [s]	Presnosť alt.1	Presnosť alt.2
gestalt8 - quick_ratio	0.019007	100%	98.75%
gestalt8 - ratio	0.020845	100%	100%
leveshtein3	0.064311	100%	95.18%
manual_trackfinder3	0.067991	55.70%	55.70%
leveshtein5	0.056848	87.34%	82.28%
gestalt7 - quick_ratio	0.019946	100%	98.75%
gestalt7 - ratio	0.022906	60.76%	59.49%

Tab. 6.5: Finálna testovacia séria - Dráha 3. Obe prípady málo presných výsledkov sú späté s funkciami, ktoré boli vylúčené v tab.č.6.4. Potvrdila sa ich nedostatočná výkonnosť v mierke presnosti.

Dráha 3 - Sachsenheim_3_1.asc			
Funkcia	Doba behu [s]	Presnosť alt.1	Presnosť alt.2
gestalt8 - quick_ratio	0.008951	100%	98,44%
gestalt8 - ratio	0.004140	100%	100%
leveshtein3	0.020534	100%	95,45%
manual_trackfinder3	0.033877	87.3%	87.3%
leveshtein5	0.024469	100%	98,44%
gestalt7 - quick_ratio	0.009947	100%	98,44%
gestalt7 - ratio	0.011968	55.55%	55.55%

Ďalšou zaznamenanou chybou sú možné nesprávne detekované začiatky a konce liniek. Klastrovacie algoritmy potrebovali byť stavané na to, aby vždy zachytili čo najväčší úsek. Pretože iba tak sa dokázali dostať k celej autobusovej linke, inak by nachádzali iba tie najzaujímavejšie úseky týchto liniek. Tým, ako sa navrhnuté algoritmy snažia tlačiť do bodu, kedy identifikujú čo najväčšiu linku, stáva sa, že sice detekujú 100% linky ktorú detekovať mali, ale pretečú aj do ďalšej iterácie prechodu danou linkou. Táto chyba sa vyskytuje iba pri 3 a viac konzekutívnych iteráciách na rovnakej linke. Zadanie práce si žiada vytvoriť modely liniek a tie budú vytvorené správne aj pri tomto pretečení, pre ďalší potenciálny vývoj to je ale vhodné vedieť.

Tab. 6.6: Finálna testovacia séria - Dráha 2. Vylúčenie ďalších riešení levenshtein5 a gestalt7 - quick_ratio.

		Dráha 2 - Linka 1		**Dráha 2 - Linka 2**	
Funkcia	Beh	Alt.1	Alt.2	Alt.1	Alt.2
gestalt8 - quick_ratio	0.035344	100%	100%	100%	96.51%
gestalt8 - ratio	0.040127	100%	100%	100%	98.8%
leveshtein3	0.137811	100%	100%	100%	94.31%
manual_trackfinder3	0.906753	100%	100%	59.03%	59.03%
leveshtein5	0.090422	100%	93.2%	62.65%	59.03%
gestalt7 - quick_ratio	0.063829	100%	95.24%	61.45%	59.04%
gestalt7 - ratio	0.077709	100%	96.39%	57.83%	55.42%

Tab. 6.7: Finálna testovacia séria - Dráha 4. V tomto bode sú vylučené všetky Funkcie až na prvé 3 v tejto tabuľke. Funkcia levenshtein3 mala v rámci všetkých 4 dráh len jedno zaváhanie na hranici vyhodenia a s prvými dvoma funkciami gestalt8 sa zúčastní ďalších testovaní.

		Dráha 4 - Linka 1		**Dráha 4 - Linka 2**	
Funkcia	Beh	Alt.1	Alt.2	Alt.1	Alt.2
gestalt8 - quick_ratio	0.065794	95.50%	95.5%	96.55%	96.55%
gestalt8 - ratio	0.061869	98.84%	98.84%	96.55%	96.55%
leveshtein3	0.200603	89.47%	91.4%	87.5%	87.5%
manual_trackfinder3	0.231898	98.84%	98.84%	96.55%	96.55%
leveshtein5	0.169909	95.51%	95.51%	90.32%	90.32%
gestalt7 - quick_ratio	0.062576	94.44%	94.44%	93.33%	93.33%
gestalt7 - ratio	0.056131	85.88%	87.06%	93.33%	93.33%

6.2.4 Zhrnutie

Testing klastrovacích funkcií spočíval v cielenej izolácií jednotlivých funkčných častí a použitých metód. Na základe otestovania týchto malých celkov sa získali informácie o ich výkone a teda o tom, ktoré z nich má zmysel zaradiť do ďalšieho kola. Toto testovanie prebehlo v tabuľkách č.6.2 a č.6.3.

V ďalšej iterácií testovania sa z jednotlivých funkčných častí vyskladali komplexnejšie algoritmy, ktoré boli testované aj na ďalších dvoch viac obsiahlych a komplikovanejších dátach. Obsah testovania tejto časti je v tabuľkách č.6.4, 6.5, 6.6 a 6.7. Každá tabuľka predstavuje test všetkých funkcií na jednej konkrétnej dráhe.

Tab. 6.8: Scoring - vyhodnotenie rýchlosti s výstupom toho, že 2x bola najrýchlejšia a 2x druhá najrýchlejšia funkcia gestalt8 - quick_ratio a presne rovnako to platí pre funkciu gestalt8 - ratio. Funkcia levenshtein3 bola vždy najpomalšia.

		gestalt8 - quick_ratio	**gestalt8 - ratio**	**levenshtein3**
Dráha 1	Výsledok [s]	0.019007	0.020845	0.064311
	Body	2	1	0
Dráha 3	Výsledok [s]	0.008951	0.004140	0.020534
	Body	1	2	0
Dráha 2	Výsledok [s]	0.035344	0.040127	0.137811
	Body	2	1	0
Dráha 4	Výsledok [s]	0.065794	0.061869	0.200603
	Body	1	2	0
	Súčet bodov	**6**	**6**	**0**

Tab. 6.9: Scoring - vyhodnotenie presnosti (D = dráha, L = linka). Body sa prideľovali na základe umiestnenia vo vzťahu k ostatným funkciám. Výsledok možno vidieť v poslednom riadku, kde sa funkcia gestalt8 - ratio ukázala ako najpresnejšia.

			gestalt8 - quick_ratio	**gestalt8 - ratio**	**levenshtein3**
D1	L1	Výsledok	100%, 98.75%	100%, 100%	100%, 95.18%
		Body	1	2	0
D3	L1	Výsledok	100%, 98.44%	100%, 100%	100%, 95.45%
		Body	1	2	0
D2	L1	Výsledok	100%, 100%	100%, 100%	100%, 100%
		Body	2	2	2
	L2	Výsledok	100%, 96.51%	100%, 98.8%	100%, 94.31%
		Body	1	2	0
D4	L1	Výsledok	95.50%, 95.5%	98.84%, 98.84%	89.47%, 91.4%
		Body	1	2	0
	L2	Výsledok	96.55%, 96.55%	96.55%, 96.55%	87.5%, 87.5%
		Body	1	1	0
		Súčet bodov	**7**	**11**	**2**

Na základe dát z testovania druhého kola bolo možné usúdiť, ktoré z navrhovaných funkcií majú najväčší potenciál. Kritérium tohto hodnotenia bola kvalita výsledku vyjadrená v percentách voči referencii. Výstupom boli 3 funkcie z ktorých **gestalt8 - quick_ratio** dosahovala presnosti výsledkov minimálne 95.5% na všetkých úsekoch s priemernou presnosťou 98.15%, **gestalt8 - ratio** dosiahla výsledkov kvality zhody minimálne 96.55% na všetkých 4 testovacích dráhach s priemernou

Tab. 6.10: Scoring - výsledok. Následkom vyššej presnosti v závere dosiahla najlepších výsledkov v hodnotení rýchlosti a presnosti funkcia gestalt8 s komparačným algoritmom ratio(). Tým, že v žiadnej kvalite funkcie na druhom a treťom mieste nepredbehli prvú, nie je motivácia odporúčiť za určitých podmienok použiť inú funkciu ako výhernú.

	gestalt8 - quick_ratio	**gestalt8 - ratio**	**levenshtein3**
Body za rýchlosť	6	6	0
Body za presnosť	7	11	2
Celkové hodnotenie	**13**	**17**	**2**

presnosťou 99.13% a **leveshtein3**, funkcia ktorá dosahovala výsledkov kvality konzistentne nad hranicou 87.5% s priemernou zaznamenanou presnosťou 95.06%. Na záver je v tabuľke 6.8 porovnanie a scoring výkonu týchto funkcií na základe dĺžky ich behu a v tabuľke č.6.9 porovnanie a scoring kvality ich výstupu.

Tabuľka č.6.10 zobrazuje záverečné vyhodnotenie scoringu. Najlepšie výsledky vykazovala funkcia s názvom **gestalt8()** s použitím komparačného algoritmu **ratio()**. Jej definícia je v súbore patternfinding_functions.py pod názvom **gestalt8** a pre jej volanie v hlavnom skripte stačí v súbore **constants.py** vložiť do premennej METHOD hodnotu "gestalt8".

6.3 Otestovanie na množine reálnych dát

Ako príklady výstupov navrhnutého klastrovacieho algoritmu na reálnych dátach, sa použijú linky použité pri normalizácii. Dôvod je ten, že je možnosť nahliadnuť na presné čísla chýb normalizácie a následne to spozorovať na výstupe klastrovacieho algoritmu. Zároveň sa tým dokáže funkčnosť navrhnutej metódy a jej **nezávislosť od typu vstupných vzoriek**.

Vizualizačných výstupov je pripravených viac a podrobnejšie sa budú diskutovať v sekcii 7.3 (z pohľadu vizualizácie ako prostriedku). Dáta, ktoré sa použijú vzišli zo vstupných súborov, ktoré reprezentovali jedno-dňovú jazdu autobusov a boli vybrané spomedzi viacerých nájdených. Vo veci vstupných súborov bola linka 556 identifikovaná vo vstupnom súbore s názvom **aBUS1_01_06.txt** a linky 551 a 558 v súbore **aBUS3_01_08.txt**. Tieto dva súbory obsahujúce vstupné GNSS dáta možno pre predstavu a názornú ukážku vidieť na obr.č.B.1.

Nasledujú konkrétne linky a vizualizácie výstupu riešenia tejto práce na reálnych dátach. Na obr.č.B.2 vidieť referenciu dohľadateľnej linky 558. Následne na obr.č.B.3

možno vľavo hore vidieť úsek vstupných dát - priblížený na inkriminovaný úsek kde sa bude nachádzať daná linka. Je to prakticky priblížený kúsok mapy, ktorá sa nachádza na obr.č.B.1 vľavo. Vpravo hore je ukážka normalizovaných vstupných dát daného úseku. Opakované prejazdy už nemajú vplyv a táto reprezentácia zodpovedá formátu OSM čo sa týka konkrétnych pozícií nódov. V časti vľavo dole sú zvizualizované alternatívy 1 a 2. Sú to alternatívy, ktoré úradovali v tabuľkách celej sekcie testovania klastrovania. Sú to teda dva úseky, ktoré sa vzájomne porovnávali a určila sa ich vzájomná podobnosť vo forme koeficientu komparačného algoritmu a ich ultimátna presnosť voči referencii. Na danom obrázku sú zvizualizované jedna oranžovou a druhá čiernou farbou. Vpravo dole je zvizualizovaný začiatok a koniec linky a jej priebeh načrtnutý aj farebne pre účel zvýraznenia smeru. Na obr.č.B.4 vidieť výsledný model linky 558 zvizualizovaný pomocou nástroja JOSM, ktorý okrem iného slúži aj na overenie kompatibility s OSM formátom. Farebne zvýrazní druh cesty, ak je kontribútormi táto charakteristika k dispozícii, automaticky vizualizuje nódy a pod.

Ďalej na obr.č.B.5 možno vidieť ako ukážku referenciu a výstupný model linky 551. Na obr.č.B.6 rovnako ukážka referencie a výstupného modelu, tentokrát linky 556.

7 Štruktúra projektu

7.1 Návod

Vo veci zorientovania sa v tom ako operovať nad týmto projektom bol vytvorený video-návod o dĺžke cca 14 min., ktorý zhrňuje príkladový beh programu. Nachádza sa v dokumentácii v priečinku docs, kde sa na Youtube odkaz dá dostať po otvorení súboru **documentation.html** ihneď na titulnej strane. Video-návod sledovateľa prevedie počiatočným extrahovaním mapy z webu overpass-turbo, následným predspracovaním pomocou basemap_creator.py a vytvorením náležitostí podkladovej mapy potrebných pre main.py. Vysvetlí sa základná stavba súboru constants.py a jeho vzťah k hlavnému skriptu a ukáže sa jeho príkladový beh. Na záver sa predstavia jednotlivé výstupy skriptu či už vo forme výstupných súborov alebo vizualizácií a spôsoby ktorými možno tento výstup zobraziť.

7.2 Dokumentácia

S návodom je úzko spojená samotná dokumentácia. Kód je dokumentovaný v štýle zaužívaných konvencií komentármi docstring (Documentation String) nad funkciami, a blokovými a inline komentármi nad procesom a premennými. Dokumentačný štýl bol prispôsobený nástroju Doxygen pomocou ktorého bola generovaná už spomínaná dokumentácia pod priečinkom **docs**. Z dôvodu limitovanej podpory tohto nástroja pre jazyk Python sa doinštaloval filter **doxypypy**, ktorý doplňuje funkcionalitu nástroja Doxygen pre Python. Ide najmä o to, aby generátor dokumentácie rozoznával všetky možné Pythonovské štruktúry a hlavne samotné komentáre na základe stanovených konvencií ako napr PEP 257. Dokumentácia je pod spomínaným odkazom **documentation.html**, kde na titulke okrem odkazu na video možno vidieť krátky súhrn funkcionality práce v angličtine. Z dôvodu stanovenej požiadavky na praktickú časť práce, sú dokumentácia a návod vyhotovené v anglickom jazyku.

7.3 Vizualizácia

Veľká časť obrázkov písomnej časti tejto práce je vytvorená navrhnutým vizualizačným systémom. Vizualizovať vstupné nódy a výsledky ktoré vykazuje backend práce bolo nutné od samého počiatku tohto projektu. K dispozícii je viacero knižníc pre vytváranie webových mapovacích aplikácií. Najdôležitejšie bolo nájsť a sprievozniť ľubovoľnú z nich, ktorá bude poskytovať podporu na čo možno všetky žiadúce funkcionality. Z jednoduchej rešerše možností a žiadúcich funkcionalít bola vybraná

knižnica **folium**. Táto knižnica je Pythonovský wrapper pre Leaflet.js, čo je open-source JavaScript knižnica pre interaktívne mapy. Nad konkurenciou vyhráva svojou jednoduchosťou a tým, že nemala žiadne chýbajúce funkcionality, ktoré si pýtal projekt. Druhorado úradoval aj fakt, že som s ňou bol zľahka oboznámený už pred prácou na tomto projekte. Ak by sa zvažovala alternatíva, bolo by vhodné nahliadnuť napríklad aj na framework GeoDjango.

O vizualizáciu sa stará modul s názvom **CVT.py**, ktorý možno nájsť medzi ostatnými funkciami v súhrnnom priečinku pre funkcie s názvom **functions**; projektová štruktúra je k dispozícii v prílohe A. V prvom rade bolo dôležité pochopiť ako funguje táto knižnica a ako sa do nej pridávajú jednotlivé vrstvy, ako sa s ňou pracuje a aké funkcionality vôbec poskytuje. V tejto kapitole bude rozobrané ako funguje funkcionalita hlavných vizualizácií v priečinku **visualisation**, kde sú generované dva hlavné súbory **main_vis.html** a **Map_testbed.html**. Prvý súbor je vizualizačný výstup z modulu **main.py** a druhý spomenutý je vizualizačný výstup z modulu **input_checker.py**.

Postup vytvorenia vizualizácie pozostáva z troch krokov:

1. Vytvorenie mapového základu (metóda Map()). V rámci neho sa v prípade tohto projektu nastavuje a rieši:
   - Implicitná pozícia
   - Implicitné priblíženie
   - Štýl mapy (rôzne motívy podkladu - satelitné snímky, dopravná mapa ...)
2. Implementácia vrstiev (feature groups).
   - Vizualizácia daných nódov (Circle(location, color, radius), PolyLine())
   - Tooltips (Circle(tooltip="lat lon xyz"), GeoJsonTooltip(datový balíček z OSM viď obr.č.5.2)),
   - Markers (start, stop) viď obr.B.3 vpravo dole
   - Úprava farieb - tak isto ako na referencovanom obrázku, kde bol aplikovaný gradient od zelenej k červenej, alebo na rovnakom obrázku vľavo hore gradient od fialovej k žltej. Pre daný účel bola využitá knižnica tretej strany s názvom **colour**.
3. Uzavretie mapového základu.
   - Pridanie všetkých vrstiev do vytvoreného mapového základu.
   - Umožnenie manuálnej kontroly nad vrstvami, prípadne iné dodatočné nastavenia.
   - Uloženie vizualizácie pod zadaným menom/lokáciou.

Obr. 7.1: Jednoduchá nódová vizualizácia - možnosť dynamicky meniť veľkosti a farby nódov napríklad pre zrovnanie podobnosti dvoch dráh - na tomto príklade je možnosť si všimnúť, že sú zvizualizované dve alternatívy a nie sú dokonalo podobné - líšia sa v jednom nóde vpravo dole

Tým, že sa tento proces neustále opakoval, bolo výhodné vytvoriť vizualizačné funkcie pre účely tohto projektu. Pre vytvorenie komplexnej vizualizácie teda stačí zavolať funkcie **create_mapbase()**, následne ľubovoľný počet z funkcií, ktoré pridávajú vrstvu do mapového základu, výber najpoužívanejších vytvorených funkcií:

- **fg_add_succession()** - vhodné na zobrazenie vstupných dát, pretože nódy sa v priebehu vykresľovania zmenšujú a je možné snímať danú postupnosť, ktorá je indikovaná aj zmenou farieb; viď obr.č.7.4
- **fg_add_basic()** - vhodné na zvýraznenie jednoduchých skupín nódov, ideálne podkladové mapy; tak isto vhodné na porovnanie dvoch rôznych úsekov (je možné zmeniť farbu a veľkosť vykreslených nódov) - viď obr.č. 7.1
- **add_fg_OSM_clean()** - zobrazenie výsledného modelu, získanie dát z OSM databázy do finálneho modelu - viď obr.č.7.3
- **add_fg_direction()** - zobrazenie priebehu výsledného modelu linky, štart a stop značia dva Markery - viď obr.č.7.2

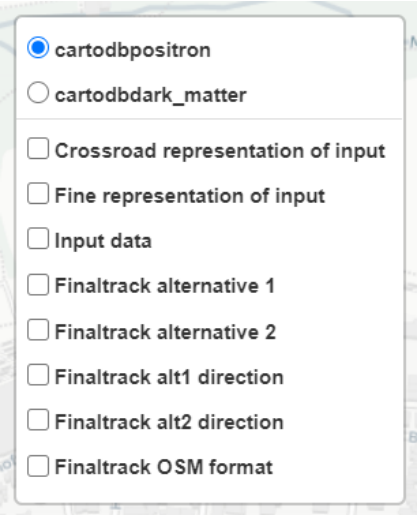
A na záver bola pre uzavretie vizualizácie vytvorená funkcia **close_mapbase()** s účelom popísaným v treťom bode hrubého funkcionálneho náčrtu popísaného vyššie. Vytvorenie tohto základu pre prácu s knižnicou folium projektu dodalo mnoho možností a výborný náhľad do skúmaných problémov.

Vpravo hore je vo vizualizáciách (.html súboroch) v ktorých je to umožnené kontrolný prvok pre prácu s jednotlivými vrstvami. Stačí myšou nájsť na prvok o vzhľade viď obr.č.7.6 a rozbalí sa ponuka, v hlavnej vizualizácii **main_vis.html** presne o vzhľade z obr.č.7.5.

Obr. 7.2: Vizualizácia so zvýraznením smeru/priebehu výsledného modelu linky

Obr. 7.3: Vizualizácia OSM formátu s prístupom k dátovému balíku na ľubovoľnom úseku

Obr. 7.4: Vizualizácia s dôrazom na zvýraznenie postupnosti. To je docielené progresívnym zmenšením nódov spolu s kontinuálnou zmenou farby.



Obr. 7.5: Kontrolný prvok pre zobrazenie žiadúcich vrstiev implementovaných v .html vizualizácii. Funguje podľa očakávania - posledná zaškrtnutá vrstva prekrýva predchádzajúce + je možnosť zobraziť ľubovoľný počet vrstiev naraz.

Obr. 7.6: Layer control - v pravom hornom rohu okna s vizualizáciou. V momente, kedy užívateľ nájde kurzorom na daný prvok, rozbalí sa ponuka na obr.č.7.5.

Obr.7.5: prvé dve možnosti prepínajú medzi svetlým a tmavým módom mapy - možnosť vybrať ten, na ktorom je lepší kontrast. Následne je možnosť zobrazenia:

- Križovatková reprezentácia vstupu (pomocou fg_add_basic())
- Hladká reprezentácia vstupu (fg_add_basic())
- Vstupné dáta (fg_add_succession())
- Alternatíva 1 z dvojice úsekov, ktoré reprezentujú model linky
- Alternatíva 2 (fg_add_basic() pre obe alternatívy)
- Alt.1 Štart/Stop (add_fg_direction())
- Alt.2 Štart/Stop (add_fg_direction())
- Finálny model v OSM (add_fg_OSM_clean()))

7.4 Testing a logging

Testovanie a overovanie výsledku je zásadné ako na nízkej úrovni, tak na úrovni vyššej abstrakcie, kde je nutnosť použitia iných metód pre zmysluplný a čo najrelevantnejší výsledok. Preto je testovanie tohto projektu rozdelené na unittesting a teda testing na nízkej úrovni, kde sa testujú čo najmenšie časti kódu a funkcie, a na testovanie vyššej úrovne, kde bolo nutné pretestovať funkcionalitu na mieru šitým spôsobom, ktorý hodnoverne odzrkadlí výkonnosť navrhnutého celistvého riešenia.

Testy unit-testingu sa nachádzajú v priečinku **tests**, kde sa v rámci 5 testovacích modulov pretestuje celkovo 14 základných funkcií. Testuje sa pomocou testovacieho balíčka **unittest**.

Testy vyššej úrovne boli navrhnuté a implementované v rámci kapitoly 6, kde sa otestovala ako normalizácia, tak klastrovanie.

Vzájomným spojením týchto dvoch prístupov by malo dôjsť k dostatočnému pokrytiu vo veci pretestovania ako funkčnosti riešenia, tak kvality výsledku.

Vo veci logovania výstupu skriptu bolo navrhnuté riešenie, ktoré čiastočne naväzuje na testovaciu časť. Pomocou balíčka **logging** zo štandardnej knižnice Pythonu sa priebežne logujú všetky dôležité dáta ohľadom behu skriptu za daných podmienok. Ukážka logu viď výpis č.7.1. Zaznamenávajú sa teda doby behov jednotlivých častí kódu, vstupné súbory a vstupné parametre, hlavné metadáta vo veci nájdených modelov liniek (Positions) a pri možnosti a zvolení danej voľby aj výstupy z testovania (track validity). Každý tuple v liste Positions zaznamenáva: (veľkosť nájdeného modelu v nódoch, pozíciu v sekvencii prvej alternatívy, pozíciu vo vstupnej sekvencii druhej alternatívy, výstupný koeficient danej použitej metódy).

Tento set dát sa vždy pripíše na koniec súboru **ccd.log** v priečinku **tmp** po každom behu skriptu main.py. Avizované pripisovanie teda umožňuje nahliadnutie na históriu behov daného skriptu. Zaznamenáva sa aj čas spustenia, takže táto história je prehľadná a poskytuje zaujímavý prehľad základných informácií jednotlivých spustení skriptu. Obrovské využitie sa mu našlo v časti, kedy sa testovala normalizácia a klastrovanie, kedy nebolo nutné si opakovane zapisovať za akých podmienok sa skript spúšťal, jednoducho si to možno pripísať do logovacieho setu a následne pracovať len s výsledným výpisom.

Výpis 7.1: Názorná ukážka logfile po jednej inštancii behu skriptu, viď príloha A zložka **tmp**, súbor ccd.log

```
1  Input file: input_files/IN_custom/Sachsenheim_4_2.asc
2  Input parameters: length_start = 20 ; length_finish = 100
3  Script execution start: Sat Apr 3 05:50:27 2021
4  Normalisation time: 2.302232 seconds
5  Method used: gestalt6
6  Pattern finding time: 72.019203 seconds
7  Tracks found: 2
8  Positions: [(100, 0, 367, 0.8530), (29, 131, 207, 0.8548)]
9  Track verified: input_files/IN_custom/verification/Sachsenheim_4_track1.asc
10 Alternative 1 validity: 88.541667 %
11 Alternative 2 validity: 86.734694 %
12 Execution time: 76.827848 seconds
```

Ako vizualizácia, tak testing a logging, rovnako ako ďalšie funkcionality programu bolo samozrejme treba naprogramovať a nie sú to automatické/generovacie nástroje. Jediný nástroj, ktorý niečo generoval je Doxygen, ktorý sa postaral o frontend dokumentácie.

7.5 Vedenie projektu a náležitosti vytvoreného nástroja

Kód je písaný vo verzii Python 3.8, to isté platí pre testing a celý projekt. Končí jej podpora v októbri 2024. Realease je pod hlavičkou PEP 569. Verzia bola zvolená na základe toho, že release Python 3.9 (najaktuálnejší v čase písania práce, ďalší release je naplánovaný na 4.10.2021) bol 5.10.2020, v čase ktorom bol vývoj projektu už v pokročilej fáze. Vývoj teda začal na najnovšej vtedajšej verzii.

Pre spustenie a využívanie projektu je teda potreba **Python 3.8**, prípadne kompatibilná verzia a použité balíčky, ktoré nie sú súčasťou štandardnej knižnice Pythonu. Tie sú súčasťou súboru **requirements.txt** vo forme v ktorej ich možno priamo nainštalovať pomocou pip install.

Projekt bol vyvíjaný pod Git verzovacím systémom s repozitárom na GitHub hostingu.

Záver

Na úvod bol objasnený zdroj GNSS dát - spôsob ich zberu. Opísal sa ich formát a potenciál či úžitok jednotlivých vlastností, ktoré sú k nim pridružené. Následne sa dostane na často skloňované pojmy a na to, aké dátové formáty alebo zdroje geodatabáz riešia spoločnosti v praxi. Boli preskúmané potenciálne možnosti a skonvergovalo sa k výberu OSM formátu. Ten sa následne rozobral čo sa týka licencie, pokrytia, potenciálu a spoľahlivosti.

Normalizovaná forma autobusovej dráhy má v dátovej reprezentácii určitý charakter. Ten trebalo vziať v dotaz a nájsť vhodné algoritmy pre porovnávanie jednotlivých úsekov tejto dráhy. Výsledkom je výpočet Levenshteinovej vzdialenosti, použitie Gestalt Pattern Matchingu a ako tretia možnosť bolo navrhnuté vlastné riešenie. Výsledkom sú tri vzájomne zameniteľné alternatívy pre vyriešenie problému komparácie trajektórií.

V tematike normalizácie bol popísaný jej význam a benefity, ktoré prináša. Naväzuje teoretický základ pre pochopenie OSM formátu, pre jej následnú čo najprecíznejšiu implementáciu. Bolo popísané akými spôsobmi a z akých zdrojov možno extrahovať požadovanú vektorovú mapu. Na záver bolo vysvetlené, ako ju možno programovo spracovať a prefiltrovať ešte v danom nástroji pre extrakciu pomocou vbudovaného dotazovacieho jazyka.

V štvrtom bode zadania bola predstavená vstupná vzorka na ktorej je postavené riešenie. Pripravená vstupná vzorka zahrňuje vlastné vygenerované dáta a reálne dáta spoločnosti Siemens. K tomu boli vyextrahované vhodné podkladové mapy pre pripravené vstupné vzorky. Výstupom je dátový set nad ktorým možno vyvíjať a testovať riešenia tohto projektu.

Riešenie tejto práce sa v samom základe delí na normalizáciu a klastrovanie. Oba tieto piliere boli implementované a popísané ako metodicky, tak z programového hľadiska. Bola vynaložená snaha metodický princíp aj zvizualizovať na originálnych vzorkách konkrétnych prípadov, ktoré riešili jednotlivé problémy navrhnutej implementácie. Výstupom je škálovateľná metóda pre normalizáciu a viacero riešení či prístupov ku klastrovaniu liniek.

Výstup z testovania normalizácie možno vidieť v tab.6.1. Výsledná chybovosť normalizácie je na úrovni 3,2%. Tabuľka zároveň poskytuje konkrétne hodnoty chybovosti na úsekoch, ktoré sú exemplárne predstavené v sekcii 6.3. Výstupy klastrovania boli pre projekt zásadnými a okrem kvality výstupu do hry vstupovala aj rýchlosť algoritmov, ktorá predstavovala zásadný implementačný fokus. Jednotlivé navrhnuté princípy boli oddelene testované aby preukázali kvalitu jednotlivých implementovaných funkcionalít. Z nich sa vystavali finálne klastrovacie riešenia, ktoré boli medzi sebou pretestované a vyhodnotené. Časovú výkonnosť troch najúspeš-

nejších možno vidieť v tab.6.8 a kvalitatívnu výkonnosť v tab.6.9. Vo veci rýchlosti samotnej bola funkcia gestalt8 s oboma alternatívami komparačných algoritmov quick_ratio() a ratio() podľa zadefinovanej metriky približne rovnako rýchla a funkcia levenshtein3 výrazne zaostávala. Najvyššiu presnosť zaznamenala funkcia gestalt8 - ratio s konzistentným výstupom nad hranicou 96.55% a priemernou presnosťou 99.13%, o čosi horšie výsledky vykazovala funkcia gestalt8 - quick_ratio s minimálnou presnosťou 95.5% na všetkých testovaných úsekoch s priemernou presnosťou 98.15% a funkcia tretia v poradí pod názvom levenshtein3 vykazovala konzistentnú presnosť minimálne 87.5% s priemernou hodnotou presnosti 95.06%. Finálny výstup testovania klastrovacej časti vyhodnocuje tab.6.10, z ktorej sa v závere usúdila najspoľahlivejšia celková výkonnosť u funkcie **gestalt8** s komparačným algoritmom **ratio()**.

Posledná, siedma kapitola sa zaoberá témami nad rámec zadania, prípadne témami, ktoré špecificky nepatrili žiadnej z predchádzajúcich šiestich kapitol vychádzajúcich zo zadania diplomovej práce. Vizualizácia a jej výstupy sa zobrazujú v mnohých častiach práce, v tejto sekcii bola dôkladne popísaná. Bola vysvetlená voľba knižnice folium a spôsob, ktorým sa k jej možnostiam pristúpilo. Ďalej sa ujasnila konkrétna implementácia výslednej vizualizácie hlavného skriptu. Popísal sa unittesting a implementácia či zmysel logovania metadát a zvolených výstupov za behu skriptu. Obsahom tejto kapitoly je aj set informácií a snáh, ktoré smerujú potenciálnemu užívateľovi tohto projektu. Medzi ne patrí návod na použitie, dokumentácia a všeobecné informácie k náležitostiam vytvoreného nástroja. Návod a dokumentácia boli vytvorené v angličtine a nachádzajú sa v priečinku **docs**. Medzi informácie o vedení projektu patria konkrétne požiadavky vo veci použitých balíčkov, knižníc alebo nástrojov a ich požadovaných verzií.

Literatúra

[1] Nolan, M.: *Fundamentals of Air Traffic Control.*
Cengage Learning, 2010. s. 200–202. ISBN 978-1-4354-8272-2.
Dostupné z URL: <https://books.google.cz/books?id=6yhTiGC3ulcC>

[2] Richard B. Langley: *Dilution of precision* [online].
University of New Brunswick,
posledná aktualizácia 05/1999 [cit. 2. 12. 2020].
Dostupné z URL: <http://gauss.gge.unb.ca/papers.pdf/gpsworld.may99.pdf>

[3] Morag Chivers: *Differential GPS Explained* [online].
Trimble, posledná aktualizácia neznáma; [cit. 2. 12. 2020].
Dostupné z URL: <https://www.esri.com/news/arcuser/0103/differential1of2.html>

[4] Mapy.cz: *Podmínky užití a licenční podmínky služby Mapy.cz* [online].
posledná aktualizácia 01/2020 [cit. 29. 12. 2020].
Dostupné z URL: <https://licence.mapy.cz/?doc=mapy_pu>

[5] Shawn Gordon: *Where in the World Does All this ESRI World Data Come from?* [online].
posledná aktualizácia 08/2014 [cit. 29. 12. 2020].
Dostupné z URL: <https://www.smartdatacollective.com/where-world-does-all-esri-world-data-come/>

[6] RON AMADEO: *Google's Street View cars are now giant, mobile 3D scanners* [online].
posledná aktualizácia 09/2017 [cit. 29. 12. 2020].
Dostupné z URL: <https://arstechnica.com/gadgets/2017/09/googles-street-view-cars-are-now-giant-mobile-3d-scanners/>

[7] Adena Schutzberg: *Esri has 40+% of GIS Marketshare* [online].
posledná aktualizácia 11/2011 [cit. 29. 12. 2020].
Dostupné z URL: <https://web.archive.org/web/20130103033512/http://apb.directionsmag.com/entry/esri-has-40-of-gis-marketshare/215188>

[8] ArcGIS Hub: *Global Roads from GRIP* [online].
posledná aktualizácia 11/2018 [cit. 29. 12. 2020].
Dostupné z URL: <https://hub.arcgis.com/datasets/

89e3fa578173483aa677d28ee474acae?geometry=17.075%2C48.139%2C17.137%2C48.159>

[9] Google: *Legal Notices for Google Maps/Google Earth and Google Maps/Google Earth APIs* [online].
posledná aktualizácia 06/2020 [cit. 29. 12. 2020].
Dostupné z URL: <https://www.google.com/help/legalnotices_maps/>

[10] The official blog for Google Maps: *It's your world. Map it.* [online].
posledná aktualizácia 03/2008 [cit. 29. 12. 2020].
Dostupné z URL: <https://maps.googleblog.com/2008/03/its-your-world-map-it.html>

[11] Here: *About us.* [online].
posledná aktualizácia neznáma; [cit. 29. 12. 2020].
Dostupné z URL: <https://www.here.com/company/about-us>

[12] Paul Sawers: *Here launches XYZ, a geospatial data management and mapping platform* [online].
posledná aktualizácia 10/2018 [cit. 29. 12. 2020].
Dostupné z URL: <https://venturebeat.com/2018/10/04/here-launches-xyz-a-geospatial-data-management-and-mapping-platform/>

[13] Leo Kent: *HERE shares how automated cars can 'heal' maps on the fly* [online].
posledná aktualizácia 06/2015 [cit. 29. 12. 2020].
Dostupné z URL: <https://360.here.com/2015/06/23/here-sensor-data-ingestion/>

[14] Leo Kent: *Autonomous cars can only understand the real world through a map* [online].
posledná aktualizácia 04/2015 [cit. 29. 12. 2020].
Dostupné z URL: <https://360.here.com/2015/04/16/autonomous-cars-can-understand-real-world-map/>

[15] Here: *HERE car drive schedule* [online].
posledná aktualizácia neznáma; [cit. 29. 12. 2020].
Dostupné z URL: <https://www.here.com/drives>

[16] Apple Inc.: *Apple Acknowledgements* [online].
posledná aktualizácia 01/2016 [cit. 29. 12. 2020].
Dostupné z URL: <https://web.archive.org/web/20160118101039/http://gspe21.ls.apple.com/html/attribution-8.html>

[17] Mark Lee: *Apple Shares Google China Map Partner in Win for AutoNavi: Tech* [online].
posledná aktualizácia 07/2012 [cit. 29. 12. 2020].
Dostupné z URL: <https://web.archive.org/web/20141018152101/http://www.bloomberg.com/news/2012-07-05/apple-shares-google-china-map-partner-in-win-for-autonavi.html>

[18] DONNA HOWELL: *Jack Dangermond's Digital Mapping Lays It All Out* [online].
posledná aktualizácia 08/2009 [cit. 29. 12. 2020].
Dostupné z URL: <https://web.archive.org/web/20100510064955/http://www.investors.com/NewsAndAnalysis/Article.aspx?id=503454>

[19] M. Haklay: *How Good is Volunteered Geographical Information? A Comparative Study of OpenStreetMap and Ordnance Survey Datasets.*
Environment and Planning B: Planning and Design, 2010. volume 37 s. 682–703.
Dostupné z URL: <https://www.ucl.ac.uk/~ucfamha/OSM%20data%20analysis%20070808_web.pdf>

[20] Mordechai Haklay, Patrick Weber: *OpenStreetMap: User-Generated Street Maps.*
IEEE Pervasive Computing, 2008. volume 7 s. 12–18.
Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4653466&isnumber=4653458>

[21] X. Xie, Y. Zhou, Y. Xu, Y. Hu and C. Wu: *OpenStreetMap Data Quality Assessment via Deep Learning and Remote Sensing Imagery.*
IEEE Access, 2019. volume 7 s. 176884–176895.
Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8924753&isnumber=8600701>

[22] J. E. Vargas Munoz, S. Srivastava, D. Tuia and A. X. Falcao: *OpenStreetMap: Challenges and Opportunities in Machine Learning and Remote Sensing.*
IEEE Geoscience and Remote Sensing Magazine, 2019.
Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9119753&isnumber=8976286>

[23] OSM wiki: *Google Maps user contributions* [online].
posledná aktualizácia 12/2020 [cit. 30. 12. 2020].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/Google_Maps_user_contributions>

[24] OSM wiki: *Comparison Google services – OSM* [online].
posledná aktualizácia 02/2018 [cit. 30. 12. 2020].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/Comparison_Google_services_%E2%80%93_OSM>

[25] J. Kaur and J. Singh: *An Automated Approach for Quality Assessment of OpenStreetMap Data.*
International Conference on Computing, Power and Communication Technologies (GUCON), 2018. s. 707–712.

Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8674899&isnumber=8674884>

[26] OpenStreetMap: *Autorská práva a licence* [online].
posledná aktualizácia neznáma; [cit. 30. 12. 2020].
Dostupné z URL: <https://www.openstreetmap.org/copyright>

[27] OSM wiki: *Potential Datasources* [online].
posledná aktualizácia 12/2020 [cit. 30. 12. 2020].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/Potential_Datasources>

[28] Frank Hofmann: *Levenshtein Distance and Text Similarity in Python* [online].
posledná aktualizácia neznáma; [cit. 31. 12. 2020].
Dostupné z URL: <https://stackabuse.com/levenshtein-distance-and-text-similarity-in-python/>

[29] John W. Ratclif: *Pattern Matching: the Gestalt Approach* [online].
posledná aktualizácia 07/1988 [cit. 01. 01. 2021].
Dostupné z URL: <https://www.drdobbs.com/database/pattern-matching-the-gestalt-approach/184407970?pgno=5>

[30] Python 3.9.1 Documentation: *difflib — Helpers for computing deltas* [online].
posledná aktualizácia neznáma; [cit. 01. 01. 2021].
Dostupné z URL: <https://docs.python.org/3/library/difflib.html#module-difflib>

[31] OSM wiki: *Map features* [online].
posledná aktualizácia 11/2020 [cit. 02. 01. 2021].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/Map_features>

[32] A. Ligocki, A. Jelinek and L. Zalud: *Brno Urban Dataset - The New Data for Self-Driving Agents and Mapping Tasks.*
2020 IEEE International Conference on Robotics and Automation (ICRA), 2020. s. 3284–3290.
Dostupné z URL: <https://ieeexplore.ieee.org/abstract/document/9197277>

[33] OSM wiki: *JOSM* [online].
posledná aktualizácia 12/2020 [cit. 02. 01. 2021].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/JOSM>

[34] OSM wiki: *Stats* [online].
posledná aktualizácia 01/2021 [cit. 03. 01. 2021].
Dostupné z URL: <https://wiki.openstreetmap.org/wiki/Stats>

[35] Martijn van Exel: *An intro to OpenStreetMap for the Geography department of the University of Utah.* [online].
posledná aktualizácia neznáma; [cit. 07. 04. 2021].
Dostupné z URL: <https://www.slideshare.net/mvexel/openstreetmap-9819440>

[36] PIZUR, Jaroslav: *Klasifikace linek MHD z GNSS dat* [online].
Semestrální práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. [cit. 26. 4. 2021].
Dostupné z URL: <https://www.vutbr.cz/studenti/zav-prace/detail/130816>

[37] Sammie Bae: *Big-O Notation. In: JavaScript Data Structures and Algorithms.* Apress, Berkeley, CA, 2019. [cit. 15. 04. 2021]
Dostupné z URL: <https://doi.org/10.1007/978-1-4842-3988-9_1>

Zoznam symbolov a skratiek

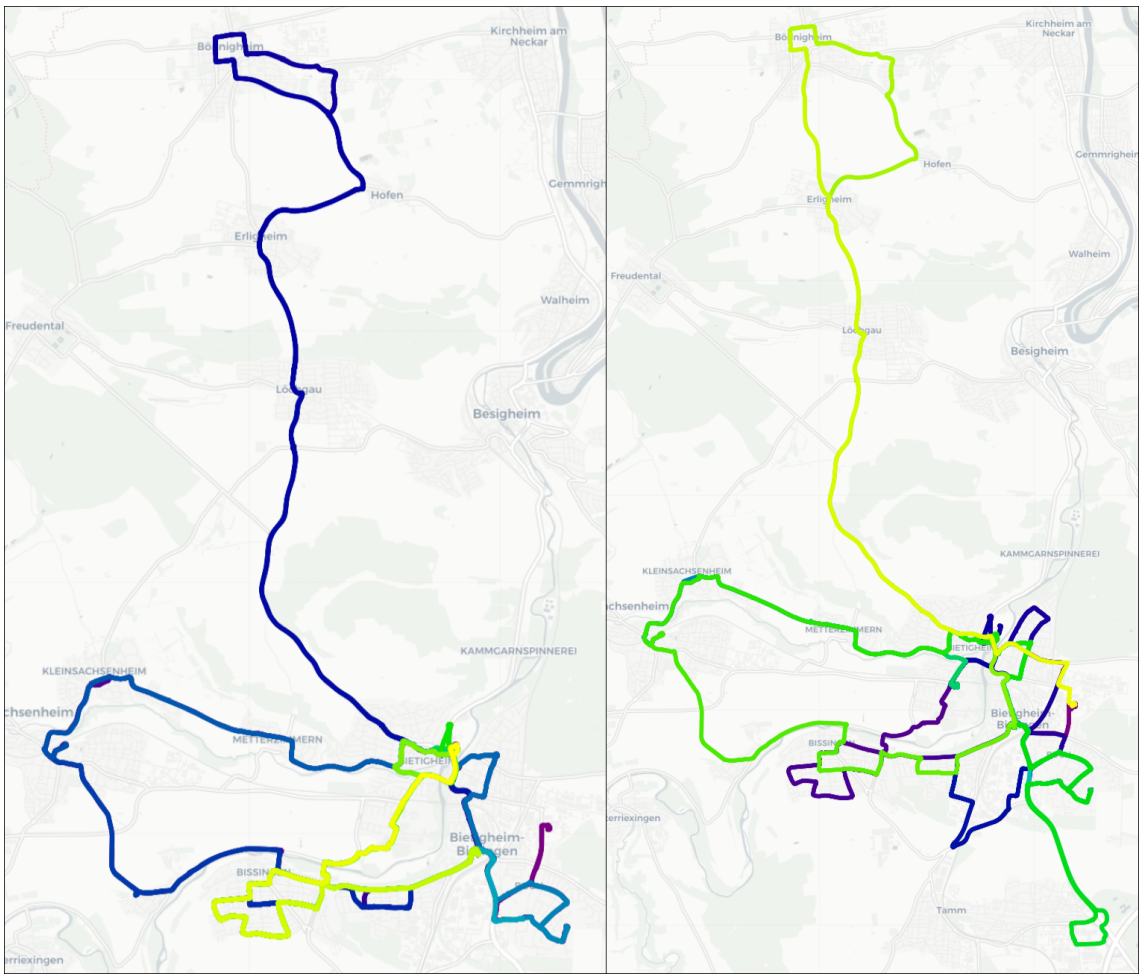
OSM	Open Street Map
JOSM	Java Open Street Map - Editor
GNSS	Global Navigation Satellite System - Globálny družicový polohový systém
QL	Query Language - Vyhľadávací/Dopytovací jazyk (obe formy sú spisovné a správne)
XML	Extensible Markup Language - Rozšíriteľný značkovací jazyk
DOP	Dilution of Precision - parameter presnosti (Terminologická komise ČÚZK)
GPS	Global Positioning System - Globálny Polohový Systém
DGPS	Differential GPS - Diferenciálne GPS
GIS	Geographic Information System
LiDAR	laser imaging, detection, and ranging
VGI	Volunteered geographic information

A Orientácia v projektových súboroch

S funkcionalitou a obsahom samozrejme súvisia aj mnohé iné súbory a zložky ktoré nie sú v tomto zozname. Nadchádzajúci zoznam obsahuje súbory a zložky, ktoré sa zvážili ako najkľúčovejšie.

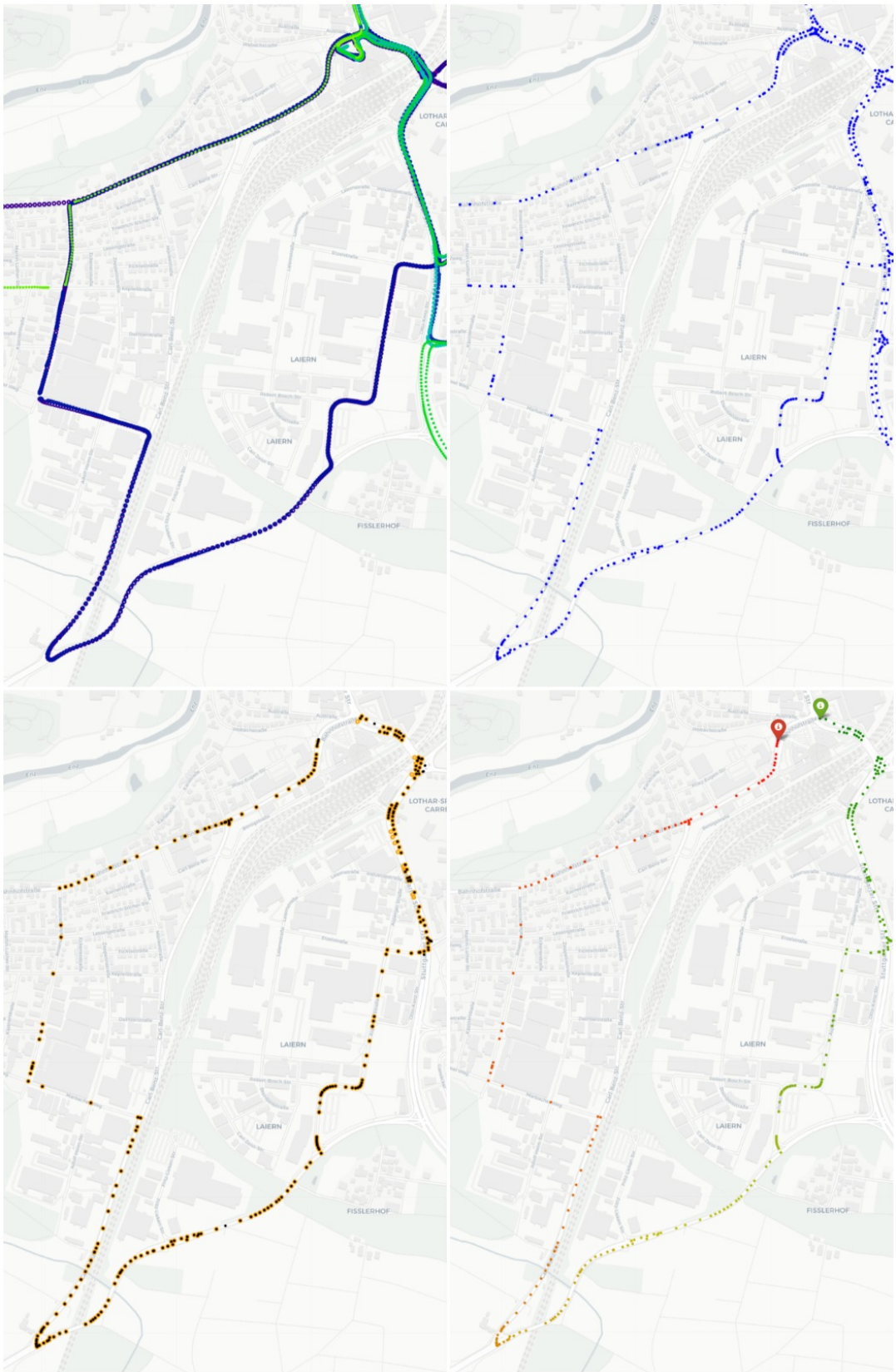
- `/` — koreňový adresár
  - **`docs`**
    - `Doxygen_misc` — náležitosti dokumentácie a nástroja Doxygen
      - `Doxyfile` — config pre Doxygen
      - `py_filter.bat` — filter pre Doxygen
    - `documentation.html` — dokumentácia programu a videonávod na použitie
  - `osmProjekt` — zdrojové súbory
    - `deps`
      - `constants`
        - **`constants.py`**
      - **`functions`** — všetky referencované .py moduly z main.py
      - `non_package_deps`
        - `basemap_dep`
          - `Basemaps` — podkladové mapy
          - `Basemap_script_output` — výstup skriptu basemap_creator
        - `input_preprocessor`
          - `input_data_preprocessor.py`
    - `input_files`
      - `IN_custom` — vlastné/vygenerované vstupné dáta
        - `verification` — referencia pre vygenerované linky (testing)
      - `IN_sie` — reálne vstupné dáta
    - `output_files` — výstupy skriptu main.py
      - `data` — basemapové výstupy
      - `track_alternatives` — nájdené modely liniek
      - `visualisation` — výstupy všetkých vizualizačných tendencií
        - `main_vis.html` — hlavná výstupná vizualizácia
    - `tests` — unit testy
      - `run_all_tests.py`
    - `tmp` — logfile
    - **`basemap_creator.py`** — vytvorenie náležitostí z podkladovej mapy
    - `input_checker.py` — vizualizácia vstupného súboru
    - **`main.py`** — hlavný modul
    - `normalisation_tester.py`
  - `.gitignore` — výnimka pre súbory output zložky
  - `README.md`
  - `requirements.txt` — zoznam použitých balíčkov

B Vybrané vizualizácie liniek



Obr. B.1: Ukážka vstupných GNSS dát jedno-dňových dráh; aBUS1_01_06.txt vľavo a aBUS3_01_08.txt vpravo. Ich účelom je nadobudnúť vizuálnu predstavu týchto dát, a spätne na nich nahliadnuť pri identifikácii liniek, ktoré obsahujú. Späť na sekciu 6.3.

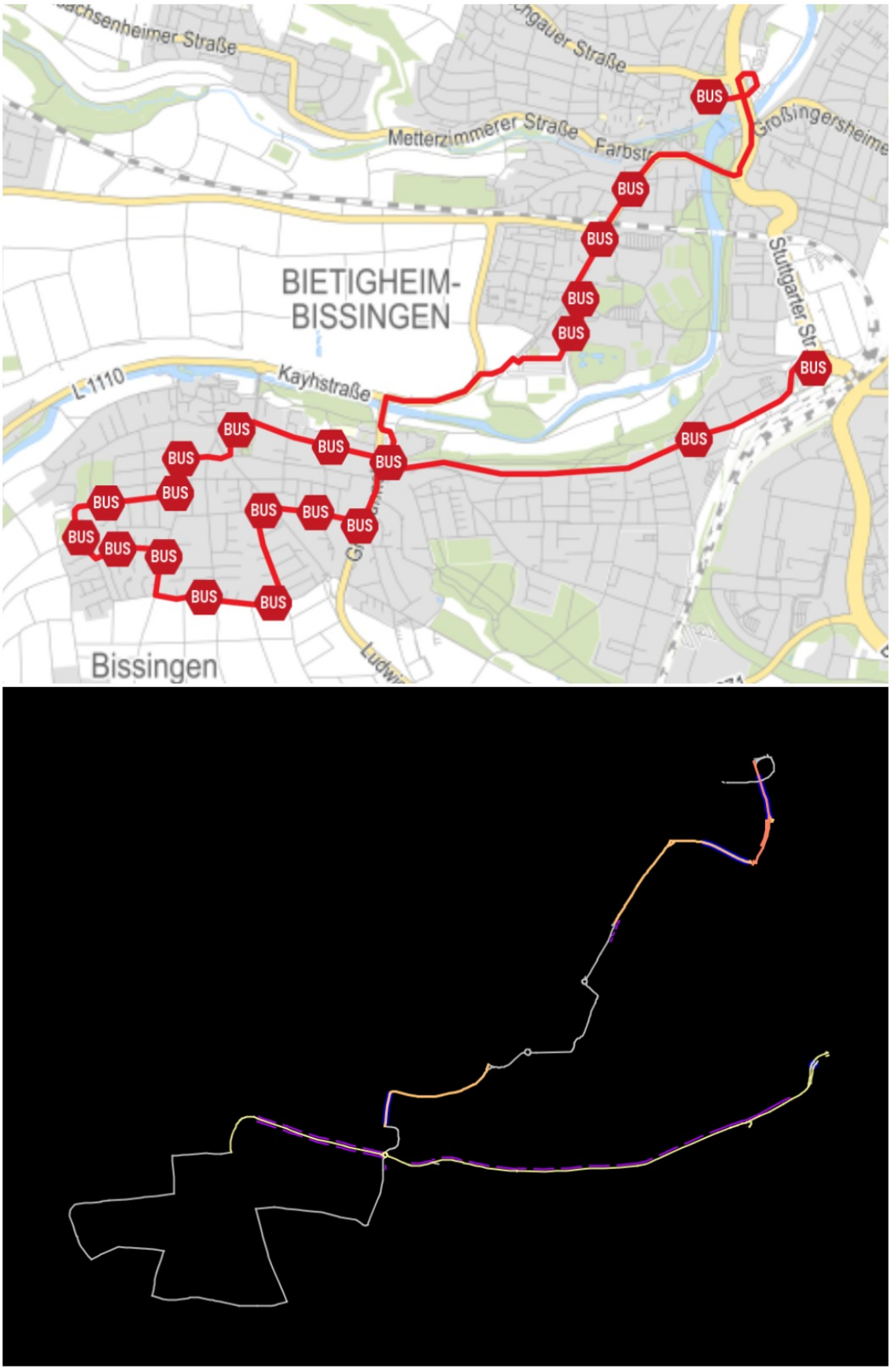
Obr. B.2: Linka 558 - referencia/vizualizácia aktuálne zaevidovanej linky; zdroj @OpenStreetMap contributors. Späť na sekciu 6.3.



Obr. B.3: Vizualizácia vybraných iterácií z procesu normalizácie a klastrovania linky 558. Vizualizácie spodnej polovice vznikli na dátach, ktoré boli výstupom algoritmu gestalt8 s komparačným algoritmom ratio(). Späť na sekciu 6.3.

Obr. B.4: Záverečný model linky vizualizovaný v nástroji JOSM pre overenie spätnej kompatibility OSM formátu; chyba normalizácie 5.69% (výstup manuálneho testovania normalizácie na danom konkrétnom úseku); použitá funkcia gestalt8 - ratio(). Späť na sekciu 6.3.

Obr. B.5: Referencia (hore) a výstupný model linky 551 vizualizovaný v JOSM (dole); chyba normalizácie 4.11%. Späť na sekciu 6.3.



Obr. B.6: Referencia (hore) a výstupný model linky 556 vizualizovaný v JOSM (dole); chyba normalizácie 1.84%. Späť na sekciu 6.3.