VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

DOLOVÁNÍ NEOBVYKLÉHO CHOVÁNÍ V DATECH TRAJEKTORIÍ

MINING ANOMALOUS BEHAVIOUR IN TRAJECTORY DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. PETR KOŇÁREK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. JAROSLAV ZENDULKA, CSc.**
SUPERVISOR

BRNO 2017

Zadání diplomové práce/19550/2016/xkonar08

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Koňárek Petr, Bc.**
Obor: Informační systémy
Téma: **Dolování neobvyklého chování v datech trajektorií**
Mining Anomalous Behaviour in Trajectory Data
Kategorie: Data mining

Pokyny:

1. Seznamte se s problematikou dolování v datech trajektorií pohybujících se objektů.
2. Podrobněji prostudujte přístupy k dolování neobvyklého chování z dat trajektorií dopravních prostředků.
3. Po dohodě s vedoucím práce specifikujte přesněji dolovací úlohu, na kterou se zaměříte a přístup, který použijete.
4. Navrhněte a implementujte metodu pro řešení dané úlohy.
5. Na vhodných datech experimentálně ověřte vlastnosti navržené metody.
6. Zhodnoťte dosažené výsledky a diskutujte možná vylepšení navržené metody.

Literatura:

- Yu Zheng:Trajectory Data Mining: An Overview. *ACM Transactions on Intelligent Systems and Technology*, Vol. 6, No. 3, Article 29, May 2015. Available at https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/TrajectoryDataMining-tist-yuzheng_published.pdf.
- Cai, Y. et al.: Trajectory-based anomalous behaviour detection for intelligent traffic surveillance. *IET Intelligent Transport Systems*. 2015, Vol. 9, Iss. 8, pp. 810-816. Available at http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7274499.
- Brun, L. at al.: Detection of anomalous driving behaviors by unsupervised learning of graphs. In: Proc. of the 11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2014, pp.405 - 410. Available at http://ieeexplore.ieee.org/document/6918702/?arnumber=6918702&tag=1.

Při obhajobě semestrální části projektu je požadováno:

- Body 1 až 3

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Zendulka Jaroslav, doc. Ing., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem práce je vytvoření přehledu problematiky dolování neobvyklého chování v datech trajektorií. Následuje návrh dolovací úlohy pro detekci odlehlých trajektorií a výběr vhodných metod k této úloze, které jsou detailněji popsány. Vybrané metody jsou implementovány jako aplikace pro detekci odlehlých trajektorií.

Abstract

The goal of this work is to provide an overview of approaches for mining anomalous behavior in trajectory data. Next part is proposes a mining task for outliner detection in trajectories and selects appropriate methods for this task. Selected methods are implemented as application for outliner trajectories detection.

Klíčová slova

trajektorie, dolování dat v trajektoriích, detekce odlehlých trajektorií, detekce anomálií v trajektoriích, GPS, mapování trajektorií, TOP-EYE, iBoat, TRAOD

Keywords

trajectory, trajecory data mining, trajectory outlier detection, trajectory anomaly detection, GPS, trajectory maping, TOP-EYE, iBoat, TRAOD

Citace

KOŇÁREK, Petr. *Dolování neobvyklého chování v datech trajektorií.* Brno, 2017. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Jaroslav Zendulka, CSc.

Dolování neobvyklého chování v datech trajektorií

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doc. Ing. Jaroslava Zendulky, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Koňárek
22. května 2017

Poděkování

Děkuji mému vedoucímu doc. Ing. Jaroslavu Zendulkovi, CSc. za rady a pomoc při řešení této diplomové práce.

Obsah

Kapitola 1

Úvod

V dnešní době je pokročilá lokalizační technika, jako americká GPS, ruský GLONASS nebo evropský GALILEO, či úplně jiné technologie, v téměř každém zařízení, ať už jde o chytrý mobilní telefon, automobil nebo vojenskou techniku. Používáme je při navigaci, snímání cesty během výletů či sportovních aktivit. Služby tyto údaje používají k nabízení možnosti obchodů v blízkosti a k cílené reklamě, či dokonce k zasílání zpráv, že se nacházíte nedaleko nějaké události. Data o pohybu používáme i mimo přímé lidské potřeby. Různé lokátory můžeme použít i ke sledování migrace zvířat. Můžeme si také zaznamenávat směry tornád, bouřek, pohyby vesmírných těles.

Díky tomuto velkému množství zaznamenaného pohybu z těchto zařízení, máme ohromné množství dat, která lze využít mnoha způsoby. Můžeme dolovat mnoho užitečných informací. Jako příklad lze uvést zjišťování hustoty provozu, nejčastější trasy z bodu A do bodu B, shlukování podobných trajektorií za účelem určení jejich podobnosti atd.

Zaměření této práce je ovšem oblast anomálních vlastností a dat, které můžeme z trajektorií získat. Tato data nám pomůžou lépe zhodnotit mnoho věcí. Dolování zde můžeme rozdělit na dvě hlavní kategorie, anomální trajektorie a ostatní anomálie jako zácpy, snížení provozu atd, tedy především anomálie v dopravě. Oba typy znalostí budou dále zmíněny, ale tato práce se bude především zajímat o anomální trajektorie.

Pro tento účel dnes existuje mnoho různých přístupů a řešení. V práci stručně popíšeme několik metod jak pro anomální trajektorie, tak anomálie provozu. Speciálně vybrané poté budou popsány podrobně, jelikož budou později implementovány a testovány. Vybrané metody jsou *iBoat*, *TRAOD* a *TOP-EYE*. Byly vybrány vzhledem k jejich výkonu, práci v online režimu a možnosti detekce části trajektorie, která je anomální. Díky tomu že řeší podobný problém, bude porovnávání značně zjednodušeno.

Cílem práce je kromě poskytnutí přehledu postupů k řešení problematiky odlehlých trajektorií implementace, ověření a porovnání vybraných metod. Pro implementaci byl nejdříve zvolen jazyk Python pro jeho velkou podporu v oblasti dolování dat. Ukázalo se ale, že pro práci s rozsáhlými daty je tato implementace pomalá. Proto byly metody následně naprogramovány v jazyce C++ s využitím rámce Qt5. Data zvolená pro testování metod jsou Portugalské taxíky[1] a databáze hurikánů[2], aby bylo možné porovnat různé typy trajektorií.

[1] https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015

[2] http://weather.unisys.com/hurricane/atlantic/

Práce je rozdělena do šesti kapitol. První kapitolou je úvod, kde je popsána motivace dolování v trajektoriích pohybujících se objektů. Druhá kapitola přibližuje obecnou práci s trajektoriemi jako předzpracování, komprese a obecné dolovací techniky jak nad obecnými daty, tak nad daty trajektorií. Třetí kapitolou začíná jádro práce, kde jsou definovány různé typy neobvyklé chování v trajektoriích a ke každému typu je popsáno několik metod, které se zabývají jejich řešením. Ve čtvrté kapitole jsou detailněji specifikovány vybrané metody pro danou dolovací úlohu. Pátá kapitola pokračuje popsáním implementace vybraných metod. Předposlední šestá kapitola vysvětluje výsledky experimentů dolování. Závěrečná sedmá kapitola je závěr, kde je shrnut obsah práce a výsledky dolování.

Kapitola 2

Trajektorie pohybujících se objektů

Dolování dat z trajektorií je náročný proces a k výsledku se dostaneme až po provedení několika kroků. Jednotlivé fáze zpracování si ukážeme v této kapitole. Převážná část informací pro tuto kapitolu ohledně trajektorií byla čerpána z [25, 26, 7].

Jako první je nutné provést předzpracování trajektorií dle potřeb dolovací úlohy. Konkrétně se bude jednat o filtrování, kdy bod značně vybočuje z cesty. Dále můžeme uvažovat o kompresi trajektorií. Snižováním počtu bodů v trajektorii totiž dosáhneme především zrychlení dolování. Takové zrychlení je velice vhodné při online zpracování. Dalším krokem je trajektorie mapovat do reálného zobrazení např. na segmenty silnic, či do jednoduché mřížky atd. Můžeme pak tyto vlastnosti využít při dolovací úloze.

Další část analýzy se bude zabývat základními dolovacími úlohami v obecných datech a poté v oblasti trajektorií. Jde např. o klasifikaci, shlukování, vzory atd.

Následně již přejdeme k užšímu tématu této práce. Vysvětlíme si, co to je problematika dolovaní neobvyklého chování, co z těchto dat můžeme získat a jaké existují metody a na jakém principu pracují.

2.1 Trajektorie

Je křivka generována v prostoru pohybujícím se objektem např. vozidlem, ptákem, tornádem. Tato křivka je většinou definována pomocí několika bodů v prostoru, které mají časovou známku a podle času jsou také seřazeny do sekvence.

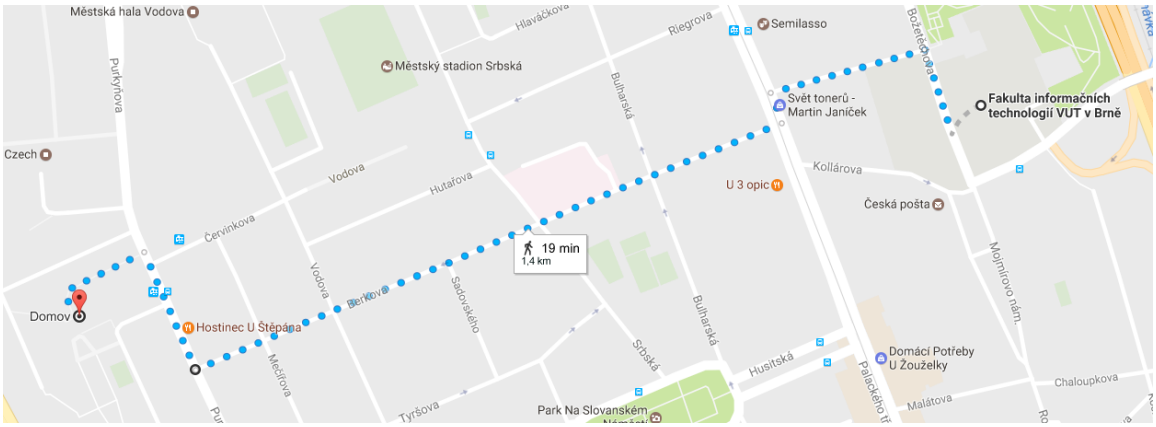
Definice 2.1. ***Bod*** *trajektorie je dvojce* $p = (t, pos)$*, kde* t *je čas zaznamenání pozice a* pos *je fyzická lokace.*

Definice 2.2. ***Trajektorie*** *obsahuje* $\langle p_1, p_2, ..., p_n \rangle$*, kde* $p_i (i = 1, ., n)$ *je bod trajektorie a* n *je počet bodů v trajektorii. Body jsou sekvenčně seřazené podle času.*

Dnes generujeme trajektorie prováděním našich každodenních činností, jako např. na obrázku 2.1, který ukazuje moji trajektorii z kolejí na FIT VUT ve městě Brno. Zdrojem dat trajektorií jsou zejména chytré mobilními telefony, které v sobě skrývají různé technologie na připojení, především WiFi, Bluetooth a v první řadě satelitní poziční systémy např. GPS. Různá sledovací zařízení však můžeme připojit na zvířata nebo objekty sledovat radarem apod. Tímto získáme ohromné množství dat, které lze zpracovávat a informace z nich získané využít k různým aplikacím.

Získaná data lze následovně využít pro sledování způsobu pohybu lidí mezi obchody, domovem a prací. Tedy zobrazovat cílené reklamy, dopravní situace atd. nebo analyzovat

trajektorie po dopravních komunikacích a plánovat objížďky v případě dopravní zácpy, výstavby nových silnic atd.



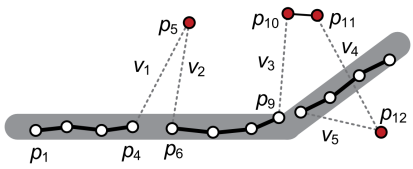
Obrázek 2.1: Moje cesta z kolejí na FIT VUT v Brně.

2.2 Předzpracování

I když máme dnes mnoho zařízení schopných snímat naši pozici, ne vždy je to s dobrou přesnosti, kde např. poziční systémy typu GPS, mohou mít i několik desítek metrů nepřesné údaje. U jinak přesného systému, může také dojít k různým odchylkám měření. Proto se tato sekce zaměří na předzpracování trajektorií, abychom tyto chyby odstranili a dále trajektorie upravovali. Jelikož dnešní svět generuje příliš mnoho trajektorií a abychom dosáhli rozumné rychlosti zpracovaní, musíme data redukovat, komprimovat. Pro získání dalších vlastností můžeme trajektorie dělit na segmenty. Každá metoda používá vlastní druh segmentace, podle potřeb úlohy, proto jsou metody segmentace popsány až u jednotlivých metod dolování neobvyklého chování. Pro některé další úlohy je nutné trajektorie mapovat na dopravní síť, či jinou reprezentaci.

2.2.1 Vyhlazování

V naměřených bodech trajektorií se může vyskytovat šum čili že některé body jasně vybočují. Tento šum totiž naznačuje, že nejspíše byla naměřena špatná hodnota. Tyto hodnoty buď potřebujeme eliminovat nebo nahradit přibližnou hodnotou, která správně zapadá do trajektorie [26] (strana 20-33).



Obrázek 2.2: Chybně naměřené hodnoty v trajektorii [25].

Průměr a medián

Nejjednodušším způsobem korekce bodu se šumem je využití průměru nebo mediánu předchozích naměřených bodů. Počet předchozích bodů, které se využijí k výpočtu, záleží na velikosti okna. Velikost okna si určíme dle potřeb aplikace. Medián vychází lépe, pokud jde o body, které leží extrémně mimo trajektorii čili je robustnější vůči šumu. Pokud je ovšem šum několik bodů následujících po sobě, jako jsou body p_{10}, p_{11}, p_{12} v obrázku 2.2, výkon filtrů záleží na velikosti okna, které je použito. Více bodů použitých pro korekci, ale znamená náročnější výpočet.

Kalmanův a částicový filtr

Kalmanův filtr ke korekci bodů využívá odhad následující hodnoty. Filtr se skládá ze dvou modelů. Model měření a dynamický model. K odhadu následující hodnoty využívá několika hodnot jako vzdálenost, akceleraci, rychlost ale především bere v potaz zákony fyziky jako např. gravitaci.

Částicový filtr je podobný Kalmanově. Také používá dva stejné modely - měření a dynamický. Je ovšem více obecný, ale na úkor obecnosti také výpočetně náročnější.

Detekce a odstranění odlehlých bodů

Nesprávně naměřenou hodnotu zjistíme porovnáním vůči předchozím hodnotám, především rychlosti a vzdálenosti. Spočítáme vzdálenost nebo rychlost mezi všemi body. Tyto hodnoty následně porovnáváme s určitým prahem a pokud přesahují. Bod detekovaný jako odlehlý odstraníme z trajektorie.

2.2.2 Redukce bodů

Pokud bychom chtěli zpracovávat přímo všechny jednotlivé body trajektorií, tak výpočetní systémy, které zrovna máme k dispozici, takové objemy dat nemusí zvládnout nebo chceme omezit využití úložného prostoru. Je proto v některých případech nutnost redukovat počty bodů. Této redukce musíme dosáhnout tak, abychom vůbec nebo jenom minimálně změnili trajektorii a tím zachovali její datovou hodnotu.

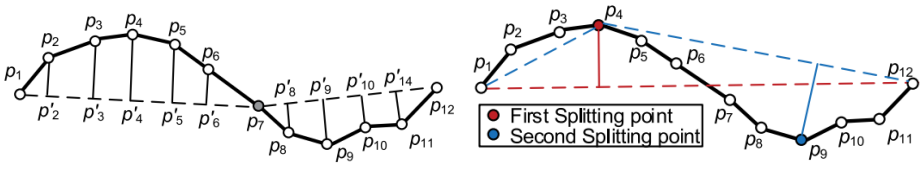
Dávková redukce (offline)

Redukce se provádí, až když máme celou trajektorii k dispozici. Nejjednodušší metoda je ponechání každé i-té hodnoty. Tím ovšem můžeme ztratit velmi důležité informace. Proto se využívají následující metody:

Douglas-Peucker dělí trajektorii na dvě subtrajektorie a poté porovnává jejích chybovost s předchozí pomocí eukleidovské vzdálenosti viz obr. 2.3. Pokud nesplňuje daný práh chyby, rekurzivně dělí dále, a to podle bodu, který se nejvíce podílí na vypočítané chybě. Algoritmus je zobrazen na obrázku 2.3, kde první bod nesplnil práh a proto je přidán druhý.

Top-down time-ratio je modifikace předchozího algoritmu, kde pro výpočet chyby se bere v úvahu i časová rovina.

Bellman je další vylepšení, které zajišťuje že dělení je nejlepší možné. Aplikuje dynamickou techniku pro aproximaci spojité funkce podle konečného počtu segmentů.



Obrázek 2.3: Eukleidovská vzdálenost a Douglas-Peucker algoritmus [25].

Online redukce

Některé případy potřebují redukci již během postupného získávání trajektorie. K této problematice nelze samozřejmě použít předchozí algoritmy. Všechny následují algoritmy pro online redukci používají nějaký způsob redukce pomocí okna.

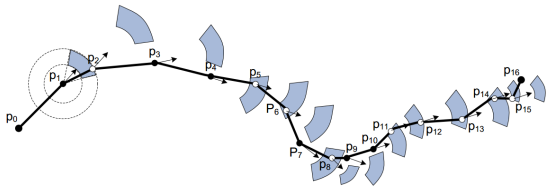
Reservoir sampling využívá tzv. rezervoáru určité velikosti R. Tato velikost není překročena. Při příchodu nového bodu trajektorie je rozhodnuto, jestli bude vložen do rezervoáru. Pokud ano a je místo, tak je vložen, pokud místo v rezervoáru není, tak je náhodně vybrán jeden bod a ten je nahrazen bodem novým. Hlavní nevýhoda je náhodnost odstranění bodů z rezervoáru.

Sliding window do kterého přidáváme postupně body trajektorie. Při každém přidání spočítáme rozdíl mezi trajektorií v okně a prvním a posledním bodem v okně. Pokud splňuje práh chyby daný uživatelem, přidá se do okna další bod a krok se zopakuje. Pokud práh chyby nesplňuje, první bod okna se zapíše do výsledné trajektorie a poslední bod okna se stává prvním.

Open window Stejné jako předchozí metoda jenom pro určení bodu s maximální prahovou hodnotou používá Douglas-Peucker algoritmus.

Redukce závislá na rychlosti nebo směru

Předchozí algoritmy byly založené na vzdálenosti. Existují ovšem i další možnosti. Prahově založené vzorkování (Threshold-guided sampling) je jednou z nich. Metoda je založená na směrové a rychlostní toleranci. Nový bod je predikován, v jaké oblasti by se měl přibližně nacházet. Pokud je predikce správná, následující bod není přidán do výsledné trajektorie. K výpočtu se používají poslední dva přidané body. Dobře zobrazené je to na následujícím obrázku 2.4.



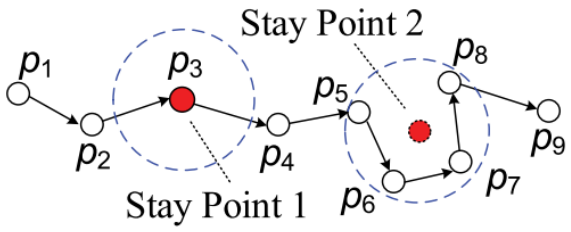
Obrázek 2.4: Prahově založené vzorkování [26].

Body zastavení

I když detekce bodů zastavení 2.5 není přímo metoda k redukci, je zařazena sem, jelikož se dá i k této funkci využít. Ale většinou je využita na detekci zájmových oblastí, kde trávíme

více času a tím pádem více časově po sobě jdoucích bodů trajektorie je na přibližně jednom místě. Pokud cestujeme autem může se jednat i o zácpu, dopravní nehodu apod.

Jak bylo řečeno, jde o více bodů na stejném místě čili tyto body můžeme nahradit pouze jedním bodem, pokud to nezhorší výsledky dolovací úlohy. Příklad je na obrázku 2.5, kde bod zastavení 2 může nahradit body, které se nachází okolo.



Obrázek 2.5: Detekce bodů zastavení [25].

2.2.3 Mapování

Pokud chceme využít topologii města, kde se trajektorie nacházejí, musíme je namapovat na silniční síť města. Tento úkon není triviální, jelikož zde narážíme na nepřesnost získávání jednotlivých bodů trajektorie, především pomocí satelitních lokalizačních systémů. Body totiž nejsou přímo na silnici, po které jedeme. Dalším problémem je, že snímání pozice nemusí mít vhodnou frekvenci a tím pádem mezi body může být několik segmentů silnic mezera. K tomu se připočítá další problém, že mezi naměřenými body existuje několik možných cest. Jelikož tedy jde o velice složitou věc, bude zde vypsán pouze přehled některých základních způsobů a detailněji popsané budou případně až přímo u konkrétních metod dolování neobvyklého chování, které je používají. Pro více informací doporučují přehled, ze kterého bylo čerpáno [25]. Existují tři základní metody mapování:

Geometrická mapuje body na nejbližší segmenty. Jednoduchá metoda, ale vhodná pouze když máme velkou hustotu bodů.

Topologická při výpočtu bere v potaz i strukturu cesty. Některé metody využívají Fréchetovu vzdálenost pro výpočet vhodnosti mezi trajektorií a cestou složenou ze segmentů silnic [17].

Pravděpodobnostní metody vytváří několik možných cest mezi body a dle pravděpodobnosti vybírá tu správnou.

2.3 Přehled obecných dolovacích technik

V této sekci budou stručně přiblíženy některé všeobecné metody pro dolování v obecných datech a jejich specifikace pro data trajektorií. Půjde o klasifikaci a predikci, dolování frekventovaných vzorů a asociačních pravidel, shlukování a odlehlé hodnoty. Čerpáno bylo především z těchto dvou publikací [8, 19].

2.3.1 Klasifikace a predikce

Pomocí klasifikace přidáváme datům sémantickou hodnotu. Toho dosáhneme analýzou a následným rozdělením dat do různých tříd dle jejich hodnot. Pro to, aby bylo možné data

správně klasifikovat, nejdříve potřebujeme vytvořit tzv. klasifikační model. Model vytváříme z trénovacích dat. Jde o sadu, kde známe skutečnost, do jakých tříd data patří. Následně nad další testovací množinou, kde opět víme, do jakých tříd data spadají, rozčleníme data do tříd pomocí vytvořeného modelu a porovnáme, jestli model správě přiřadil třídy. Tímto způsobem zjistíme, jestli byl model korektně vytvořen. Díky přiřazením dat do tříd můžeme analyzovat dříve skryté poznatky, jako např., že nejvíce elektroniky nakupují 30 letí muži s vysokým příjmem.

Predikce je proces zjištění dalších dat nad daným objektem. Jeden z typů predikce je vlastní klasifikace popsána výše. Druhým typem je predikce hodnot dle spojité funkce. Např. kolik by mohl daný zákazník utratit při těchto slevách na základě jeho starších dat.

Pro klasifikaci a predikci dat existuje mnoho metod. Zde uvedeme některé z nejčastěji používaných se stručným popisem. Pro detailní popis a další metody viz. [8, 19].

Rozhodovací strom je struktura, kde každý uzel představuje hodnotu nebo rozsah hodnot dle které se rozhoduje při klasifikaci. Koncové listy stromu poté představují již jednotlivé třídy klasifikace.

Bayesovská klasifikace rozhoduje o příslušnosti ke třídě na základě statistické pravděpodobnosti příslušnosti k dané třídě.

Pravidly řízená klasifikace je založena na podmínkách které rozhodují o přiřazení do konkrétní třídy. Podmínky jsou často generovány z rozhodovacího stromu.

Metoda podpůrných vektorů (SVM: Support Vector Machines) je vhodná pro klasifikaci jak lineárních, tak nelineárních vícedimenzionálních dat. Mapuje data na vícedimenzionálního prostor a následně hledá optimální rovinu, která odděluje třídy největší vzdáleností.

Neuronové sítě se zpětným šířením chyby se také mohou použít ke klasifikaci. Model se upřesňuje pomocí úprav vah vstupů jednotlivých neuronů.

Regrese je využívána při predikci. Nejčastěji se využívá lineární regrese, kde tvoříme křivku, které je vytvořená metodou nejmenších čtverců nebo násobnou lineární regresí nad již získanými daty. Existují také nelineární regrese, jako třeba polynomiální.

2.3.2 Frekventované vzory a asociační pravidla

Vzory, jež se v datech objevují přesněji nazýváme frekventované vzory. Tento pojem se často používá se spojením s transakčními databázemi, ale může jít i o frekventovaný podgraf, podstrom a další. Nalezení těchto vzorů je důležité pro nalezení asociací, korelací a dalších vztahů mezi daty.

Pro tvorbu asociačních pravidel potřebujeme právě frekventované vzory, jež vydolujme např. z množiny nákupních transakcí. Pokud se nám objevuje často nějaký vzor můžeme zjistit další informace. Třeba zjistíme že jeden z frekventovaných vzorů v nákupech je digitální fotoaparát a paměťová karta. Díky tomu můžeme vytvořit asociační pravidlo mezi těmito položkami nákupů. Pro tuto analýzu používáme tzv. podporu (anglicky *support*) a spolehlivost (anglicky *confidence*). Podpora znamená počet nákupů v %, kde se spolu objevily fotoaparát a karta. Důvěra označuje v kolika % případů při koupi fotoaparátu byla koupena i karta. Asociace zapisujeme následovně:

$$fotoaparát \Rightarrow karta \quad [support = 10\%, confidence = 85\%] \tag{2.1}$$

Jelikož nás ovšem nezajímají všechny frekventované množiny a všechna asociační pravidla, dolujeme pouze ty, které splňují tvz. minimální podporu (*minimal support*) a minimální spolehlivosti (*minimal confidence*). Tyto hodnoty jsou zpravidla v zadány v % z celkového počtu transakcí, položek v databázové tabulce.

2.3.3 Shluky

Proces, který objekty přiřazuje do stejné třídy neboli shluku, se nazývá právě shlukování. Objekty přiřazené do jednoho shluky si musí být značně podobné, ale značně rozdílné oproti objektům z ostatních shluků. Na rozdíl od klasifikace, třídy nejsou předem známy a vytváří se až právě během shlukování. Což je značná výhoda, když nemusíme třídy vytvářet, jelikož u většího množství dat jde o náročnou práci. Shlukování můžeme využít i k detekci odlehlých hodnot. Takové hodnoty jsou totiž příliš vzdálené k jakémukoliv shluku.

Některé ze shlukovacích metod jsou následující. Pro více detailů viz. [8, 19].

Hierarchické shlukování rekurzivně rozděluje prostor hodnot atributů objektů tzv. shora-dolů nebo zdola-nahoru. Shora-dolů znamená, že všechny objekty na začátku patří do jednoho shluku a ten dále dělíme na menší shluky podle určitého prahu podobnosti objektů ve shluku. Zdola-nahoru je metoda kdy každý objekt na začátku patří do svého shluku a ty nejpodobnější jsou následně spojování do větších celků.

Metody rozdělování přemísťují objekty z jednoho shluku do druhého, dle podobnosti objektu se shlukem. U této metody shlukování typicky musí uživatel zvolit počet shluku, což je značně nevýhodné (nutnost znalost dané oblasti apod.). Následně se do každého shluku vloží náhodně jeden objekt. Ostatní se dle podobnosti přidávají. Do shluku se přidává např. podle průměrné hodnoty objektů ve shluku nebo podle objektu, který je nejblíže středu apod. Jelikož se tak shluk mění, mohou některé body padnou v další iteraci do jiného shluku.

Metody založené na hustotě shlukují objekty, jež jsou blízko u sebe ve větším množství a od ostatních odděleny prostorem s řídkým obsazením objektů. Tímto způsobem vznikají shluky přirozenějšího tvaru a dokážeme také dobře detekovat odlehlé hodnoty.

Shlukování pomocí mřížky , nad kterou jsou prováděny operace shlukování. Tato mříž dělí prostor na konečný počet buněk. Díky tomu jsou operace rychlejší, jelikož jsme závislí pouze na počtu buněk, a ne počtu objektů. Čili jde především o zrychlení shlukování.

Shlukování dle modelů zajišťuje, aby shluky byly co nejvíce podobny nějakému matematickému modelu.

Metody soft-computingu znamená využití neuronových sítí pro výpočet shluků (fuzzy a evoluční techniky).

2.3.4 Detekce odlehlých hodnot

Odlehlá hodnota je taká hodnota, která se odlišuje od většinové množiny ostatních hodnot, ať už vlastnostmi nebo v případě trajektorií prostorovou vzdáleností. Většina dolovacích technik vidí odlehlé hodnoty jako šum, které dokáží znehodnocovat jejich výsledky, a proto se je snaží detekovat a poté smazat nebo nahradit nějakou jinak běžnou hodnotou ve fázi

předzpracování dat např. průměrem. V některých případech však odlehlá hodnota naznačuje, že může být něco v nepořádku, např. úniky elektrického napětí v přístrojích, peněžní podvody apod. Následují některé přístupy k dolování odlehlých hodnot dle [8, 19].

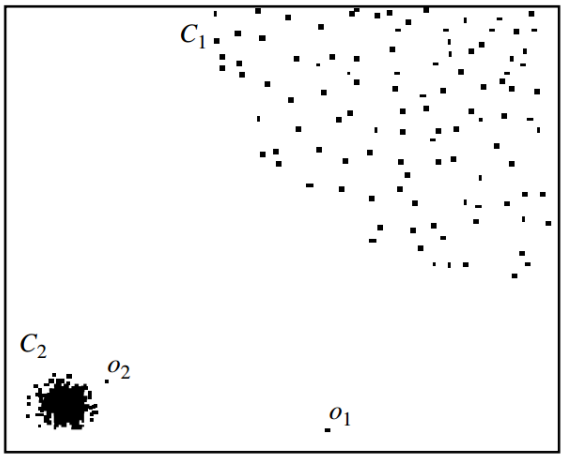
Statistické metody hledají odlehlé hodnoty dle nějakého pravděpodobnostního rozložení (např. Poissonovo). Potíž je v tom, že ne vždy předem víme, jaké pravděpodobnostní rozložení data mají.

Podle vzdálenostní metody určíme počet sousedů. Pokud nalezneme méně sousedů v okolí daném parametrem pro vzdálenostní funkci, je hodnota označena jako odlehlá.

Pomocí shlukování lze také nalézt odlehlé hodnoty. Dle vybrané metody shlukování, jsou odlehlé hodnoty ty, které nepatří do žádného shluku nebo jsou nejdále od středu či průměru shluku.

Metody založené na hustotě určují odlehlost na základě lokálního sousedství. Tím dosáhneme detekci odlehlých hodnot i v různorodě hustých datech. Příklad je ukázán na obrázku 2.6. Body o_1 a o_2 jsou označeny jako odlehlé. Ovšem metoda založená např. na vzdálenosti by detekovala pouze bod o_1

Metody založené na odchylce analyzují objekty v dané skupině a pokud se objekt odlišuje od skupiny je prohlášen jako odlehlý.



Obrázek 2.6: Ukázka různých odlehlých hodnot (o_1, o_2) [8].

2.4 Dolování v datech trajektorií

Při dolování v datech trajektorií si musíme uvědomit o jaký typ dat jde a co za informace nám může nabídnout. Nejjednodušší reprezentace trajektorie je pomocí bodů ve 2D prostoru. Jenže k tomu nám může přibýt další rovina jako čas, třetí prostor (výška u letadel), rychlost, úhel atd. S takovými daty tak můžeme pracovat jako s prostorovými nebo časoprostorovými. Díky tomu obecné metody bývají nedostatečně efektivní nebo nedokáži potřebné informace vydolovat. Proto se vytvářejí speciální techniky pro dolování v trajektoriích, potažmo všeobecně v časoprostorových datech.

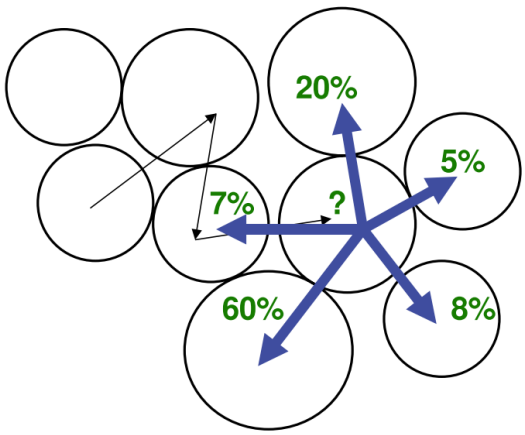
2.4.1 Klasifikace a predikce

Díky klasifikaci můžeme u trajektorií zjišťovat specifické chování aut na silnici nebo lidí, kteří zastavují u různých obchodů, nebo zpracování statistik ze sportovních aktivit. Tyto trajektorie potřebujeme nějak rozčlenit, klasifikovat. Jako příklad můžeme uvést jakým prostředkem se osoba zrovna pohybuje, jestli v autě, na kole, v autobuse či jde pěšky. Klasifikace trajektorií většinou prochází těmito fázemi. První fáze je dělení trajektorie na segmenty, kde segment může být několik po sobě jdoucích bodů, ale i jenom jeden. Druhá fáze je extrakce vektoru vlastností ze segmentů. Poslední třetí váze je vytvoření modelu a klasifikace segmentů do tříd. Můžeme tedy použít vytváření modelů pomocí známých technik, případně jejich upravených verzí přímo pro trajektorie. Tyto techniky můžeme použít, protože trajektorie je ve skutečnosti sekvence. Mezi tyto techniky patří skryté Markovovy modely (HMM), dynamické Bayesovské sítě (DBN), podmíněné náhodné pole (CRF) a jiné.

Často pro klasifikaci trajektorií používáme jejich shluky, jelikož můžeme přidat trajektoriím další hodnoty, které později použijeme v klasifikátorech. Jako například v práci [11] se vytvářejí dva různé typy shluků. Hierarchický založený na oblastech a poté shluky založené na trajektoriích. Díky dodatečně získaných informací ze shluků, je vytvořen vektor hodnot pro každou trajektorii a ty jsou následně klasifikovány pomocí SVM. Pro více informací doporučují přehled, ze kterého bylo čerpáno [25].

Predikce trajektorií nám umožňuje např. zabránit srážkám lodí, letadel, zefektivnit následující cestu nalezením kratší cesty, která bude bez provozu apod. Predikovat můžeme několik vlastností jako poloha, hustota provozu či možných událostí jako zácpa způsobených nadměrným množstvím pohybujících se objektů.

Pro predikci hustoty můžeme využit rozdělení zkoumané oblasti do mřížky nebo na shluky a poté na základě historie hustoty předpovídat jaká hustota bude v následujících časových slotech. S předpovědí hustoty můžeme zjistit případnou dopravní zácpu. Další možností je predikce dalšího bodu pohybu trajektorie. Nejjednodušší způsob je získání rychlosti a směru objektu, a tak vypočítat další bod. Tato metoda, ač velice jednoduchá není příliš přesná, jelikož objekt může kdykoliv změnit směr. Další možností je opět rozdělení prostoru na mříž nebo nalezení shluků a vytvořit pravděpodobnostní model, kde vypočítáme pravděpodobnost přechodu mezi sousedními shluky či buňky mříže jako v obrázku 2.7. Na podobném principu pracuje metoda TOP-EYE popsána v sekci 4.3.



Obrázek 2.7: Predikce pohybu trajektorie na základě pravděpodobnosti pohybu mezi shluky [7].

2.4.2 Frekventované vzory a asociační pravidla

Frekventovaných vzorů můžeme využít i u trajektorií, jelikož i ty mohou mít společné, frekventované atributy, jako body, rychlost, tvar, periodické opakování průchodu atd. Tyto vzory nám mohou pomoci detekovat dopravní situace, zajímavá místa nebo odlehlé hodnoty. Vzory se dají rozčlenit do následujících skupin, spolu pohybující se vzory, sekvenční a periodické vzory. Asociační pravidlo je podobné jako nákup u obecných technik. Může jít třeba o podporu a spolehlivost bodů pohybujících se bodů, kde zjistíme, jaká je podpora výskytu dvou bodů v trajektoriích a jaká je důvěra že tyto body se vyskytují v sekvenci ihned za sebou. Toto pravidlo je vhodné pro predikci směru na základě historických dat.

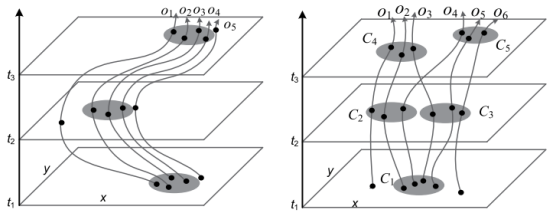
Skupinové vzory

Skupiny objektů, které se spolu pohybují mohou mít nějaký význam. Musíme tedy zjistit, které objekty se takto ve skupinách pohybují. Byly vytvořeny různé typy pohybujících se shluků jako např. stádo *(flock)*, konvoj *(convoy)*, roj *(swarm)*, cestující společník *(traveling companion)*, shromáždění *(gathering)*.

Stádo jsou objekty v určitém prostorovém rozsahu cestující spolu minimálně určitou dobu. Kde stádo je limitováno určitou velikostí, tak konvoj používá na hustotě založené shlukování. V konvoji musí být každý objekt schopný se spojit s jiným bodem přes ostatní, kde vzdálenost mezi dvěma nejbližšími body není menší než určitý práh. Jak u stáda, tak konvoje musí být všechny body po celou dobu cesty spolu časově spojené. Toto omezení ovšem neplatí pro roj, kde je dovoleno, aby se objekt dočasně od roje odklonil.

Všechny předchozí metody potřebují mít k dispozici celou trajektorii. Metoda *traveling companion* si uchovává svoji speciální strukturu a v ní hledá konvoj, roj.

Shromáždění detekuje shluky bodů, které mají určitou minimální velikost a jsou si také prostorově blízké. Tato shromáždění se mohou neustále měnit. Lze tak detekovat různé události jako oslavy, dopravní nehody atd.



Obrázek 2.8: První část ukazuje stádo, konvoj, roj, druhá část je shromáždění [25].

Sekvenční a periodické vzory v trajektoriích

Další možnou informaci, kterou lze dolovat z trajektorií jsou sekvenční vzory. Jde o objekty, které spolu cestují přes podobná místa trasy v přibližně stejný čas. Nemusí jít o přesně stejná místa a mezi nimi se mohou nacházet i ostatní, ovšem musí být ve stejném pořadí a přibližně stejném časovém rozmezí, které si uživatel nastaví jako určitý práh. Znalost lze poté využít k predikci cest, k doporučení míst atd.

Pro detekci musíme nejdříve zjistit důležité body. To lze vydolovat z různých turistických map, kde památky a jiná významná místa označíme právě jako důležité body. Nebo analyzujeme databázi trajektorií a nalezneme tato místa pomocí některé ze shlukovacích metod nebo můžeme použít body zastavení 2.2.2.

Dalším vzorem je vzor periodický. Jde o opakující se událost jako každodenní jízda autobusem do práce, nakupovaní apod. Díky tomu můžeme předvídat budoucí pohyb nebo detekovat změny v chování. Díky fuzzy povaze dat jde o celkem složitý problém. Trasy se mohou překrývat, být různě rozdělené v čase atd.

2.4.3 Shlukování

Jelikož trajektorie objektů jsou většinou ve 2D nebo ve 3D prostoru, tak se často využívá vzdálenost bodů nebo jejich segmentů. Tu můžeme vypočítat pomocí eukleidovské vzdálenosti nebo manhattanské vzdálenosti. Shlukování pomocí vzdálenosti segmentů řeší J. LEE a ostatní [12], pomocí Hausdorffovy vzdálenosti pro trajektorie. Další z obecných metod je vytvoření vektoru vlastností z dat trajektorie a měřit vzdálenost mezi těmito vektory. Není ovšem lehké vytvořit takový vektor, který by dobře reprezentoval trajektorii a zároveň šel porovnávat s ostatními vektory. To z důvodů, že trajektorie jsou značně proměnlivé. Mají různou délku, tvar, frekvenci bodů a jejich vzdálenost mezi sebou atd. Další možností hledání shluků [18] je namapovaní trajektorií na silniční síť a tím získáme graf. Nad takto vytvořeným grafem poté můžeme používat shlukovací metody nad grafy.

2.4.4 Detekce odlehlých hodnot

Odlehlé hodnoty v trajektoriích jsou jednoznačně samotné trajektorie (obrázek 3.1) nebo jejich body či segmenty, které vyčnívají ze shluků, tak jak byly odlehlé hodnoty popsány výše. Ovšem může se jednat i o tzv. neobvyklé chování, které již není tak jednoduše detekovatelné, jelikož je skryté právě v chování trajektorií. Pro dolování odlehlosti se využívají dříve popsané metody. V [10] využívají vzdálenostní metodu spolu s hustotou. V [2] používají dělení prostoru do mřížky. Detailnějšímu popisu se věnuje následující kapitola 3.

Kapitola 3

Neobvyklé chování

V této kapitole začíná jádro práce, dolování neobvyklého chování v trajektoriích. Konkrétní zaměření jsou odlehlé trajektorie, ale budou zde stručně popsány i některé metody na dolovaní jiných druhů neobvyklého chování v dopravě. Vybrané metody, které byly implementovány a následně na nich prováděny testy, jsou popsány v kapitole 4.

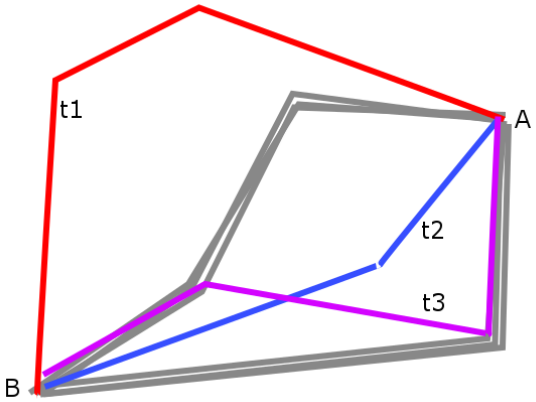
V mnoha aplikacích nám přijde vhod hledání neobvyklých vzorů, chování, odlehlých trajektorií, díky kterým můžeme detekovat nebo i predikovat různé události. Tyto informace nejsou však často na první pohled vidět, a proto existuje celá řada metod a algoritmů díky kterým, je můžeme získat. V následujících sekcích se nejprve podíváme na odlehlé trajektorie a následně na ostatní typy neobvyklé chování. U každé sekce je uvedeno, o jakou problematiku se jedná a je prezentováno několik různých metod, které ji řeší.

3.1 Odlehlé trajektorie

Jeden z hlavních typů neobvyklého chování v trajektoriích jsou tzv. odlehlé trajektorie, anglicky *outlier*. Jde o takové trajektorie, které jasně vybočují z cesty, kde se pohybují ostatní objekty. Odlehlé trajektorie se mohou lišit nejenom tím, že mají značně odlišnou trasu nežli ostatní, ale také rychlostí, časem průjezdu, tvarem. Díky detekci takových nezapadajících trajektorií můžeme detekovat podvádějící taxikáře, neočekávané odklonění dopravy nebo i to, že jezdec zabloudil. Dolování lze i aplikovat na jiné jevy, jako vychýlená tornáda, zvířata, jenž migrují do jiných oblastí atd.

Na následujícím obrázku 3.1 můžeme vidět několik druhů odlehlých trajektorií z A do B (vyznačený barevně), kde některé se nám takové na první pohled zdát nemusí. Třeba t_1 je jasně detekovatelná, t_2 a t_3 už nemusí být tak jednoduché rozpoznat. Trasa t_2 je evidentně zkratka, ale na druhou stranu stále značně mimo ostatní cesty a t_3 částečně využívá cestu, kde se pohybují ostatní objekty. Obzvláště t_3 je těžké detekovat, jelikož se pouze na chvíli stane odlehlou a před tím i potom opět využívá cestu, která se jasně používá nejčastěji. Pokud máme daný určitý práh tak t_3 může být považována za normální trajektorii.

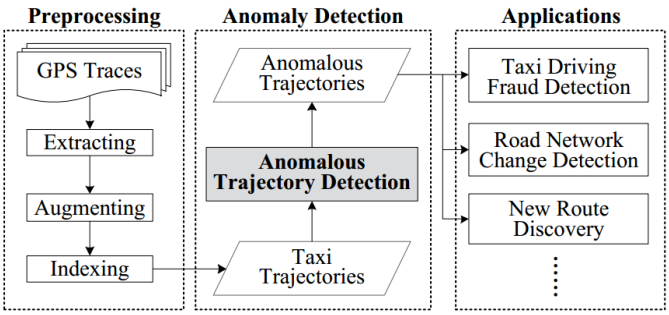
V následujících podkapitolách jsou popsány metody reprezentující některé z různých přístupů k detekci odlehlých trajektorií.



Obrázek 3.1: Ukázka odlehlých trajektorií.

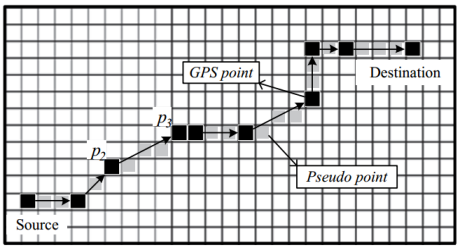
3.1.1 Metoda iBat

Zhan a ostatní [24] navrhli metodu založenou na izolování anomálních trajektorií *(Isolation Based Anomalous Trajectory)* neboli *iBat*. Metoda se skládá ze tří kroků zobrazené na obr. 3.2. První krok je rozdělení mapy pomocí mříže, kde všechny buňky mají stejnou velikost. Následně se trajektorie konvertují na posloupnost buněk, kde trajektorie seskupíme podle počáteční a cílové buňky. Z toho vyplývá, že detekce bude probíhat pouze mezi trajektoriemi, které mají stejný počátek a konec. Druhým krokem je samotná metoda detekce, kde se využívá jejich vlastností "few and diffrent", čili několik a rozdílné. Třetí krok je použití vydolovaných dat v nějaké reálné aplikaci.



Obrázek 3.2: Koncepce metody iBAT [24].

Při převodu trajektorií na buňky mříže se může stát, že ne všechny buňky budou propojené neboli trajektorie bude spojitá. Příklad mapování trajektorie je ukázán na obr. 3.3, kde jsou zobrazeny i umělé body. Existuje ovšem jednoduchá metoda dosáhnutí spojitosti. Vloží se umělé body dle přímky, která vzniká mezi reálně naměřenými. Po převodu potřebujeme uložit novou reprezentaci tak, aby v ní šlo lehce najít všechny trajektorie, které začínají v určité buňce a končí v jiné. Metoda proto používá invertované indexování, kde máme uložené pro každou buňku, jaká trajektorie jí prochází a kolikátá to je buňka v pořadí. Tento způsob byl také vybrán s ohledem na to, že velmi dobře řeší problematiku, pokud neexistuje mnoho trajektorií, které ve vybraných buňkách začínají a končí. Takhle můžeme jednoduše do výsledku započítat i trajektorie, které body pouze procházejí.



Obrázek 3.3: Mapování trajektorií na buňky mřížky [24].

Pro samotnou detekci je použit upravený algoritmus *iForest* [3], který právě využívá vlastnosti "few and diffrent". Aplikuje náhodné rozdělovaní, dokud nejsou jednotlivé trajektorie izolované nebo se dále nedá dělit. Tímto postupem se vytváří struktura *iTree*.

Úprava, kterou *iBat* zavedl, je "lazy learning". Místo aby vytvářel celý *iTree*, zjišťuje odlehlost pouze na jedné trajektorii. Vezme z trajektorie náhodně jednu buňku a odstraní zbytek trajektorií, které tuto buňku neobsahují. Tento postup opakuje, dokud je trajektorie osamocena nebo nejde odstranit další trajektorie.

Pro celkovou funkčnost pomocí *iForest* je nutné zadat parametr $n(t)$, který určuje počet buněk nutný k izolaci trajektorie a **skóre anomálnosti** danou následujícím vzorcem, jako normalizaci průměrného počtu použitých bodů.

$$s(t, N) = 2^{-\frac{E(n(t))}{c(N)}} \tag{3.1}$$

kde $E(n(t))$ je průměrný počet buněk použitý pro izolaci trajektorie t, N je počet trajektorií oddělených od t a $c(N)$ je průměr $n(t)$ pro dané N. Tedy,

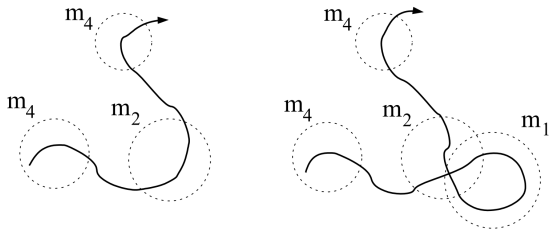
$$c(N) = 2H(N-1) - 2(N-1)/N \tag{3.2}$$

kde $H(i)$ může být odhadované jako $ln(i) + Eulerovakonstanta$. Anomální trajektorie je tak označena pokud platí $E(n(t)) \to 0, s(t, N) \to 1$.

Výhoda *iBat* spočívá především v tom, že není nutné počítat žádné vzdálenosti, shluky atd., které jsou výpočetně náročnější. Dále bylo zjištěno že není nutné kalkulovat se všemi trajektoriemi procházejícími určenými body, ale stačí pouze menší podmnožina. Metoda *iBat* byla následně vylepšena na metodu *iBoat* popsanou v 4.1, která dokáže online detekci.

3.1.2 Metoda ROAM: Rule- and Motif-based Anomaly detection in Moving objects

Metoda představená jako ROAM [13], je založená na pravidlech a motivech trajektorií. Motiv zde představuje fragment trajektorie. Z těchto motivů jsou extrahovány vlastnosti, které poté klasifikujeme podle hierarchického klasifikátoru založeného na pravidlech. Motivy reprezentují různé vlastnosti trajektorií jako zatáčky, křížení, smyčky viz. obr. 3.4. Pro extrahování motivů je použito okno určité velikosti. Okno extrahuje vektor od svého prvního bodu do posledního. Tyto vektory jsou poté "poskládány"na sebe a pomocí shlukování zjištěny podobnosti, jež se stanou právě motivy. Využívá se eukleidovská vzdálenost. Po extrakci různých typů motivů prozkoumáme trajektorie znovu. Opět využijeme stejně velké okno w jako v předchozím případě a pokud $||w-m|| < e$, kde e je zadáno uživatelem, potom tuto část trajektorie lze reprezentovat motivem m. Pro každý motiv je získáno i několik potřebných atributů dle aplikace. Může jít třeba o průměrnou rychlost, dobu trvání atd. Tato část se v rámci ROAM nazývá extrakce motivů.



Obrázek 3.4: Ukázka možných motivů [13].

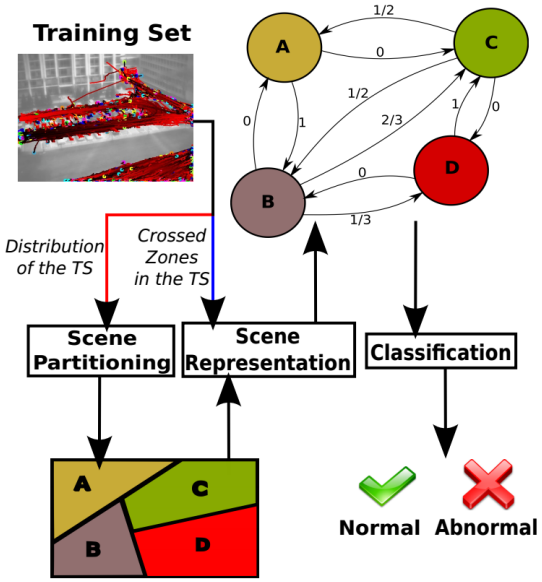
Aby klasifikátor zvládl pracovat, musíme mu dodat správná data. K tomu slouží generátor vlastností, který mapuje jednotlivé motivy s jejich atributy do jednotlivých vlastností. Tvůrci používají i generalizační metodu, která zmenšuje prostor. Nahrazují například rozsah časů jako pozdní odpoledne atd. Po generalizaci se pomocí shlukování vytvoří hierarchický strom, kde rodičovský uzel shlukuje data jeho potomků. Jako příklad můžeme uvést rychlost, kde uzel s rychlostí označenou jako pomalá rychlost, obsahuje dva potomky, kde první má rychlost 1-5 km/h a druhý 6-10 km/h.

Pro samotné hledání odlehlosti, tvůrci navrhují na pravidle založený klasifikátor CHIP *(Classification using Hierarchical Prediction Rules)*. Vyhledávání je iterativní a hladové, dokud nejsou všechny případy uspokojeny a není nalezeno co nejlepší pravidlo.

3.1.3 Metoda detekce anomálního chování pomocí učení bez učitele na grafech

Následující způsob detekce [1] je založený na ohodnoceném grafu a dělení prostoru na fixní počet zón, které se stávají uzly grafu viz. obr. 3.5. Mezi jednotlivými uzly grafu zjišťujeme pravděpodobnost přesunu z uzlu do uzlu. Po vygenerování grafu klasifikujeme trajektorie na normální a anomální.

Pro vytvoření ohodnoceného grafu nejprve potřebuje scénu rozdělit na zóny podle trénovací množiny. Nejlepší přesnosti by detekce dosáhla, pokud by co pixel, to jedna zóna. To ovšem není reálné možné, jelikož by to příliš zatížilo výpočetní prostředky a doba výpočtu by byla v praxi nepoužitelná. Také by v grafech byla přílišná variace a detekce by nemusela být příliš úspěšná. Proto navrhují vlastní metodu hierarchického dělení, kde se začíná od jedné zóny a postupně se dělí dle hustoty pohybu trajektorií.



Obrázek 3.5: Přehled rámce [1].

Když máme scénu rozdělenou, přistoupíme k vytvoření ohodnoceného orientovaného grafu. Graf $G = (V, E)$, kde $V = \{z_1, ..., z_L\}$ je množina vrcholů, kde vrchol z_i reprezentuje jednu zónu a každá hrana $e_{ij} \in E$ je propojení mezi zónami z_i a z_j. Pro všechny e_{ij} vypočítáme váhu $w(e_{ij})$ dle následujícího vzorce, kde $g(i, j)$ je počet trajektorií prochází zónami z_i a z_j a L je počet vrcholů.

$$w(e_{ij}) = \frac{g(i,j)}{\sum_{l=1}^{L} g(i,l)} \tag{3.3}$$

Klasifikace probíhá na základě pravděpodobnosti přechodu mezi zónami. Každou nově vznikající trajektorii namapujeme na sekvenci zón. Pro každou subtrajektorii t_j fixní velikosti H, která splňuje vlastnosti $H < |t|$, kde $t =< z_1, ..., z_{|t|} >$ a tedy $|t|$ je počet projetých zón, zjistíme její celkovou pravděpodobnost $P(t_j)$ *(cross-probability)*. Zvolení fixní velikosti H určuje kolik zón potřebujeme k určení anomálnosti subtrajektorie.

$$P(t_j) = w(e_{z_1;z_2}) * ... * w(e_{z_{H-1};z_H}) \tag{3.4}$$

Vypočítanou hodnotu musíme ještě ohodnotit, jestli jde o normální nebo anomální trajektorii. Metoda popisuje tři metody, jak toho dosáhnou. První je porovnání s uživatelem daným prahem, což je metoda velice jednoduchá a nezávislá na velikosti trajektorie. Dalším způsobem je *Posteriori Probability Ratio Test* a detekce s použitím *String Kernels*. Pro více detailu [1].

Výhodou je vysoká robustnost pro různá nastavení, a navíc nepotřebuje časově náročné učení, i když je nutné dělení na zóny. Navíc podporuje online zpracování, kdy potřebuje jenom část trajektorie ke klasifikování čili na objekt může upozornit dříve než opustí scénu. I když je metoda testována v [1] na trajektoriích z kamery, po zvážení by měla zvládnout i trajektorie aut ve městě.

3.1.4 Další metody

V [4] navrhují, jak detekovat odlehlé trajektorie mezi regiony, které nás zajímají. Detekcí těchto regionů se práce nezabývá. Díky hledání trajektorií pouze mezi zájmovými regiony značně zmenšíme prohledávaný prostor. Algoritmus následně detekuje odlehlé trajektorie jak prostorové, tak i časové. Je založený na počtu bodů v daném sousedství určité velikosti a času, kdy trajektorie opouští region zájmu.

Další možností je využití suppor vector machine (SMV, do češtiny překládáno jako metoda podpůrných vektorů). Kde trénujeme model SVM a ten je poté schopný klasifikovat odlehlé trajektorie, jelikož dělí prostor na dvě části, kde první jsou normální trajektorie a druhá jsou trajektorie odlehlé nebo jinak anomální.

Byla vytvořena práce na detekci podvodných taxíků [5]. Detekujeme zajímavá místa. Jde o místa, kde si lidé volají taxík a jejich cílové lokace. Trajektorie se mapují na symboly. Podle kódovací ceny symbolů detekujme důkaz trajektorie. Dalším důkazem je ujetá vzdálenost mezi body zájmu. Tyto důkazy jsou následně dále zpracovávány.

Metoda [15] využívá upravenou metodu minimální Hausdoffovy vzdálenosti (*Minimum Hausdoff Distance*). Detekce probíhá nad subtrajektoriemi určité velikosti dané uživatelem.

3.2 Anomálie provozu

Kromě odlehlých trajektorií můžeme v provozu detekovat jiné anomálie. Těmi můžou být náhlé změny hustoty a směru provozu. Čili např. dopravní nehody způsobující zácpy, různé události ve městě jako trhy apod. K detekci se přistupuje značně odlišně oproti odlehlým trajektoriím. Vždy musíme provést nějaké mapovaní, na segmenty silnice jako v kapitole 2.2.3 nebo na regiony ohraničené hlavními silnicemi. Tím získáme přehled, kolik aut či jiných objektů se v dané části města pohybovalo. Jelikož hlavní téma práce jsou odlehlé trajektorie, bude zde principiálně popsáno jenom několik metod. Další zajímavé metody lze nalézt v těchto pracích [23], [21], [14].

3.2.1 Metoda založená na odchylce s rozptylovou funkcí pro detekci dopravních anomálií

Rámec představený zde [9], využívá mapování trajektorií na segmenty silnic. Pro detekci anomálií využívá rozdílovou metodu, kde porovnává aktuální provoz vůči předpokládanému provozu založeném na minulých datech. Následně navrhuje difuzní metodu založenou na teplotním rozptylu, jelikož vytvořená anomálie ovlivňuje i ostatní segmenty a může zde vytvořit další anomálie. Díky této metodě lze identifikovat centrální anomálii. Pro samotné mapování je použita metoda pro mapování GPS trajektorií s nízkou vzorkovací frekvencí [17].

Pro každý segment silnice, kde segment je od křižovatky ke křižovatce, ukládáme dvojici (μ, σ), kde μ je předpokládaná hustota provozu a σ je směrodatná odchylka. Příklad je ukázán na obrázku 3.6, kde u druhé části vidíme, že z původní hodnoty 100 se hodnota zvýšila na 150, což překročilo odchylku a je pokládáno za anomálii. Tyto údaje jsou pro každý segment ukládány v 30 minutových časových slotech. Rámec samotný nedělí týden na jednotlivé dny, ale pouze na všední dny a víkendové dny. Tím dostává pouze $48 \times 2 = 96$ časových slotů. Díky tomuto lze lehce porovnávat rozdíly mezi časovými sloty a získat anomální segmenty. Přesný vzorec pro výpočet lze nalézt v uvedeném zdroji.

(a) 11:00am every Friday

(b) 11:00am June 1, 2012 (Friday)

Obrázek 3.6: Ukázka mapování (a), vznik anomálie (b) [9].

Velice zajímavé je ovšem řešení zjištění centra šíření anomálie pomocí teplotně difuzního modelu. Jednotlivé segmenty si mezi sebou vyměňují *teplotní energii*, která působí anomálie. Samozřejmostí je implementace procesu poklesu, který způsobují externí elementy. Tento proces simuluje "chladnutí"anomálie. Centrum anomálie je následovně zjišťováno pomocí rozdílu mezi okamžitou teplotou a jakou teplotu to mělo v předchozích stavech.

3.2.2 Metoda pro detekci davových dopravních anomálií založená na lidské pohyblivosti a sociálních mediích

Metoda [20] je založená na směrovém chovaní, např. plynulosti dopravy mezi dvěma body. Trajektorie jsou opět mapovány na jednotlivé segmenty silniční mapy. Pro každou dvojici počátečního a koncového bodu je vytvořen směrový vzor. Tento vzor obsahuje dvojice, kde první údaj je objem dopravy a druhý je procentuální zatížení cesty oproti ostatním možným cestám. Tyto dvojce jsou definovány pro každou cestu mezi počátečním a koncovým bodem.

Metoda je rozdělena na několik kroků. První je offline dolování, kde se vytváří výše zmíněné dvojice pro jednotlivé cesty. Vytváří se také dva různé indexy. Jeden offline index, jenž je obousměrná struktura mezi segmenty a trajektorií. Druhý je online index, kde pro každý segment silnice indexujeme všechny cesty, které v segmentu končí.

Druhý krok je samotná detekce anomálií. Anomálie vytváří subgraf podle šíření do okolí. Anomální segment je vybrán podle skutečnosti, jak moc se liší od minulých stavů.

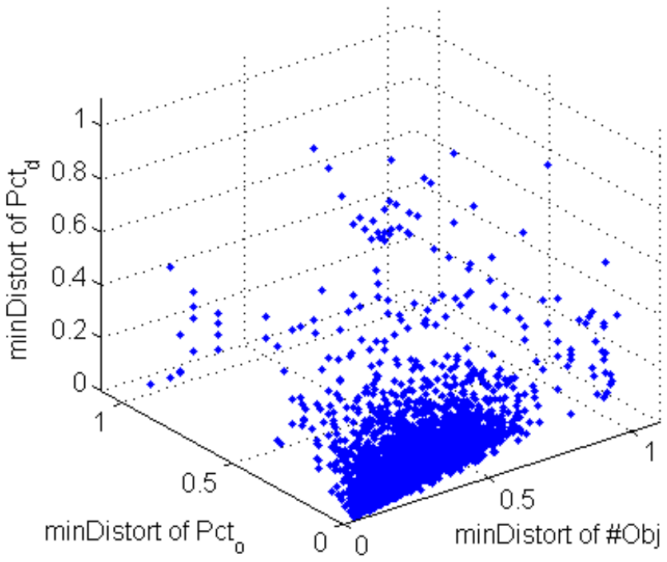
Posledním krokem je analýza anomálie. Zde se počítá, jaký dopad měla anomálie na dopravu, a to v podobě zpoždění. Metoda také navrhuje možnost propojení sociálních sítí s detekcí anomálií. Dnes totiž lze na sociální sítě umisťovat příspěvky, kde je nejen přesný čas vložení příspěvku, ale i pozice, kde jsme příspěvek vložili.

3.2.3 Metoda pro objevení běžné interakce v proudech dopravních dat

Další z možných metod [16] řešící detekci anomálního provozu je založena na detekci extrémních bodů, pomocí Mahalanobisovy vzdálenosti ve 3D prostoru.

Metoda používá mapování na regiony, jež jsou ohraničeny hlavními silnicemi. Mezi těmito regiony je vybudován orientovaný graf provozu. Propojení se vytváří mezi počátečním a koncovým regionem. Propojení obsahuje tři atributy. První atribut je množství všech aut, pohybující se z počátečního do koncového regionu. Druhý je celkový počet aut, odjíždějící z počátečního regionu a třetí parametr obsahuje celkový počet aut, přijíždějící do koncového regionu. Propojení je navíc rozděleno pomocí časových slotů určeného rozsahu.

Pro detekci je použita metoda minimálního zkreslení *(minimum distortion)*. Metodu zavoláme nad každým atributem vektoru a porovnáváme s předchozími časovými sloty, např. pátek v 10:00 s minulými pátky ve stejný čas. Vytvořený 3D prostor je zobrazený na následujícím obrázku 3.7.



Obrázek 3.7: Ukázka 3D prostoru [16].

Pomocí Mahalanobisovy vzdálenosti detekuje odlehlé body. Značná výhoda této detekce je, že nezáleží pouze na jednom parametru, ale odlehlost může být dána např. pouze zvýšením odjezdů aut z regionu. Buduje se také strom anomálií, kde dle času výskytu anomálie se rozhoduje, co je čí otcovský uzel. Otcovský uzel se synem také musí být v prostorovém sousedství. Díky stromu dokážeme zjistit počátek anomálie a její postupné rozšíření.

Kapitola 4

Vybrané metody

Aby bylo možné porovnávat výsledky metod, byly zvoleny následující metody: *iBoat*, *TRAOD* a *TOP-EYE*. Všechny detekují nejen odlehlé trajektorie, ale i která část či části způsobují anomálnost. Tato schopnost dává možnosti přesné detekce oblasti, kde nastává problém tím, že se zde bude vyskytovat více anomálií v krátkém času. Dále všechny pracují s určitým typem souřadnicového systému. Neměl by tedy být problém je použít na trajektoriích taxíků, kde body jsou dány GPS souřadnicí.

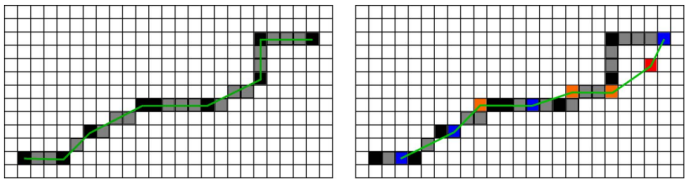
4.1 Metoda iBoat

Chen a ostatní [2] navrhují rámec založený na izolování odlehlých trajektorií *(isolation-based online anomalous trajectory)* neboli *iBoat*. Jde o následovníka, na podobném principu založené metodě *iBat* popsané v sekci 3.1.1, který je značně vylepšen především o online zpracování.

4.1.1 Problém mapování a anomálnost trajektorie

Opět se zde využívá mapovaní na mřížku stejně velikých buněk, jako právě v případě *iBat* popsané v sekci 3.1.1.

Vyskytuje se ovšem problém s nepřesným mapováním, kde i auto, které projede stejnou cestu, může mít výslednou mapovanou cestu jinou. Jde o chybu, že měření GPS souřadnic není přesné a i čas snímání pozice je v mnoha případech odlišný. Navíc mapovaní není příliš přesné díky velikosti buněk a přidávání bodů kvůli dosaženi spojitosti mapované trajektorie. Proto cestu prohlašujeme za stejnou i když je pouze v sousedství buněk. Příklad je ukázán na následujícím obrázku 4.1, kde vlevo je jedna z namapovaných trajektorií. Černé buňky znamenají mapované GPS body a šedé jsou buňky přidané, které jsou doplněné podle přímky vedené mezi body. Vpravo je druhá trajektorie, jejíž GPS body jsou modré, oranžové a červené. Modré buňky jsou stejné jako v předchozí trajektorii a oranžové buňky označují ty které jsou v sousedství. U těch ještě počítáme, že tato část trajektorie je stejná. Kdežto červená nemá žádnou buňku z předchozí trajektorie. Tuto buňku nemůžeme vůči trajektorii vlevo prohlásit za blízkou.



Obrázek 4.1: Vlevo: mapování trajektorií na buňky mřížky. Vpravo: Porovnání dvou trajektorií [2].

Pro následují vysvětlení si definujeme tyto množiny. T je množina všech mapovaných trajektorií a G je množina všech buněk v mřížce rozdělení.

Aby se rámec vypořádal s problémem nepřesnosti a malé frekvence získávání GPS souřadnice, zavedl funkci $pos : \mathbb{T} \times G \rightarrow \mathbb{N}^+$, která zjistí prvně se vyskytující pozici buňky v trajektorii a $N(g) \in t$, která vrací první buňku z t, která sousedí s buňkou g.

$$pos(t, g) = \begin{cases} argmin_{i \in \mathbb{N}^+}\{t_i = g\} & \text{když } g \in t \\ \infty & \text{jinak} \end{cases} \tag{4.1}$$

Tedy $pos(t, N(g))$ vrátí první index buňky z t sousedící s g. Tím dosáhneme, že trajektorie budou brány jako stejné, i když jsou oranžové body mimo původní jak je ukázáno na obr. 4.1. Pokud tedy jako příklad budeme mít tyto hodnoty $t_1 = \langle g_1, g_2, g_5, g_6, g_8 \rangle$, tak $pos(t_1, g_5) = 3$, a podobně pokud máme druhou trajektorii $t_2 = \langle g_1, g_2, g_4, g_6, g_8 \rangle$, tak $pos(t_1, N(g_4)) = 3$, jelikož nejbližší soused g_4 je v t_1 bod g_5.

Tyto funkce jsou následně využity ve funkci $hasPath(T, t)$, která vrací všechny další trajektorie z T, které obsahují všechny body z t a zároveň tyto body musí být ve stejném pořadí, ale mohou mezi nimi být i jiné body.

$$hasPath(T, t) = \left\{ t' \in T \;\middle|\; \begin{array}{l} (i) \forall 1 \leq i \leq n . N(g_i) \in t' \\ (ii) \forall 1 \leq i \leq j \leq n . \\ pos(t', N(g_i)) < pos(t', N(g_j)) \end{array} \right\}. \tag{4.2}$$

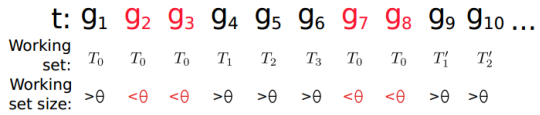
Následně je definována $\theta -$ anomálnost trajektorie t pro daný práh $0 \leq \theta \leq 1$ s ohledem na sadu trajektorií T pokud splňuje následující podmínku.

$$support(T, t) = \frac{|hasPath(T, t)|}{|T|} < \theta \tag{4.3}$$

4.1.2 Algoritmus

Když jsou definovány všechny potřebné předpoklady, přejdeme k samotnému algoritmu. Data jsou opět ukládána stejně jako v předchozí metodě *iBat* popsané v sekci 3.1.1, pomocí invertovaného indexu. Základní myšlenka detekce anomální trajektorie je potom následující. Udržuje se adaptivní pracovní okno. Když přijde nový bod, je přidán do okna a je vypočtena nová množina trajektorií pomocí *hasPath*. Následně pokud výpočet anomálnosti pomocí funkce *support* je nad daným prahem θ, se pracovní množina trajektorií T změní na množinu, kterou vrací funkce *hasPath* a s oknem bodů se nic nedělá. Jestliže hodnota klesne pod práh θ, okno zredukujeme pouze na poslední přijatý bod a pracovní množina T se resetuje na původní hodnotu, tedy všechny trajektorie. Na následujícím obrázku 4.2 je vidět testování anomálnosti bodů g a změny pracovní množiny T. Při prvním bodu testovací množina obsahuje všechny trajektorie. Jelikož, ale bod $g1$ není zhodnocen jako anomální, je pracovní

množina zmenšena na množinu, kterou vrací funkce $hasPath$. Další bod $g2$ je detekován jako odlehlý, a proto je tak označen a pracovní množina T je resetována na všechny trajektorie. Tímto postupem odhalíme odlehlé body, které jsou na obrázku označeny červeným písmem.



Obrázek 4.2: Ukázka práce adaptivního okna a změn pracovních množin [2].

Pro každou trajektorii uchováváme její skóre anomálnosti. Jde o výpočet na základě délky a hustoty anomálních bodů, jež se vyskytují v trajektorii. Následuje vzorec pro výpočet, pokud je trajektorie kompletní, kde λ je teplotní konstanta vyvažující funkci $support$, která pro hodnoty nad daný práh θ tyto hodnoty snižuje blíže k 0. Díky vyvažující funkci mají více anomální body větší skóre. Funkce $dist(p_i, p_{i-1}$ značí vzdálenost mezi předchozím bodem a bodem anomálním.

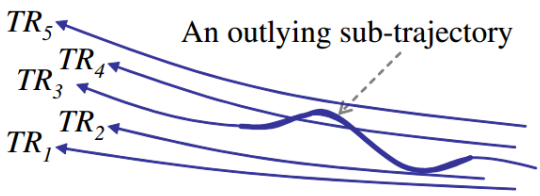
$$score = score(n) = \sum_{i=2}^{n} \frac{1}{1 + e^{\lambda(support(i)-\theta)}} dist(p_i, P_{i-1}) \tag{4.4}$$

Představený rámec je velice výkonný v online zpracování trajektorií. Zásadní výhoda oproti většině ostatních metod je že není nutné počítat vzdálenosti a hustoty trajektorií, jako u následující metody TRAOD, které jsou velice výpočetně náročné. Uplatnit rámec lze v několika aplikacích jako detekce podvodných taxíků nebo zjištění změny mapy cest měst.

4.2 Metoda TRAOD

Lee a ostatní [10], vyvinuli způsob detekce odlehlých trejektorií postavený na dělení a detekci (partition-and-detect framework TRAOD). Jak vyplývá z názvu, algoritmus obsahuje dvě části. První část je dělení, které segmentuje trajektorie na segmenty. Tyto segmenty v druhé části podrobíme analýze, jestli jsou anomální, čili jde o část detekční. Detekce je složení dvou klasických přístupů a to založených na hustotě a vzdálenosti segmentů a trajektorií, jejíchž jsou součástí.

Když porovnáváme trajektorii jako celek, nemusí být detekován jako anomální. Proto se porovnávají segmenty trajektorie a tím dosáhneme detekce i anomálních částí. Ukázka na obrázku 4.3 zobrazuje tuto situaci. Pokud by byly trajektorie mnohem delší, než je ukázáno, anomálie by při porovnání celé cesty zanikla.



Obrázek 4.3: Možná detekce anomální subtrajektorie [10].

4.2.1 Detekce odlehlých trajektorií

Odlehlá trajektorie nebo její segment je definována převážně podle vzdáleností. Přesněji se jedná o identifikaci odlehlého segmentu, na základě počtu dalších segmentů do určité vzdálenosti. Před vysvětlením samotného algoritmu si musíme vypsat nejdůležitější definice uvedené v [10] (Jsou vypsány jenom nejdůležitější.). Než začneme s definicemi je nutné si definovat uživatelem zadávané parametry, které se budou dále objevovat. Prvním parametr D určuje, do jaké vzdálenosti mezi segmenty jsou segmenty považovány za blízké. Parametr F určuje kolik trajektorie musí obsahovat anomálních segmentů, aby sama byla anomální. Posledním parametrem je p, které určuje kolik blízkých trajektorií je potřeba, aby segment nebyl určen jako anomální. Definice začneme definováním vzdálenosti, na jejímž výpočtu je celá metoda založená.

Jako první definujeme, jaké segmenty můžeme považovat za blízké vůči jiné trajektorii.

Definice 4.1. *Trajektorie TR_i je **blízká** segmentu $L_j \in P(TR_j)$ pokud $(TR_i \neq TR_j)$ a $\sum_{L_i \in CP(TR_i, L_j, D)} len(L_i) \geq len(L_j)$, parametr D je zvolen uživatelem. Kde $P(TR_j)$ je množina všech segmentů trajektorie TR_j. $CP(TR_i, L_j, D)$ je množina všech segmentů z TR_i vzdálená méně než D od segmentu L_j.*

Následně definujeme funkce pro detekci odlehlého segmentu a trajektorie.

Definice 4.2. *Segment L_i je shledán **odlehlým** pokud následující rovnice je pravdivá. $\mathcal{I}$ je počet všech trajektorií, p je parametr dán uživatelem a funkce $CTR(L_i, D)$ je množina blízkých trajektorií k trajektorii L_i podle parametru vzdálenosti D.*

$$|CTR(L_i, D)| \leq \lceil (1-p)|\mathcal{I}| \rceil \tag{4.5}$$

Definice 4.3. *Trajektorie TR_i je shledána **odlehlou** pokud následující rovnice je pravdivá. Kde $P(TR_i)$ je množina všech segmentů trajektorie, $OP(TR_i, D, p)$ je množina všech odlehlých segmentů a F je parametr dán uživatelem.*

$$Ofrac(TR_i) = \frac{\sum_{L_i \in OP(TR_i, D, p)} len(L_i)}{\sum_{L_i \in P(TR_i)} len(M_i)} \geq F \tag{4.6}$$

Abychom předešli problémům, které způsobují regiony s odlišně hustou dopravou a tím pádem různě početné trajektorie v dané oblasti, musíme na tento problém brát ohled a zakomponovat ho do řešení, jelikož předchozí definice neberou problém hustoty v potaz.

Jako první definujeme hustotu pro segment $density(L_i)$ a následně korekční koeficient $adj(L_i)$, kterým násobíme hodnotu $|CTR(L_i, D)|$ a tím dosáhneme korekce v hustých nebo naopak řídkých oblastech.

Definice 4.4. ***Hustota** segmentu L_i je dána počtem segmentů ve vzdálenosti σ od segmentu L_i, kde σ je směrodatná odchylka vzdálenosti mezi dvěma segmenty, vypočítána pro všechny dvojce segmentů mezi všemi trajektoriemi.*

$$densiny(L_i) = |\bigcup_{TR_j \in \mathcal{I}} CP(TR_j, L_i, \sigma)| \tag{4.7}$$

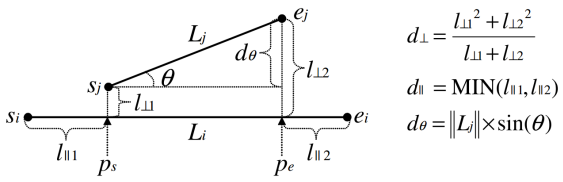
Definice 4.5. ***Korekční koeficient** segmentu L_i je poměr průměrné hustoty a hustoty segmentu L_i*

$$\begin{aligned} adj(L_i) &= \frac{\sum_{L_j \in \mathcal{L}} density(L_i)/|\mathcal{L}|}{density(L_i)} \\ kde\ \mathcal{L} &= \bigcup_{TR_k \in \mathcal{I}} P(TR_k) \end{aligned} \tag{4.8}$$

Tímto jsme zajistili, že hustota nebude mít na výsledky vliv. Navíc zajišťuje, že uživatelem definované konstanty F, D, p, nemají na hustotu vliv. Nyní stačí definovat pouze výpočet vzdálenosti mezi dvěma segmenty.

Výpočet je zobrazen na obrázku 4.4. Skládá se ze tří částí. Kolmá vzdálenost $d_\perp$ *(perpendicular distance)*, Paralelní vzdálenost $d_\parallel$ *(parallel distance)* a úhlová vzdálenost d_θ *(angle distance)*. Díky tomu že bereme různé typy vzdáleností v potaz, dosahujeme lepších výsledků a můžeme následovně rozlišovat mezi odlehlým segmentem pozičním nebo úhlovým. Tuto schopnost nám dávají váhy, které jsou ve finálním vzorci pro výpočet vzdálenosti. Nastavujeme je podle určení aplikace.

$$dist(L_i, L_j) = w_\perp \cdot d_\perp + w_\parallel \cdot d_\parallel + w_\theta \cdot d_\theta \tag{4.9}$$

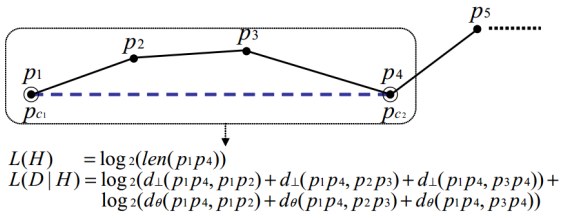


Obrázek 4.4: Výpočet vzdálenosti mezi segmenty L_i a L_j [10].

4.2.2 Segmentace trajektorie

Segmentace trajektorie je potřeba, aby optimalizovaná verze metody pracovala správně a efektivně. Pro verzi popsanou v [10] je potřeba použít jejich segmentační algoritmus, jelikož později k odhalení, jestli při následné detekci použít jemné segmenty, využívá určité poznatky, které se získají právě při segmentaci. Po patřičné úpravě algoritmu by však mělo jít použít jakýkoliv segmentační algoritmus vhodný pro trajektorie. Navíc mohou postačovat pouze body trajektorie, které ji přirozeně segmentují, jak používá základní verze metody. Může jich být ovšem mnoho nebo naopak málo, a tím se detekce zpomalí nebo se stane nepřesnou.

Tvůrci sami navrhli dvoufázový segmentační algoritmus, jenž je součástí jejich jiné práce [12], který urychlí detekci, a použili jej ve vylepšené verzi algoritmu. První, hrubá fáze, je založena na minimální popisné délce (MDL) *minimum description length*. Díky ní z několika jemných segmentů vytvoříme jeden segment hrubý. Při jeho tvorbě využíváme některé z částí pro výpočet vzdálenosti definovanou dříve. Jde o úhlovou a kolmou vzdálenost. Při výpočtu, jestli do hrubého segmentu vložit další jemný segment, se vypočítá MDL mezi hrubým segmentem a všemi již přidanými segmenty a novým segmentem. Příklad je zobrazen na obrázku 4.5.



Obrázek 4.5: Ukázka hrubého segmentu a výpočet MDL ceny [12].

Jelikož by výpočet segmentů dle ideální MDL ceny byl výpočetně zdlouhavý, tvůrci používají aproximační metodu. Ten vypočte dvě MDL ceny. Jednu normální (MDL_{par}), jak je uvedeno výše a druhou (MDL_{nopar}) takovou, u které neuvažujeme ostatní body, ale pouze počáteční a koncový bod segmentu. Pokud $MDL_{par} <= MDL_{nopar}$, tak jemný segment přidáme do hrubého a přidáváme segmenty dál dokud platí předchozí podmínka.

Při tvoření hrubých segmentů si o nich ukládáme určité informace, které se využívají v části detekce odlehlých segmentů. Jde o maximální a minimální délku jemného segmentu, maximální kolmou a úhlovou vzdálenosti mezi hrubým segmentem a jeho jemnými segmenty. Tyto informace jsou využity k výpočtu vrchní a dolní meze. Tyto meze následně rozhodují, jestli použít všechny jemné segmenty, žádné nebo je podrobit výpočtu na vzdálenost.

4.2.3 Algoritmus

Pro funkčnost základní verze algoritmu stačí předchozí definice bez segmentace, neboli segmenty definují základní naměřené body trajektorie. Pro každý takový segment vypočítáme všechny blízké trajektorie dle definice 4.1, což vzhledem k nutnosti výpočtu směrodatné odchylky a počtu jemných segmentů, je výpočetně zdlouhavé. Pro základní verzi následně zjistíme, jestli je segment odlehlý podle definice 4.2, kde $CTR(L_i, D)$ upravíme pomocí korekční konstanty $adj(L_i)$. Konstantou zajistíme, že hustota trajektorií neovlivňuje detekci, jak bylo popsáno definicí 4.5. Finálně pro každou trajektorii zjistíme její anomálnost dle definice 4.3.

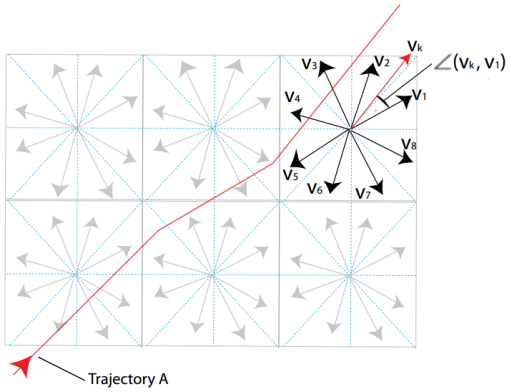
U optimalizované verze metody je nutná segmentace, jak ji popsali autoři a jak je popsána v předchozí sekci 4.2.2. Následně vypočítáme vrchní a dolní meze mezi hrubými segmenty. Dle těchto mezí rozhodujeme, jestli jsou segmenty blízko sebe dle vzdálenostního parametru D. Jestliže splňují podmínky, tak jsou jejich jemné segmenty vloženy do množiny, nad kterou se provede detekce jako u základní verze metody.

Výhody algoritmu spočívají v detekci odlehlých částí trajektorie čili segmentů. Dále, že trajektorie nemusíme nijak mapovat např. na segmenty silnic, které by však šly využít jako segmenty ve výpočtu. Další výhodou je možnost použití různých segmentačních algoritmů. I když to v práci není řečeno, myslím si, že by metoda šla použít i jako online. Pro online detekci musíme předpokládat okamžitý výpočet, jestli je segment odlehlý ihned po příchodu nové lokace, která je základní jednotkou segmentu.

Značná nevýhoda algoritmu je celková rychlost výpočtu, kdy časová složitost je $O(n_t^2)$, kde n_t je počet segmentů. Pokud tedy máme město s tisící, ne-li milióny trajektoriemi, není to příliš dobrá volba. Vylepšená verze ovšem přináší celkem značné zrychlení. Další nevýhodou je velké množství uživatelem zadávaných parametrů.

4.3 Metoda TOP-EYE

Vyvíjející se detekce odlehlých trajektorií jménem TOP-EYE [6] je další z možných detekcí anomálních trajektorií. Zaměřuje se na co možná nejvčasnější detekci odlehlosti právě vznikající trajektorie. Jde tedy především o zpracování v reálném čase. Metoda postupně akumuluje anomálnost trajektorie vůči ostatním trajektoriím nasbíraných v minulosti. Bere v potaz současnou, ale i minulou anomálnost trajektorie, a kvůli tomu zavádí exponenciální pokles (exponential decay), pro postupné snížení dopadu anomálního pohybu v předchozích stavech. Odlehlost trajektorie může být definována na základě hustoty a směru.



Obrázek 4.6: Ukázka rozdělení dle směru [6].

4.3.1 Kalkulace skóre

Metoda rozděluje prostor do mřížky malých buněk. Nad tímto rozdělením vytváří pravděpodobností model na základě směru trajektorie. Buňku rozdělí na osm sektorů dle rozsahu $\pi/4$, a vypočítá, na základě historických dat, pravděpodobnost pohybu v daném směru. Každá buňka je tedy definována jako vektor, kde p_i je frekvence pohybu ve směru sektoru.

$$g = (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) \tag{4.10}$$

Jednotlivým buňkám přidáme dále parametr hustoty provozu dle počtu trajektorií, které buňkou procházejí.

Pro hledání anomálií dle směru pohybu je navrhnuta následující funkce pro výpočet anomálního skóre. Mějme $\mathcal{K}$ směrů v jedné buňce pro novou trajektorii. Díky tomu můžeme skóre vypočítat následovně.

$$OScoreDir = 1 - \sum_{k=1}^{\mathcal{K}} q_k \sum_{i=1}^{8} p_i \cdot \cos \angle(v_k, v_i) \tag{4.11}$$

Kde $\cos(v_k, v_i)$ je úhel mezi směry v_k a v_i, jak je zobrazeno na obrázku 4.6 a q_k je $1/\mathcal{K}$.

Skóre dle hustoty provozu je velice jednoduché, jelikož je založeno pouze na zvoleném prahu a hodnotě, kterou chceme anomálii přiřadit.

$$OScoreDen = \begin{cases} s & \text{když hustota} < \tau \\ 0 & \text{jinak} \end{cases} \tag{4.12}$$

4.3.2 Detekční metoda

TOP-EYE představuje metodu, která kumuluje skóre anomálnosti v reálném čase tvořené trajektorie. Řeší se problémy kombinace aktuálního skóre odlehlosti a skóre, jež bylo dosaženo v minulosti. Nejspíše bylo přistoupeno k tomuto řešení, jelikož trajektorie mohla být v minulosti odlehlá, ale její nejaktuálnější trasa již není, a proto by neměla být takto detekována neboli neustále spouštět upozornění. Z požadované funkčnosti můžeme odvodit dva problémy. První je vytvoření funkce exponenciálního poklesu pro historickou odlehlost. Druhá je kumulace skóre anomálnosti.

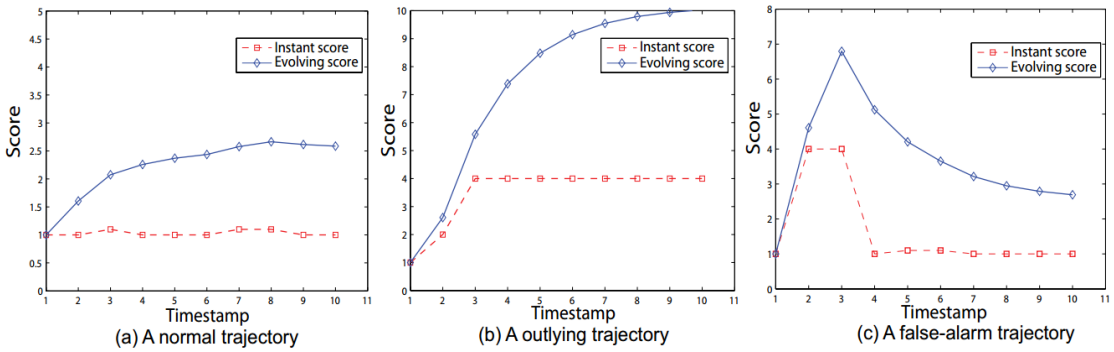
Pro řešení prvního problému je využita exponenciální funkce $exp(-\lambda\delta t)$. Kde λ je stupeň poklesu definován uživatelem a δt je časový rozdíl mezi aktuálním měřením a minulým časem, kdy byla vypočítáváno anomální skóre. Tato exponenciální funkce snižuje dopady předchozích odlehlých bodů. Pokud tedy trajektorie projde buňkou mřížky, vypočítáme skóre dle pravděpodobnosti směru nebo hustoty provozu pro aktuální čas t_0. Trajektorie dále pokračuje a chceme zjistit akumulované skóre v další buňce v čase t_1. Tohoto skóre dosáhneme následovně.

$$S_{t_1}^{\sum} = S_{t_1} + S_{t_0} * exp(-\lambda\delta t_0) \tag{4.13}$$

Pro výpočet skóre v jakémkoliv čase použijeme následující vzorec.

$$S_{t_i}^{\sum} = S_{t_i} + S_{t_{i-1}} * exp(-\lambda\delta t_{i-1}) + S_{t_{i-2}} * exp(-\lambda\delta t_{i-2}) + ... + +S_{t_0} * exp(-\lambda\delta t_0) \tag{4.14}$$

Díky kumulativnímu počítání a poklesové funkci se skóre vyvíjí s časem. Pokud přesáhne určitý práh daný uživatelem, odlehlá trajektorie je detekována velice brzo. Na následujícím obrázku 4.7 vidíme postupnou změnu dle anomálnosti v čase. Lze vidět, že pokud aktuální skóre přetrvává na stejné úrovni, tak celkové skóre s postupem času také dále neroste. Díky tomu můžeme detekovat různě úrovně anomálnosti.



Obrázek 4.7: Změna skóre odlehlosti v čase [6].

Jasná výhoda metody spočívá v reálném čase zpracovávaných trajektorií a brzké detekce anomálních trajektorií. Je celkově také dost odolná vůči šumu.

Tvůrci algoritmus testovali na trajektoriích z kamery, kde si husté rozdělení dle mříže mohou dovolit, jelikož i tak buněk nebude příliš mnoho. Navíc z kamery dostáváme spojitý pohyb objektu, narozdíl od GPS souřadnic. Bude tedy zajímavé sledovat výkonnost metody na datech GPS, na kterých budou provedeny testy, jelikož pokud zvolíme hustou síť nad celým městem, bude detekce odlehlých trajektorií trvat déle díky mnohem větší množině stavů.

Kapitola 5

Implementace

V této kapitole bude popsána implementace aplikace, která obsahuje dvě metody iBoat a TRAOD. Jako první se podíváme, proč a jaký jazyk byl nakonec zvolen. Následně pro obě metody popíšeme jejich vnitřní reprezentace dat a implementaci algoritmu. U metody iBoat jde o verzi s adaptivním oknem a u metody TRAOD se jedná o implementaci optimalizované metody s jimi navrženou metodou segmentace. Nakonec si popíšeme jejich spojení a vytvoření uživatelského prostředí pro zobrazení výsledků pomocí grafů.

Implementace metody TOP-EYE byla převzata z diplomové práce [22]. Do aplikace bylo přidáno načítání dat implementací rozhraní `MovingObjectsData`. Poté byla provedena menší úprava grafického rozhraní pro výběr dat, aby bylo možno specifikovat cestu k souboru s daty trajektorií.

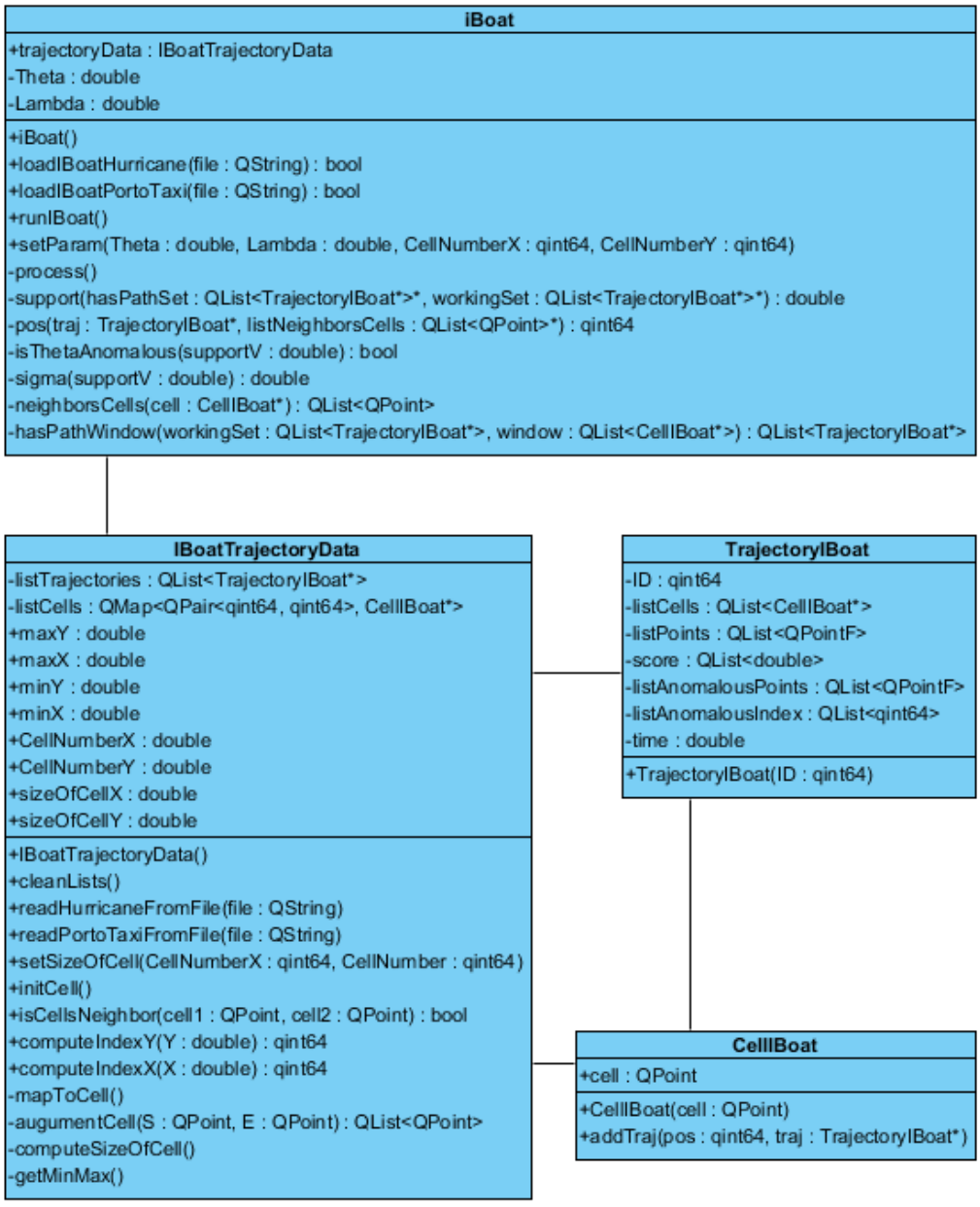
Implementace je vytvořena způsobem, aby bylo možné jednoduše přidávat další zdroje trajektorií. I přesto musí člověk zasáhnout do kódu nebo alespoň trajektorie přeformátovat do již podporovaného formátu. Každá metoda požaduje na vstupu různou strukturu, ale do těchto struktur stačí pouze načíst pro každou trajektorii její poziční body do struktury seznamu daného objektu trajektorie. O vytváření segmentů, či mapování na buňky sítě, se již daná metoda postará sama. Dále je nutné přidat tlačítko do rozhraní pro spuštění metody a předání uživatelem zvolených parametrů, případně volat funkce pro vykreslení grafů. Vše je možno jednoduše okopírovat z již existujícího řešení.

5.1 Výběr jazyka

Jako první byl vybrán jazyk Python pro jeho podporu v rámci dolování dat. Byly v něm napsány první prototypy a předzpracování dat. Jak se ovšem u prototypů ukázalo, implementace v Pythonu, i přes rychlé naprogramování prototypů, nebyla příliš dobrá volba, a to díky jeho rychlosti. Hlavně metoda TRAOD byla značně pomalá, především kvůli výpočtu směrodatné odchylky a hustoty segmentů. Bylo totiž nutné provést kalkulace vzdáleností každého segmentu s každým a tato operace si vyžádala příliš výpočetního času, i při aproximovaném výpočtu směrodatné odchylky ze zlomku náhodně vybraných dat. U hurikánů šlo o několik hodin a u taxíků, kde se v nejmenší skupině nacházelo 700 trajektorií, šlo již o několik málo desítek hodin. Jelikož však nebyly použity žádné speciální dolovací knihovny, byly metody přeprogramovány do C++, konkrétně Qt rámce verze 5, aby bylo možno jednoduše vytvořit malé uživatelské prostředí a aplikace zůstala multiplatformní. Implementace v C++ přinesla zrychlení až o dva řády. To vyplývá i z různých testů, kde Python je většinou 100-500 pomalejší než kompilované C++. Předzpracování dat bylo ponecháno v Pythonu.

5.2 Implementace metody iBoat

V této sekci bude popsána implementace metody iBoat a to v následujícím pořadí. Jako první je popsán způsob ukládání dat do struktur vytvořených pro řešení algoritmu. Poté je uvedeno, jak se trajektorie načítají a mapují na mříž dělící prostor do buněk. Poslední podsekce je zaměřená na jádro metody a to detekci odlehlých bodů a trajektorií pomocí verze s adaptivním oknem. Na obrázku 5.1 je zobrazen diagram tříd pro část metody iBoat.



Obrázek 5.1: Diagram tříd pro část řešící metodu iBoat

5.2.1 Datová struktura

Hlavní třída, která zapouzdřuje data je `IBoatTrajectoryData`. Operace v této třídě jsou popsány v další sekci 5.2.2. V této třídě definujeme strukturu seznamu `QList` jménem `listTrajectories`, která jako prvky seznamu obsahuje ukazatele na objekty trajektorií třídy `TrajectoryIBoat`. Prostor dělený mříží je reprezentován pomocí `listCells`, což je Qt slovníková struktura `QMap`. Jako klíč je použita struktura dvojce celočíselných hodnot `QPair`, aby bylo možné buňky vyhledávat pomocí souřadnic X a Y. Původní návrh byl použít strukturu bodu `QPoint` jako klíč. Ta bohužel není povolena, jelikož neumí některé z nutných operaci jako třeba porovnání menší, větší. Hodnota ve slovníku je ukazatel na objekt buňky.

Pro uložení trajektorií v rámci iBoat byla vytvořena třída `TrajectoryIBoat` a pro reprezentaci buněk sítě třída `CellIBoat`. Každý objekt trajektorie obsahuje jedinečnou identifikaci `ID`, která byla přidána pouze pro pozdější identifikaci trajektorie v souboru a zobrazení jejich odlehlých bodů v grafu. Nejdůležitějším prvkem třídy trajektorií je seznam bodů `listPoints`, kde bod je reprezentován strukturou `QPointF`. Pro uložení odlehlých bodů je zde seznam `listAnomalousPoints` a také seznam `listAnomalousIndex`, kde jsou uloženy pozice do seznamu bodů trajektorie. Seznam s indexy anomálních bodů byl zvolen kvůli pozdějšímu vykreslování odlehlých trajektorií v grafickém rozhraní, abychom věděli, které odlehlé body spojit. Skóre anomálnosti trajektorie, které s každým bodem roste, je uloženo v seznamu `score`. Máme tedy k dispozici skóre pro každý bod trajektorie. Poslední položkou je seznam buněk `listCells`, na které je trajektorie namapována. Seznam obsahuje ukazatele na buňky, aby se zbytečně nezabíralo místo kopírováním hodnoty a měli jsme k dispozici vždy aktuální obsah buňky.

Buňky jsou reprezentovány objekty třídy CellIBoat. Ta obsahuje souřadnice definované jako bod `QPoint`. Dále definuje `listTrajectory` jako slovníkovou strukturu `QMap`. Slovník ukládá jako hodnotu seznam ukazatelů na trajektorii, kde klíčem je pozice buňky v trajektorii. Díky tomu, že máme uloženy jak buňky v objektech trajektorie, tak i trajektorie v objektech buněk, jsme vytvořili obousměrnou indexovou strukturu, v metodě nazvanou jako invertovaný index. Pomocí tohoto indexu můžeme rychle vracet všechny trajektorie v rámci buňky. To se hodí, pokud hledáme trajektorie ve velké sadě dat. Jelikož pro testování z důvodů šetření paměti a rychlosti výpočtu, byly použity pouze konkrétní zdroj a cíl trajektorie, není invertovaný index nijak použit.

5.2.2 Načítání a mapovaní trajektorií

Všechny zmíněné metody v této části patří do třídy `IBoatTrajectoryData`, pokud nebude řečeno jinak. Pro načtení jednotlivých trajektorií slouží metody `readHurricaneFromFile` a `readPortoTaxiFromFile`. Obě metody jako jediný parametr potřebují cestu k souboru řetězec typu `QString`. V těchto metodách se zpracovávají vstupní data a pro každou trajektorii je vytvořen objekt třídy `TrajectoryIBoat`, kde po načtení bodů do seznamu je vložen do seznamu všech trajektorií. Navíc je trajektorie vložena do buňky metodou objektu buňky `addTraj`. První bod na pozici 0 předáme prvním parametrem. Druhým parametrem je ukazatel na objekt trajektorie. Dále pokračujeme inicializováním mříže pomocí metody `initCell`, která dle zadané velikosti mříže, naplní slovník objekty buněk. Na konci inicializace funkce zavolá další metodu `mapToCell`, která se stará o namapování GPS bodů trajektorie na síť buněk.

Před samotným mapováním je nutné zjištění maximální a minimální hodnoty souřadnic X a Y. O tento problém se stará metoda `getMinMax`. Po zjištění těchto maximální a minimál-

ních hodnot následně vypočítáme velikost jednotlivých buněk. Toho dosáhneme v metodě `computeSizeOfCell`, v níž provedeme tento jednoduchý výpočet, kde $CellNumberY$ je počet buněk ve směru osy Y a $maxY$ a $minY$ je maximálním respektive minimální hodnota Y souřadnice. Funkce `qFabs` je funkce rámce Qt pro výpočet absolutní hodnoty. Analogicky pro rovinu X.

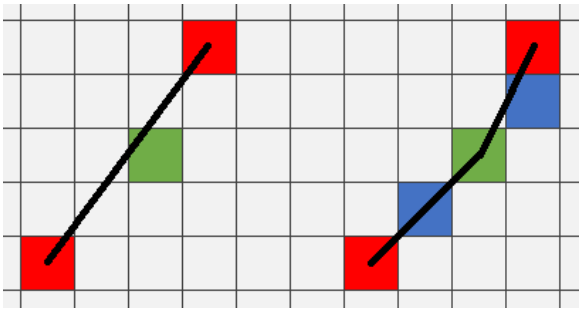
$$sizeOfCellY = qFabs(maxY - minY)/CellNumberY \tag{5.1}$$

Nyní je vše připravené pro zavolání metody `mapToCell`. V ní pro každou trajektorii provedeme mapování bodů na buňky a případné vytvoření přidaných buněk, abychom zajistili celistvost trajektorie. Pro první bod zjistíme indexy buňky, na kterou bude mapován. Index je reprezentován bodovou strukturou `QPoint`. Toho dosáhneme pomocí metod `computeIndexX` a `computeIndexY`, které přijímají jako parametr X a Y souřadnici bodu a vrací index v požadovaném směru osy. Pro výpočet je použit tento kód, kde `qRound64` je funkce rámce Qt pro zaokrouhlení číselných hodnot.

$$index = qRound64((Y - minY)/sizeOfCellY) \tag{5.2}$$

Po vypočtení prvního indexu buňky, kterou nyní určíme jako předchozí index, se vnoříme do cyklu nad body trajektorií, který začíná od druhého bodu. Na začátku cyklu převedeme bod na index výše zmíněnými metodami. Následně pomocí metody `isCellsNeighbor`, která bere jako parametry předchozí a nynější indexy buňky a vrací, jestli jsou buňky v sousedství, určíme sousednost buněk. Pokud spolu buňky sousedí a nejsou tyto buňky stejné, jelikož může nastat případ, že např. taxík zůstal po dobu více než jednoho snímání polohy na podobném či dokonce stejném místě, tak provedeme přidání ukazatele na buňku do listu buněk trajektorie a ukazatele trajektorie do buňky. Nynější buňka se stává buňkou předchozí. Pokud buňky spolu nesousedí, musíme nalézt buňky mezi předchozí a nynější buňkou.

Na to je určená metoda `augumentCell` ukázána na obr. 5.2, která jako parametry bere dva indexy buněk, které jsou na obrázku červené, a vrací seznam indexů buněk mezi buňkami v parametrech. Metoda vypočítá index, který je uprostřed mezi indexy parametrů na obrázku zelený. Poté vyzkouší sousednosti vypočteného indexu a prvního parametru. Pokud spolu nesousedí, zavolá metoda sama sebe z parametru vypočteného indexu a prvního parametru. Analogicky to funguje pro druhý index. Tato rekurze pokračuje do té doby, dokud spolu vypočtený index a ty předané v parametrech nesousedí, což je splněno při výpočtu modrých buněk. Při vynořování se spojují indexy do seznamu. Tento seznam zpracujeme v cyklu, kde u každé položky vložíme ukazatel na buňky z hlavního seznamu buněk do seznamu buněk dané trajektorie a do buňky vložíme trajektorii.



Obrázek 5.2: Ukázka výpočtu přídavných bodů. Vlevo první výpočet. Vpravo rekurzivní zanoření zanoření.

5.2.3 Detekce

O chod celé detekce se stará metoda `process` třídy `iBoat` a všechny následující metody patří do této třídy pokud nebude řečeno jinak. Nad všemi trajektoriemi v cyklu probíhá algoritmus, jak je popsaný přímo v práci o metodě iBoat [2]. Na začátku u každé trajektorie nastavíme pracovní množinu `workingSet` na všechny trajektorie. Z pracovní množiny odstraníme právě zpracovávanou trajektorii, jelikož by nám to mohlo ovlivnit výsledky. Po inicializaci spustíme cyklus nad každým bodem trajektorie. U každého bodu se provede mapovaní na objekt buňky, který jako ukazatel vložíme do adaptivního okna zastoupeného proměnou `window` typu seznam. Nyní potřebujeme získat trajektorie, které mají stejnou cestu jako buňky v seznamu `window`. K tomu slouží metoda `hasPathWindow`, která pro svoji funkci potřebuje na vstupu aktuální pracovní množinu a seznamu adaptivního okna.

V metodě `hasPathWindow` provedeme nad všemi trajektoriemi ve vstupním parametru `workingSet` test, jestli v nich existují buňky stejné nebo alespoň sousedící s buňkami adaptivního okna `window`, předané jako druhý parametr. Nestačí však, aby buňky byly přítomny v trajektorii, musí také splňovat stejné pořadí výskytu. Poté je trajektorie prohlášena za tu sdílející cestu a přidána do návratového seznamu. Tento test je proveden nad jednotlivými buňkami trajektorie a okna ve vnořených cyklech. V nich se zkouší sousednost buněk pomocí metody `isCellsNeighbor` třídy `IBoatTrajectoryData`. Pokud nalezneme, že všechny buňky z okna jsou přítomny nebo alespoň sousedí s některou z buněk trajektorie přejdeme k druhé části. Pokud alespoň jedna z buněk nesousedí s trajektorií, není dále nijak testována, jelikož porušila jedno z pravidel. V druhé části testujeme, jestli jsou nalezené buňky trajektorie ve stejném pořadí jako buňky v okně. K tomu potřebujeme vrátit pozici buňky v trajektorii, o což se stará metoda `pos`. Ta jako parametr přijímá trajektorii a seznam sousedních buněk včetně buňky, u které chceme zjisti pozici. Tento seznam vrací metoda `neighborsCells`, jíž jako parametr předáme buňku, jejíž sousedy chceme. Metoda `pos` vrací pozici první buňky trajektorie, která je shodná s některou ze seznamu sousedících buněk. Takto postupně vyzkoušíme pro buňky v seznamu okna, jestli předchozí má menší nebo stejnou pozici jako buňky následující. Pokud trajektorie splní jak sousednost, tak pořadí, je přidána do návratového seznamu.

Díky seznamu trajektorií, které mají stejnou část cesty jako adaptivní okno, můžeme vypočítat, jaká je podpora pro daný bod. Toho dosáhneme pomocí metody `support`, která přijímá dva parametry. Jedním je množina trajektorií, který nám vrátila metoda `hasPathWindow`, a druhým je aktuální pracovní množina trajektorií. Pokud je podpora nižší než zadaná θ, je aktuální bod prohlášen za odlehlý a proběhne inicializování hodnot. Inicializováni znamená, že pracovní množina trajektorií obsahuje všechny trajektorie a v

adaptivním okně smažeme všechny buňky kromě poslední přidané. Pokud je podpora větší, je pracovní množina trajektorií změněna na množinu, co vrátila metoda `hasPathWindow`. Tato množina je maximálně stejně velká jako předchozí, ale často dochází k jejímu zmenšení, např. na křižovatkách, odbočkách, a tím pádem k budoucímu ušetření výpočetního času. Jako poslední je vždy pro každý bod vypočítáno jeho skóre a uloženo do seznamu, kde se skóre značně zvětšuje pouze pro anomální body.

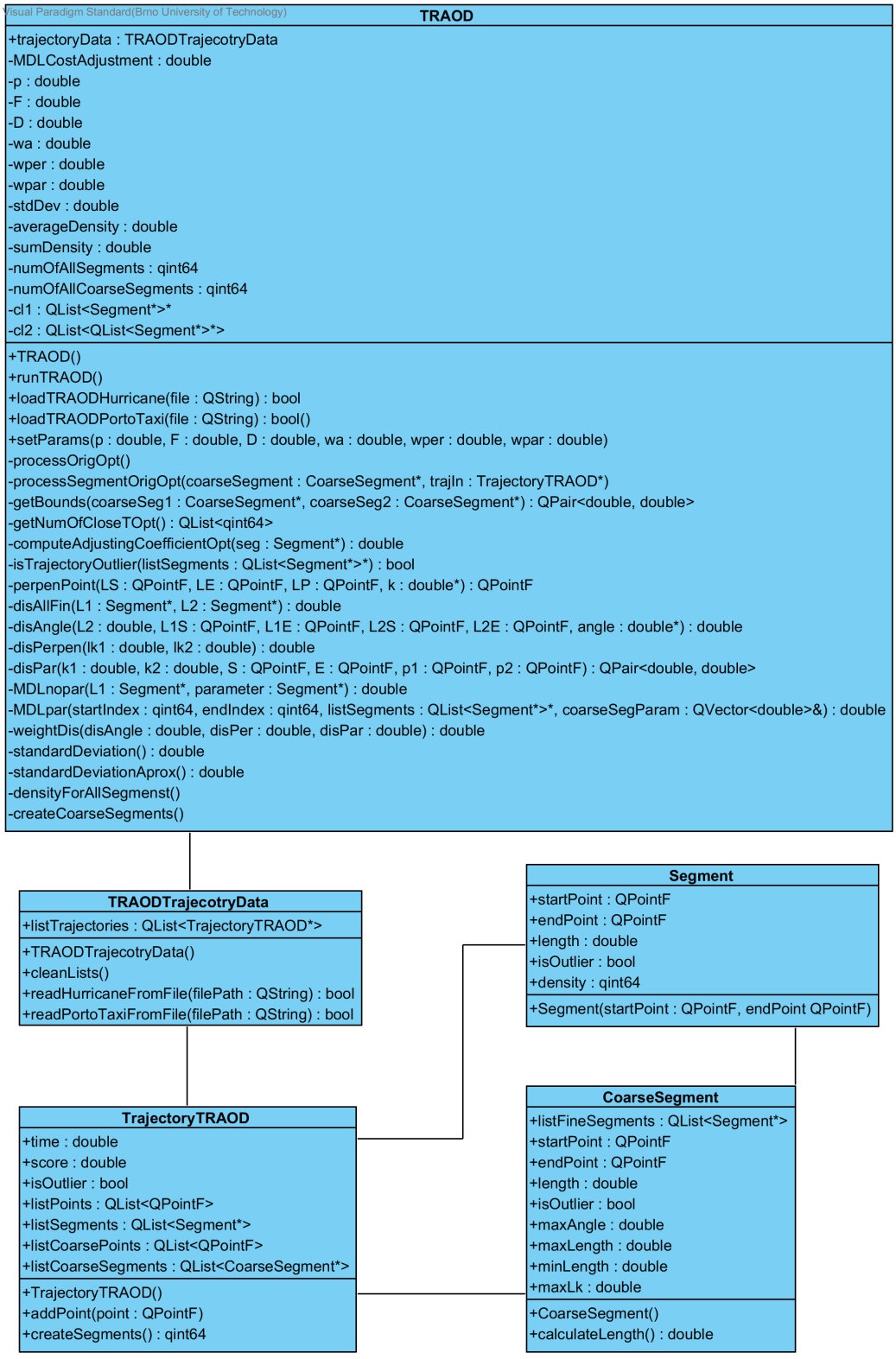
Tímto jsme dosáhli detekce anomálních bodů v trajektoriích pomocí metody iBoat. Jak lze vidět, metoda není na implementaci nijak zvlášť náročná. Přitom její detekční schopnosti jsou velice dobré. Nastavení pomocí několika parametrů je také velice snadné, kde asi nejtěžší je odhadnout správný počet buněk.

5.3 Implementace metody TRAOD

V této sekci bude popsána implementace metody TRAOD a to v následujícím pořadí. Jako první je popsán způsob ukládání dat do struktur vytvořených pro řešení algoritmu, potom následuje implementace výpočtu vzdálenosti mezi segmenty. Poté je uvedeno, jak se vytvářejí hrubé segmenty pro použití v algoritmu. Poslední podsekce je zaměřená na jádro metody. Konkrétně na její optimalizovanou verzi pro využití hrubých segmentů. Veškeré metody dále uvedené jsou členy třídy `TRAOD`. Na obrázku 5.3 je zobrazen diagram tříd pro část metody TRAOD.

K dispozici jsem měl originální zdrojové kódy metody TRAOD[1]. Z této implementace jsem využil způsoby výpočtů všech tří vzdáleností mezi segmenty, které jsem upravil, abych je mohl použít s ostatními částmi kódů, které již byli vytvořeny pouze dle popisu metody.

[1] http://dm.kaist.ac.kr/jaegil/



Obrázek 5.3: Diagram tříd pro část řešící metodu TRAOD

5.3.1 Datová struktura

Třída `TRAODTrajecotryData` obsahuje seznam ukazatelů na objekty trajektorie, dvě metody pro načtení trajektorií a metodu pro vyčištění všech struktur seznamů a smazání objektů, aby šla metoda znovu použít s jinými daty.

Hlavní objekty trajektorie definuje třída `TrajectoryTRAOD`. Ta obsahuje několik listů. První list jsou načtené body trajektorie. Druhým seznamem jsou body, které jsou součástí hrubších segmentů. Poslední dva seznamy obsahují ukazatele na objekty jemných a hrubých segmentů. Jemné segmenty jsou definované třídou `Segment`. Segment obsahuje počáteční a koncový bod popsaný pomocí struktury bodu `QPointF`. Následují hodnoty délky segmentu, jestli je segment anomální a jakou má segment hustotu, neboli kolik segmentů je ve vzdálenosti směrodatné odchylky. Třída `CoarseSegment`, jež popisuje objekty hrubých segmentů, má seznam, v němž jsou ukazatele na objekty jemných segmentů, ze kterých se hrubý segment skládá. Opět zde jsou dva body popisující počáteční a koncový bod hrubého segmentu. Následují pomocné proměnné pro pozdější výpočty a to maximální a minimální délka, maximální úhel a maximální kolmá vzdálenost. Veškeré hodnoty se počítají mezi hrubým segmentem a jeho jemnými segmenty.

5.3.2 Výpočet vzdálenosti

Pro výpočet celkové vzdáleností segmentů se používají tři specifické vzdálenosti, na jejichž výsledky se aplikují váhy určené uživatelem dle potřeb aplikace. Nejdříve si definujeme jednotlivé operace a následně je poskládáme do finální metody `disAllFin`. Všechny metody jsou součástí třídy `TRAOD`, pokud nebude řečeno jinak.

Jako první popíšeme vzdálenost úhlovou, kterou řeší metoda `disAngle`. Jako parametr ji předáváme délku druhého segmentu a počáteční a koncové body obou segmentů. Přes poslední parametr si vracíme vypočítaný úhel, který potřebujeme v některých situacích použít dále. Pro výpočet úhlu použijeme vytvoření vektoru z bodů segmentu. Následně z těchto vektorů vypočítáme skalární součin d a velikost vektorů l. Z těchto hodnot dostaneme úhel $cos\theta = d/l$. Problém je, že pro výpočet úhlové vzdálenosti potřebujeme sinus a ne cosinus. Sinus vypočteme jako $sin\theta = \sqrt{1 - cos\theta^2}$. Tímto výsledkem na konci vynásobíme délku předanou parametrem a máme výsledek úhlové délky, kterou metoda vrací. Zároveň změníme hodnotu úhlu v proměnné předané parametrem, abychom ji mohli použít po dokončení metody.

Pro kolmou a paralelní vzdálenost potřebujeme vypočítat tzv. projekční body. Jde o body p_1 a p_2 z obrázku 4.4. Tyto body získáme metodou `perpenPoint`. Parametry metody jsou body segmentu a bod, ze kterého budeme hledat nejkratší vzdálenost k druhému segmentu, neboli projekci mezi bodem a přímkou. Musíme tedy metodu volat dvakrát abychom získali oba body. Jednou s počátečním a podruhé s koncovým bodem druhého segmentu. Přes další parametr k si vracíme koeficient k vzdálenosti projekčního bodu od počátečního bodu prvního segmentu, jelikož jej potřebujeme dále k výpočtu paralelní vzdálenosti. Výpočet probíhá opět vypočtením vektorů z bodů tentokrát ovšem trochu pozměněný. První vektor $v1$ je vypočítán mezi počátečním a koncovým bodem segmentu a druhý vektor $v2$ mezi bodem z druhého segmentu a koncovým bodem segmentu. Dále vypočítáme dva skalární součiny. První $s1$ je mezi oběma vektory $v1$ a $v2$ a druhý $s2$ se vypočítá pomocí $v1$ sám mezi sebou. Koeficient k poté spočítáme $k = s1/s2$. Ve finále proběhne výpočet projekčního bodu a to následovně $x = x1 + k * v1[0]$ a $y = y1 + k * v1[1]$, kde $x1$ a $y1$ jsou souřadnice počátečního bodu segmentu.

Vypočtené projekční body upotřebíme ve výpočtu eukleidovské vzdálenosti, kterou získáme paralelní a kolmé délky. Paralelní délka je od projekčního bodu k počátku nebo konci segmentu, na kterém kolmý bod leží. O který jde bod, se určí dle dříve vypočítaného koeficientu k. O tento výpočet se stará metoda `disPar`. Ta jako parametry bere projekční body a jejich koeficienty spolu s body počátku a konce segmentu. Vrací dvojici čísel, které udávají paralelní délky. Pro výpočet kolmé délky použijeme eukleidovské vzdálenost mezi body `euclideanDistance`. Body jsou projekční bod a bod, ze kterého byla projekce hledána. Následují výpočty kolmé a paralelní vzdálenosti. Kde paralelní vzdáleností je menší z hodnot paralelních délek. O výpočet kolmé vzdálenosti se stará funkce `disPerpen`.

Celý postup je poté sestaven v metodě `disAllFin`. Jediný problém, který nastává, že musíme určit na kterém segmentu z dvojce budeme vypočítávat projekční body. Řešením je, že projekční body budou na delším ze segmentů. Tím dosáhneme uspokojivých výsledků, jelikož delší segment má větší pravděpodobnost obepínat druhý segment tak, aby nejkratší vzdálenost byla kolmice a tím pádem projekční body nebyly body počátku nebo konce delšího ze segmentů. Všechny získané vzdálenosti ke konci zpracujeme metodou `weightDis`, která vzdálenosti upraví pomocí zadaných vah.

5.3.3 Hustota a korekční koeficient

Pro řešení problému hustoty trajektorií v určité oblasti potřebujme zjistit korekční koeficient, abychom dle něho určili hustotu okolí jednotlivých segmentů. Pro měření hustoty dle definice 4.4, potřebujeme zjistit směrodatnou odchylku. K tomu potřebujeme průměrnou vzdálenost mezi segmenty. To je výpočetně velice náročné, jelikož počet různých dvojic může růst do obrovských čísel. Proto bylo přistoupeno k aproximačnímu řešení metodou `standardDeviationAprox`, kde při výpočtu bereme pouze 10% náhodně vybraných segmentů, u kterých vypočítáme vzdálenosti vůči všem zbylým segmentům. Následně pro každý segment spočítáme počet segmentu, které jsou ve vzdálenosti dané směrodatnou odchylkou a nazveme to hustota segmentu. Tuto hustotu vypočítává metoda `densityForAllSegmenst`. Jde o výpočetně nejnáročnější část metody kvůli vysokému počtu segmentů.

5.3.4 Tvorba hrubých segmentů

Důležitou částí metody TRAOD je tvorba hrubých segmentů. V programu se o tuto část stará metoda `createCoarseSegments`. Zde zpracujeme jemné segmenty každé trajektorie, nad kterými probíhá iterace od prvního segmentu. Než začneme s iterací, nastavíme hodnotu aktuálního indexu jemného segmentu do proměnné `startIndex`. Hodnota bude 0, jelikož indexujeme od 0. Dále nastavíme délku `length` na 1. Tato délka udává, kolik segmentů od `startIndex` budeme používat při výpočtu MDL hodnoty. Při vnoření do cyklu si pro pomocné účely vypočítáme aktuální index `currentIndex`, jenž označuje, který jemný segment je posledním segmentem v hrubém segmentu, a dostaneme jej sečtením `startIndex` a `length`. Nyní potřebujeme MDL hodnoty, jak byly popsány v sekci 4.2.2. Hodnoty `costPar` a `costNoPar` vrací metody `MDLpar` a `MDLnopar`, které budou popsány v dalším odstavci. Po vypočítání MDL hodnot provedeme jejich porovnání. Pokud je hodnota z `costPar` větší než `costNoPar` ukončíme přidávání dalších segmentů do hrubého, jelikož to znamená, že bychom ztratili již příliš informací o trajektorii. Index `startIndex` označující počátek přepíšeme hodnotu indexu `currentIndex`, abychom začali u nového hrubého segmentu segmentem, který již bylo nevýhodné vložit do předchozího hrubého segmentu, a `length` inicializujeme na hodnotu 1. Když `costPar` není větší, potom nynější jemný segment určený `currentIndex` vložíme do hrubého segmentu a uložíme si dodatečné informace

získané při výpočtu MDL hodnot. Hodnotu `costNoPar` navíc zvyšujeme, abychom zvýšili kompresní poměr. Jinak nám totiž vždy vyjde, co jemný segment to hrubý segment z důvodu postupu při aproximaci MDL hodnot.

Pro určení, jestli ještě vložit jemný segment do hrubého potřebujeme dvě různé MDL hodnoty. Ty dostaneme z metod `MDLpar` a `MDLnopar`. První `MDLnopar`, vrací pouze eukleidovskou vzdálenost od prvního k poslednímu bodu. Ve druhé metodě `MDLpar` potřebujeme vypočítat úhlovou a kolmou vzdálenost. Ve vektoru `coarseSegParam` si vracíme hodnoty pro pozdější vypočítání mezí mezi hrubými segmenty. Úhlové a kolmé vzdálenosti vypočítáme mezi hrubým segmentem a všemi jemnými segmenty, které obsahuje. Vzdálenosti potom upravíme logaritmem o základu dva a všechny sečítáme do finální sumy. Ve vektoru `coarseSegParam` vracíme 4 hodnoty a to tyto: maximální a minimální délka, maximální úhel a maximální kolmá vzdálenost jemného segmentu. Než výslednou MDL hodnotu vrátíme, přičteme k ní ještě hodnotu z metody `MDLnopar`. Tímto dosáhneme aproximačního segmentování pomocí MDL.

5.3.5 Detekce

Detekce probíhá v metodě `processOrigOpt`. Zde pro všechny hrubé segmenty trajektorie probíhá následující. Jako první pomocí metody `processSegmentOrigOpt` získáme všechny jemné segmenty aktuálně zkoumané trajektorie, a to v globálním seznamu `cl1` a globální seznam `cl2` jemných segmentů ostatních trajektorií, které by mohly být v dosahu segmentů v `cl1` dle parametru D. Naplnění těchto seznamů záleží na dolní a vrchní hranici hrubých segmentů, které vrací metoda `getBounds`. V ní vypočítáme, podobně jako vzdálenosti, dolní a vrchní meze pro každou vzdálenost, tedy úhlovou, paralelní a kolmou. K výpočtu kromě projekčních bodů a všech délek, jejichž získaní je stejné jako v sekci 5.3.2, potřebujeme hodnoty které byly uloženy během vytváření hrubých segmentů. Poté proběhne výpočet dolní a vrchní meze pro každou vzdálenost a jejich úprava váhami.

Po získání mezí pokračuje metoda `processSegmentOrigOpt` porovnáváním mezí s parametrem vzdálenosti D. Jestliže je dolní mez větší než D, pokračujeme dalším segmentem. Pokud je horní mez větší nebo rovna D, vložíme všechny jemné segmenty do seznamů `cl1` a `cl2`. Pokud není ani jedna z podmínek pravdivá, provedeme mezi jemnými segmenty hrubých segmentu výpočet vzdálenosti. Pokud je vzdálenost menší než D vložíme oba jemné segmenty do jejich korespondujících seznamů.

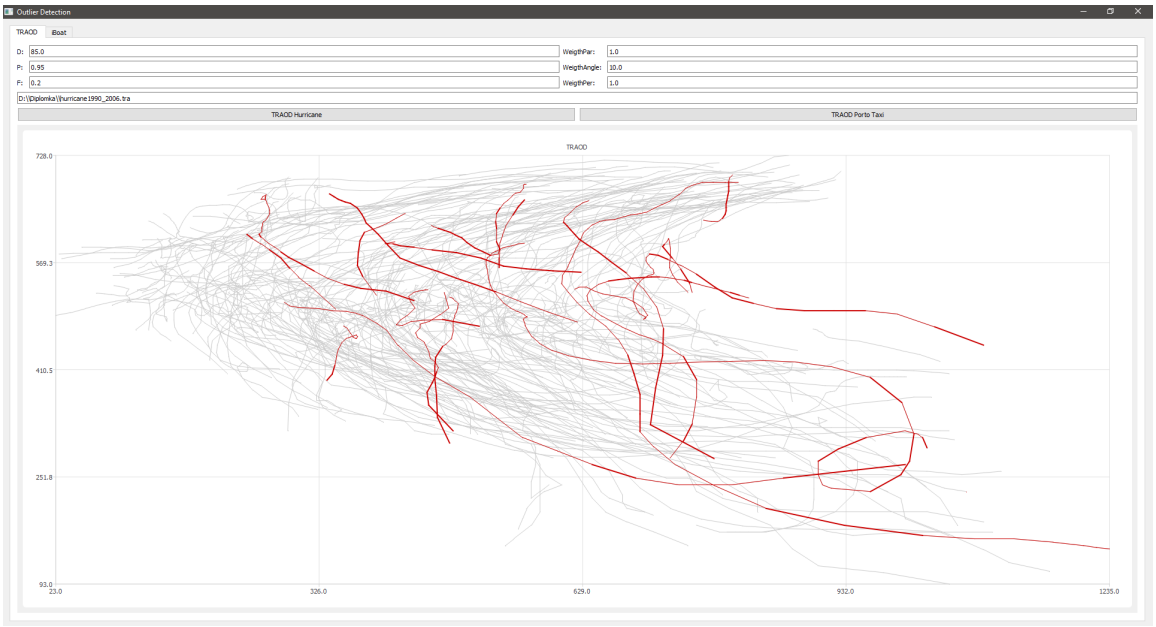
Ukončením metody `processSegmentOrigOpt` získáme všechny jemné segmenty, které mohou ovlivnit výpočet počtu blízkých trajektorií vůči segmentu. O tento počet se stará metoda `getNumOfCloseTOpt`, která nám vrátí seznam počtu blízkých trajektorií vůči segmentům v `cl1`. Pro každý segment v `cl1` zjistíme jestli je trajektorie v `cl2` blízká segmentu. Trajektorie je blízká, jestliže délka všech segmentů trajektorie se vzdáleností menší než D, je větší než délka segmentu, k němuž testujeme blízkost trajektorie. Pokud je trajektorie blízká zvýšíme počet blízkých trajektorií v seznamu, který vracíme.

Posledním krokem je výpočet, jestli je tedy segment odlehlý nebo ne. To provedeme vzorcem 4.2, kde levou stranu vynásobíme korekčním koeficientem, který vrací metoda `computeAdjustingCoefficientOpt`. Tento fakt si zaznačíme v objektu pro segment pro pozdější možnost označení při zobrazení. Ve finále spočítáme délku všech odlehlých segmentů v trajektorii metodou `isTrajectoryOutlier` a jestli je poměr všech segmentů vůči odlehlým menší než parametr F, je také trajektorie prohlášena za odlehlou.

5.4 Uživatelské rozhraní

Pro zobrazení výsledků a zadávaní parametrů bylo vytvořeno jednoduché uživatelské rozhraní pomocí `Qt5 widgets`. Aplikace obsahuje jedno okno s dvěma záložkami, pro každou metodu jednu. V každé záložce je několik grafických prvků typu `lineEdit` pro zadávání parametrů a cesty k souboru s daty. Dále jsou přítomné tlačítka pro spuštění metody, kde každé tlačítko specifikuje zdroj. ze kterého bude načítáno.

U metody TRAOD máme plochu pro vykreslení trajektorií, kde se šedou barvou zobrazují všechny trajektorie a červenou trajektorie odlehlé. Odlehlé segmenty v rámci odlehlé trajektorie se vyznačují tučnější červenou barvou. Na obrázku 5.4 je zobrazena ukázka při detekci anomálií v datech hurikánů.



Obrázek 5.4: Záložka pro metodu TRAOD se zpracovanými trajektoriemi hurikánů.

Jelikož metoda iBoat počítá jednotlivé skóre anomálnosti pro každý bod v trajektorii, je v její záložce přítomna tabulka seřazena podle největší anomálnosti. Opět zde také máme plochu pro vykreslení trajektorií. Šedá barva označuje trajektorie a malé červené kruhové body označují odlehlé body trajektorie. Pokud si chceme zobrazit odlehlé body pro jednotlivé trajektorie, stačí kliknout na příslušnou buňku v tabulce. Vybráním se také zobrazí graf vývoje skóre anomálnosti pod tabulkou trajektorií viz. následující obrázek 5.5.

Obrázek 5.5: Záložka pro metodu iBoat se zpracovanými trajektoriemi taxíků.

Jak zobrazení vývoje anomálního skóre u metody iBoat, tak i vykreslování trajektorií u obou metod, je prováděno pomocí třídy `myChart` a `myChartView`. Tyto třídy dědí z Qt5 tříd `QChart` a `QChartView`, aby bylo možné implementovat schopnost přibližování, oddalování a pohyb v grafech. Přibližování je prováděno pomocí vybrání oblasti myší nebo pomocí klávesy "+". Oddalování je ovládáno pravým tlačítkem myši nebo klávesou "-". Pro pohyb v grafu lze použít klávesové šipky. Toto řešení bylo převzato z rámce Qt z příkladů pro grafy[2].

[2] https://doc.qt.io/qt-5/qtcharts-zoomlinechart-example.html

Kapitola 6

Dolování a vyhodnocení

Nyní bude popsána dolovací úloha a poté jednotlivé výsledky daných metod, jejichž implementace byla popsána v předchozí kapitole. Nejdříve si popíšeme data, nad kterými bylo prováděno dolování a jaké předzpracování nad nimi bylo provedeno. Následně si popíšeme dolovací úlohu a jaké výstupy očekáváme. Poté provedeme popis výsledků jednotlivých metod a na konci budou metody porovnány.

6.1 Dolovací úloha

Jelikož vybrané metody detekují anomální trajektorie nebo jejich segmenty či body, bude tato problematika i náplní dolovací úlohy. Půjde tedy o nalezení takovýchto anomálních trajektorií v široké databázi reálných trajektorií taxíků a hurikánů. Funkčnost vlastní implementace byla nejdříve testována na syntetických datech, kde byly vloženy různé druhy anomálních trajektorií, jelikož v reálných datech si jejich pozicí nejsme jisti a hledat ručně anomální trajektorie v tak velkém množství, je prakticky nemožné. Tímto způsobem byla otestována základní funkčnost implementací.

Cílem je tedy nalézt odlehlé trajektorie či jejich části a porovnání výsledků mezi jednotlivými metodami a různými datovými soubory. Také bylo porovnáváno, jak je obtížné metody nastavit, aby jsme dostávali uspokojivé výsledky a stručně si řekneme, jaké tyto nastavení mají na výsledky vliv. Posledním sledovaným a velice důležitým aspektem je rychlost provedení detekce odlehlosti.

6.1.1 Data

Pro tento účel dolování jsou zvoleny datové soubory trajektorie Portugalských taxíků[1] a hurikánů[2]. Data hurikánu byla použita ve zpracované podobě, jaká je k dispozici ke stáhnutí s originálními zdrojovými kódy metody TRAOD[3]. V nich jsou k dispozici pouze souřadnice pohybu hurikánu, kdežto v originální databázi zde udávají jak tlak a rychlost větru, tak i třeba jméno a typ hurikánu. Datový soubor taxíků ve formátu csv byl také upraven. Nadbytečná data jako jestli je taxík obsazený, jeho konkrétní identifikace atd., byla odstraněna. Ponechána byla jenom časová známka začátku trajektorie a souřadnice bodů trajektorie.

[1] https://archive.ics.uci.edu/ml/datasets/Taxi+Service+Trajectory+-+Prediction+Challenge,+ECML+PKDD+2015

[2] http://weather.unisys.com/hurricane/atlantic/

[3] http://dm.kaist.ac.kr/jaegil/

Formát souboru hurikánů, který přijímá aplikace, je následující. Na prvním řádku je počet dimenzí, jelikož originální implementace nejspíše zvládá i více dimenzionální data. Na druhém řádku je počet trajektorií v souboru. Od třetího řádku začínají samotné trajektorie. Všechna jednotlivá čísla jsou oddělená mezerou. První číslo značí identifikaci hurikánu a druhé číslo počet bodů v trajektorii. Následují dvojce hodnot udávající pozici.

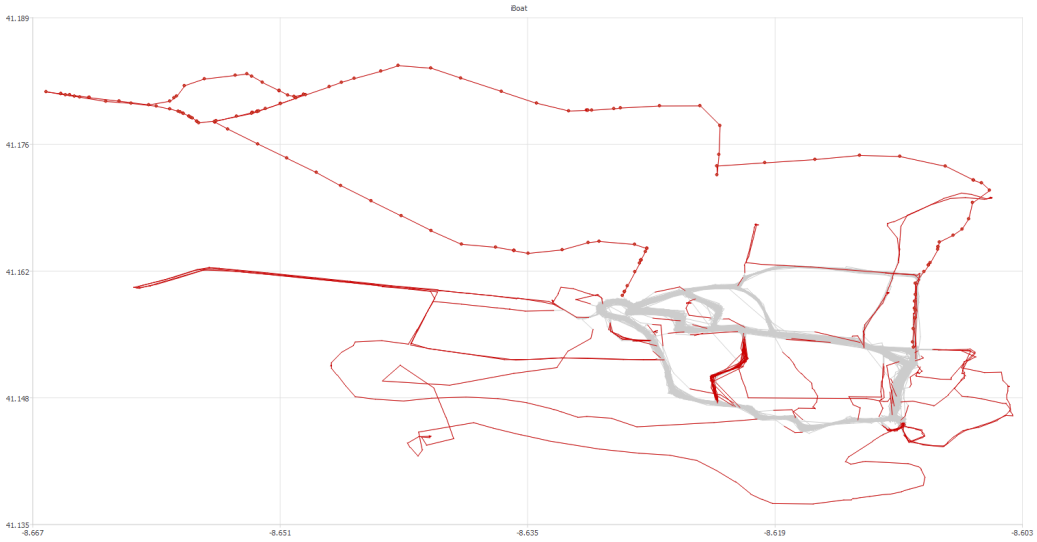
Formát souboru Portugalských taxíků, který přijímá aplikace, je následující. Na každém řádku je jedna trajektorie. První číslo je časová známka, kdy začala být trajektorie měřena. Poté následují GPS souřadnice. Čas i jednotlivé souřadnice jsou oddělené středníkem a body samotné čárkou. Původní datový soubor obsahuje téměř 2 milióny taxíků. To je opravdu mnoho a z výsledků by toho nejspíš vydedukovat mnoho nešlo. Proto byly vybrány pouze trajektorie, jejichž počátky a cíle byly ve stejných 2x2 buňkách dle mapování do buněk v metodě iBoat. Velikost sítě byla 400x400. Následně bylo vybráno 10 sad s nejvíce trajektoriemi.

6.2 Výsledky metody iBoat

Nyní budou popsány výsledky metody iBoat. Budou zobrazeny detekce trajektorií, nejdříve pro data taxíků a následně hurikánů. U taxíku si řekneme co má na detekci vliv a jaké výsledky z nich plynou. Půjde například o hustotu mříže, co má vliv na rychlost apod. U dat hurikánů si jenom zobrazíme příklad výsledků dolování a popíšeme, jestli je metoda pro tento typ trajektorií vhodná. Parametry metody iBoat jsou θ, určující kdy je bod klasifikován jako odlehlý a λ je teplotní konstanta pro výpočet skóre anomálnosti. Dalším parametrem je počet buněk v osách X a Y.

6.2.1 Taxíky

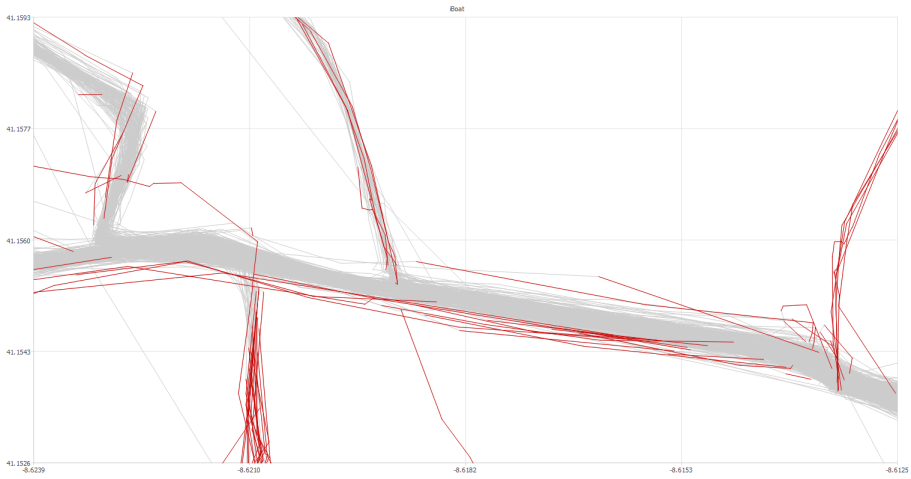
Jako první byla testována datová sada taxíků. Zde iBoat fungoval dobře a odhalil různé odlehlé body trajektorií jako např. na obrázku 6.1. Na něm vidíme přesnou detekci opravdu odlehlých hodnot pro parametry $\theta = 0.05$, $\lambda = 150$, počet buněk v X je 400 a v Y 300. Počet buněk v Y ose je menší, jelikož i rozsah hodnost v ose Y je menší. Červeně jsou označeny odlehlé segmenty a trajektorie s červenými body má největší skóre anomálnosti. Ve středu obrázku, který je hustěji pokryt anomálními částmi trajektorie, vidíme kde pár taxikářů volí zkratku nebo možná objížďku dopravního incidentu.



Obrázek 6.1: Odlehlé části trajektorií pro sadu č. 2.

Parametr metody θ má na výsledky jasný dopad, jelikož jde o poměr blízkých trajektorií vůči trajektoriím v pracovní množině. Čím větší je hodnota θ tím méně trajektorií jedoucích vedlejší cestou bude odhaleno. To znamená, že se nám sníží schopnost odhalit cesty, které se nevyužívají tolik jako hlavní cesty, ale přesto zde jezdí nezanedbatelné množství taxikářů.

Velikost buněk také značně ovlivňuje detekci. Jde především o detekci na okraji naměřených souřadnic v hlavních proudech, pokud jsou buňky příliš malé, jak je ukázáno na obrázku 6.2, kde parametry jsou $\theta = 0.05$, $\lambda = 150$, počet buněk v X je 600 a v Y 400. Této detekci se chceme vyhnout, jelikož evidentně označuje částí trajektorií, které jsou normální. Pokud bychom zvolili příliš velkou hustotu sítě, budou všechny body detekovány jako odlehlé, protože nemají dostatečnou podporu z okolních buněk.



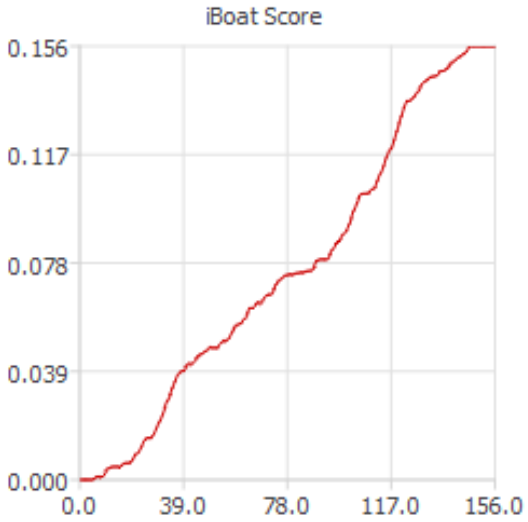
Obrázek 6.2: Vliv hustoty mříže na detekci okrajových hodnot v hlavních proudech.

Další vlastnost, jež ovlivňuje počet buněk, je čas trvání zpracování trajektorie. Kde logicky, čím hustější síť, tím více buněk k porovnávání, a tedy delší výpočetní čas. Ovšem při experimentech bylo naměřeno, že rozdíl není příliš velký. Pro sadu taxíků č. 7, kde

síť byla nastavena 300x200 trval výpočet 54 sekund a pro 1000x800, byla délka ukončení výpočtu 66 sekund, pro všechny trajektorie. Rozdíl tedy není příliš velký, i když je buněk mnohonásobně více. Je to způsobeno tím, že při příliš husté sítí, je každý bod detekován jako odlehlý, jelikož nemá dostatečnou podporu okolí, a tím pádem má adaptivní okno stále velikost pouze dvou prvků. Poté je porovnávání značně rychlejší, když není přítomno více buněk v okně. Na druhou stranu, pokud je mříž nastavena dobře a adaptivní okno má více prvků, tak se nám zmenšuje pracovní množina a opět máme méně porovnávání. Byl zkoušen i vliv počtu trajektorií a počtu jejich bodů. To se také ukázalo jako zásadní faktor pro rychlost detekce. V sadě č. 4, kde je přes dva tisíce trajektorií a cesta mezi počátkem a koncem je delší, a tedy i počet bodů, které testujeme, je také větší. Doba otestování všech trajektorií byla 5 a půl hodiny.

Co je ovšem problém, že při stávající implementaci požadavky na paměť jsou celkem vysoké a pro síť 10000x8000 zabírá téměř 5GB. Ovšem pro experimenty, jež byly provedeny, čili se sítí 300x200 až 1000x800 se spotřeba paměti pohybovala od 20MB do 120MB pro sadu taxíků č. 7, kde trajektorií je 701, což je na dnešní velikosti pamětí více než přijatelná hodnota. Z toho vyplývá že nemůžeme metodu používat nad příliš velkým prostorem, kde aby vyhovovala hustota, byla nutnost vytvořit velkou síť.

Parametr λ pouze ovlivňuje růst skóre anomálnosti. Toto skóre můžeme využít např. pro spuštění nějakého upozornění, pokud skóre přeroste určitou hranici. Hodnota je také ovlivněna funkcí `dist`, což je vzdálenost mezi předchozím a anomálním bodem. V implementaci je použita eukleidovská vzdálenost. Takže u taxíků bude skóre malé, jelikož GPS body v tomto výřezu se od sebe liší pouze v setinách spíše tisícinách. Vhodnější by bylo použít přepočet na skutečnou vzdálenost např. metrech. Na obrázku 6.3 vidíme růst skóre pro trajektorii označenou v obrázku 6.1, která má největší skóre anomálnosti. Na ose x je číslo buňky a na ose y je skóre anomálnosti.

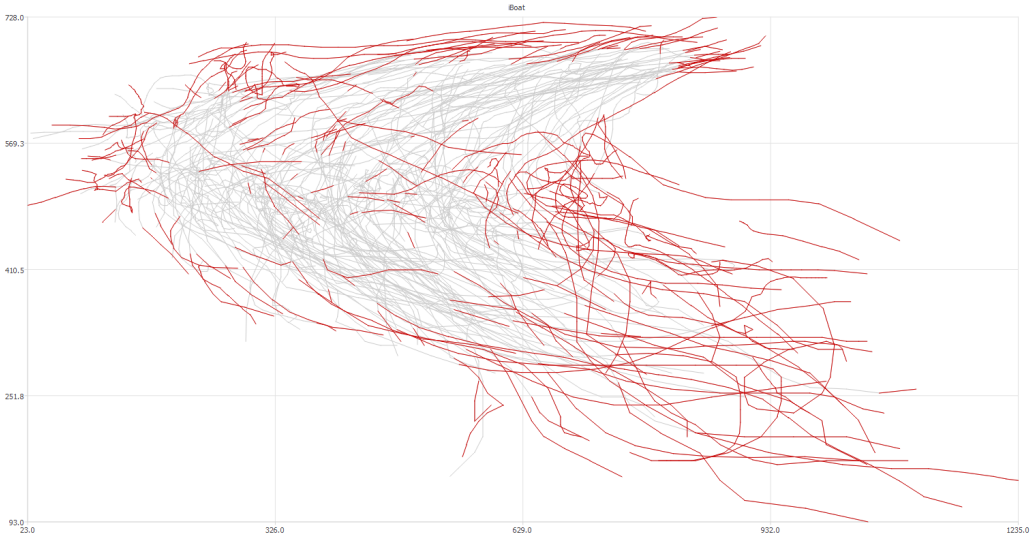


Obrázek 6.3: Růst anomálního skóre.

6.2.2 Hurikány

Při experimentech nad daty hurikánu bylo shledáno, že pro tento typ trajektorií není metoda vhodná. Jelikož zde nejsou trajektorie mají souřadnice počátku a cíle blízko u sebe, aby byli

ve stejné nebo alespoň sousedící buňce, tak mnoho trajektorií nemá dostatečnou podporu. Na obrázku 6.4 je zobrazen výsledek pro parametry $\theta = 0.1$, $\lambda = 150$, počet buněk v X je 80 a v Y 60. Lze opět pozorovat, že okrajové trajektorie jsou označeny jako odlehlé, i když pro tento případ by takto označeny být neměly. Uprostřed oblasti, kde je podpora dostatečná, vidíme detekované hurikány, které bychom označili za anomální, jelikož nedodržují obecný směr.



Obrázek 6.4: Odlehlé části trajektorií pro data hurikánů.

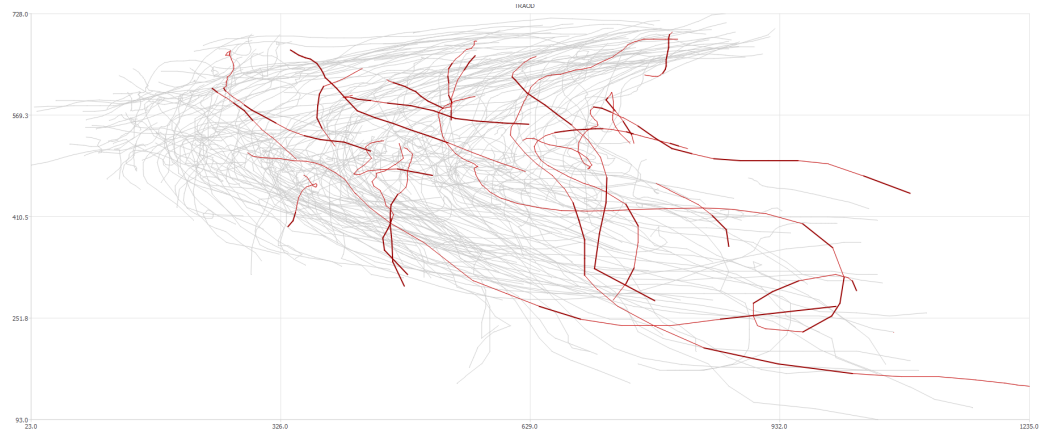
6.3 Výsledky metody TRAOD

V této sekci si popíšeme výsledky experimentů s metodou TRAOD. Budou zobrazeny detekce trajektorií, nejdříve pro data hurikánů a následně taxíků. U hurikánů popíšeme, co má na detekci vliv a jaké výsledky z toho plynou. Půjde například o nastavení jednotlivých vah, co má vliv na rychlost metody apod. U dat taxíků si jenom zobrazíme výsledek dolování a popíšeme jestli je metoda pro tento typ trajektorií vhodná.

Parametry pro metodu TRAOD je několik. Prvním je D určující maximální vzdálenost, kdy dva segmenty jsou prohlášeny za blízké. Parametr F určuje kolik trajektorie musí obsahovat anomálních segmentů, aby sama byla anomální. Posledním parametrem je p, které určuje kolik blízkých trajektorií je potřeba, aby segment nebyl určen jako anomální. Výpočet celkové vzdálenosti mezi segmenty, určují váhy k jednotlivým dílčím vzdálenostem.

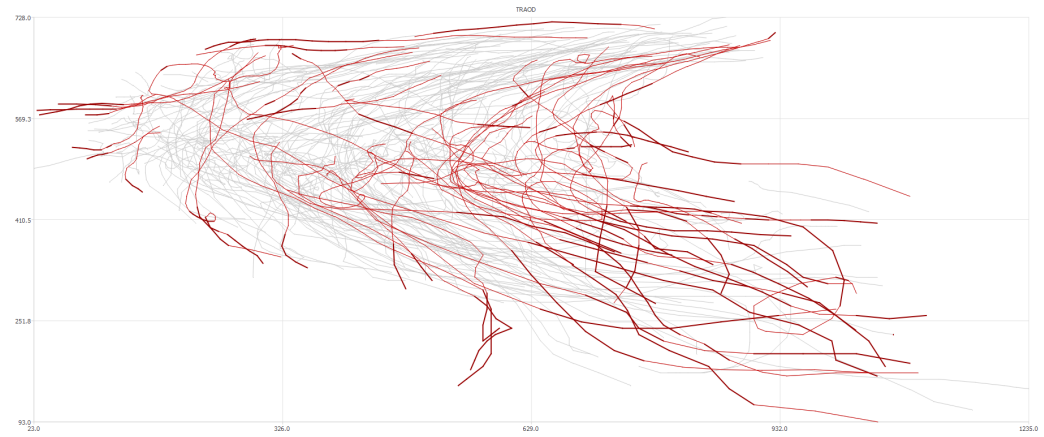
6.3.1 Hurikány

Po několika experimentech nad daty hurikánů bylo shledáno, že metoda je pro tento typ dat vhodná. Díky korekčnímu koeficientu pro hustotu neoznačuje okrajové hodnoty, které dodržují obecný směr ostatních hurikánů, za odlehlé. Označuje ty segmenty podle nastavených vah. Celkové nastavujeme tři různé váhy. Jako první začneme s váhou úhlovou. Díky ní jsou segmenty označeny jako vzdálené pokud spolu svírají velký úhel. Tím dokážeme odhalit trajektorie, které nedodržují obecný směr ostatních hurikánů. Výsledek můžeme vidět na obrázku 6.5, kde parametry jsou následující $D = 85$, $P = 0.95$, $F = 0.2$, úhlová váha 10 a zbylé dvě váhy mají hodnotu 1.



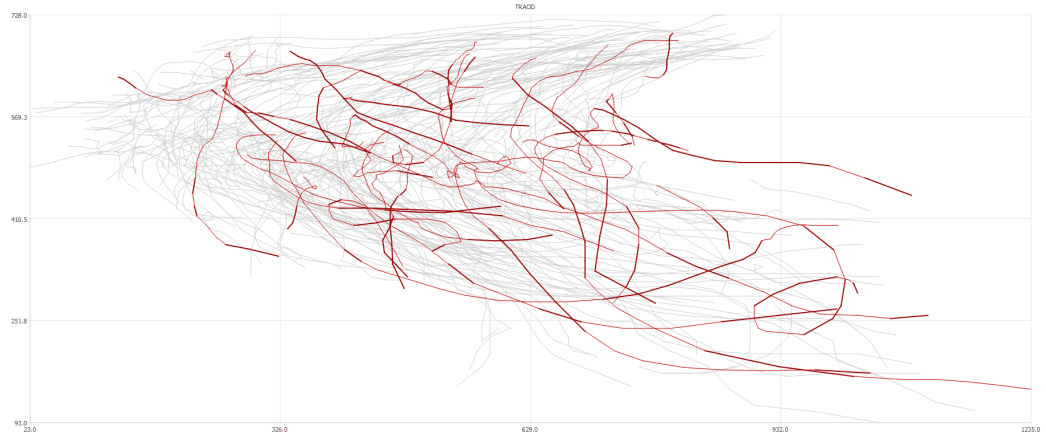
Obrázek 6.5: Odlehlé části trajektorií hurikánů dle zvýšené úhlové váhy.

Další váha ovlivňuje paralelní vzdálenost. Ta určuje, jak vzdálený segment je vůči druhému když mají rozdílnou velikost. Ovlivňuje tedy, aby malé segmenty prostorově blízko nebyly označený jako blízké u velkých segmentů. Poslední váhou je kolmá váha. Ta ovlivňuje projekční vzdálenost, čili vzdálenost jak normálně chápeme. To ovlivňuje detekci odlehlých segmentů vzdálených od hlavních shluků. Příklad je na obrázku 6.6, kde parametry jsou $D = 110$, $P = 0.95$, $F = 0.2$, kolmá váha 5 a zbylé dvě váhy mají hodnotu 1.



Obrázek 6.6: Odlehlé části trajektorií hurikánů dle zvýšené kolmé váhy.

Dalším parametrem je vzdálenost D, která ovlivňuje jestli je segment prohlášen z blízký nebo ne. Musíme jej zadávat dle vah. Protože pokud zvýšíme některou z vah, zvýší se i hodnota vzdálenosti, se kterou D porovnáváme. Celkově však čím nižší D, tím více odlehlých segmentů a poté tedy i trajektorií je detekováno. Parametr P ovlivňuje, při kolika sousedech v dosahu D, je segment prohlášen za odlehlý. Parametr F značí kolik segmentů musí být odlehlých, aby i trajektorie byla prohlášena za odlehlou. Počítají se délky segmentů a ne jejich počet. Na následujícím obrázku 6.7 jsou parametry stejné jako u obrázku 6.5 pouze parametr D byl zmenšen na 80. Rozdíl mezi obrázky je právě počet detekovaných odlehlých trajektorií, kde u obrázku 6.7 je jich více oproti obrázku 6.5, právě díky menší hodnotě parametru D.



Obrázek 6.7: Odlehlé části trajektorií hurikánů dle zmenšeného parametru D.

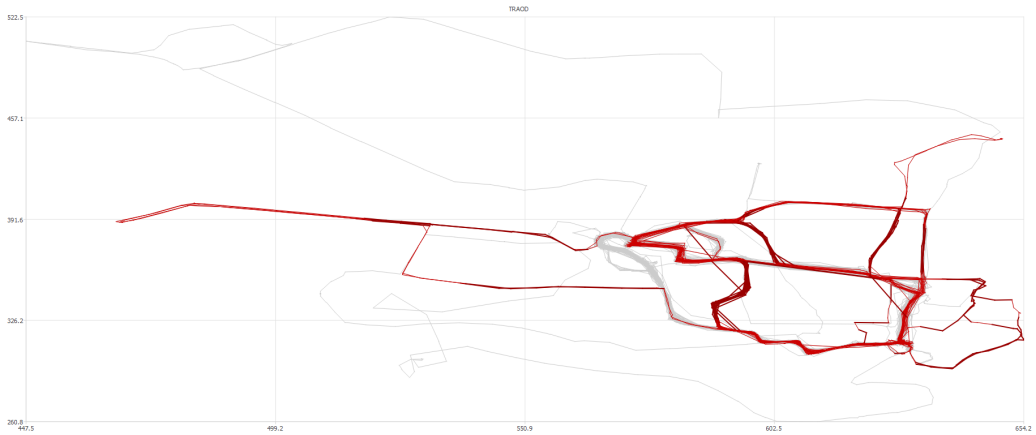
Co však není u této metody příliš dobré, je rychlost výpočtu. Jelikož potřebujeme vypočítat směrodatnou odchylku, hustotu každého segmentu a nakonec samotnou detekci. Doba výpočtu směrodatné odchylky se dala zmenšit její aproximací. Bohužel hustota by takto vylepšit nešla, jelikož tuto vlastnost potřebujeme u všech segmentů. Délka času pro výpočet hurikánů, kde je 221 trajektorií je následující. Doba výpočtu směrodatné odchylky aproximovanou metodou je 6 sekund, pro hustotu to je 57 sekund a pro samotnou detekci 10 sekund. Tyto časy nevypadají, že by to bylo dlouhá doba výpočtu pro takový úkol. Jenomže pokud použijeme stejnou sadu č. 2 Portugalských taxíků jako u metody iBoat, kde u nich byla doba výpočtu 54 sekund, tak pro metodu TRAOD je to značně déle. Aproximace směrodatné odchylky trvala 2 a půl minuty. Výpočet hustoty trval 27 minut a samotná detekce trvala dalších 15 minut. Dohromady to tedy dělá téměř tři čtvrtě hodiny. Výsledky TRAODu nad daty taxík jsou ukázány na obrázku 6.8. S vyšším počtem trajektorií, tedy celkem zásadně roste doba zpracování.

6.3.2 Taxíky

Experimenty nad daty taxíků byly velice zdlouhavé díky délce výpočtu. Projevila se zde i potřeba znalosti dat při zadávaní parametrů. Po prvních několika experimentech a neschopnosti nastavit parametry tak, aby pokaždé nebyla nalezena žádná odlehlá trajektorie nebo naopak byly prohlášené všechny za odlehlé, se hodnoty GPS bodů normalizovaly do rozmezí hodnoty, jaké měli hurikány. Tím byla dosažena alespoň rámcová představa, jaké parametry, konkrétně váhy a vzdálenost D, se mají nastavit. Po několika dalších pokusech vznikl výsledek na obrázku 6.8. Parametry jsou $D = 8$, $P = 0.95$, $F = 0.2$ a kolmá vzdálenost byla nastavena na 2 a zbylé dvě váhy na 1.

Při interpretování výsledku se ukázalo, že metoda TRAOD není vhodná pro tento typ shluklých trajektorií dle počátku a konce, jelikož nedetekuje jasně odlehlé trajektorie. Za to, že nebyly odhaleny výrazně odlehlé trajektorie, může korekční koeficient, který právě vyrovnává hustotu v okrajových oblastech. Je možné, pokud bychom nechali metodu zpracovat nad celou množinou taxíků, že by našla nějaké anomální trajektorie. Tento experiment však nelze provést s prostředky, kterými jsem disponoval z důvodu přílišné výpočetní náročnosti. Je také možnost, že s jiným nastavením parametrů, by se dosáhlo lepší detekce, ovšem díky celkové obtížnosti nastavení parametrů a délce výpočtu nebylo možné vyzkoušet větší počet kombinací. I přesto však, že některé evidentně odlehlé hodnoty odhaleny nebyly, tak

metodou byly označeny jako odlehlé tři přejezdy mezi hlavními silnicemi (tmavě červená barva). Tento další experiment jenom dále ukazuje, že každá metoda je pro jiná data.



Obrázek 6.8: Odlehlé části trajektorií taxíků.

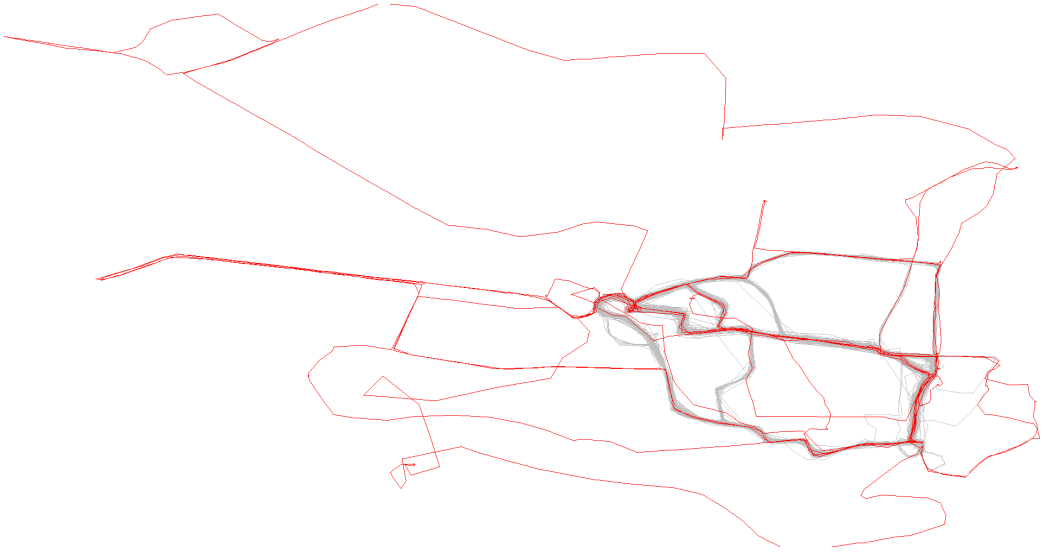
6.4 Výsledky metody TOP-EYE

Nyní se podíváme na poslední metodu. Opět si okomentujeme výsledky nad datovými sadami. Jaké parametry mají vliv na výsledky a další detaily jsou popsány v diplomové práci [22]. Metoda umí dva detekční přístupy jeden na základě hustoty okolí a druhý dle pravděpodobnosti směru pohybu. Pro každou datovou sadu si ukážeme oba případy. Nastavení parametrů bylo velice jednoduché a pro zobrazené výsledky bylo třeba pouze několika málo experimentů. Také rychlost metody je velice dobrá, kde i pro největší sadu taxíků trvala pár sekund. Nejdéle se načítaly trajektorie ze souboru do vnitřní reprezentace.

Parametry pro metodu TOP-EYE jsou následující. Velikost buňky, kterou se poté rozdělí prostor a namapují trajektorie. Parametr λ ovlivňuje úpadkovou funkci pro snížení vlivu předchozí odlehlosti. Dále pro směrovou detekci je zde práh, kdy je trajektorie detekována jako anomální. Pro detekci dle hustoty jsou celkem 3 parametry. Prvním je opět práh detekce, kdy je trajektorie prohlášena za anomální. Poslední dva jsou pro výpočet skóre, kde první je skóre, které bude přiřazováno pokud je hustota menší než druhý parametr.

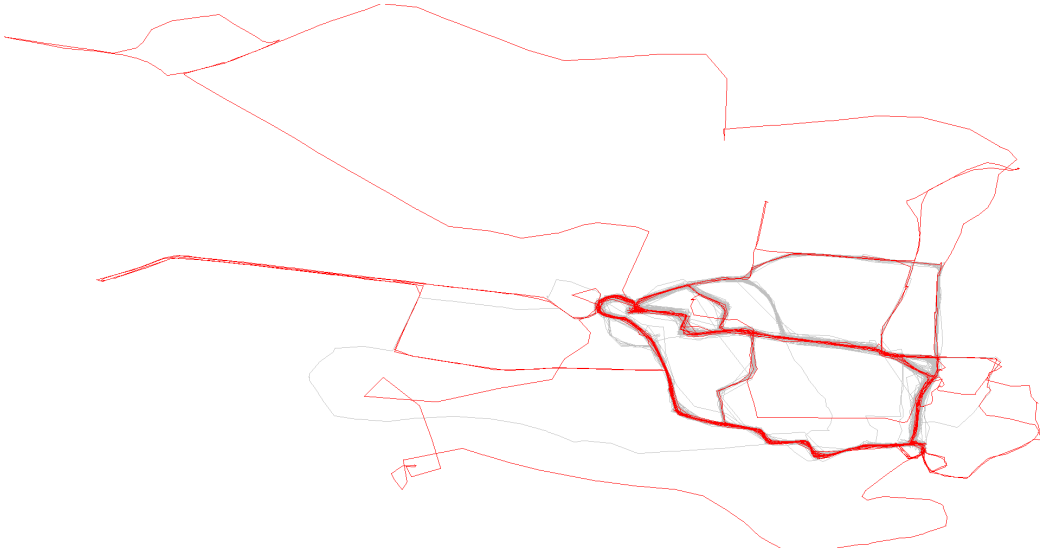
6.4.1 Taxíky

Při experimentování nad daty taxíků bylo zjištěno, že metoda může být pro tento typ použitelná, kde nejlepší výsledky podávala verze, která odlehlost počítá dle hustoty. Výsledek pro hustotu je zobrazen na obrázku 6.9, kde parametry jsou následující. Velikost mřížky 0.002, $\lambda = 0.75$, $s = 1$, $\tau = 5.0$ a práh pro směr je 2.0. Tato odlehlost detekuje opravdu odlehlé trajektorie, jak si je pod tímto pojmem představíme, čili vzdálené od hlavních proudů v prostoru.



Obrázek 6.9: Odlehlé části trajektorií taxíků na základě hustoty.

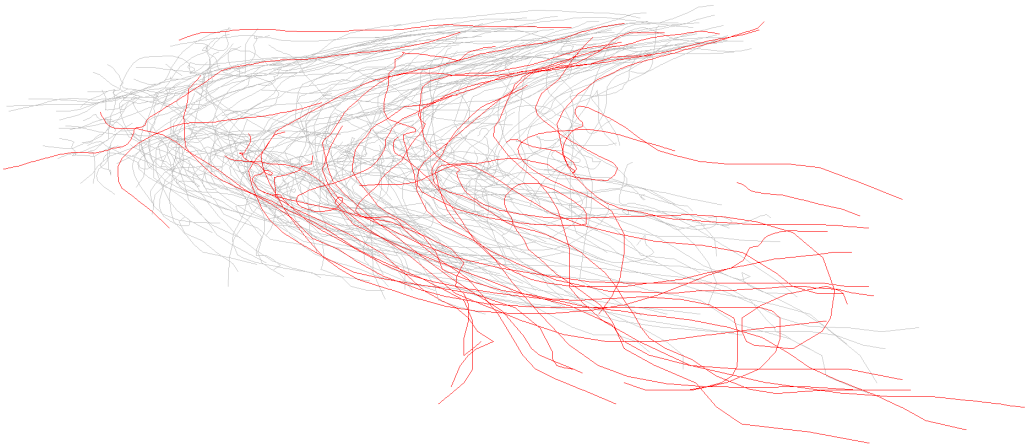
Druhá, a řekněme hlavní část metody, je detekce dle pravděpodobnosti směru trajektorie zobrazena na obrázku 6.10, kde parametry byly stejné jako u předchozího obrázku a práh pro hustotu byl nastaven na 1.5. Díky tomu můžeme detekovat trajektorie, které okolo sebe sice mají dostatečnou podporu dle hustoty, ale vyznačují se, že sice používají často využívané komunikace, ale v jiné kombinaci než ostatní. Mohou mít také jiný směr, což pokud jde o silniční komunikace, znamená jízda v opačbém směru.



Obrázek 6.10: Odlehlé části trajektorií taxíků na základě směru.

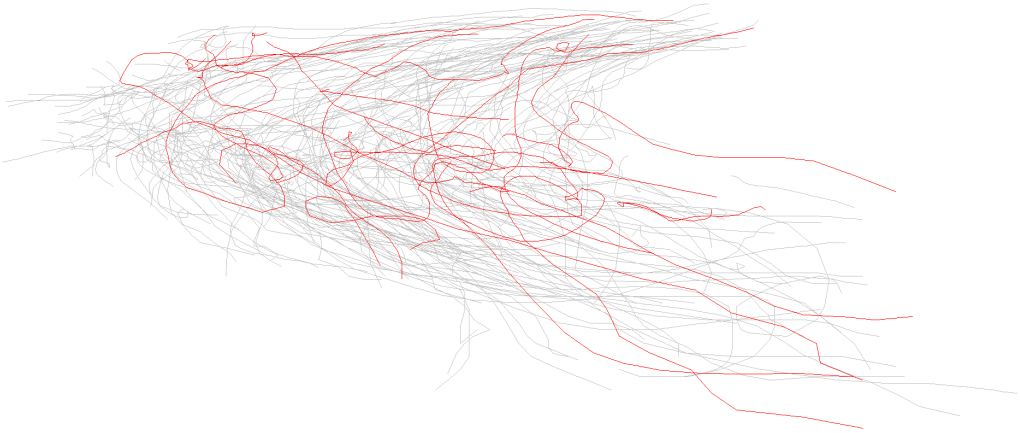
6.4.2 Hurikány

Pro dolování v datech hurikánů byly ponechány stejné parametry jako u dat taxíků, kromě velikosti buněk, které byly změněna na 40. Na obrázku 6.11 jsou zobrazeny výsledky pro detekci dle hustoty. Hurikány, jež pomocí hustoty byly detekovány, by nejspíše i člověk takto označil.



Obrázek 6.11: Odlehlé části trajektorií hurikánů na základě hustoty.

Dle pravděpodobnosti směru byly detekovány odlehlé trajektorie ukázané na obrázku 6.12. Vidíme že detekovány byly hurikány, jež často mění směr a nedodržují celkový obecný směr ostatních hurikánů. Tato metoda je skvěla pro detekci tohoto typu dat, jelikož nás mohou upozornit na hurikán, jenž změnil směr a tím pádem může ohrozit obyvatelstvo nějakého města.



Obrázek 6.12: Odlehlé části trajektorií hurikánů na základě směru.

6.5 Shrnutí a porovnání

Nyní si shrneme schopnosti všech metod použitých při experimentování a porovnáme je mezi sebou. Jako první si vezmeme metodu iBoat. Metoda vynikající pro detekci neobvyklých trajektoriích či jejich částí. Ovšem pouze v trajektorií, které se pohybují na silniční sítí nebo po velice podobné trase. Omezení také vyplývá, že funguje pouze dobře mezi vybranými trajektoriemi, které mají podobný počátek a konec, tak aby se namapoval do sousedních

buněk, proto pro jiné typy trajektorií jako hurikány již metoda není vhodná. Dobrá výkonnost nad objekty pohybujícími se značně podobnou cestou, je dána již tím, že metoda byla vyvinuta jako jedna z možností detekce podvodných taxíků, které nejezdí optimální cestou aby zákazník zaplatil co nejméně. Rychlost je dobrá, když vezmeme v potaz, že metoda je navržená pro online detekci, kde potřebujeme klasifikovat pouze aktuální bod a ne celou množinu trajektorií.

Metoda TRAOD vyšla ze všech metod jako nejhorší. Její nastavení parametrů je příliš složité a vyžaduje zdlouhavé experimentování nebo, jak říkají i autoři, vyžaduje doménového experta, jenž datům a požadavkům případné aplikace rozumí. Detekční schopnosti jsou dobré nad daty typu hurikánů. Ovšem u taxíků detekce neměla dobré výsledky. Nedetekovala ani vyloženě odlehlé trajektorie. To je ovšem dané jejím korekčním koeficientem, který koriguje hustoty okolí. Rychlost je také značnou slabinou, především kvůli nutnosti spočítat směrodatnou odchylku a podle ní hustotu kolem každého segmentu.

Poslední metodou, nad kterou byly provedeny experimenty, je TOP-EYE. Celkově vyšla jako nejlepší a to především díky rychlosti, jednoduchému nastavení a celkově i dobrým výsledkům, jak nad hurikány, tak nad taxíky. Další značnou výhodou je použití detekce, jak na základě hustoty tak i na základě pravděpodobnosti směru. Kde detekce dle hustoty se právě více hodí u dat taxíků, jelikož se pohybují pouze v určitých částech vyhrazených silniční sítí a směrová detekce zase u hurikánů, kde se hurikány pohybují pouze ve všeobecně stejném směru.

Celkový závěr tedy zní, že každá metoda se hodí na určitý typ trajektorií a nejlepší z testovaných je metoda TOP-EYE, jak svojí univerzalitou, tak i rychlostí a nastavením.

Kapitola 7

Závěr

Cílem práce byla problematika dolování neobvyklého chování v datech trajektorií. Tento cíl byl rozdělen do několika podcílů, kde prvním z nich byla definice trajektorie samotné a jejich předzpracování pro lepší nebo rychlejší výsledky dolování. Dalším krokem bylo popsání obecného dolovaní nad daty a specializovanějšího dolování nad daty trajektorií, jakožto typu časoprostorových dat. Poté přichází jádro práce. Jde o seznámení s neobvyklým chováním a to ve formě odlehlých trajektorií a anomálního provozu. Touto problematikou se během let již zabývalo několik výzkumných prací, především v oblasti detekce odlehlých trajektorií. Ke každé oblasti je k dispozici jejich popis a několik různých přístupů v podobě existujících metod řešících tuto problematiku.

Druhou částí práce je praktické vyzkoušení vybraných metod. Metody, které byly vybrány, jsou iBoat, TRAOD, TOP-EYE. Nejdříve každá z metod byla detailněji popsána. Jak funguje a na jakém principu je detekce založena. Pro samotnou implementaci byly vybrány metody iBoat a TRAOD. Pro metodu TOP-EYE byla zvolena, již existující implementace v jazyce Java [22]. Pro implementaci byl zpočátku vybrán jazyk Python z důvodů jeho velké podpory v oblasti dolování. Ovšem po vytvoření prototypů se ukázalo, že implementace je příliš pomalá pro reálné experimentování. Proto finální implementace je napsána v C++ s pomocí rámce Qt5, kde rychlost se zlepšila až o dva řády. V obou jazycích byla zajištěna i multiplatformnost aplikace. Finálním výsledkem je grafická aplikace pro detekci odlehlých trajektorií dle vybrané metody a sady dat. Implementované třídy metod se dají použít i v jiných aplikacích, které potřebují detekci odlehlých trajektorií.

Finálním podcílem byly experimenty nad implementovanými metodami. Byla specifikována dolovací úloha pro odhalení odlehlých trajektorií. Následovaly výsledky jednotlivých metod nad reálnými daty hurikánů a taxíků. U každé metody je popsáno, jaké odlehlé trajektorie detekuje a jaký vliv má nastavení parametrů na detekci či její rychlost provedení. Celkově nejlépe vyšla metoda TOP-EYE. Ovšem pro detekci trajektorií objektů pohybujících se po silniční sítě vyšla lépe metoda iBoat.

Literatura

[1] BRUN, L.; CAPPELLANIA, B.; SAGGESE, A.; aj.: Detection of anomalous driving behaviors by unsupervised learning of graphs. In *11th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2007.

[2] CHEN, C.; ZHANG, D.; CASTRO, P. S.; aj.: iBOAT: Isolation-Based Online Anomalous Trajectory Detection. In *IEEE Transactions on Intelligent Transportation Systems*, 2013.

[3] Fei Tony LIU, K. M. T.: Isolation Forest. In *8th IEEE International Conference on Data Mining*, 2008.

[4] FONTES, V. C.; de ALENCAR, L. A.; RENSO, C.; aj.: Discovering Trajectory Outliers between Regions of Interest. In *GEOINFO*, 2013.

[5] GE, Y.; XIONG, H.; LIU, C.; aj.: A Taxi Driving Fraud Detection System. In *11th International Conference on Data Mining*, 2011.

[6] GE, Y.; XIONG, H.; ZHOU, Z.-H.; aj.: TOP-EYE: Top-k Evolving Trajectory Outlier Detection. In *19th ACM International Conference on Information and Knowledge Management*, 2010.

[7] GIANNOTTI, F.; PEDRESCHI, D. (editoři): *Mobility, Data Mining and Privacy*. Springer, 2008, ISBN 978-3-540-75176-2.

[8] HAN, J.; KAMBER, M. (editoři): *Data Mining: Concepts and Techniques*. Diane Cerra, 2006, ISBN 978-1-55860-901-3.

[9] LAN, J.; LONG, C.; WONG, R. C.-W.; aj.: A New Framework for Traffic Anomaly Detection. In *SIAM International Conference on Data Mining*, 2014.

[10] LEE, J.-G.; HAN, J.; LI, X.: Trajectory Outlier Detection: A Partition-and-Detect Framework. In *24th International Conference on Data Engineering*, 2008.

[11] LEE, J.-G.; HAN, J.; LI, X.; aj.: TraClass: Trajectory Classification Using Hierarchical Region-Based and Trajectory-Based Clustering. In *VLDB Endowment*, 2008.

[12] LEE, J.-G.; HAN, J.; WHANG, K.-Y.: Trajectory Clustering: A Partition-and-Group Framework. In *2007 ACM SIGMOD international conference on Management of data*, 2007.

[13] LI, X.; HAN, J.; KIM, S.; aj.: ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets. In *7th SIAM International Conference on Data Mining (SDM)*, 2007.

[14] LI, X.; LI, Z.; HAN, J.; aj.: Temporal Outlier Detection in Vehicle Traffic Data. In *25th International Conference on Data Engineering*, 2009.

[15] LIU, L.; FAN, J.; QIAO, S.; aj.: Effciently Mining Outliers from Trajectories of Unrestraint Movement. In *3rd International Confrence on Adanced Computer Theor and Engineering(ICACTE)*, 2010.

[16] LIU, W.; ZHENG, Y.; CHAWLA, S.; aj.: Discovering Spatio-Temporal Causal Interactions in Traffic Data Streams. In *17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011.

[17] LOU, Y.; ZHANG, C.; ZHENG, Y.; aj.: Map-Matching for Low-Sampling-Rate GPS Trajectories. In *18th ACM SIGSPATIAL Conference on Advances in Geographical Information Systems*, 2009.

[18] MAHRSI, M. K. E.; GUIGOURES, R.; ROSSI, F.; aj.: Co-Clustering Network-Constrained Trajectory Data. In *8th International Conference on Fuzzy Systems and Knowledge Discovery*, 2011.

[19] MAIMON, O.; ROKACH, L. (editoři): *Data Mining and Knowledge Discovery Handbook*. Springer, 2010, ISBN 978-0-387-09822-7.

[20] PAN, B.; ZHENG, Y.; WILKIE, D.; aj.: Crowd Sensing of Traffic Anomalies based on Human Mobility and Social Media. In *21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013.

[21] PANGA, L. X.; CHAWLA, S.; LIUB, W.; aj.: On Detection of Emerging Anomalous Traffic Patterns Using GPS Data. In *Data & Knowledge Engineering*, ročník 87, 2013.

[22] PESŠEK, M.: *Získávání znalostí z časoprostorových dat*. Diplomová práce, Vysoké učení technické Brno, 2011.

[23] Sanjay CHAWLA, Y. Z.; HU, J.: Inferring the Root Cause in Road Traffic Anomalies. In *12th International Conference on Data Mining*, 2012.

[24] ZHANG, D.; LI, N.; ZHOU, Z.-H.; aj.: iBAT: Detecting Anomalous Taxi Trajectories from GPS Traces. In *13th international conference on Ubiquitous computing*, 2011.

[25] ZHENG, Y.: Trajectory Data Mining: An Overview. *ACM Transaction on Intelligent Systems and Technology*, 2015.
URL https://www.microsoft.com/en-us/research/publication/trajectory-data-mining-an-overview/

[26] ZHENG, Y.; ZHOU, X.; HAN, J. (editoři): *Computing with Spatial Trajectories*. Springer, 2011, ISBN 978-1-4614-1628-9.

Příloha A

Obsah CD

- Zdrojové soubory nástroje v adresáři /src/
  - Zdrojové soubory Qt5 aplikace (TRAOD a iBoat) qt_src/
  - Zdrojové soubory Java aplikace (Top-EYE) java_src/
- Zkompilované aplikace v adresáři /app/
  - Zkompilovaná Qt5 aplikace pro systém Windows (TRAOD a iBoat) qt_app/
  - Zkompilovaná Java aplikace (Top-EYE) java_app/
- Data použitá při experimentech /data/
- Technická zpráva ve formátu PDF v adresáři /thesis/
- Zdrojové soubory technické zprávy v adresáři /thesis_latex/