ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA ELEKTROTECHNICKÁ

DIPLOMOVÁ PRÁCA

Analýza dopravy na základe nameraných dát plávajúcich vozidiel

Študijný program:	Elektrotechnika a informatika
Katedra:	Katedra riadiacej techniky
Vedúci diplomovej práce:	Ing. Michal Kutil, Ph.D.

Praha 2011

Bc. Marek RIEDEL

Prehlasujem, že som predloženú prácu vypracoval samostatne a že som uviedol všetky použité informačné zdroje v súlade s Metodickým pokynom o dodržiavaní etických princípov pri príprave vysokoškolských záverečných prác.

V Prahe dňa 3. januára 2011 Bc. Marek Riedel

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra řídicí techniky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Bc. Marek Riedel**

Studijní program: Elektrotechnika a informatika (magisterský), strukturovaný
Obor: Kybernetika a měření, blok KM1 - Řídicí technika

Název tématu: **Analýza dopravy na základě naměřených dat plovoucích vozidel**

Pokyny pro vypracování:

1. Seznamte se s metodami měření městské dopravy pomocí plovoucích vozidel a následným zpracováním naměřených dat.
2. Sestavte matematický model dopravní oblasti na základě vstupních naměřených dat, s ohledem na jejich budoucí využití pro potřeby modelování a optimalizace.
3. Navrhněte a implementujte aplikaci pro ověření výsledků.

Seznam odborné literatury:

Dodá vedoucí práce

Vedoucí: Ing. Michal Kutil

Platnost zadání: do konce letního semestru 2010/2011

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

L.S.

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 4. 2. 2010

Ďakujem vedúcemu diplomovej práce Ing. Michalovi Kutilovi, Ph.D. za odborné vedenie a pomoc, ktorú mi poskytol pri vypracovaní diplomovej práce. Zároveň by som rád poďakoval všetkým, predovšetkým rodine, ktorí ma po celú dobu štúdia podporovali.

Abstrakt

Cieľom tejto práce bolo analyzovať namerané GPS dáta pomocou plávajúcich vozidiel a na ich základe vytvoriť matematický model oblasti vo forme orientovaného ohodnoteného grafu. V práci sú popísané postupy filtrácie dát, spresňovanie nameraných dát (mapovací algoritmus), výber pracovných dát (zhlukovací algoritmus), vyhľadávanie križovatiek (vrcholy grafu), ulíc (hrany grafu). Mapové podklady pre vizuálne overenie výsledkov analýz sa sťahujú vo forme PNG obrázkov zo serveru Google Maps. Výsledky analýzy a dáta pre matematický model oblasti sa ukladajú do navrhnutej databázy.

Abstract

The aim of this study was to analyze GPS data measured by floating vehicles and to create a mathematical model of the area in form of oriented weighted graph. The paper describes data filtering, measured data refinement (map-matching algorithm), working dataset selecting (clustering algorithm), intersections finding (vertices), street finding (edges). Map sources for visual verification of the results are downloaded in the form of PNG images from Google Maps server. Results of data analysis and the mathematical model of the area is stored in the database.

1 Obsah

1 Obsah **5**
2 Zoznam ilustrácií **7**
3 Zoznam tabuliek **9**
4 Zoznam skratiek a značiek **10**
5 Slovník termínov **11**
6 Úvod **12**
7 Súčasný stav riešenej problematiky **13**
7.1 Spresňovanie merania polohy 13
7.2 Získavanie pracovných dát 18
7.3 Matematický model máp 19
8 Spracovanie vstupných dát **21**
8.1 Vstupné dáta 21
8.2 Mapový podklad Google Maps 22
8.3 Mercatorova projekcia 22
8.4 Porovnanie Windows Forms a WPF na vykreslenie vstupných dát 24
8.5 Filtrácia vstupných dát 26
8.5.1 Určenie následnosti vstupných bodov 27
8.5.2 Rozlíšenie ulíc z mapového podkladu 28
8.5.3 Bresenhamov algoritmus 29
8.5.4 Voľba priblíženia mapového podkladu 30
8.5.5 Mapovací algoritmus bodov na ulicu 31
8.6 Minimalizácia bodov – zhlukovanie 33
8.6.1 Porovnanie algoritmov Fuzzy K-Means, Fuzzy C-Means 33
8.6.2 Voľba počtu zhlukov 34
8.6.3 Priemerovanie zhlukov 35
8.7 Spojovanie zhlukov 38
8.8 Vyhľadávanie križovatiek 39
8.9 Vyhľadávanie ulíc 42
9 Popis aplikácie **47**
9.1 Hlavné okno 47
9.1.1 Hlavné menu 47
9.1.2 Aktívna zobrazovacia oblasť 48

9.1.3 Menu aplikácie ... 48
9.2 Stavové informačné okná ... 49
10 Databáza ... 50
10.1 Vytvorenie aplikačnej databázy ... 50
10.1.1 Použitie dodaného SQL skriptu ... 50
10.1.2 Pripojenie dodanej predspracovanej databázy ... 53
10.2 Nastavenie spojenia ... 54
10.2.1 Nastavenie spojenia pre skompilovanú aplikáciu ... 54
10.2.2 Nastavenie spojenia v prostredí Microdoft® Visual Studio® 2010 ... 55
10.3 Štruktúra databázy ... 56
10.3.1 Tabuľka receiver ... 57
10.3.2 Tabuľka point ... 57
10.3.3 Tabuľka intersection ... 58
10.3.4 Tabuľka intersectionExit ... 58
10.3.5 Tabuľka street ... 58
10.3.6 Tabuľka streetJourney ... 58
11 Systémové požiadavky ... 60
11.1 Hardvér ... 60
11.2 Softvér ... 61
12 Záver ... 62
13 Zoznam použitej literatúry ... 64
14 Prílohy ... 67

2 Zoznam ilustrácií

Obr. 1 Bernstein a Kornhauser [1]: mapovanie krivka-krivka 16
Obr. 2 Porovnanie dvoch zdrojov máp z hľadiska presnosti [22] 17
Obr. 3 Greenfeldov [7] vážený topologický mapovací algoritmus 18
Obr. 4 Analógia Mercatorovej projekcie 23
Obr. 5 Mercatorova projekcia 24
Obr. 6 Zobrazenie 292 000 záznamov na mapovom podklade 26
Obr. 7 vstupných dát (urban canyons) 27
Obr. 8 Nájdenie ulíc z mapového podkladu 29
Obr. 9 Bresenhamov algoritmus rastrového vzorkovania úsečky 30
Obr. 10 Veľké priblíženie 31
Obr. 11 Chybné mapovanie v mieste križovatky 32
Obr. 12 Správne mapovanie v mieste križovatky 32
Obr. 13 Namapované body(modrá) pri oddialení 33
Obr. 14 Nájdené zhluky metódou Fuzzy C-Means 34
Obr. 15 Chybný počet zhlukov (čierny kruh) 35
Obr. 16 Nájdené zhluky v miestach s veľkým počtom bodov 36
Obr. 17 Problém určenia príslušnosti bodu pri priemerovaní 36
Obr. 18 Fuzzy C-Means výstup po spriemerovaní 37
Obr. 19 Príslušnosť bodov k stredu zhluku 38
Obr. 20 Prepojenie nesusedných stredov zhluku 39
Obr. 21 Správne prepojenie susedných stredov zhlukov 39
Obr. 22 Miesta, ktoré by mohli predstavovať križovatku 40
Obr. 23 Nájdené križovatky 41
Obr. 24 Dialógové okno križovatky 42
Obr. 25 Prejazdy ulicami rozdelené križovatkou 43
Obr. 26 Okno ulice so zoznamom prejazdov 45
Obr. 27 Graf rýchlosti prejazdu ulicou 46
Obr. 28 Hlavné okno aplikácie 47
Obr. 29 Príklady informačných dialógových okien 49
Obr. 30 Vytvorenie novej prázdnej databázy 50
Obr. 31 Zadanie názvu a umiestnenia novej databázy 51
Obr. 32 Otvorenie súboru s SQL skriptom 52
Obr. 33 Spustenie SQL skriptu 52

Obr. 34 Pripojenie predspracovanej databázy 53
Obr. 35 Vloženie predspracovanej databázy 54
Obr. 36 Nastavenie databázového spojenia v prostredí Ms Visual Studio® 2010..... 55
Obr. 37 Štruktúra databázy 56

3 Zoznam tabuliek

Tab. 1 Použitý hardvér 60
Tab. 2 Minimálne požiadavky na hardvér 60
Tab. 3 Použitý softvér 61
Tab. 4 Minimálne softvérové požiadavky 61

4 Zoznam skratiek a značiek

.NET („dotnet“ podľa anglického dot NET = bodka NET, NET pochádza z network, sieť) je zastrešujúci názov pre súbor technológií v softvérových produktoch, ktoré tvoria celú platformu, ktorá je dostupná nielen pre Web, Windows a Pocket PC. ...

WPF (Windows Presentation Foundation) grafický subsystém .NET platformy

MVC (Model View Controller) je softvérová architektúra, ktorá oddeľuje dátový model aplikácie, užívateľské rozhranie a riadiacu logiku do troch nezávislých komponent tak, že úprava niektorej z nich má minimálny vplyv na ostatné.

WWW (World Wide Web) je globálny informačný priestor, ktorý slúži na distribúciu informácií vzájomne medzi používateľmi.

HTTP (Hypertext Transfer Protocol) je primárna metóda prepravy informácií na world wide webe (WWW). Pôvodný účel bol poskytovať prostriedky pre publikáciu a získavanie HTML stránok.

API (Application Programing Interface) aplikačné programové rozhranie

PNG (Portable Network Graphics – prenosná sieťová grafika) je grafický formát určený pre bezstratovú kompresiu rastrovej grafiky. Bol vyvinutý ako zdokonalenie a náhrada formátu GIF, ktorý bol patentovo chránený.

SQL (Structured Query Language – štruktúrovaný dotazovací jazyk) je neprocedurálny jazyk pre popis databáz a pohľadov do nich. Umožňuje rozličnú manipuláciu s dátami uloženými v databázach.

XML (eXtensible Markup Language - rozšíriteľný značkovací jazyk) je jazyk vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium) ako pokračovanie jazyka SGML a HTML. Umožňuje jednoduché vytváranie konkrétnych značkovacích jazykov na rôzne účely a široké spektrum rôznych typov údajov.

5 Slovník termínov

Proces je postupnosť či rad časovo usporiadaných udalostí tak, že každá predchádzajúca udalosť sa zúčastňuje na determinácii nasledujúcej udalosti.

Mapovací algoritmus (map-matching algorithm) je postup, ktorý sa pokúsi upraviť polohu meraného GPS bodu tak, aby ležal na mapovom segmente existujúcej známej mapy oblasti.

On-line v kontexte tejto práce znamená, že ide o spracovávanie v reálnom čase

Off-line je opak on-line spracovávania – nie v reálnom čase.

Fuzzy logika je odbor matematiky odvodený z teórie fuzzy množín, v ktorom sa logické výroky ohodnocujú stupňom príslušnosti, ktorého hodnoty sú v intervale od 0 do 1 z oboru reálnych čísel.

Heuristika (z gréckeho *heuriskó* – nájsť, objaviť) znamená skusmé riešenie problémov, pre ktoré neexistuje presný algoritmus alebo metóda riešenia. Heuristické riešenie je často iba približné, založené na odhade. Heuristika nikdy nezaručuje najlepšie riešenie.

6 Úvod

Systémy monitorovania a sledovania pohybu vozidiel ponúkajú veľké množstvo užitočných informácií pre správu a plánovanie dopravných systémov, aktualizáciu existujúcich máp o nové cestné spojenia, plánovanie optimálnych trás navigačných systémov. Väčšina týchto systémov pracuje s virtuálnymi mapami, ktoré sú popísané matematickým modelom v tvare orientovaného ohodnoteného grafu.

Cieľom tejto práce je na základe nameraných GPS dát z plávajúcich vozidiel zostaviť matematický model dopravnej oblasti, v ktorej sa vozidla pohybovali.

V časti *Súčasný stav riešenej problematiky* sú uvedené nájdené pramene, v ktorých sa autori zaoberajú riešením čiastkových problémov pri spracovávaní nameraných GPS dát, akými sú spresňovanie získaných nameraných dát a redukcia vstupných dát výberom menšieho množstva pracovných dát.

V časti *Spracovanie vstupných dát* sú postupne popísané jednotlivé fázy spracovania vstupných dát v navrhnutej aplikácii, ktorých výsledkom je matematický model dopravnej oblasti pozostávajúci z nájdených polôh stredov križovatiek spolu s tvarom križovatiek (počet výjazdov s ich vzájomné rozmiestnenie) a zoznamu jednotlivých prejazdov vozidiel medzi križovatkami, ktoré spolu tvoria ulice.

Nadväzujúcou prácou by mal byť návrh aplikácie, ktorá sa na základe vytvoreného matematického modelu dopravnej oblasti pokúsi o efektívne riadenie dopravy v tejto oblasti.

7 Súčasný stav riešenej problematiky

Správny odhad rýchlosti, akou je možné sa dostať z miesta A do miesta B, je veľmi cennou informáciou z pohľadu zlepšovania navigačných systémov a optimalizácie navrhnutých trás, ktoré výrazne dokážu ušetriť čas a náklady. Takúto informáciu dokážeme získať zo systémov monitorujúcich pohyb vozidiel, ktoré nám môžu poskytnúť hlbšie informácie o správaní sa cestujúcich z dlhodobého hľadiska a jeho dôsledkoch na dopravný systém. Ako vedľajší produkt môžeme z týchto systémov získať napríklad informácie o množstve emisií exhalátov a hluku v danom uzle dopravného systému.

Statické meranie rýchlostí motorových vozidiel v uliciach nie je pre tento účel efektívne vzhľadom na potrebné množstvo senzorov rýchlosti a ich cenu. Pravidelné meranie rýchlostí pomocou veľkého množstva kalibrovaných vozidiel je z hľadiska ceny vybavenia vozidiel vo väčšine prípadov rovnako neefektívne. S klesajúcou cenou zariadení vybavenými GPS prijímačmi sa zlepšili možnosti zberu dát z dopravných systémov a následná dostupnosť k nameraným dátam. Dáta je možné získavať z celej dĺžky dopravných systémov oproti statickým senzorom a s väčšou hustotou oproti malému počtu kalibrovaných vozidiel. Nazbierané dáta sú pre ďalšie spracovanie ukladané do takzvaných databáz pohybujúcich sa objektov (Moving Objects Databases).

Štúdium databáz pohybujúcich sa objektov priťahuje stále väčšiu pozornosť komunity Geografických informačných systémov (GIS). Vo väčšine prípadov sú trajektórie pohybujúcich sa objektov zaznamenané ako konečné postupnosti časopriestorových bodov so vzorkovacou frekvenciou 0,1Hz až 1Hz. Existujú rôzne spôsoby rekonštrukcie trajektórie objektov z jednotlivých záznamov týchto bodov. Najčastejšie ide o jednoduché spájanie bodov alebo je použitá niektorá z interpolačných metód.

7.1 Spresňovanie merania polohy

Najväčším problémom, ktorý je nutné v databázach pohybujúcich sa objektov riešiť, je problém nepresnosti a neurčitosti pochádzajúci z rôznych zdrojov, akými sú napríklad chyby merania vznikajúce kvôli aktuálnym atmosférickým podmienkam, prípadne kvôli úzkym uliciam s vysokými budovami okolo (urban canyons). Minimalizácia chyby sa dosahuje napríklad použitím diferenčného merania GPS

polohy. K tomuto účelu sa využívajú statické pozemné stanice, ktorých presná poloha je známa a do svojho okolia vysielajú aktuálnu hodnotu rozdielu svojej nameranej GPS polohy a známej presnej polohy. Odčítaním chyby merania polohy od aktuálnej polohy je možné dosiahnuť presnosť v jednotkách metrov. V prípade integrovaných GPS zariadení vo vozidlách sa na spresnenie meranej polohy využívajú senzory rýchlosti, vzdialenosti, prípadne aj natočenia kolies vozidla. Výhodou týchto systémov je možnosť dostatočne presného zberu dát polohy vozidla aj v prípade dočasnej straty GPS signálu napríklad pri prejazdoch tunelom.

V navigačných zariadeniach najnižšej cenovej kategórie určených prevažne pre motorové vozidlá, sa na spresnenie polohy používajú algoritmy mapovania meranej GPS polohy na polohu najbližšej známej ulice (map-matching algorithm), ktorej poloha je uložená v pamäti zariadenia. Ide o on-line mapovací algoritmus, u ktorého je očakávaná veľká rýchlosť s nižším dôrazom na presnosť. Naopak v off-line mapovaní a spresňovaní nameraných GPS dát nám ide často o vysokú presnosť. K vysokej presnosti niektorých off-line algoritmov napomáha nielen informácia o bodoch predošlých, ale aj o bodoch nasledujúcich k aktuálne mapovanému bodu. Informáciu o bode budúcom v prípade on-line algoritmov nemáme.

Úlohou niektorých mapovacích algoritmov je spresňovanie existujúcich máp alebo vytváranie nových. Spoločnosť TomTom prišla na trh so zariadeniami vybavenými aplikáciou TomTom Map Share (zdieľanie máp). Slúži na opravy už existujúcich mapových podkladov poskytovaných spoločnosťou (zmena smeru jednosmeriek, obchádzky a uzávierky). V tomto prípade nedochádza k tvorbe nových máp. Opakom je napríklad webová stránka verejného projektu OpenStreetMap [18], kde dochádza k zberu súkromných GPS dát, pričom následné doplnenie nových ulíc závisí na zvážení a potvrdení užívateľov týchto stránok. Podobným projektom je YouTrace. Opäť umožňuje nahrávanie užívateľských GPS dát, pričom užívateľ ako protihodnotu dostane aktuálne dostupnú mapu. Dáta, ktoré sú vnútorne namapované na existujúce známe ulice, slúžia na ich spresňovanie, zvyšné dáta sa agregujú a vytárajú nové ulice. Viac informácií o projekte YouTrace je možné nájsť v [5].

Quddus s kolektívom [23] rozdelili mapovacie algoritmy na 4 skupiny:

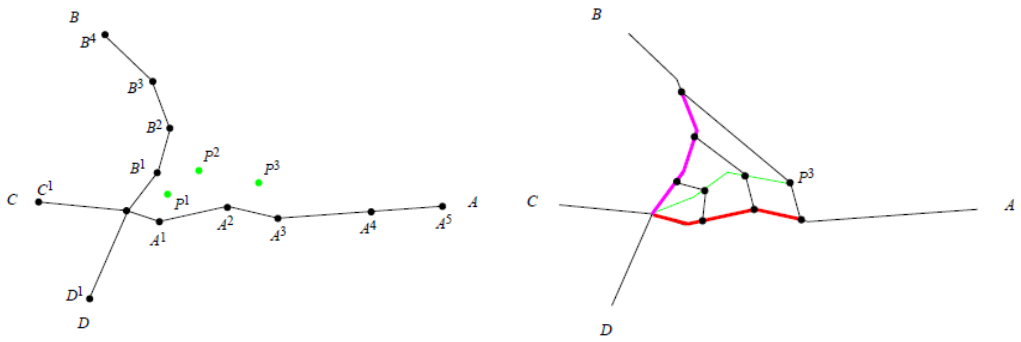
1. *Geometrické algoritmy* – pri mapovaní meraného bodu na existujúcu mapu sa prizerá len na geometrický tvar cestných segmentov s preferenciou najbližšieho segmentu k mapovanému bodu, čo často vedie k chybám. Vo všeobecnosti ich

vieme rozdeliť na algoritmy *bod-bod*, ktoré hľadajú najbližší bod cestného segmentu k mapovanému bodu. Druhou skupinou sú algoritmy *bod-krivka*, ktoré mapujú bod na najbližšie spojenie dvoch bodov cestného segmentu. Treťou skupinou sú algoritmy *krivka-krivka*, ktoré sú založené na algoritme *bod-bod* pre 2 po sebe merané polohy hľadaných bodov. Podľa [25] je najlepším z týchto algoritmov algoritmus *bod-krivka* a naopak najhorším algoritmus *krivka-krivka,* pretože algoritmus *bod-bod* generuje väčšinou veľké množstvo chýb a algoritmus *krivka-krivka* je veľmi náročný na výpočet a systémové prostriedky.

2. *Topologické algoritmy* – využívajú topologickú infomáciu (napr. smer jednosmerky) z existujúcich máp tvorených matematickým orientovaným grafom, kde vrcholy tvoria križovatky a orientované hrany tvoria ulice. Vo všeobecnosti tieto algoritmy ignorujú aditívne informácie dostupné z GPS zariadení, akými sú okamžitá rýchlosť a smer. Udržujú veľmi dobrú kontinuitu namapovaných bodov, ale sú značne citlivé na chybne namerané GPS polohy (outliers). Topologické algoritmy pracujú v dvoch krokoch. Najskôr sa vyhľadajú najbližšie cestné spojenia a následné sa medzi nimi rozhodne na základe známych topologických informácií. Quddus s kolektívom [23] poukázali, že tieto algoritmy môžu mať problémy v miestach križovatiek a vyžadujú ďalšie upresňovanie, ktoré je dostupné iba v off-line spracovaní. Preto sa tieto algoritmy nehodia pre on-line aplikácie v reálnom čase (real-time applications).
3. *Pravdepodobnostné algoritmy* – tieto algoritmy využívajú „chybovú oblasť“ v tvare kruhu, elipsy alebo štvorca okolo mapovaného bodu, v ktorom hľadajú cestné spojenia s prihliadnutím na ich smer a vzdialenosť. Niektoré algoritmy generujú „chybovú oblasť“ pre každý bod [5], iné iba pre body v oblastiach križovatky [23] pre zvýšenie výkonu algoritmu.
4. *Pokročilé algoritmy* – využívajú rôzne iné techniky oproti spomenutým alebo kombinujú jednotlivé jednoduché algoritmy. Často využívajú informáciu aj o aktuálnej rýchlosti, smere jazdy, kvalite meraných dát, počte dostupných GPS satelitov alebo aj dáta z iných senzorov polohy (napr. diferenčné meranie GPS). Najčastejším prístupom je využitie modelov fuzzy logiky, Kalmanových filtrov, rozšírených Kalmanových filtrov a ďalších hlavne pre prepojenie GPS dát

s dátami z ostatných senzorov pred samotným začatím mapovania bodov [17][26].

Príklad geometrického mapovacieho algoritmu *krivka-krivka* je na Obr. 1.



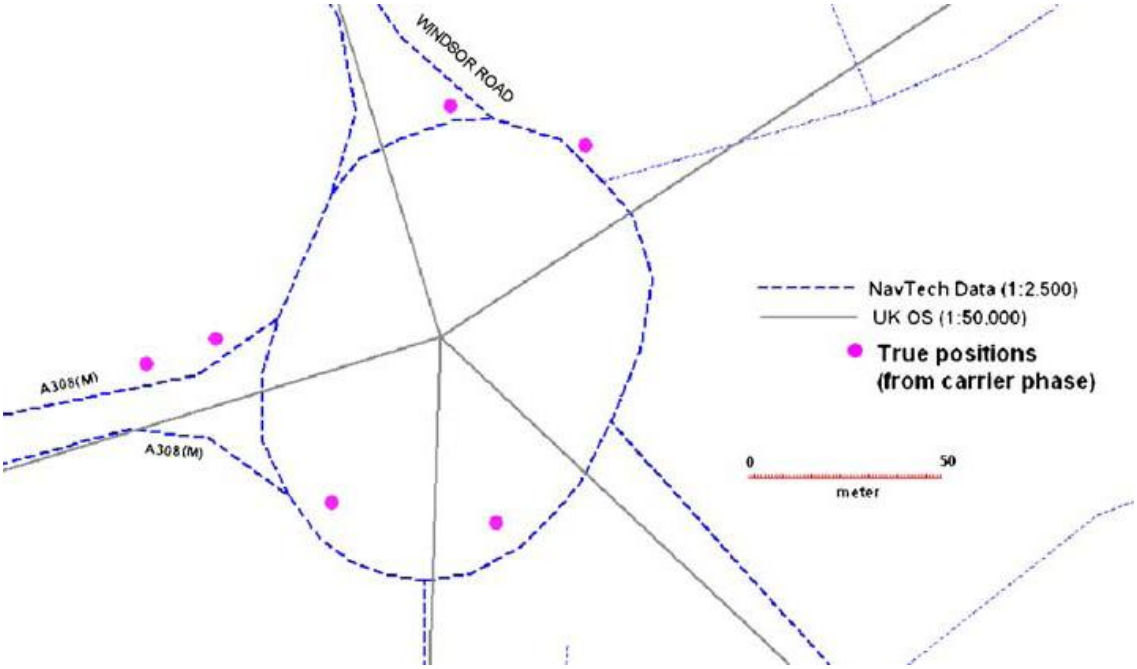
Obr. 1 Bernstein a Kornhauser [1]: mapovanie krivka-krivka

Počiatočný bod P_0 trasy je v strede križovatky. Namerané body trasy $P_1 - P_3$ sú označené zelenou. Prvým krokom je spojenie nameraných bodov. Následne sa nájdu priľahlé segmenty ulíc. Segmenty ulíc sa rozdelia na rovnako dlhé úseky tak, aby počet úsekov zodpovedal počtu nameraných bodov mimo križovatky $P_1 - P_3$. Následne sa spočítajú pre každý nájdený bod vzdialenosti od segmentov ulíc, nižšia hodnota súčtu vzdialeností rozhoduje o smere mapovania k segmentu ulice.

Autonómne off-line vytváranie nových máp na základe informácii z existujúcej mapy predstavili Câmara s tímom [2]. Ich algoritmus nazvali M-GEMMA a vznikol spojením vylepšeného Marchalovho algoritmu [16], ktorý umožňuje prácu s neúplnými topologickými mapami a Genetického mapovacieho algoritmu (GEnetic Map Matching Algorithm - GEMMA), ktorý sami navrhli.

Bernstein a Kornhauser [1] popísali niekoľko geometrických algoritmov pre mapovanie nameraných bodov na sieť reprezentujúcu dopravný systém ulíc. Z ich štúdie sú zjavné 2 hlavné závery. Prvým je náročnosť úlohy mapovania. Mapovanie *bod-bod* alebo *bod-krivka* nebude fungovať dobre, ak sú chyby merania už v sieti reprezentujúcej dopravný systém alebo sú značné chyby merania v nameraných bodoch. V týchto prípadoch je nutné použiť zložitejšie algoritmy mapovania. Druhým záverom je nutnosť zahrnutia topologických informácii pre spresňovanie nameraných GPS

bodov. Bernstein a Kornhauser preukázali, že čím je väčší dôraz kladený na presnosť topologických informácii, tým presnejšie sa chovajú navrhnuté mapovacie algoritmy.

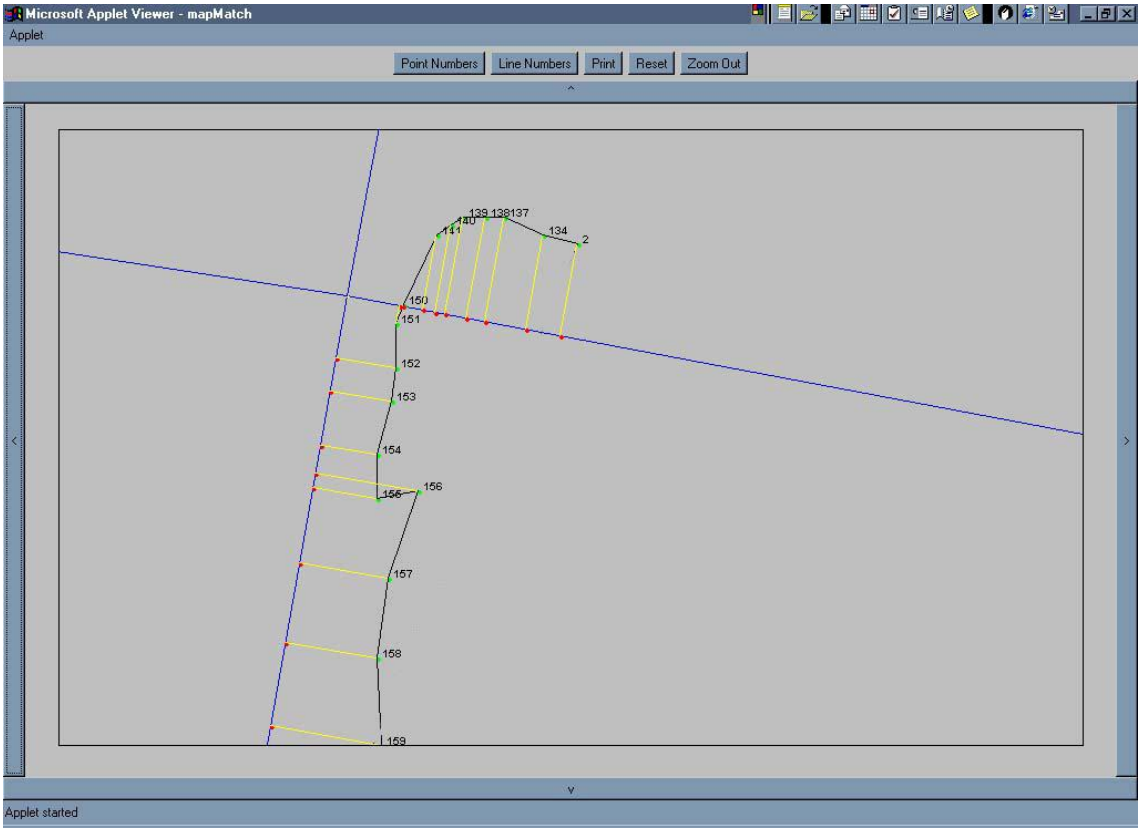


Obr. 2 Porovnanie dvoch zdrojov máp z hľadiska presnosti [22]

White s kolektívom [27] tiež popísal niekoľko mapovacích algoritmov, pričom 4 z nich boli aj overené. Pretože najväčšie zmeny v trasách sú na križovatkách, táto štúdia kladie najväčší dôraz na problémy mapovania vznikajúce na križovatkách v mestách. Napriek overeniu ich algoritmov na niektorých reálnych situáciách, sami nedošli k jednoznačným obecným záverom, pretože ich predpoklady neboli overené na širšej a obecnejšej mestskej oblasti.

Greenfeld [7] zhodnotil niektoré už existujúce mapovacie algoritmy a navrhol vážený topologický algoritmus. Je založený na ohodnocovaní podobností medzi hlavnými charakteristikami topologických sietí ulíc a tvarom postupnosti nameraných bodov. Tento algoritmus si dokáže relatívne dobre poradiť aj so značnými chybami nameraných dát. Tento mapovací algoritmus patrí medzi algoritmy využívajúce informáciu o predošlej polohe mapovaného bodu. Kľúčovým problémom takto založených algoritmov je správne namapovanie predchádzajúceho bodu. Ak sa prvý bod namapuje na nesprávny segment ulice, všetky nasledujúce body sa budú rovnako chybne mapovať. Quddus s kolektívom [24] rozšírili tento algoritmus o kombináciu

algoritmov od Bernsteina a Kornhausera [1], pričom predpokladali použitie spresneného diferenčného merania GPS polohy.



Obr. 3 Greenfeldov [7] vážený topologický mapovací algoritmus

7.2 Získavanie pracovných dát

Jednou z najbežnejších techník získavania pracovných dát z nameraných vzoriek je použitie zhlukovacích metód. Táto metóda rozdelí vstupné dáta do samostatných zoznamov tak, že všetky body v zozname sú si navzájom „podobné“ a súčasne sa „odlišujú“ od ostatných bodov z iných zoznamov. V kontexte pohybujúcich sa objektov vytvorí zhlukovacia metóda zoznamy bodov, ktoré tvoria rovnakú trajektóriu pohybu. Zhlukovacie algoritmy môžeme rozdeliť do 3 skupín:

1. *Segmentačné algoritmy (partitioning clustering)* – väčšina z nich je založená na známom zhlukovacom algoritme K-Means. Tento algoritmus musí mať dopredu známy počet výsledných nájdených zhlukov, čo v niektorých prípadoch nie je možné dopredu odhadnúť. Pre odhad počtu zhlukov bolo vytvorených pre rôzne situácie rôzne heuristiky.

2. *Algoritmy založené na hustote (density-based clustering)* – zhluky sú vytvárané na základe hustoty (množstva) susedných bodov. Využívajú sa na extrakciu často navštevovaných bodov [6][19]. Vyžadujú kontinuálne merané dáta bez prerušení.
3. *Algoritmy založené na čase (time-based clustering)* – hľadajú zhluky na základe časovej osi. Zhlukuje miesta s časom ležiacim v rámci určitého časového pásma. Vhodné na extrakciu často navštevovaných miest za predpokladu nepretržitého zberu dát [10].

Výber vhodnej zhlukovacej metódy nie je vždy jednoznačný. Skúmaním chovania samotných chodcov a následnou predikciou ich ďalších krokov sa zaoberali v [8]. Cieľom zhlukovacej metódy bolo nájsť pravidelne navštevované miesta. Segmentačné zhlukovanie založené na metóde K-Means neprichádzalo do úvahy, pretože pred samotným vyhľadávaním navštevovaných miest nie je známy ich počet. Pri tejto metóde sú všetky body priradené k niektorému zo zhlukov, takže k často navštevovaným bodom by sa priradili aj body, ktoré predstavujú cestu chodca k danému miestu. Zhlukovanie založené na hustote nie je napriek svojim niektorým výhodám vhodné pre použitie s dátami nameranými pomocou GPS zariadení kvôli častým výpadkom signálu. Za najvhodnejšiu metódu sa ukázalo zhlukovanie na základe času. Algoritmus však musel byť rozšírený tak, aby dokázal fungovať v prípade výpadku meraných dát.

Zhlukovacími metódami vhodnými pre spracovanie nameraných GPS dát pohybujúcich sa objektov (vozidiel) sa zaoberal napríklad Ing. Milan Janeček [9]. Sú nimi segmentačné zhlukovacie metódy, ktorých výsledkom je zoznam polôh zhlukových centier a bodov k nim prislúchajúcich. Vhodným spojením zhlukových centier získame akúsi stredovú čiaru cesty (road centerline), ktorá v sebe nesie informáciu o základnom tvare tejto cesty.

7.3 Matematický model máp

Na úvod uvedieme niekoľko základných termínov. GPS *body* sú prvkami priestoru $\mathbb{R} \times \mathbb{R}$ a *cestné segmenty* sú tvorené dvojicami týchto *bodov*. *Časový bod* $p = (x, y, t)$ má globálne GPS súradnice x a y spolu s časovou značkou $t \epsilon \mathbb{R}$, ktorou je absolútny čas

meraného bodu. *Časový cestný segment* je tvorený dvojicou *časových bodov*. Trasa T je usporiadaná postupnosť *časových bodov* $p_1 = (x_1, y_1, t_1), \dots, p_n = (x_n, y_n, t_n)$ a to tak, že čas t_i vzostupne rastie. *Časová cesta* $P = s_1, \dots, s_{n-1}$ je pridružená usporiadaná postupnosť *časových cestných segmentov* $s_i = (p_i, p_{i+1})$. Okamžitá rýchlosť v jednotlivých *časových cestných segmentoch* je priamym dôsledkom predchádzajúcich definícií.

Orientovaný graf $G = (V, E)$ je definovaný pomocou množiny vrcholov V a množiny hrán E, ktoré tvoria usporiadané dvojice $e = (v_i, v_j), \ v_i \epsilon V$. *Trasový graf* $G_T = (V, E, d, t)$ je ohodnotený orientovaný graf definovaný takými vrcholmi $v = (x_v, y_v) \epsilon V$, ktoré patria do trasy T, hranami $e = (u, v) \in E$, kde u, v sú 2 po sebe nasledujúce časové body $p_u = (x_u, y_u, t_u)$ a $p_v = (x_v, y_v, t_v)$ v trase T, $d(e) = \|u - v\|_2$ je Euklidova vzdialenosť bodov hrany a $t(e) = t_v - t_u$ je časový rozdiel bodov hrany.

Cestovný graf (matematický model oblasti z nameraných GPS dát) tvorí *trasový graf*, ktorý je doplnený o informáciu o križovatkách I na tejto trase. Pre jednoduchosť sú križovatky tvorené súradnicami jediného bodu (stredom križovatky). Križovatkový bod $(x_I, y_I) \epsilon I$ nemá časovú značku. Množina trás, ktoré majú aspoň počiatočný bod alebo koncový bod v niektorej z križovatiek $i \in I$ tvoria jednotlivé prejazdy ulicou.

Úloha zostavenia matematického modelu mapy oblasti z nameraných GPS dát sa skladá z už spomenutého spresňovania nameraných dát, výberu (zhlukovania) pracovných dát a následným vyhľadávaním cestných segmentov a trás. Následnými algoritmami vyhľadávania križovatiek a ulíc sa zostaví cestovný graf s rôznymi prídavnými informáciami [4][9][10] [28].

8 Spracovanie vstupných dát

Cieľom tejto práce je na základe vstupných nameraných GPS dát zostaviť matematický popis oblasti, v ktorej sa plávajúce vozidlo (vozidlo vybavené GPS záznamovým zariadením) pohybovalo.

Na overenie výsledkov bola vytvorená aplikácia, ktorá načíta vstupné dáta, vyfiltruje ich pomocou mapovacieho algoritmu, zostaví matematický model oblasti ako ohodnoteného orientovaného grafu vyhľadaním križovatiek (vrcholov) a ulíc medzi nimi (hrán).

8.1 Vstupné dáta

Vstupné GPS dáta aplikácie sú uložené v textovom súbore v DOS (CR/LF) formáte s ASCII kódovaním. Príklad vstupných dát:

```
31225022 51.4870911256 -0.133118955418 28.21 130.92 3 1
31225023 51.4870119585 -0.132961794733 27.8 129.32 3 1
31225024 51.4869292711 -0.132802287116 27.6 128.96 3 1
31225025 51.4868490981 -0.132642360403 27.83 129.49 3 1
31225026 51.4867651115 -0.132480673491 28.95 131.12 3 1
31225027 51.4866755927 -0.132316974922 29.83 132.24 3 1
31225028 51.4865816316 -0.132153192534 29.88 132.91 3 1
```

K vstupným dátam nie je dostupná informácia o presnom význame uložených dát, ale na prvý pohľad je zrejmé, že oddeľovačom jednotlivých záznamov je koniec riadku a oddeľovačom nameraných hodnôt záznamu je medzera.

V druhom a treťom stĺpci je uložená zemepisná šírka a dĺžka GPS súradníc. Prvý stĺpec dát by mohol predstavovať ID záznamu, ale po prevedení dát do databázy sa ukázalo, že dáta obsahujú 526 z celkových 392 195 záznamov s rovnakou hodnotou prvého stĺpca, čo by odporovalo podmienke jedinečnosti ID. Ďalším skúmaním sa zistilo, že ide o čas v sekundách, čo potvrdil aj výpočet rýchlosti z dvoch po sebe idúcich záznamov polohy. Porovnaním vypočítanej rýchlosti so štvrtým stĺpcom zistíme, že v tomto stĺpci je uložená okamžitá rýchlosť. Čas je meraný ako relatívna doba, ktorá uplynula od neznámej časovej značky. Ak by to bola UNIX časová značka (1.1.1970), prvý záznam by vznikol 28. 12. 1970 9:36:08 a posledný záznam 15. 4. 1971 7:05:55. Tento čas pôsobí nereálne, preto za časovú známku zvolíme dátum 1.1.2005 00:00:00.

Piaty stĺpec obsahuje hodnoty 0 až 360. Ide o okamžitý smer jazdy v stupňoch. Výpočtom je to opäť možné overiť. Význam posledných dvoch stĺpcov sa nepodarilo jednoznačne určiť. Posledný stĺpec má vždy hodnotu 1, pravdepodobne sa bude jednať o ID zariadenia. Predposledný stĺpec nadobúda hodnoty 1 až 10. Takúto hodnotu nadobúda v GPS zariadeniach najčastejšie počet dostupných GPS vysielačov.

Pri prenose nameraných GPS dát v reálnom čase z mobilných zariadení sa snažíme minimalizovať množstvo prenášaných dát. Pre záznam polohy stačí iba informácia o čase a okamžitej polohe. Zvyšné hodnoty je možné dopočítať z týchto informácii. Preto sa pre ďalšie spracovanie bude predpokladať, že máme k dispozícii iba prvé tri stĺpce vstupných dát.

8.2 Mapový podklad Google Maps

Vizuálne overenie vstupných dát umožňuje preloženie jednotlivých bodov na mapový podklad. Za mapový podklad bol zvolený obrázok vo formáte PNG, ktorý sa získava prostredníctvom http protokolu zo serveru Google Maps. Oproti mapám napríklad zo serveru Bing nie je nutná registrácia užívateľa.

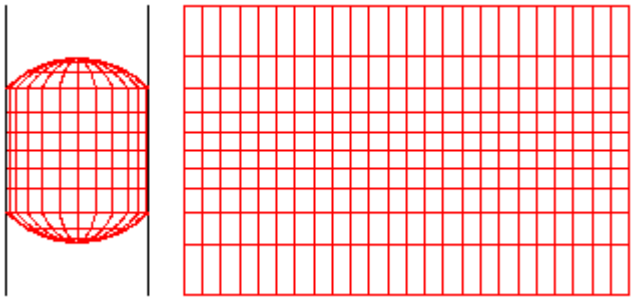
Obrázkové mapové podklady sa získavajú prostredníctvom takzvaného statického rozhrania Google Maps API, ktoré predstavuje URL adresa so vstupnými parametrami. Vstupnými parametrami sú GPS súradnice stredu mapového obrázku, rozmer (max. 640x640px), formát obrázku a zvolené priblíženie (zoom). Ďalšími vstupnými parametrami môžu byť napríklad body, ktoré sa vykreslia do obrázku. Vzhľadom na odporúčanú maximálnu dĺžku URL adresy do 2000 znakov, nie je možné prostredníctvom tohto rozhrania vykresliť veľké množstvo bodov. O vykreslenie bodov na mapový podklad sa musí starať samotná aplikácia.

8.3 Mercatorova projekcia

Projekciou vo všeobecnosti rozumieme zobrazenie priestoru väčšej dimenzie do priestoru nižšej dimenzie. V našom prípade pôjde o zobrazenie trojdimenzionálneho priestoru do roviny.

Mercatorova projekcia je valcová mapová projekcia, ktorú predstavil v roku 1569 belgický geograf a kartograf Gerardus Mercator. Stala sa štandardnou mapovou

projekciou pre námorníctvo, pretože zachovávala azimut medzi ľubovoľnými dvoma bodmi. Vykazuje však značné skreslenie vo vyšších zemepisných šírkach. Ako každá mapová projekcia má všetky poludníky a rovnobežky rovnobežné a navzájom kolmé. Aby to bolo možné dosiahnuť, dochádza k nerovnomernému rozťahovaniu v smere juh-sever, z čoho vyplýva, že nemá všade rovnakú mierku. Laickou analógiou tejto transformácie by mohlo byť nafúknutie balónu v tvare zeme vo vnútri valca, ako je zobrazené na obrázku Obr. 4.



Obr. 4 Analógia Mercatorovej projekcie

Rovnice, pre určenie polohy bodu [x, y] na Mercatorovej mape sú:

$$x = \theta - \theta_0$$

$$y = ln\left(tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right)$$

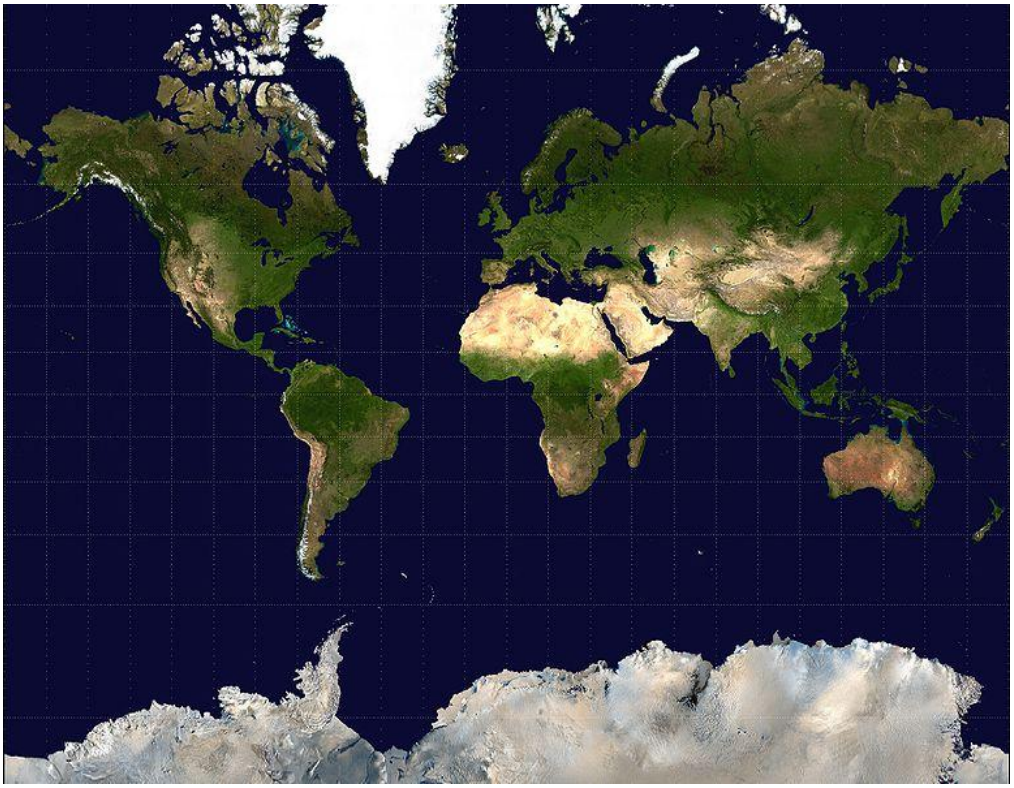
kde θ_0 je zemepisná dĺžka stredu mapy, θ je zemepisná dĺžka a φ zemepisná šírka bodu. Zodpovedajúce inverzné rovnice sú:

$$\theta = x + \theta_0$$

$$\varphi = tan^{-1}(sinh(y))$$

Google mapy (Google Maps) využívajú univerzálnu priečnu Mercatorovu projekciu (UTM - Universal Transverse Mercator coordinate system) pre zobrazenie mapových obrázkov. Mapa zeme je rozdelená na menšie štvorcové časti (tiles) podľa zvoleného priblíženia (zoom) s veľkosťou 256 pixelov. Vynechaná je len oblasť severného a južného pólu, presnejšie od hranice $\varphi = \pm 85{,}05113°$ zemepisnej šírky. Oblasti za

touto hranicou sú zobrazované odlišnými vzťahmi. Táto práca však nepočíta so spracovávaním bodov za touto hranicou, preto neuvádza bližšie detaily.



Obr. 5 Mercatorova projekcia

Pre zobrazenie nameraných bodov na mapovom podklade bolo nutné navrhnúť metódy na prevod sférických GPS súradníc do roviny pomocou Mercatorovej transformácie a následný prevod globálnej polohy v rovine na polohu bodu v pixeloch na obrázku mapového podkladu podľa zvoleného priblíženia (zoom) mapového podkladu. Neskôr bol navrhnutý aj reverzný prevod z polohy v pixeloch do sférických súradníc. Výhodou navrhnutých metód je ich všeobecné použitie s Google Maps API, rovnako ako aj s napríklad Bing API od Microsoftu. Mnohí poskytovatelia mapových podkladov totiž využívajú rovnakú Mercatorovu transformáciu, líšia sa len umiestnením počiatočného bodu súradnicového systému.

8.4 Porovnanie Windows Forms a WPF na vykreslenie vstupných dát

Úlohou užívateľského rozhrania aplikácie je zobrazenie vstupných dát, čiastočných a výsledných výsledkov analýz. Vzhľadom na množstvo zobrazených dát je vhodné

jednotlivé výsledky zobrazovať do samostatných vrstiev, ktoré je možné jednoducho skryť a znovu zobraziť. Pre vývoj aplikácie bol zvolený Microsoft .NET framework.

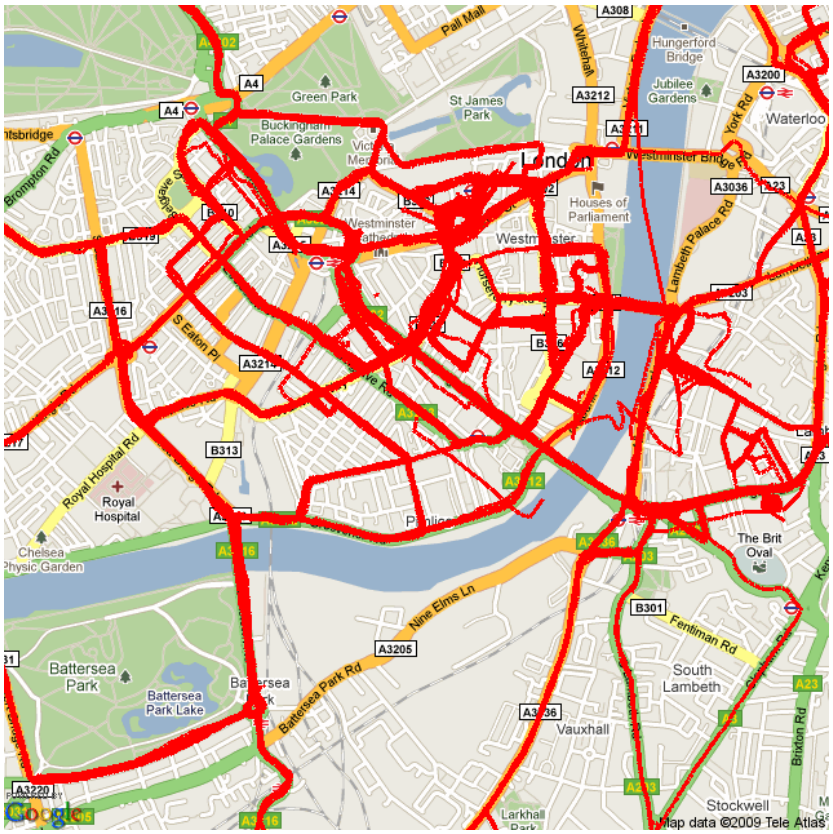
Prvý návrh programu bol vytvorený ako Windows Forms aplikácia s vykresľovaním bodov priamo do získaného mapového obrázku. Tento spôsob bol spomedzi 3 návrhov najpomalší a zároveň skrývanie vrstvy vyžadovalo opätovné pomalé prekreslenie obrázku. Vykreslenie 5000 bodov trvalo 4 sekundy.

Druhý návrh bol takmer rovnaký s prvým návrhom. Líšil sa iba tým, že sa namerané body vykresľovali do vygenerovaného transparentného obrázku, ktorý sa preložil cez mapový podklad. Rýchlosť vykreslenia ani záťaž systému počítača sa nezmenila. Zrýchlilo sa iba skrývanie vrstvy bez nutnosti prekresľovania obrázka.

Tretí návrh bol vytvorený ako Windows Presentation Foundation (WPF) projekt. WPF je grafický subsystém pre generovanie užívateľských rozhraní Windows-based aplikácií. WPF je navrhnuté tak, aby bolo nezávislé na zastarávajúcom grafickom GDI subsystéme, nahradil ho priamy prístup ku grafickým prostriedkom systému cez DirectX, ktorý umožňuje hardvérovú akceleráciu. Umožňuje nové vlastnosti ako je napríklad priehľadnosť, farebné gradienty alebo 3D grafiku. Za veľkú výhodu je možné považovať možnosť oddeleného vývoja grafického vzhľadu od samotnej logiky aplikácie (MVC model). Grafické rozhranie je písané prevažne v XAML (eXtensible Application Markup Language) jazyku, logika aplikácie je písaná v niektorom z .NET jazykov (C#, Visual Basic .NET), v našom prípade C#.

WPF návrh aplikácie vykreslil porovnávaných 5000 bodov zo vstupných dát za menej než 1 sekundu. Zmena polohy niektorých bodov nevyžaduje prekreslenie celej vrstvy, záťaž procesora klesla pri porovnávacom teste o viac než polovicu.

Rýchlosť odozvy WPF návrhu a jednoduchosť vývoja bola rozhodujúcim kritériom pre zvolenie tohto návrhu pre výslednú aplikáciu.

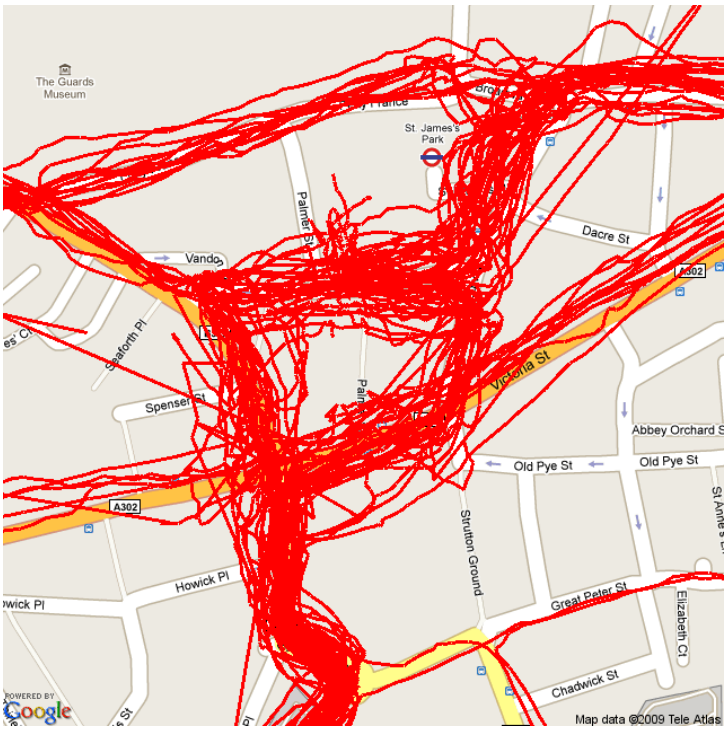


Obr. 6 Zobrazenie 292 000 záznamov na mapovom podklade

8.5 Filtrácia vstupných dát

Na príklade vykreslených vstupných dát na Obr. 7 je možné vidieť, že vstupné GPS dáta obsahujú šumy a chyby merania. Veľmi zreteľne je možné pozorovať vplyv takzvaného mestského kaňonu (urban canyon) na meranie polohy pomocou GPS zariadení. Mestský kaňon vzniká ako dôsledok vysokých budov pozdĺž úzkych mestských ulíc, kvôli ktorým nemajú GPS zariadenia dostatočný výhľad na GPS navigačné družice a dochádza k nepresnému určeniu polohu.

Pri meraní našich vstupných dát nebola použitá žiadna technika spresňovania merania. S dátami s takýmto rozptylom a šumami nie je veľmi možné presne pracovať. Preto aplikujeme filter, ktorý sa pokúsi odstrániť nerelevantné šumové dáta a zvyšné dáta preloží k sebe bližšie. Využijeme k tomu algoritmus mapovania bodov na body ulice (map matching algorithm). Pretože sa nepodarilo nájsť legálne voľne dostupnú digitálnu mapu Londýna, využijeme k tomu získaný mapový podklad získaný pomocou Google Maps API.



Obr. 7 Chyby vstupných dát (urban canyons)

8.5.1 Určenie následnosti vstupných bodov

Vzhľadom na veľké množstvo vstupných dát je z hľadiska výkonu výhodnejšie pracovať vždy len s časťou dát. Pracovať budeme vždy len s dátami patriacimi do zobrazenej časti mapového podkladu. Tieto dáta sa vždy nahrajú do pamäte z databázy po získaní nového obrázku mapového podkladu.

Prvú filtráciu vstupných dát dosiahneme určením predchodcu pre každý nameraný bod. Dva po sebe idúce body musia spĺňať:

1. Časový rozdiel dvoch po sebe idúcich bodov musí byť menší alebo rovný 3 sekundám.
2. Vzdialenosť medzi bodmi môže byť najviac 105 metrov a zároveň rýchlosť medzi bodmi nesmie byť väčšia než 35 m/s (126km/h)
3. Zmena okamžitej rýchlosti medzi bodmi nesmie byť väčšia než 25 m/s (90 km/h)

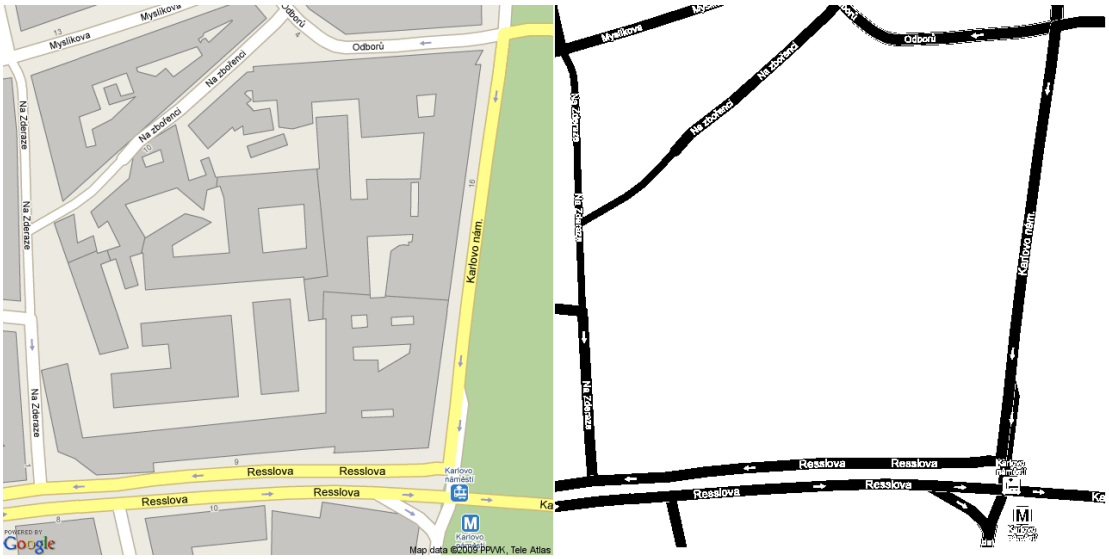
Prvá podmienka vylučuje spojenie bodov, ktoré vznikli napríklad meraním z predošlého dňa alebo došlo k výpadku merania. Tolerované sú najviac dvojsekundové výpadky. Druhá podmienka vylučuje body, ktoré boli merané v rovnakom čase na inom vzdialenejšom mieste. Touto podmienkou sa podarilo odchytiť napríklad body, ktorým

vychádzala okamžitá rýchlosť vyššia než je hranica nadzvukovej rýchlosti. Tieto body mohli vzniknúť meraním v inom zariadení, ale vzhľadom na ich nespojitosť sú považované za šum. Tretia podmienka vylučuje súslednosť bodov z hľadiska zrýchlenia. Monoposty formule F1 dosahujú pri akcelerácii zrýchlenie $14ms^{-2}$ a pri decelerácii $38ms^{-2}$. Môžeme predpokladať, že dáta v Londýne neboli merané takýmto vozidlom a ani iným športovým vozidlom, ktoré by dokázalo zrýchliť alebo spomaliť z rýchlosti 90km/h do 1 až 3 sekúnd. Táto podmienka odfiltrovala iba malý počet bodov. Po preložení na mapový podklad sa zistilo, že išlo o body merané v momente, keď vozidlo stálo. Zaznamenávali sa body v okruhu vozidla a niektoré odskočili do vzdialenosti 30 metrov od vozidla.

Po splnení týchto podmienok sa uloží do lokálnej pamäte aj do databázy informácia o predchodcovi každého bodu. Hľadanie predchodcu sa vykonáva zakaždým pri vykresľovaní vstupných dát. Ak už bod obsahuje predchodcu, vyhľadávanie sa preskočí pre zvýšenie rýchlosti aplikácie.

8.5.2 Rozlíšenie ulíc z mapového podkladu

Prvým krokom bude rozlíšenie ulíc a ostatných plôch mapového obrázku. K tomu bude nutné získať obrazové dáta z obrázku v PNG formáte. O dekompresiu formátu PNG a následný preklad vrstiev sa postará trieda *FormatConvertedBitmap*, ktorá sa objavila spolu s WPF subsystémom. Takto získame maticu veľkosti 639x639 pixelov (rozmer získavaného obrázku mapového podkladu), kde každý pixel je zakódovaný ako 32 bitové číslo skladajúce sa zo 4 bajtov. Vyššie 3 bajty majú význam farieb červená, zelená, modrá a najnižší bajt predstavuje alfa kanál (miera priehľadnosti). Google Maps obrázky využívajú pre ulice len niekoľko málo farebných kombinácii, ktoré sú bez tieňovania a majú teda vždy konštantnú farebnú kombináciu. Táto vlastnosť nám umožní veľmi rýchlo rozlíšiť ulicu. Menšou nevýhodou však je, že rovnaké farebné kombinácie využíva aj na vykreslenie iných topologických symbolov ako napríklad zastávok MHD. Príklad rozlíšených ulíc pre Karlovo náměstí v Prahe je na Obr. 8.



Obr. 8 Nájdenie ulíc z mapového podkladu

Ďalším krokom je presun rozptýlených mapových bodov na najbližšiu nájdenú ulicu (map-matching). Ak je rozptýleným bodom samostatne stojací bod, považuje sa za šum a nepresúva sa. Z dvoch po sebe idúcich bodov už je možné určiť smer jazdy a presunúť tieto body v smere kolmom na tento smer jazdy. Preto nájdeme z oboch bodov množinu pixelov, ktoré sú kolmé na smer pohybu a sú od presúvaného bodu vzdialené najviac o definovanú konštantu. V analytickej geometrii by sme hľadali úsečky kolmé na smer pohybu, ktoré začínajú v nájdených susedných bodoch. Úsečky obsahujú nekonečne veľa bodov, zatiaľ čo na rastrovom obrázku zloženého z pixelov môžu obsahovať len konečný počet bodov.

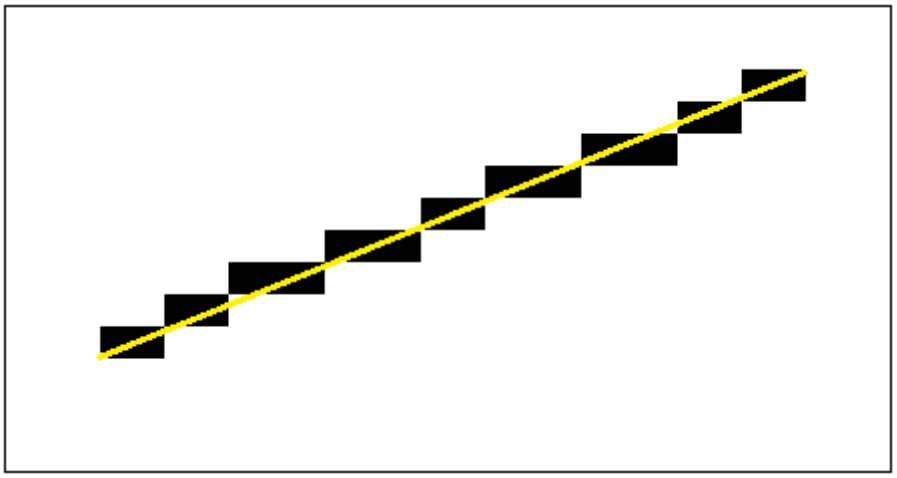
8.5.3 Bresenhamov algoritmus

Algoritmov na prevod nekonečného množstva bodov úsečky na konečný počet bodov rastrového obrázku existuje viac. Zvolíme Bresenhamov algoritmus, ktorý poznáme z jednoduchých kresliacich nástrojov, akým je napríklad aj Maľovanie (mspaint) operačného systému Windows.

Rastrový rozklad spojitej úsečky sa zakladá na vzorkovaní s konštantným krokom 1 pixel podľa osi x alebo y. Závisí to od sklonu úsečky reprezentovaného smernicou m. Ak je $|m| < 1$, tak úsečka má sklon k osi x menší ako 45°, a preto vzorkujeme podľa osi x s krokom 1 pixel. Ak je $|m| > 1$, vzorkujeme podľa osi y. Úsečka so smernicou 1

je diagonálna a vzorkujeme ju ľubovoľnou z osí. Os, podľa ktorej vzorkujeme, nazvime riadiacou osou.

Pre každú vzorkovanú polohu riadiacej osi spočítame prírastok v smere druhej vedľajšej osi. Celočíselným zaokrúhlením prírastku získame polohu pixelu v smere vedľajšej osi. Príklad výsledku vzorkovania je na Obr. 9.



Obr. 9 Bresenhamov algoritmus rastrového vzorkovania úsečky

8.5.4 Voľba priblíženia mapového podkladu

Keď už vieme určiť množinu pixelov predstavujúcich úsečku kolmú na smer pohybu vozidla, musíme si zvoliť, do akej vzdialenosti budeme vyhľadávať najbližšie ulice. Pri veľkom oddialení mapového podkladu budeme pracovať naraz s veľkým množstvom nameraných GPS bodov, ale zároveň stratíme presnosť určenia polohy ulice, pretože sa celá šírka ulice vmestí napr. do jednotiek pixelov. Naopak pri veľkom priblížení budeme pracovať s veľmi nízkym počtom bodov a všetky body patriace ulici sa nemusia zmestiť na mapový podklad (Obr. 10). Kompromisným riešením je približne stredná hodnota. Pre hodnotu tejto konštanty sú dopočítané všetky zvyšné konštanty mapovacieho algoritmu bodov na ulicu.

Obr. 10 Veľké priblíženie

8.5.5 Mapovací algoritmus bodov na ulicu

Ako už bolo spomenuté, mapovací algoritmus pracuje vždy s dvojicou po sebe nasledujúcich bodov. Pre každý bod sa najskôr overí, či daný bod už nebol namapovaný na ulicu. Ak už bol dvakrát namapovaný na ulicu, považujeme jeho namapovanie za úspešné a ďalšie mapovanie tohto bodu sa nevykonáva. Ak ešte nebol bod úspešne namapovaný, nájdeme jeho predchodcu. Ak nebol pri filtrácii v 8.5.1 nájdený predchodca, bod sa nemapuje.

Po získaní dvojice aktuálneho bodu a jeho predchodcu nájdeme z oboch bodov kolmice na smer pohybu vozidla. Potom pomocou Bresenhamovho algoritmu nájdeme zoznam pixelov, ktoré predstavujú úsečky s rovnakou smernicou, akú majú kolmice k smeru pohybu a počiatok majú vždy v jednom z bodov. Dĺžka zoznamu pixelov na jednu stranu je 35 pixelov, čo zodpovedá pre zvolené pracovné priblíženie vzdialenosti 55 metrov. Takto získame pre každý bod 2 zoznamy pixelov, ktoré budeme skúmať a porovnávať ich farebný kód s kódom ulice.

Bod sa budeme snažiť namapovať vždy k najbližšej nájdenej ulici (geometrický mapovací algoritmus *bod-bod*). Ak by sme pri hľadaní ulice brali do úvahy iba najkratšiu vzdialenosť bodu k ulici, v mieste križovatky by mohlo dôjsť k chybnému namapovaniu, ako je to možné vidieť na Obr. 11. Preto druhým kritériom mapovania bude aj dodržanie smeru úsečky namapovaných bodov so smerom vozidla. Tretím kritériom bude aj prihliadanie na polohu namapovanej polohy predchádzajúceho bodu,

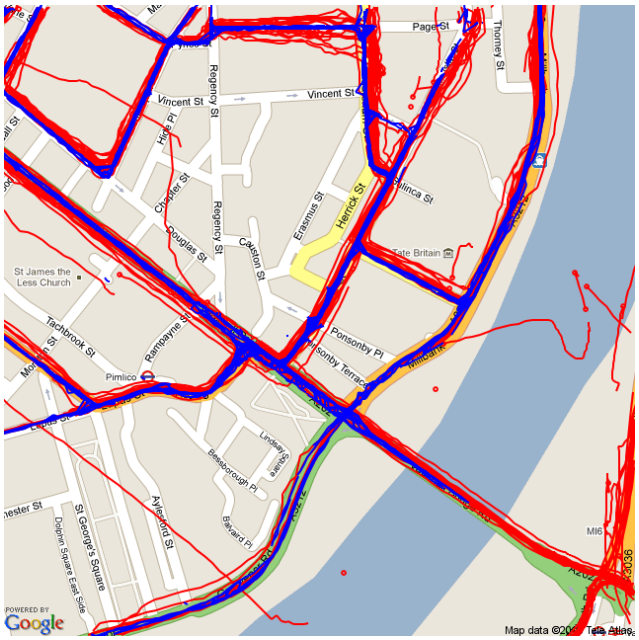
čo znamená, že preferovaný posun bodov bude v smere k namapovanému bodu predchodcu.

Každá namapovaná poloha bodu sa uloží do databázy spolu s počtom pokusov o namapovanie, aby nedochádzalo v budúcnosti k duplicitným pokusom o namapovanie a zvýšila sa tak rýchlosť aplikácie. Pretože mapovanie bodov prebieha len na základe mapového podkladu, nie je možné namapovať všetky body v databáze naraz. Aplikácia umožňuje spustiť automatický proces, ktorý vyhľadáva ešte nenamapované body a pokúša sa ich namapovať. Proces je založený na algoritme prehľadávania priestoru do šírky, kde za počiatok hľadania sa berie vždy bod, ktorý ešte nebol namapovaný úspešne, zároveň sa preferuje bod, u ktorého bol aspoň jeden pokus o namapovanie. Podmienka aspoň jedného pokusu o namapovanie nám zaručí preferenciu smeru posunu bodov mapovacieho algoritmu.

Obr. 11 Chybné mapovanie v mieste križovatky

Obr. 12 Správne mapovanie v mieste križovatky

Po namapovaní bodov na ulice môžeme predpokladať, že už máme dáta s dostatočnou presnosťou. V prípade, že už od začiatku vieme, že máme vstupné dáta namerané s dostatočnou presnosťou a bez šumov meraní, môžeme mapovanie ulíc vynechať. Mapovanie bodov na ulice je jediný proces, ktorý v našej aplikácii vyžaduje mapový podklad. Ďalšie kroky analýzy a tvorby virtuálnej mapy pracujú s namapovanými bodmi ako so vstupnými dátami s presne nameranými polohami.



Obr. 13 Namapované body(modrá) pri oddialení

8.6 Minimalizácia bodov – zhlukovanie

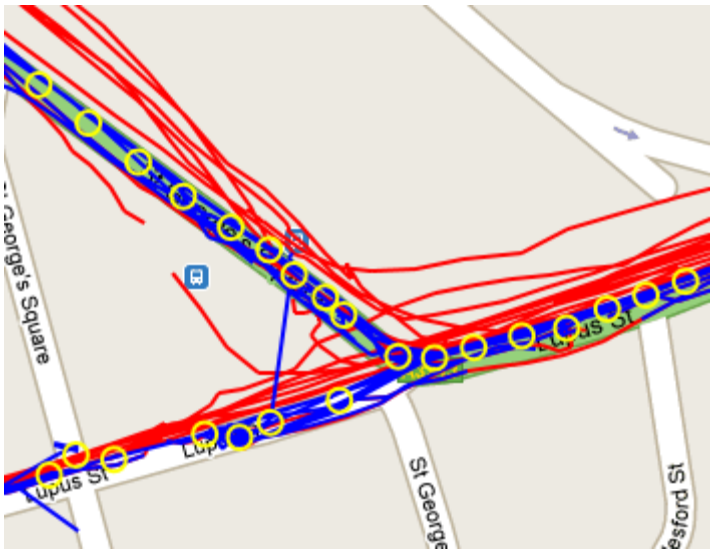
Vzhľadom na veľké množstvo vstupných dát je vhodné pre rýchlejšiu analýzu minimalizovať ich počet. Z výsledkov práce [9] vyplýva, že sa pre našu úlohu najlepšie osvedčili zhlukovacie algoritmy Fuzzy K-Means a Fuzzy C-Means.

8.6.1 Porovnanie algoritmov Fuzzy K-Means, Fuzzy C-Means

Zhlukovací algoritmus Fuzzy K-Means je podľa výsledkov [9] asi 5x rýchlejší než algoritmus Fuzzy C-Means. Preto prepíšeme tieto dva algoritmy z prostredia Matlab do C# kódu našej aplikácie a porovnáme. Po presnom prepísaní algoritmov boli nájdené hlavne v prípade Fuzzy C-Means zbytočné opakovacie slučky, ktoré bolo možné zlúčiť

a takto optimalizovať. Pre porovnanie si pripravíme 10 000 bodov, nastavíme počet hľadaných zhlukov na 500 a v kóde si pripravíme premennú, do ktorej si uložíme čas vykonávania algoritmu. Pre každý algoritmus opakujeme meranie päťkrát, aby sa vylúčila chyba prideľovania systémových prostriedkov od operačného systému.

V prípade Fuzzy K-Means bola nameraná priemerná doba zhlukovania 1925ms a v prípade Fuzzy C-Means bola priemerná doba 5717ms. Z meraní vyplýva, že je Fuzzy C-Means iba trikrát pomalší než Fuzzy K-Means. Pravdepodobne sa pod to podpísala spomínaná optimalizácia.



Obr. 14 Nájdené zhluky metódou Fuzzy C-Means

Porovnávaním oboch algoritmov sa ukázalo, že aj keď je Fuzzy K-Means rýchlejší, nie je veľmi vhodný pre zhlukovanie bodov ulíc, ktoré predstavujú pri našom zvolenom priblížení podlhovasté zhluky. Vzdialenosti medzi zhlukovými centrami boli nerovnomernejšie rozložené než v prípade Fuzzy C-Means (Obr. 14). Nerovnomernosť spôsobovala problémy hlavne v nadväzujúcich procesoch analýzy, preto bol za zhlukovací algoritmus zvolený Fuzzy C-Means.

8.6.2 Voľba počtu zhlukov

Druhým krokom návrhu zhlukovacieho algoritmu je správna voľba počtu zhlukov. K tomuto účelu môžeme využiť informáciu o počte nameraných bodov prislúchajúcich k získanému mapovému podkladu. V prípade, že zvolíme malý počet zhlukov, môže

dôjsť k nesprávnemu určeniu stredu zhlukov ako je to v príklade na Obr. 15, kde žlté body predstavujú namerané body a čierny bod predstavuje stred nájdeného jediného zhluku.

Obr. 15 Chybný počet zhlukov (čierny kruh)

Pre určenie počtu zhlukov sa v našom prípade najlepšie osvedčila deterministická voľba podľa počtu zhlukovaných bodov:

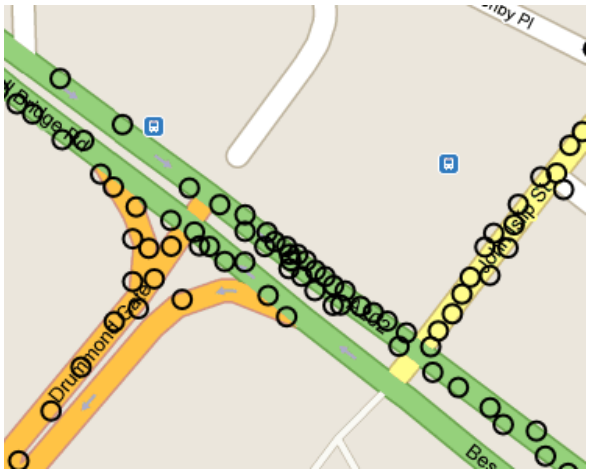
viac než 400 bodov: 1/15 celkového počtu zhlukovaných bodov

od 26 do 400: 1/6 celkového počtu zhlukovaných bodov

do 26: 1/4 celkového počtu zhlukovaných bodov, minimálne však 1

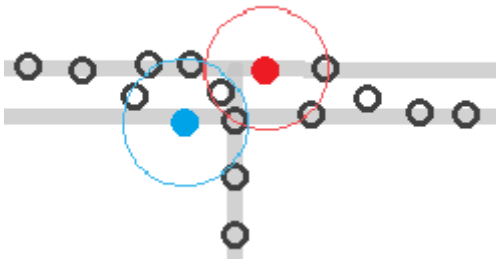
8.6.3 Priemerovanie zhlukov

Fuzzy C-Means algoritmus vráti v miestach veľkej hustoty bodov veľký počet centier zhlukov v tejto oblasti Obr. 16. To spôsobuje v ďalších krokoch analýzy problémy, hlavne pri hľadaní následnosti jednotlivých stredov a ich prepojení. Preto sa pokúsime tieto body veľmi blízko seba spriemerovať tak, aby boli čo najrovnomernejšie rozložené.



Obr. 16 Nájdené zhluky v miestach s veľkým počtom bodov

Opätovné použitie algoritmu Fuzzy C-Means na tieto zhlukové stredy neprinieslo použiteľný výsledok. Bolo nutné nájsť algoritmus, ktorý by nájdené body spriemeroval a rozložil čo najviac rovnomerne. Prechádzanie všetkých stredov zhlukov a priemerovanie so všetkými stredmi, ktoré sa nachádzajú vo vzdialenosti definovanej konštantou neprinieslo očakávaný výsledok. Problémom bolo nejednoznačné určenie príslušnosti bodov, ak boli na rozhraní medzi dvomi priemerovanými stredmi (Obr. 17).

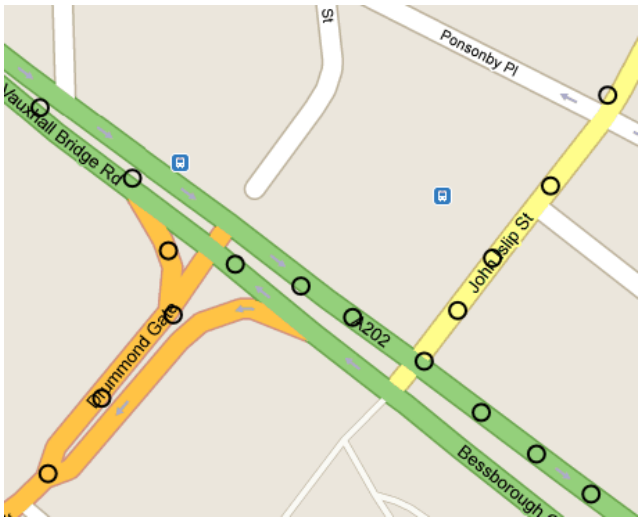


Obr. 17 Problém určenia príslušnosti bodu pri priemerovaní

Pri priemerovaní sa zachoval iba nový spriemerovaný bod a priemerované body sa odstránili. Takto ďalším priemerovaným bodom mizli body, ktoré by mohli tiež ovplyvniť ich spriemerovanú polohu. Aby nedochádzalo k miznutiu bodov, ktoré mohli ovplyvniť ostatné body, bol navrhnutý deterministický algoritmus rozhodovania jednotlivých situácii, kedy môže byť bod odstránený a kedy nie. Obsiahnuť všetky situácie bolo však takmer nemožné a aj prvých 5 základných situácii viedlo k príliš dlhému a neudržateľnému kódu. Rýchlosť výpočtu bola tiež príliš nízka.

Po skúsenosti s predošlým návrhom bol navrhnutý podobný, ale omnoho efektívnejší spôsob priemerovania. Jeho výhodou je väčšia univerzálnosť, a preto bol neskôr znovu použitý na priemerovanie polôh nájdených stredov križovatiek. Vstupnými parametrami sú opäť množina bodov (v tomto prípade stredov zhlukov), ktoré sa budú priemerovať a maximálna euklidovská vzdialenosť bodov, v ktorej sa budú vyhľadávať body na priemerovanie. Tretím nepovinným parametrom je funkcia, ktorá sa má vykonať po spriemerovaní bodov pre jeden zvolený bod (používa sa len v prípade priemerovania nájdených križovatiek).

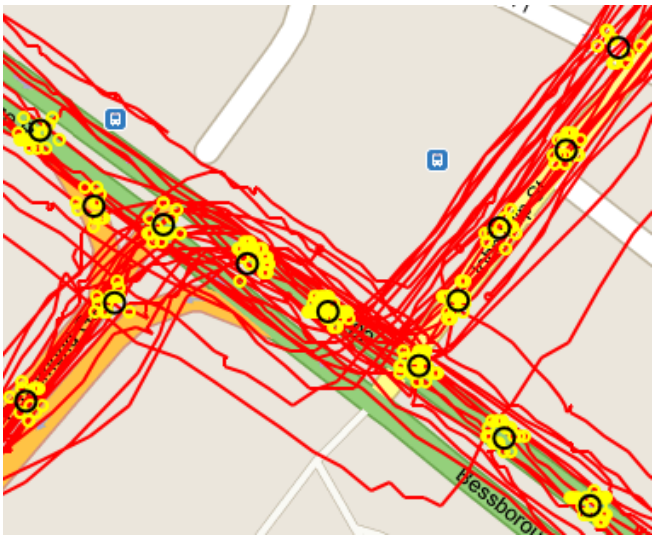
Základom priemerovacej funkcie je objekt, v ktorom sú pre každý bod dopredu uložené referencie susedných bodov aj s ich vypočítanými vzdialenosťami, čo výrazne urýchli ďalšie výpočty. Súčasne je v objekte uložená aj váha bodu. Algoritmus volí postupne hodnotu parametru maximálnej euklidovskej vzdialenosti s krokom 1 meter od hodnoty 1 až po zo vstupu zvolenú maximálnu hodnotu. Pre každú zvolenú hodnotu parametru vzdialenosti nájde algoritmus pre všetky body ich susedné body, ktoré spĺňajú podmienku maximálnej vzdialenosti a spočíta vážený priemer bodov. Nové spriemerované stredy sa uložia a priemerované body sa zahodia. Pre všetky susedné body zahodených bodov sa dopočíta vzdialenosť od nového spriemerovaného stredu. Výsledok algoritmu je na Obr. 18.



Obr. 18 Fuzzy C-Means výstup po spriemerovaní

8.7 Spojovanie zhlukov

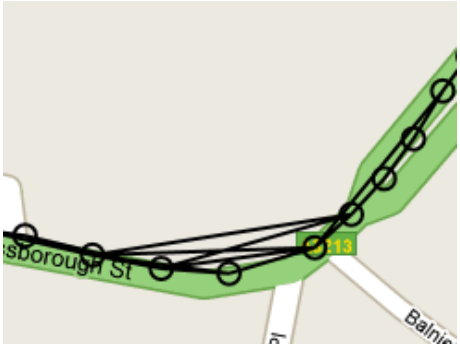
Keď už máme nájdené rovnomerne rozložené stredy zhlukov, musíme nájsť ich vzájomné prepojenie, aby sme si vytvorili virtuálnu mapu ulíc a ich vzájomných križovatiek. Pre každý stred zhluku nájdeme množinu bodov, ktoré sú od stredu zhluku vzdialené najviac o definovanú konštantu 13 metrov (Obr. 19).



Obr. 19 Príslušnosť bodov k stredu zhluku

Pre každý bod z množiny hľadáme susedný bod, z jeho susedného bodu rekurzívne jeho suseda až do nastavenej hĺbky najviac 4 susedov alebo kým niektorý zo susedov nepatrí do množiny bodov iného stredu zhluku.. Hĺbka hľadania 4 susedov sa ukázalo byť dostatočným počtom pre nájdenie susedného zhluku, zároveň to zvyšuje výkon aplikácie a nedochádza k prepojeniu nesusedných stredov zhlukov. Napriek obmedzeniu počtu preskúmavaných susedov môže stále dôjsť k nechcenému prepojeniu nesusedných stredov zhluku (Obr. 20). Dochádza k tomu hlavne v oblasti križovatiek a ohybov ulíc.

Nechcenému prepojeniu stredov zabránime tým, že pre každé novo nájdené prepojenie z daného stredu zhluku overíme, či v danom smere (s toleranciou ± 45°) už neexistuje nejaké prepojenie. V prípade, že už existuje, spočíta sa, ktoré z prepojení je kratšie. Kratšie prepojenie sa zachová, dlhšie sa zahodí. Výsledok je na Obr. 21.

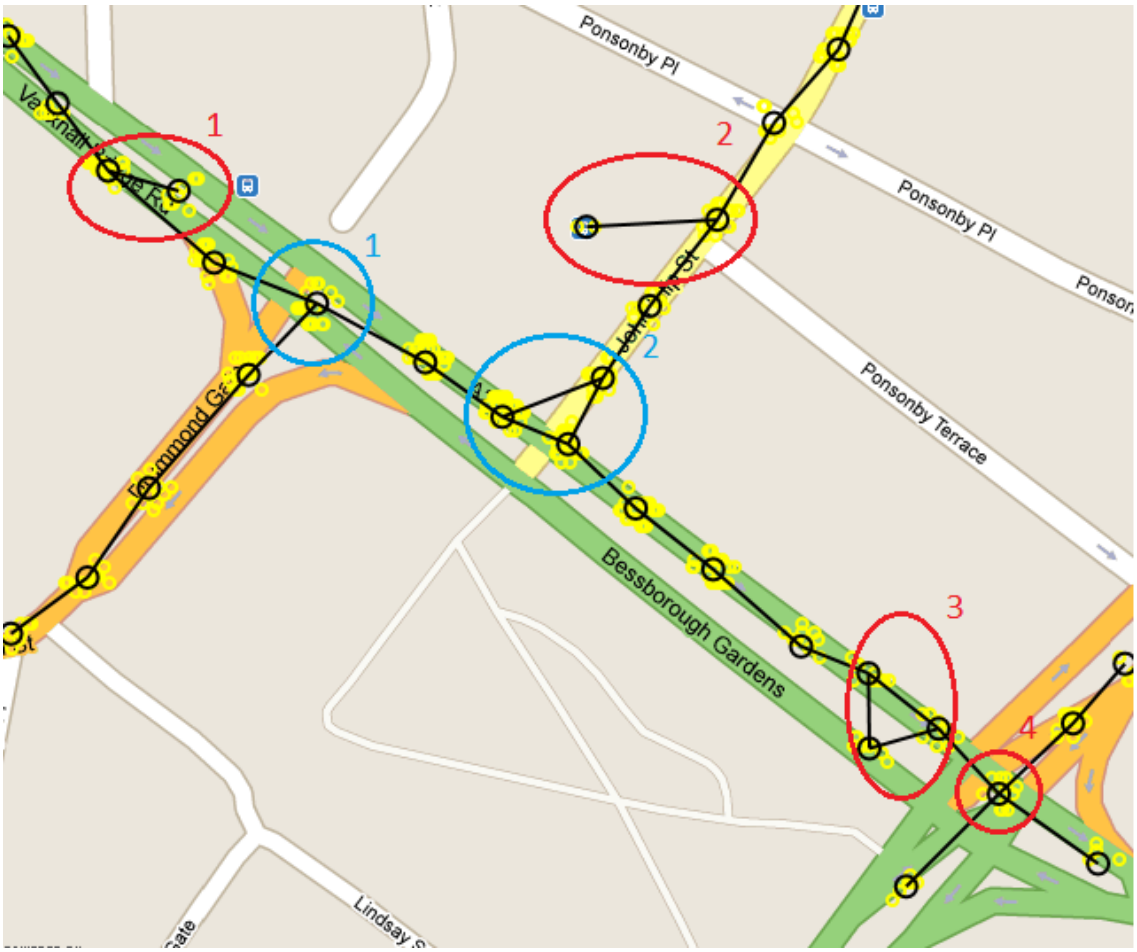


Obr. 20 Prepojenie nesusedných stredov zhluku

Obr. 21 Správne prepojenie susedných stredov zhlukov

8.8 Vyhľadávanie križovatiek

Za križovatku môžeme považovať oblasť v okolí zhlukového stredu, ktorý je spojený s aspoň 3 susednými zhlukovými stredmi. Tento základný predpoklad však budeme musieť rozšíriť o niekoľko ďalších doplňujúcich podmienok, aby nedochádzalo k chybnému určeniu polohy križovatky. Prípady, ktoré spĺňajú podmienku, ale pritom nemôžu byť križovatkou, sú vyznačené na Obr. 22 červenou farbou a miesta, ktoré môžu byť križovatkou, sú označené modrou farbou.



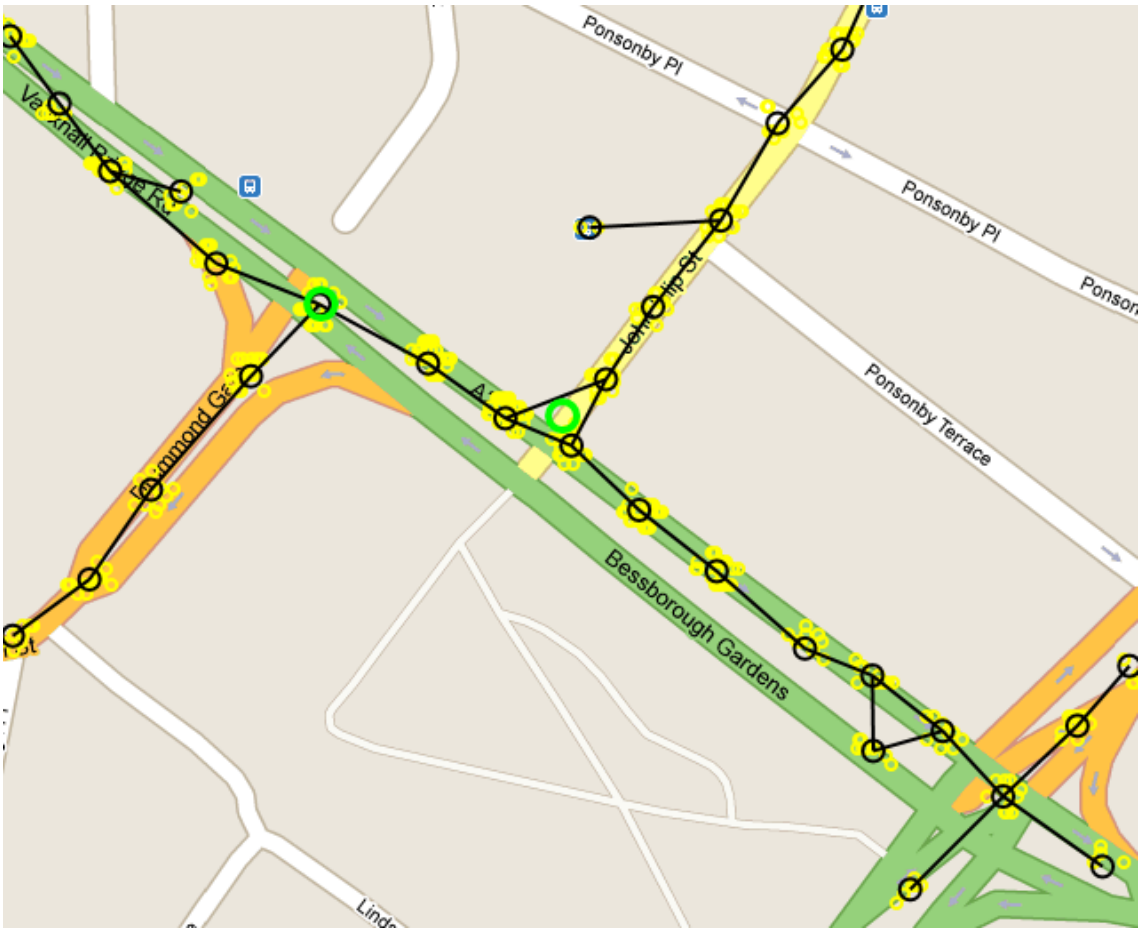
Obr. 22 Miesta, ktoré by mohli predstavovať križovatku

Prípady *1* a *3* označené červenou farbou vznikli v dôsledku širokej ulice, ktorá má jazdné pruhy pre každý smer oddelené. Pri mapovaní bodov na ulicu nebolo možné jednoznačne určiť, do ktorého jazdného pruhu z tejto ulice daný bod patrí. Prípad *2* označený červenou vznikol v dôsledku namapovania bodov na symbol zastávky autobusu, ktorý v mapovom podklade používa rovnakú farebnú výplň ako ulice. Prípad *4* označený červenou by mohol byť križovatkou, ale v tomto mieste ešte nevieme, ktorým smerom ulice pokračujú, preto nesmie byť v tomto mieste označená križovatka.

Prípadom označeným červenou farbou sa vyhneme tým, že pre každý zhlukový stred, ktorý spĺňa podmienku aspoň 3 susedov, overíme, či aj títo susedia majú svojich susedov. V prípade, že majú svojich susedov rôznych od predpokladaného stredu križovatky, overíme, či vzájomný uhol vektorov tvorených stredom križovatky a susedných susedov zviera aspoň 45°.

V prípade *1* označenom modrou farbou je určenie križovatky jednoznačné, spĺňa všetky podmienky uvedené vyššie. V prípade *2* označenom modrou spĺňajú tieto

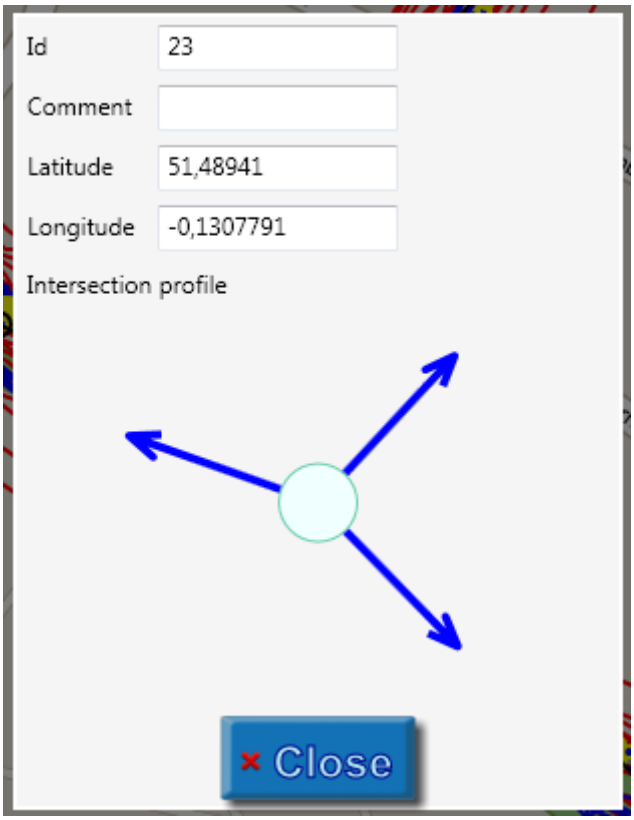
podmienky hneď 3 body, pričom je zjavné, že ide len o jednu križovatku. Túto situáciu vyrieši priemerovanie bodov, ktoré už bolo použité pri priemerovaní stredov zhlukov metódy Fuzzy C-Means v kapitole 8.6.3. Za polomer priemerovania bola zvolená hodnota 80 metrov, čo si môžeme preložiť tak, že každé 2 križovatky musia byť od seba vzdialené viac než 160 (2x80) metrov. Výsledok vyhľadávania križovatiek označených zeleným kruhom je na Obr. 23.



Obr. 23 Nájdené križovatky

Určený tvar križovatky sa ukladá do databázy pre neskoršie využitie pri vyhľadávaní ulíc. Tento tvar je možné vidieť po kliknutí na zelený kruh križovatky. Otvorí sa dialógové okno (Obr. 24) križovatky, ktoré okrem základných údajov o križovatke zobrazuje aj jej určený tvar. V tomto okne je možné ku križovatke pridať poznámku alebo kliknúť na zvolený smer a dozvedieť sa informácie o nájdenej ulici v tomto smere.

Okrem tvaru križovatky sa po nájdení križovatiek uloží príslušnosť GPS bodov ku križovatke. Sú to všetky body, ktoré sú vzdialené od stredu križovatky najviac o 55 metrov. Z týchto bodov sa neskôr začínajú vyhľadávať ulice, ktoré vedú z alebo do križovatky.



Obr. 24 Dialógové okno križovatky

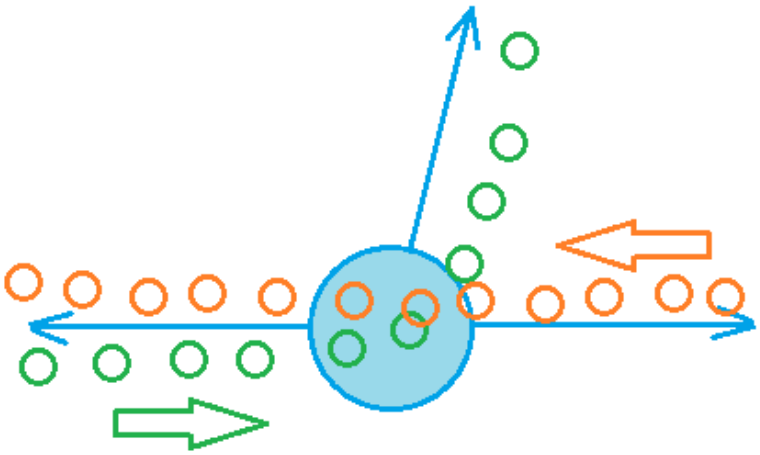
8.9 Vyhľadávanie ulíc

Matematický model dopravnej oblasti predstavuje ohodnotený orientovaný graf, kde vrcholy tvoria križovatky ulíc, orientované hrany tvoria ulice s určeným smerom jazdy v nich a hodnotu hrán predstavuje napríklad kapacita ulice.

Pre tento model už máme nájdené križovatky s GPS bodmi, ktoré prislúchajú ku križovatke. Z týchto bodov sa pokúsime nájsť ulice medzi jednotlivými križovatkami. Pre každú ulicu určíme zoznam prejazdov vozidla ulicou. Pre každý prejazd ulicou si uložíme zoznam bodov prislúchajúcich k tomuto prejazdu a aspoň jednu z križovatiek, z ktorej jazda začína alebo končí.

Ulicu bude definovať aspoň jeden prejazd ulicou. Počiatočná a koncová križovatka prejazdu ulicou zároveň určuje orientáciu ulice. Pre každú križovatku sa ukladá jej tvar ako zoznam križovatkových výjazdov, ktoré majú definovaný uhol výjazdu. Ulice môžu začínať alebo končiť iba v týchto križovatkových výjazdoch. Pri vyhľadávaní ulíc je nutné riešiť 3 situácie:

V okolí nájdenej križovatky nie sú ulice, ktoré by mohli končiť v inej nájdenej križovatke. Napríklad našli sme v celej oblasti jedinú samostatnú križovatku. V tomto prípade pre každý križovatkový bod hľadáme jeho susedný bod a od neho jeho susedný bod atď. podľa zvoleného smeru vyhľadávania. Vyhľadávame vždy v smere jazdy von z križovatky a potom v smere dnu do križovatky. Ak nájdeme susedný bod, ktorý už nie je križovatkovým bodom, našli sme prvý bod prejazdu ulicou. Postupne prehľadávame susedné body a priradzujeme ich k prejazdu ulicou, až kým nenarazíme na bod, ktorý už nemá suseda alebo sused končí v križovatke. Pre každý vytvorený nový prejazd ulicou hľadáme existujúcu ulicu podľa uhla križovatkového výjazdu a uhla prejazdu ulicou. Ak nenájdeme existujúcu ulicu, vytvoríme novú.



Obr. 25 Prejazdy ulicami rozdelené križovatkou

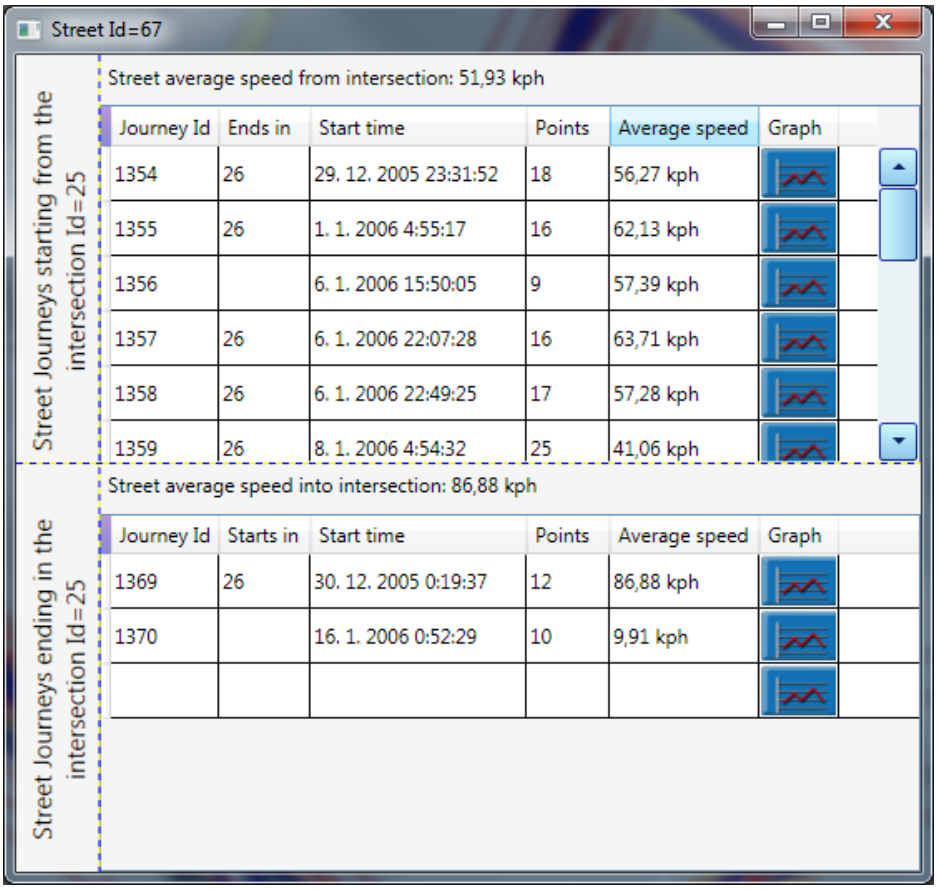
Pri vyhľadávaní ulíc z nájdenej križovatky narazíme už na existujúcu ulicu, ktorá však začína alebo ústi v inej križovatke. V tomto prípade musí dôjsť k spojeniu ulíc do jedinej. Pri spájaní ulíc sa vždy zachová staršia už existujúca ulica. Zároveň sa musia všetky prejazdy ulicou zlúčiť do starších existujúcich prejazdov a preniesť všetky prislúchajúce body. Rovnaký postup zlučovania ulíc využijeme aj na konci každého procesu vyhľadávania ulíc, kde sa pokúsime nájsť pre danú dopravnú oblasť všetky prejazdy ulicou, ktoré neústia z oboch strán do niektorej z križovatiek. Ak takéto

prejazdy nájdeme, pokúsime sa nájsť ich zvyšné chýbajúce body a prípadne, ak to bude nutné, zlúčime niektoré prejazdy.

Križovatka bola nájdená na mieste už existujúcej nájdenej ulice. V tomto prípade musí dôjsť k rozdeleniu ulice. Pri nájdení križovatky sa všetky GPS body v jej okolí označia ako body prislúchajúce ku križovatke. V túto chvíľu sú tieto body označené ako body križovatky a zároveň aj body prejazdov ulice. Z týchto bodov budeme vyhľadávať prvé susedné body, ktoré už nie sú križovatkovými bodmi. Od týchto susedných bodov rozdelíme ulicu a prejazdy v nej.

Celý proces vyhľadávania ulíc je pre zvýšenie výkonu vykonávaný len na základe referencií uložených objektov v pamäti. Existujúce objekty sa v pamäti vytvárajú pri nahratí GPS bodov z databázy pre zobrazenú dopravnú oblasť. Po skončení procesov sa nový výsledok zapíše späť do databázy. Výrazne sa týmto zníži počet SQL dotazov do databázy a rýchlosť vyhľadávania sa zvýši približne dvojnásobne.

Výsledok nájdených ulíc a ich prejazdov je možné si prezrieť po kliknutí na niektorú z križovatiek a následne v okne križovatky na niektorý z výjazdov križovatky. Zobrazí sa okno ulice (Obr. 26).



Obr. 26 Okno ulice so zoznamom prejazdov

V záhlaví okna je uvedené Id ulice, podľa ktorého je možné ulicu rýchlejšie vyhľadať v databáze. Okno je rozdelené vo vertikálnom smere na polovicu. V hornej polovici sú zobrazené informácie o prejazdoch ulice, ktoré začínajú vo vybranej križovatke a v dolnej časti o prejazdoch, ktoré ústia do vybranej križovatky zvolenou ulicou. Zoznam prejazdov ulicou je uvedený v tabuľkách, kde v prvom stĺpci je uvedené Id prejazdu pridelené databázou, následne Id križovatky, v ktorej prejazd začína alebo končí. Tento údaj je k dispozícii len v prípade, že je prejazd ulicou z oboch strán ukončený križovatkou. Čas začatia prejazdu udáva nameraný čas prvého bodu prejazdu ulicou. Z počtu bodov prejazdu a priemernej rýchlosti spočítanej z okamžitých rýchlosti týchto bodov je možné vytušiť tvar grafu rýchlosti (Obr. 27), ktorý je možné zobraziť si po stlačení tlačidla grafu. Nad každou tabuľkou je zobrazená priemerná rýchlosť v ulici pre daný smer, ktorá je spočítaná len pre prejazdy, ktoré sú z oboch strán ukončené križovatkou. Neukončené prejazdy ulicou môžu vzniknúť chybami merania a následného mapovania alebo zastavením vozidla v tejto ulici a ukončením zberu GPS dát.

Obr. 27 Graf rýchlosti prejazdu ulicou

Graf rýchlosti vozidla zobrazuje na osi X čas a na osi Y okamžitú rýchlosť vozidla v čase. Graf je možné priblížiť alebo oddialiť a posúvať v prípade, že je viditeľná časť grafu väčšia než polovica zobrazovacej oblasti grafu. Okrem aktuálnej rýchlosti prejazdu ulicou zobrazuje graf aj okamžitú rýchlosť prejazdu počiatočnou a koncovou križovatkou. Z nich sa dá vyčítať napríklad, či vozidlo pred križovatkou stálo a postupne sa rozbiehalo alebo prešlo križovatkou bez zastavenia.

9 Popis aplikácie

Vzhľadom na rozsah aplikácie sa v tejto kapitole zameriame iba na užívateľský popis aplikácie. Použité algoritmy boli vysvetlené v predošlej kapitole spolu s oknami križovatiek, ulíc a grafov.

9.1 Hlavné okno

Hlavné okno aplikácie je možné rozdeliť do troch častí.

Obr. 28 Hlavné okno aplikácie

9.1.1 Hlavné menu

Hlavné menu obsahuje možnosť importu vstupných dát, uloženie aktuálne zobrazeného mapového podkladu spolu so zobrazenými vrstvami nad ním ako bitmapový obrázok, a ukončenie aplikácie.

9.1.2 Aktívna zobrazovacia oblasť

Aktívna zobrazovacia oblasť zobrazuje mapový podklad spolu so zvolenými vrstvami. Pohyb po mape je umožnený kliknutím do aktívnej zobrazovacej oblasti (na mapu), následne sa dané miesto stáva novým stredom mapy. Aktuálna poloha stredu mapy a kurzoru sa zobrazuje v dolnej časti *Menu aplikácie*. Na vrstvu s označením nájdených križovatiek je možné kliknúť a zobraziť bližšie informácie o križovatke.

9.1.3 Menu aplikácie

Aplikačné menu pozostáva z dvoch zoznamov zaškrtávacích polí rozdelených výberom zobrazovaného priblíženia. Všetky zaškrtávacie polia fungujú iba pre zvolené priblíženie 18, ktoré je definované konštantou a všetky konštanty aplikácie sa vzťahujú k tomuto priblíženiu.

Horný zoznam zaškrtávacích polí slúži na spúšťanie aplikačnej logiky implementovanej na pozadí:

- *Auto consolidation* – spustí automatické vyhľadávanie ešte nenamapovaných GPS bodov a pokúsi sa ich namapovať
- *Consolidate while drawing* – povoľuje mapovanie bodov na ulicu počas vykresľovania GPS bodov na zvolený mapový podklad. Ak nebude pole zaškrtnuté, budú sa používať hodnoty namapovaných bodov z databázy.
- *Analyse streets* – povoľuje algoritmus vyhľadávania ulíc a križovatiek. Po zaškrtnutí je nutné zvoliť nový stred mapy, aby sa analýza spustila.

Dolný zoznam zaškrtávacích polí slúži na zobrazenie alebo skrytie zobrazovanej vrstvy:

- *Original data*, *Consolidated data* – vrstva s nameranými (červená) a namapovanými (modrá) bodmi
- *Fuzzy centers*, *Fuzzy centers* members – vrstva s nájdenými stredmi zhlukov Fuzzy C-Means algoritmu (čierna) a vrstva s GPS bodmi prislúchajúcimi k týmto stredom (žltá)
- *Fuzzy centers linkage* – vrstva spojení medzi nájdenými stredmi zhlukov
- *Intersections* – vrstva s nájdenými križovatkami (zelená)

9.2 Stavové informačné okná

Vzhľadom na množstvo spracovávaných dát trvajú niektoré výpočty jednotky sekúnd. Aby užívateľ nemal pocit, že došlo v aplikácii k nejakému zacykleniu alebo chybe, zobrazuje sa pre každý krok informačné dialógové okno s popisom aktuálnej činnosti.

V prípade, že nie je možné dopredu odhadnúť množstvo krokov, zobrazuje sa iba nekonečný stavový ukazateľ (Obr. 29 vľavo). Ak je možné dopredu určiť počet krokov, zobrazuje sa poradové číslo aktuálneho kroku z celkového počtu. Tomu percentuálne zodpovedá aj stavový ukazateľ (Obr. 29 vpravo).

Obr. 29 Príklady informačných dialógových okien

10 Databáza

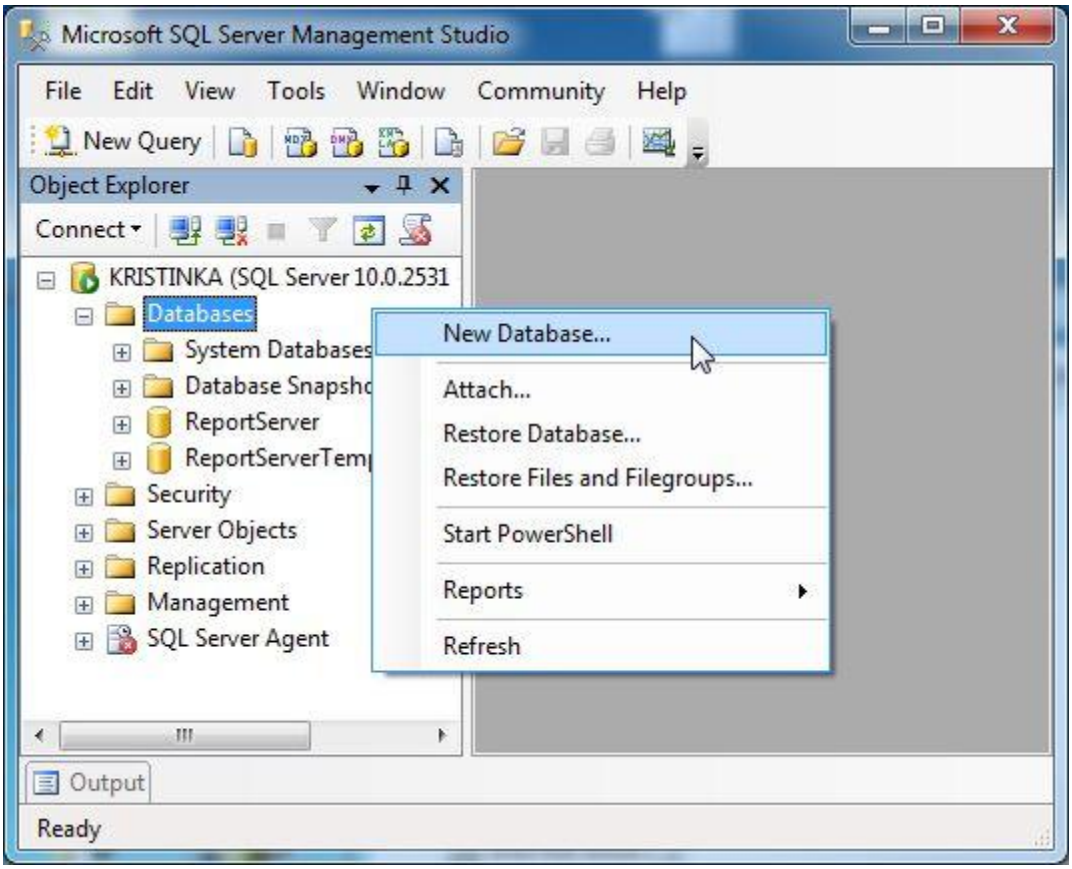
V tejto kapitole je popísané, ako je možné vytvoriť databázu potrebnú pre chod aplikácie, nastavenie spojenia s databázou a význam jednotlivých tabuliek a stĺpcov v nej. Predpokladá sa, že už je nainštalovaná databáza Microsoft® SQL Server® 2008 spolu s aplikáciou Microsoft® SQL Server® 2008 Management Studio.

10.1 Vytvorenie aplikačnej databázy

Pri vytváraní databázy je možné si zvoliť, či si vytvoríme prázdnu databázu pomocou dodaného SQL skriptu alebo využijeme dodanú už predvyplnenú databázu.

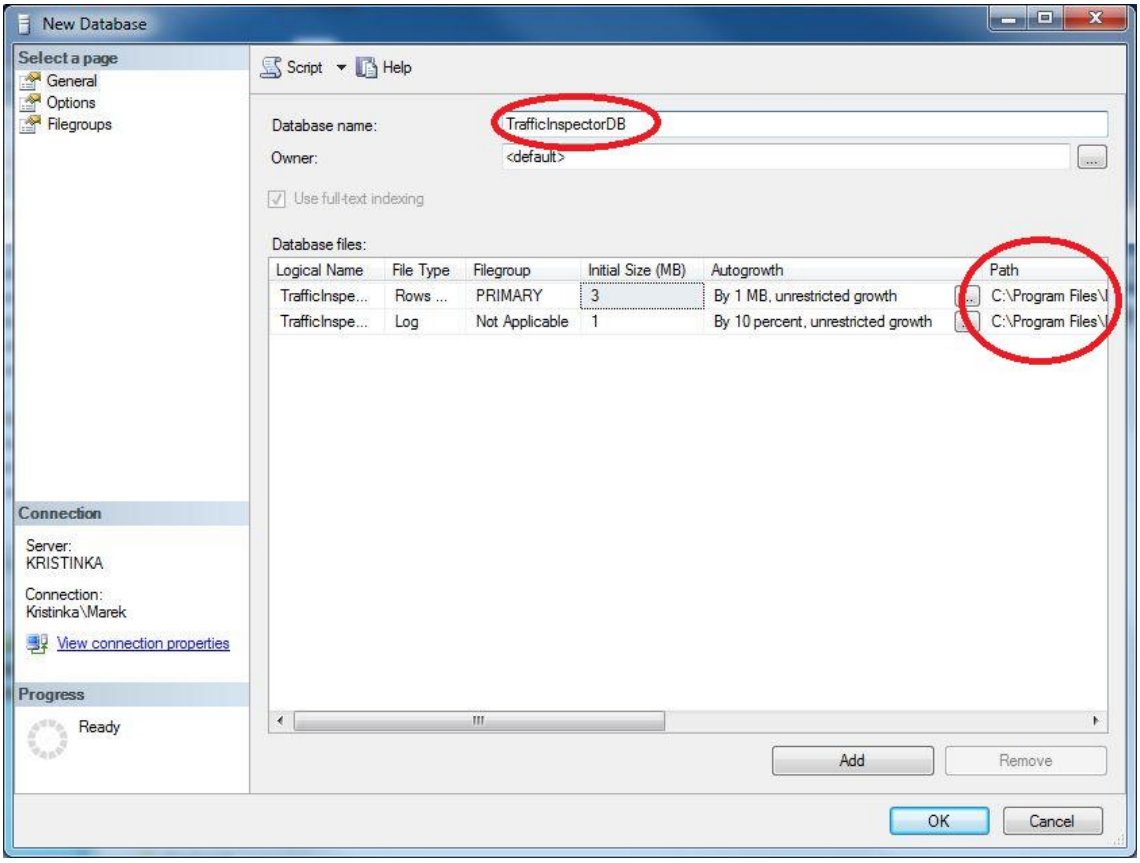
10.1.1 Použitie dodaného SQL skriptu

Tento postup umožní vytvorenie prázdnej databázovej štruktúry. SQL skript neobsahuje príkaz na vytvorenie novej databázy, aby si užívateľ mohol túto databázu umiestniť tam, kde mu to vyhovuje a prípadne nastaviť ďalšie nastavenia. Prvým krokom bude vytvorenie novej prázdnej databázy (Obr. 30).



Obr. 30 Vytvorenie novej prázdnej databázy

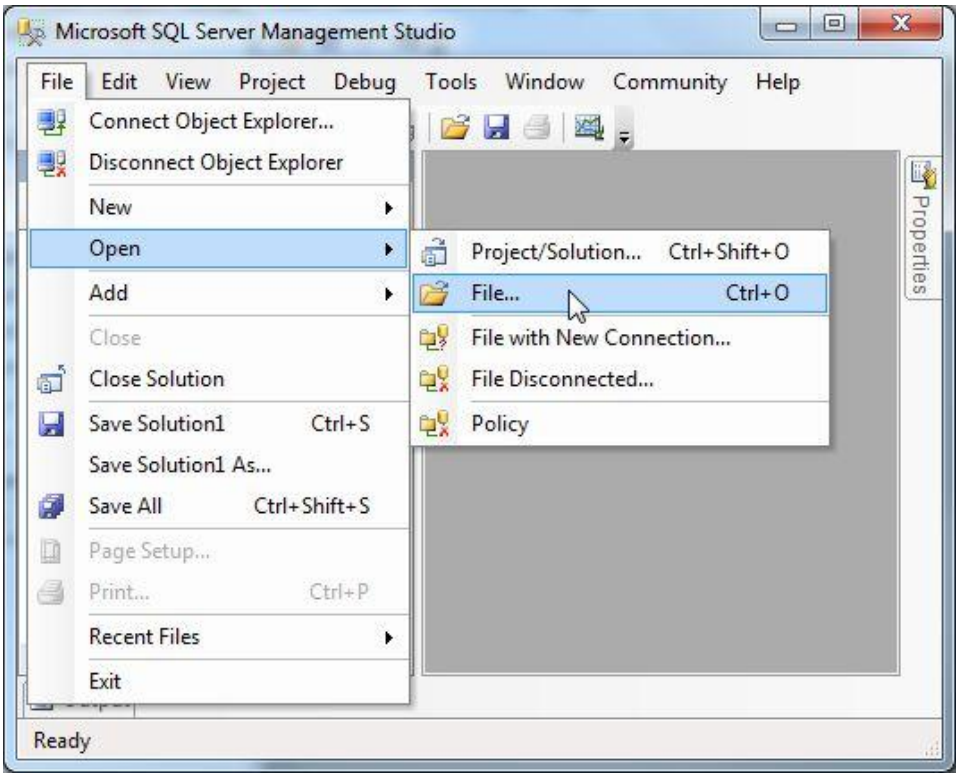
Spustíme Microsoft® SQL Server® 2008 Management Studio a pripojíme sa k nášmu databázovému serveru. V našom prípade má názov *KRISTINKA*. Klikneme pravým tlačidlom na *Databases* a zvolíme *New Database...* (Obr. 30).



Obr. 31 Zadanie názvu a umiestnenia novej databázy

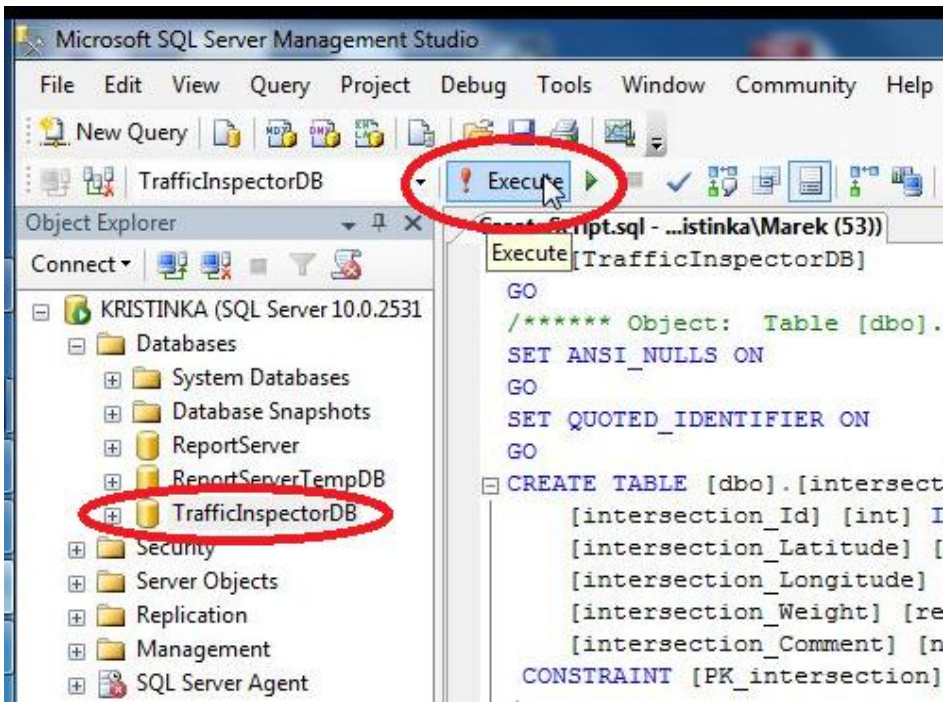
V okne *New Database* (Obr. 31) vyplníme meno databázy, ktoré musí byť presne *TrafficInspectorDB*. Tento názov využíva dodaný SQL skript a až po aplikovaní skriptu bude možné databázu premenovať. V tomto okne môžeme zmeniť aj umiestnenie databázy v stĺpci *Path*. Zvyšné nastavenia môžeme ponechať v implicitnom nastavení a uzavrieť okno tlačidlom *OK*.

Následne otvoríme súbor s dodaným SQL skriptom. Zvolíme *File* → *Open* → *File...* (Obr. 32) a nájdeme dodaný súbor *CreateScript.sql*.



Obr. 32 Otvorenie súboru s SQL skriptom

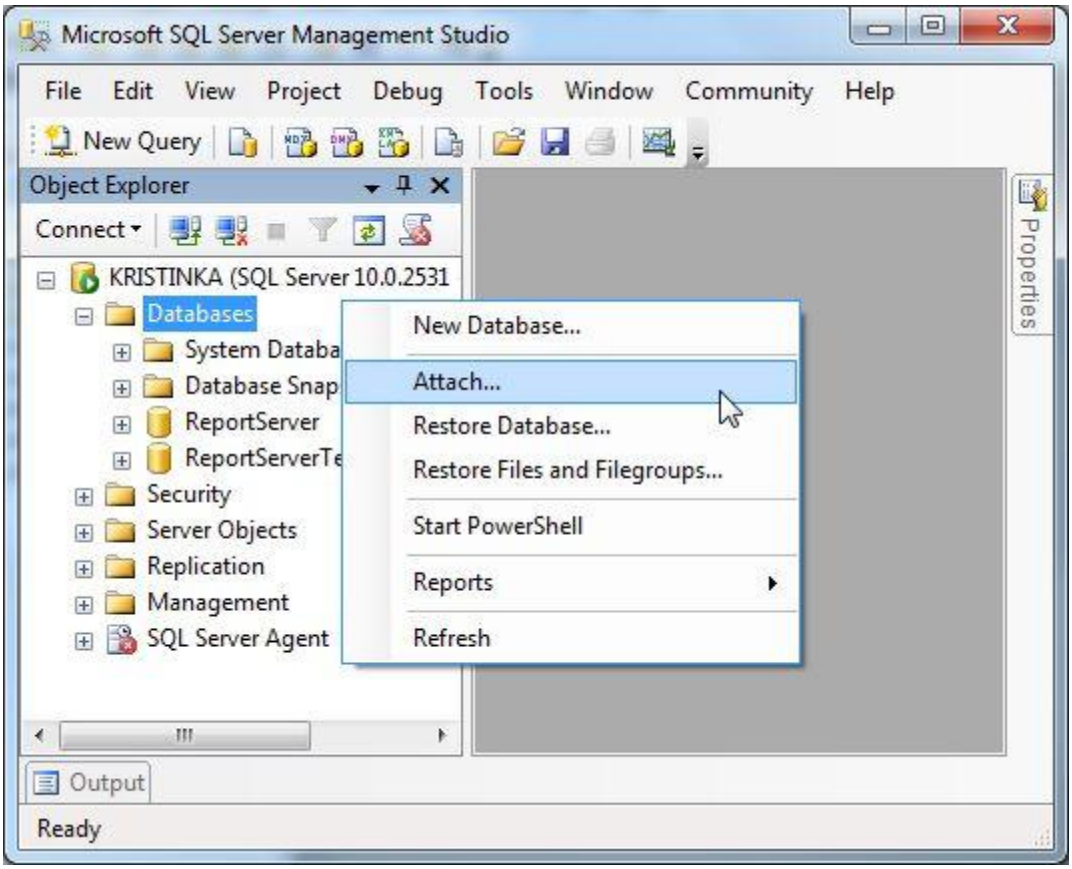
Po otvorení súboru s SQL skriptom klikneme na tlačidlo *Execute* (Obr. 33). V prípade nahlásenia chýb skriptu sa odporúča skontrolovať názov databázy.



Obr. 33 Spustenie SQL skriptu

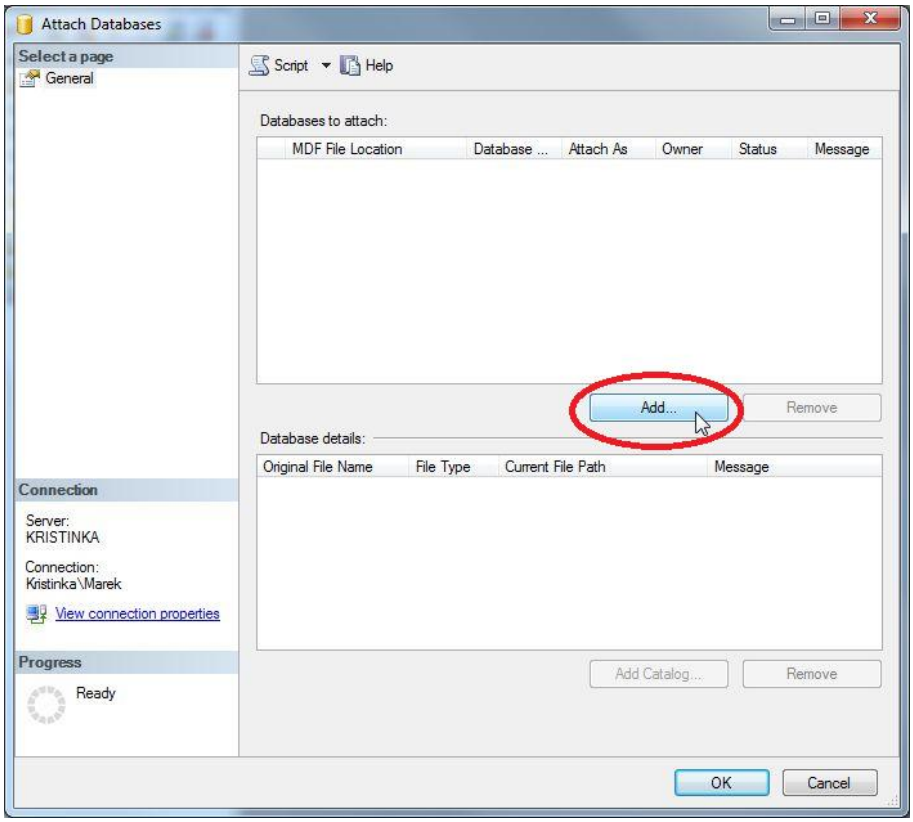
10.1.2 Pripojenie dodanej predspracovanej databázy

Predspracovaná databáza obsahuje naimportované vstupné dáta spolu s namapovanými bodmi pre takmer všetky vstupné dáta. Najskôr umiestnite dodané súbory *TrafficInspectorDB.mdf* a *TrafficInspectorDB_log.ldf* na zvolené miesto s potrebnými prístupovými právami pre databázový server.



Obr. 34 Pripojenie predspracovanej databázy

Spustíme Microsoft® SQL Server® 2008 Management Studio a pripojíme sa k nášmu databázovému serveru. V našom prípade má názov *KRISTINKA*. Klikneme pravým tlačidlom na *Databases* a zvolíme *Attach...* (Obr. 34). Následne klikneme na tlačídlo *Add* (Obr. 35) a nájdeme uložený súbor *TrafficInspectorDB.mdf*.



Obr. 35 Vloženie predspracovanej databázy

10.2 Nastavenie spojenia

Nastavenie spojenia je možné viacerými spôsobmi. Táto kapitola popisuje potrebné zmeny v konfiguračnom súbore skompilovanej aplikácie a uvádza príklad zmeny nastavení v prostredí Microdoft® Visual Studio® 2010.

10.2.1 Nastavenie spojenia pre skompilovanú aplikáciu

Po skompilovaní aplikácie sa vytvorí spustiteľný súbor TrafficInspectorWPF.exe spolu s ďalšími knižnicami s príponou *.dll. Okrem týchto súborov sa vytvorí aj konfiguračný súbor TrafficInspectorWPF.exe.config. Tento súbor je štandardný súbor v XML formáte a obsahuje iba nastavenie databázového spojenia a nastavenie knižnice pre záznam chybových správ (logger). Nastavenia knižnice chybových správ nie je nutné meniť.

Nastavenie databázového spojenia sa nachádza v uzle connectionStrings. Môže obsahovať viacero nastavení spojení. Predpokladáme však, že máme finálnu verziu konfiguračného súboru a obsahuje už len jediné aplikáciou používané spojenie. V prípade, že je na databázovom serveri povolená Windows autorizácia, stačí v dodanej

konfigurácii zmeniť iba Data Source, aby používal správny databázový server. Názov spojenia sa nesmie zmeniť, v opačnom prípade toto spojenie aplikácia nenájde. Ak nie je povolená Windows autorizácia, úpravu pre connectionString nájdeme na stránkach Microsoftu.

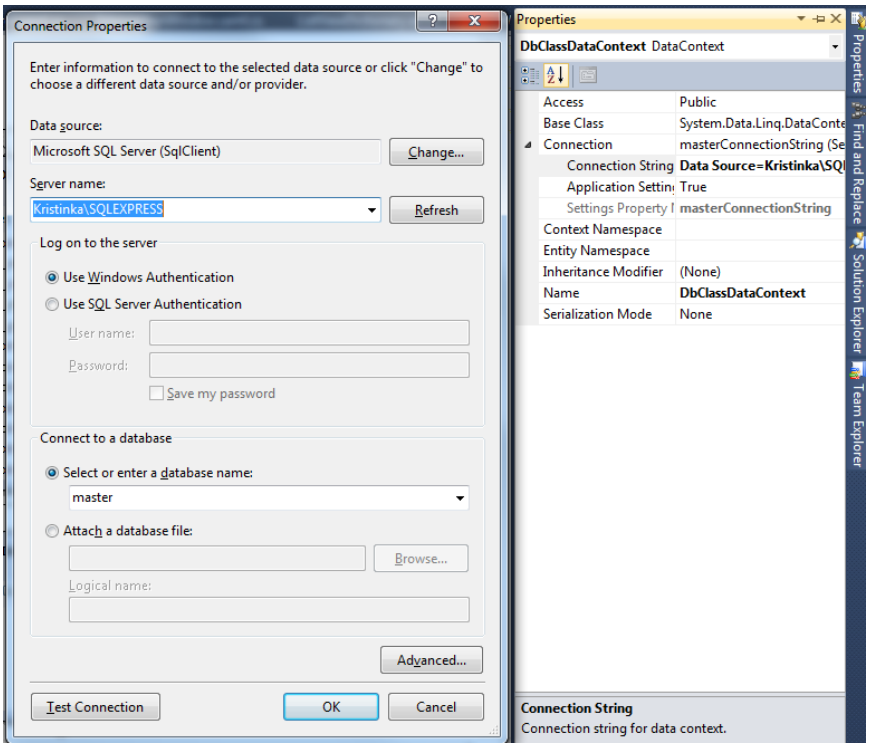
Dodaná konfigurácia databázového spojenia:

```
<connectionStrings>
<add
    name = "TrafficInspectorWPF.Properties.Settings.TrafficInspectorDBConnectionString2"
    connectionString = "Data Source=Kristinka; Initial Catalog=TrafficInspectorDB; Integrated
    Security = True" providerName = "System.Data.SqlClient"
/>
 </connectionStrings>
```

10.2.2 Nastavenie spojenia v prostredí Microdoft® Visual Studio® 2010

Nastavenie databázového spojenia je možné upraviť rovnakým spôsobom ako v prípade skompilovanej aplikácie. Rozdiel je iba v tom, že názov neskompilovaného konfiguračného súboru je *app.config*.

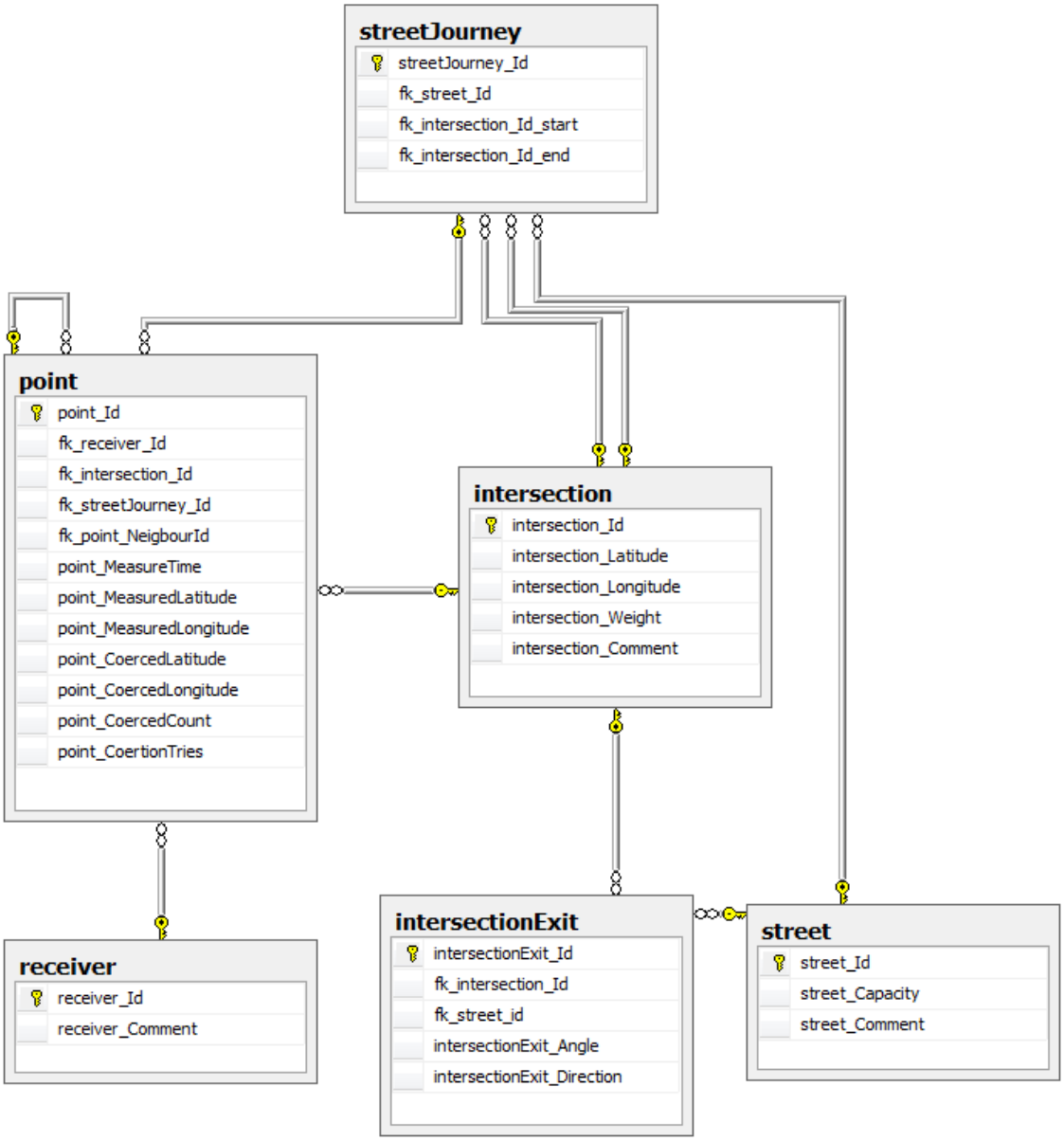
Ďalším spôsobom je otvorenie súboru *DbClass.dbml*. V jeho *Properties* je možné rozbaliť položku *Connection* a nastaviť Connection String prostredníctvom okna *Connection Propeties* (Obr. 36).



Obr. 36 Nastavenie databázového spojenia v prostredí Ms Visual Studio® 2010

10.3 Štruktúra databázy

Štruktúra databázy vychádza z funkcie a požiadaviek na aplikáciu. Jej štruktúra je na Obr. 37. Vstupné dáta sú uložené v tabuľke *point*, pričom duplicitné dáta sú odlíšené rôznym ID prijímača (tabuľka *receiver*). Zvyšné tabuľky slúžia na uloženie analyzovaných dát. Cudzie kľúče sú v tabuľke vždy nazvané s predponou *fk_* a vlastné stĺpce tabuľky sú vždy nazvané predponou tvorenou z názvu tabuľky.



Obr. 37 Štruktúra databázy

10.3.1 Tabuľka receiver

Slúži na odlíšenie GPS prijímačov.

- *receiver_Id* – primárny kľúč tabuľky, môže obsahovať všetky znaky, pretože označenie zariadenia môže byť ako číselné, tak aj nečíselné,
- *receiver_Comment* – možné pridať až 50 znakov poznámky z prijímača.

10.3.2 Tabuľka point

Obsahuje uložené vstupné GPS dáta spolu s dopočítanými dátami z mapovacieho algoritmu.

- *point_Id* – celočíselný primárny kľúč tabuľky s nastaveným automatickým zvyšovaním hodnoty pri vkladaní hodnoty
- *fk_receiver_Id* – obsahuje primárny kľúč niektorého z prijímačov, určuje príslušnosť bodu ku GPS prijímaču
- *fk_intersection_Id* – ak obsahuje primárny kľúč niektorej z križovatiek, znamená to, že tento bod je križovatkovým bodom
- *fk_streetJourney_Id* – ak obsahuje primárny kľúč niektorého prejazdu ulicou, znamená to, že ide o bod mimo križovatku a patrí do niektorej z určených ulíc
- *fk_point_NeighbourId* – ak má bod svojho určeného predchodcu, je v tomto stĺpci jeho primárny kľúč
- *point_MeasureTime* – obsahuje čas merania GPS bodu získaný zo vstupných dát
- *point_MeasuredLatitude, point_MeasuredLongitude* – poloha GPS bodu získaná zo vstupných dát
- *point_CoercedLatitude, point_CoercedLongitude* – poloha GPS bodu získaná pomocou mapovacieho algoritmu
- *point_CoercedCount* – počet uspešných mapovaní bodu na ulicu
- *point_CoertionTries* – počet pokusov mapovania bodu na ulicu

10.3.3 Tabuľka intersection

Tabuľka slúži na uloženie dát nájdených križovatiek.

- *intersection_Id* – celočíselný primárny kľúč tabuľky s nastaveným automatickým zvyšovaním hodnoty pri vkladaní hodnoty
- intersection_Latitude, intersection_Longitude – poloha určeného stredu križovatky
- *intersection_Weight* – váha určeného stredu križovatky. Udáva počet určení polohy križovatky. Táto hodnota sa využíva pri váženom priemerovaní polohy križovatky.
- *intersection_Comment* – možnosť vložiť 50 znakov poznámky

10.3.4 Tabuľka intersectionExit

Táto tabuľka obsahuje zoznam výjazdov z križovatiek, kde počet záznamov pre jednu križovatku určuje počet výjazdov križovatky.

- *intersectionExit_Id* – celočíselný primárny kľúč tabuľky s nastaveným automatickým zvyšovaním hodnoty pri vkladaní hodnoty
- *fk_intersection_Id* – primárny kľúč križovatky, ku ktorej výjazd prislúcha
- *fk_street_Id* – ak obsahuje primárny kľúč niektorej z ulíc, táto ulica začína v tomto výjazde križovatky
- *intersectionExit_angle* – uhol v radiánoch, udáva smer výjazdu križovatky

10.3.5 Tabuľka street

Obsahuje zoznam nájdených ulíc.

- *street_Id* – celočíselný primárny kľúč tabuľky s nastaveným automatickým zvyšovaním hodnoty pri vkladaní hodnoty

10.3.6 Tabuľka streetJourney

Obsahuje prejazdy ulicou.

- *streetJourney_Id* – celočíselný primárny kľúč tabuľky s nastaveným automatickým zvyšovaním hodnoty pri vkladaní hodnoty

- *fk_street_Id* – udáva príslušnosť prejazdu ulicou k niektorej z ulíc
- *fk_intersection_Id_start* – ak obsahuje primárny kľúč križovatky, tento prejazd ulicou začína v tejto križovatke
- *fk_intersection_Id_end* – ak obsahuje primárny kľúč križovatky, tento prejazd ulicou končí v tejto križovatke

11 Systémové požiadavky

V tejto kapitole je popísané hardvérové a softvérové vybavenie, ktoré bolo použité pre vývoj aplikácie. Zároveň popisuje odhadované minimálne systémové požiadavky pre správny chod aplikácie.

11.1 Hardvér

Detailný popis použitého hardvéru je uvedený v tabuľke Tab. 1. Všetky namerané doby výpočtu spomenuté v práci sa vzťahujú k tejto systémovej konfigurácii.

Tab. 1 Použitý hardvér

Čipová sada	Intel® GM45 Express
Procesor	Intel® Centrino® Core™2 Duo P8400 2,26GHz
RAM	2x 2GB DDR2, 667MHz
Grafická karta	Integrovaná Intel® Mobile GMA 4500MHD
Pevný disk	Hybridný Seagate® Momentus® XT, 4GB SSD + 500GB 7200 ot./min
Rozlíšenie monitora	1280 x 800

Predpokladané minimálne hardvérové požiadavky sú uvedené v tabuľke Tab. 2. Zohľadňujú súčasný beh aplikácie a databázového serveru Microsoft® SQL Server® 2008 na rovnakom počítači.

Tab. 2 Minimálne požiadavky na hardvér

Procesor	Dvojjadrový procesor 1,6GHz
RAM	2 GB
Grafická karta	Karta s grafickým akcelerátorom podporujúca aspoň DirectX 9.0
Pevný disk	1000 MB voľného miesta na disku (inštalácia MS SQL servera)
Rozlíšenie	1024 x 768

11.2 Softvér

Pri vývoji a testovaní aplikácie bol použitý softvér uvedený v tabuľke Tab. 3.

Tab. 3 Použitý softvér

Operačný systém	Microsoft® Windows® 7 Professional 64bit
.NET framework	3.5 SP1
Vývojové prostredie	Microdoft® Visual Studio® 2010 Ultimate
SQL server	Microsoft® SQL Server® 2008
SQL Browser	Microsoft® SQL Server® 2008 Management Studio

Minimálne softvérové požiadavky pre spustenie aplikácie sú uvedené v tabuľke Tab. 4.

Tab. 4 Minimálne softvérové požiadavky

Operačný systém	Microsoft® Windows® XP / Mono for desktop
.NET framework	3.5 SP1
Vývojové prostredie	Microdoft® Visual Studio® 2010 Express
SQL server	Microsoft® SQL Server® 2005
SQL Browser	Microsoft® SQL Server® 2005 Management Studio Express

12 Záver

Analýzou vstupných dát bolo zistené, že čas merania GPS bodov je relatívny k neznámej časovej značke, preto za počiatočnú časovú značku bola zvolená polnoc 1.1.2005. Pokúsili sme sa na základe pozorovania rýchlosti prejazdu vozidla ulicou o korekciu tohto počiatočného času. Predpokladali sme, že týždeň má 5 pracovných dní, počas ktorých by sa mala hlavne v dopoludňajších a popoludňajších hodinách výrazne zhusťovať doprava. Zhustenie dopravy má priamy dopad na rýchlosť prejazdu ulicou. Vybrali sme náhodne 10 ulíc blízko seba. Korekcia počiatočného času pre jeden z prejazdov nepriniesla očakávanú korekciu pre zvyšné prejazdy ulicou. Pokúsili sme sa preto nájsť aspoň takú korekciu počiatočného času, ktorá by vyhovovala pre väčšinu z prejazdov ulicou. Zistili sme, že bez ďalších informácii o vstupných dátach to nie je možné, pretože vozidlá prechádzali vybranými ulicami takmer v každú hodinu približne rovnakou rýchlosťou, z čoho nie je možné určiť nočnú a dennú premávku.

V práci sme na spresnenie meraných bodov navrhli geometrický mapovací algoritmus bod-krivka, ktorý ako existujúcu mapu oblasti využíva rastrový obrázok Google mapy. Výhodou je, že tento obrázok môže byť nahradený ľubovoľným iným rastrovým obrázkom (napríklad mapa areálu rozsiahleho podniku), u ktorého je známa GPS poloha prislúchajúca k stredu obrázku. Geometrický mapovací algoritmus bod-krivka sme doplnili informáciou o smere jazdy vozidla, aby nedochádzalo k chybnému mapovaniu v oblasti križovatiek.

Zhlukovací algoritmus Fuzzy C-Means sme doplnili o vážené priemerovanie stredov nájdených zhlukov, ktorý výrazne zjednodušuje nájdenie ulíc a križovatiek v miestach širších ulíc s väčším počtom jazdných pruhov pre každý smer jazdy. Zoznam nájdených ulíc a križovatiek je uložený v navrhnutej aplikačnej databáze. Navrhnutá aplikácia umožňuje grafické zobrazenie polohy nájdeného stredu križovatky spolu s jej tvarom. Pre každú ulicu je možné si zobraziť zoznam prejazdov vozidla zvolenou ulicou. Pre každý prejazd ulicou je možné si zobraziť detail o okamžitej rýchlosti vozidla formou grafu.

Výstupom tejto práce je matematický model dopravnej oblasti vo forme orientovaného ohodnoteného grafu, kde vrcholmi sú nájdené stredy križovatiek a hrany tvoria ulice medzi nimi. Orientáciu hrán (smer jazdy v uliciach) je možno získať agregáciou samostatných prejazdov ulicou, ohodnotenie hrany (ulice) je vo forme

priemernej rýchlosti jazdy ulicou. Ohodnotenie hrany vo forme kapacity ulice nebolo možné odhadnúť vzhľadom na malé množstvo dát pre jednotlivé ulice.

Nadväzujúcou prácou by malo byť vytvorenie aplikácie, ktorá sa na základe nájdeného matematického modelu dopravnej oblasti pokúsi naplánovať optimálne trasy vozidiel tak, aby do svojho cieľa dorazili v čo najkratšom čase. Druhým variantom je aplikácia, ktorá sa pokúsi o optimálnejšie rozloženie dopravy nájdením nových efektívnejších dopravných spojení.

13 Zoznam použitej literatúry

[1] Bernstein D, K. A. (1996). *New Jersey TIDE Center*. Retrieved from New Jersey TIDE Center: http://www.njtide.org/reports/mapmatchintro.pdf

[2] Câmara, F. P., Costa, H., & Pereira, N. M. (2009). An off-line map-matching algorithm for incomplete map databases. *European Conference of Transport Research Institutes* .

[3] Cao, H., & Wolfson, O. (2005). Nonmaterialized motion information in transport networks. In T. Eiter, & L. Libkin, *ICDT* (s. 173-188).

[4] Edelkamp, S., & Schrödl, S. (2003). Route planning and map inference with global positioning traces. *Perspective (Ottmann Festschrift)* , 128-151.

[5] Edelkamp, S., Pereira, F., Sulewski, D., & Costa, H. (2008). Collaborative map generation—survey and architecture proposal. In F. Hoeven, J. Shaick, S. Speck, & S. MJ, *Urbanism on track—application of tracking technologies in urbanism.* Amsterdam: IOS Press.

[6] Ester, M., Kriegel, H., & Sander, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. of International Conference on Knowledge Discovery and Data Mining*, (s. 226-231).

[7] Greenfeld, J. (2002). Matching GPS observations to locations on a digital map. *Proceedings of the 81th Annual Meeting of the Transportation Research Board.* Washington DC.

[8] Chen, L., Mingqi, L., & Gencai, C. (2010). A system for destination and future route prediction based on trajectory mining. *Elsevier B.V.*

[9] Janeček, M. (2009). Identifikace dopravní infrastruktury na základě dat z GPS měření. *Diplomová práca* . České vysoké učení technické v Prahe.

[10] Kang, J., Welbourne, W., Stewart, B., & Borriello, G. (2004). Extracting places from traces of locations. *Proc. of ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, (s. 110-118).

[11] Kuijpers, B., Moelans, B., Othman, W., & Vaisman, A. (2009). Analyzing Trajectories Using Uncertainty and Background Information. In N. M. al., *SSTD* (s. 135-152).

[12] Lacko, Ľ. (2010). *Silverlight 3 a 4.* Computer Press, a.s.

[13] Lee, S. H., Walters, S. H., & Howlett, R. J. (2008). Intelligent GPS-Based Vehicle Control for Improved Fuel Consumption and Reduced Emissions. In I. Lovrek, R. J. Howlett, & L. C. Jain, *KES* (s. Part III: 701-708).

[14] MacDonald, M. (2010). *Pro WPF in C# 2010: Windows Presentation Foundation in .NET 4.* Apress.

[15] MacDonald, M. (2007). *Windows Presentation Foundation in .NET 3.0.* Apress.

[16] Marchal, F., Hackney, J., & Axhausen, K. (2005). Efficient mapmatching of large global positioning system data sets: tests on speed monitoring experiment in Zürich. *Transportation Research* , 1935:93-100.

[17] Najjar, M. E., & Bonnifait, P. (2005). A road-matching method for precise vehicle localization using belief theory and Kalman filtering. *Autonomous Robots* , 19: 173-191.

[18] *Open Street Map platform.* (30. 06 2010). Dostupné na Internete: http://www.openstreetmap.org

[19] Palma, A., Bogorny, V., Kuijpers, B., & Alvares, L. (2008). A clustering-based approach for discovering interesting places in trajectories. *Proc. of ACM Symposium on Applied Computing*, (s. 863-868).

[20] Palúch, S. (2001). *Teória grafov.* Žilina, SR: EDIS.

[21] Pereira, F. C., Costa, H., & N.M., P. (2009). An off-line map-matching algorithm for incomplete map databases. *European Conference of Transport Research Institutes*, (s. 1:107–124).

[22] Quddus, M. A., Noland, R. B., & Ochieng, W. Y. (2009). The Effects of Navigation Sensors and Spatial Road Network Data Quality on the Performance of Map Matching Algorithms. *Geoinformatica* , 13:85–108.

[23] Quddus, M. A., Ochieng, W. Y., & Noland, R. B. (2007). Current map-matching algorithms for transport applications: state-of-the art and future research directions. *Transportation Research* , Part C Emerg. Technol. 15(5):312–328.

[24] Quddus, M. A., Ochieng, W. Y., Zhao, L., & Noland, R. B. (2003). A general map matching algorithm for transport telematics applications. *GPS Solutions* , 7:157-167.

[25] Quddus, M. (2006). High integrity map-matching algorithms for advanced transport telematics applications. *PhD Thesis* . Centre for Transport Studies, Imperial College London, UK.

[26] Thiébaux, S., & Lamb, P. (2000). Combining Kalman filterig and Markov Loacalization in network-like environments. In R. Mizoguchi, & J. Slaney, *PRICAI* (s. 756-766). LNAI.

[27] White, C., Bernstein, D., & Kornhauser. (2000). Some map matching algorithms for personal navigation assistants. *Transportation Research* , C 8:91-108.

[28] Zhou, C., Shekhar, S., & Terveen, L. (2005). Discovering Personal Paths from Sparse GPS Traces. *Proc. of the JCIS.*

14 Prílohy

- Diplomová práca v elektronickej podobe
- Skript pre vytvorenie prázdnej aplikačnej databázy
- Predvyplnená aplikačná databáza
- Skompilovaná navrhnutá aplikácia