VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z ČASOPROSTOROVÝCH DAT

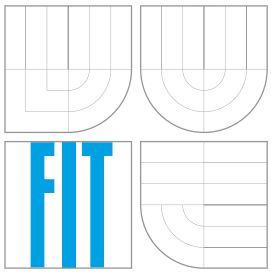
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN PEŠEK

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÍSKÁVÁNÍ ZNALOSTÍ Z ČASOPROSTOROVÝCH DAT

KNOWLEDGE DISCOVERY IN SPATIO-TEMPORAL DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN PEŠEK
AUTHOR

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

BRNO 2011

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2010/2011

Zadání diplomové práce

Řešitel: **Pešek Martin, Bc.**
Obor: Informační systémy
Téma: **Získávání znalostí z časoprostorových dat**
Knowledge Discovery in Spatio-Temporal Data
Kategorie: Data mining

Pokyny:

1. Seznamte se s problematikou získávání znalostí z časoprostorových dat
2. Po dohodě s vedoucím diplomové práce zvolte typ znalosti a dolovací algoritmus pro jeho dolování.
3. Zvolený algoritmus implementujte a experimentálně ověřte jeho vlastnosti.
4. Zhodnoťte dosažené výsledky.

Literatura:

- Hsu, Lee, Wang: Temporal and Spatio-Temporal Data Mining. IGI Publishing, 2007, 292 p., ISBN 978-1599043876

Při obhajobě semestrální části diplomového projektu je požadováno:

- Nevyžaduje se.

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 ročníkového a semestrálního projektu (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: 20. září 2010
Datum odevzdání: 25. května 2011

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

Tato práce se zabývá získáváním znalostí z časoprostorových dat, což je v současné době velmi rychle se vyvíjející oblast výzkumu v informačních technologiích. Nejprve popisuje obecné principy získávání znalostí, následně se po stručném úvodu do dolování v časových a prostorových datech soustředí na přehled a popis existujících metod pro dolování v časoprostorových datech. Zaměřuje se zejména na data pohybujících se objektů v podobě trajektorií s důrazem na metody pro detekci odlehlých trajektorií. V další části se práce věnuje postupu při implementaci algoritmu pro detekci odlehlých trajektorií nazvaného TOP-EYE. Za účelem otestování, ověření a možnosti použití tohoto algoritmu je navržena a realizována aplikace pro detekci odlehlých trajektorií. Algoritmus je experimentálně zhodnocen nad dvěma různými datovými sadami.

Abstract

This thesis deals with knowledge discovery in spatio-temporal data, which is currently a rapidly evolving area of research in information technology. First, it describes the general principles of knowledge discovery, then, after a brief introduction to mining in the temporal and spatial data, it focuses on the overview and description of existing methods for mining in spatio-temporal data. It focuses, in particular, on moving objects data in the form of trajectories with an emphasis on the methods for trajectory outlier detection. The next part of the thesis deals with the process of implementation of the trajectory outlier detection algorithm called TOP-EYE. In order to testing, validation and possibility of using this algorithm is designed and implemented an application for trajectory outlier detection. The algorithm is experimentally evaluated on two different data sets.

Klíčová slova

získávání znalostí z databází, dolování z dat, časoprostorová data, data pohybujících se objektů, trajektorie, odlehlá hodnota, anomálie, detekce odlehlých trajektorií, algoritmus TOP-EYE

Keywords

knowledge discovery in databases, data mining, spatio-temporal data, moving objects data, trajectory, outlier, anomaly, trajectory outlier detection, TOP-EYE algorithm

Citace

Martin Pešek: Získávání znalostí z časoprostorových dat, diplomová práce, Brno, FIT VUT v Brně, 2011

Získávání znalostí z časoprostorových dat

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.

.....
Martin Pešek
25. května 2011

Poděkování

Na tomto místě bych rád poděkoval svému vedoucímu doc. Ing. Jaroslavu Zendulkovi, CSc. za poskytnuté rady a ochotu při řešení této práce a Ing. Petru Chmelařovi za poskytnutí datové sady.

Obsah

Kapitola 1

Úvod

Získávání znalostí z databází je v posledních letech velmi rychle se rozvíjející oblast, která obzvláště vzhledem k obrovskému objemu různých dat nabývá stále více na užitečnosti a důležitosti. Zejména s rozvojem lokalizačních či dohledových systémů, senzorových sítí a mobilních zařízení v nedávné době začal výrazně narůstat objem ukládaných časoprostorových dat. Typicky se jedná o data pohybujících se objektů v podobě jejich trajektorií, může jít také o údaje z meteorologických a oceánografických měření, záznamy přírodních katastrof a podobně. Hlavně vzhledem ke složitosti těchto dat přináší získávání znalostí z časoprostorových dat mnohé výzvy pro současný výzkum.

Tato práce se nejprve v kapitole 2 snaží shrnout a popsat základní principy procesu získávání znalostí z databází a uvést do problematiky časových (temporálních) a prostorových dat a dolování v nich.

Kapitola 3 v úvodu vymezuje základní problémy týkající se časoprostorových dat a dolování v nich. Obsah kapitoly se dále soustředí na přehled a popis existujících technik pro dolování v datech pohybujících se objektů a v databázích časoprostorových událostí. Mezi hlavní zde popsané oblasti dolování patří časoprostorová klasifikace a predikce, časoprostorové shlukování, dolování frekventovaných pohybů objektů a dolování časoprostorových asociačních vzorů.

Mezi významné úlohy analýzy časoprostorových dat patří také detekce anomálií či odlehlých trajektorií v datech pohybujících se objektů. Obvykle se jedná o hledání pohybů, které reprezentují neobvyklé nebo podezřelé chování, tedy takových, které se výrazně odlišují od ostatních. Popisu existujících přístupů k řešení této úlohy se věnuje kapitola 4. Zaměřuje se především na metodu nazvanou TOP-EYE, která umožňuje identifikovat odlehlé trajektorie v reálném čase a vyvarovat se vysokého množství planých poplachů.

Kapitola 5 se zabývá postupem při implementaci algoritmu TOP-EYE. Popisuje nejprve zvolený přístup ke zdrojovým datům, následně jsou postupně rozebrány způsoby implementace jednotlivých kroků algoritmu.

Za účelem otestování, ověření a možnosti použití implementovaného algoritmu je v kapitole 6 navržena aplikace pro detekci odlehlých trajektorií. Cílem této aplikace je poskytnout grafické uživatelské rozhraní pro zpřístupnění funkcí algoritmu nad daty z různých zdrojů a pro prezentaci výsledků dolování.

Výsledky experimentálního ověření vlastností implementovaného algoritmu obsahuje kapitola 7. Algoritmus je ověřen a zhodnocen na základě experimentů nad dvěma různými datovými sadami zejména z pohledu získaných výsledků a časové náročnosti. V závěru je pak shrnuta celá tato práce a zhodnoceny dosažené výsledky.

Kapitola 2

Získávání znalostí z databází

Získávání znalostí z databází (anglicky *Knowledge Discovery in Databases*) [11] je proces extrakce zajímavých znalostí z velkého množství dat, přičemž jako zajímavá je zde chápána znalost, která je netriviální, skrytá, dříve neznámá a potenciálně užitečná.

V současné době existuje velké množství oblastí, ve kterých se získávání znalostí z databází výrazně uplatňuje. Příkladem je analýza trhu a management, finanční analýza, analýza rizik, detekce podvodu a neobvyklého chování, bioinformatika a mnoho dalších.

Velmi často používaným alternativním termínem pro získávání znalostí je dolování z dat (anglicky *data mining*). Ve skutečnosti je však dolování z dat pouze jedním krokem procesu získávání znalostí.

Obsahem této kapitoly je nejprve seznámení s problematikou získávání znalostí. Jsou zde stručně popsány jednotlivé kroky procesu získávání znalostí, dále jsou uvedeny druhy dat pro dolování a typy dolovacích úloh. Zbytek kapitoly je věnován časovým (temporálním) a prostorovým datům a dolování v nich.

2.1 Proces získávání znalostí

Získávání znalostí z databází je proces sestávající ze sekvence následujících kroků [11]:

1. *Čištění dat* – snaha o doplnění chybějících hodnot, odstranění šumu v datech, identifikaci odlehlých hodnot a vyřešení nekonzistence dat.

2. *Integrace dat* – sloučení dat, která pocházejí z více zdrojů. Vzhledem k tomu, že jedním ze zdrojů nekonzistence dat bývá právě sloučení dat z více zdrojů, je často krok integrace prováděn společně s krokem čištění.

3. *Výběr dat* – cílem je vybrat data, která jsou relevantní pro danou dolovací úlohu.

4. *Transformace dat* – úprava dat do podoby vhodné pro dolování. Může zahrnovat odstranění šumu, agregaci, generalizaci, normalizaci a konstrukci nových atributů. V případě potřeby může být provedena i redukce dat.

5. *Dolování z dat* – jádro procesu, kde je aplikován konkrétní algoritmus za účelem extrakce požadovaných vzorů. Algoritmy pro dolování vycházejí z různých disciplín jako je statistika, strojové učení, databázové technologie a jiné.

6. *Vyhodnocení vzorů* – identifikace skutečně zajímavých vzorů založená na určitých mírách užitečnosti. Takové míry mohou mít objektivní či subjektivní charakter.

7. *Prezentace znalostí* – cílem je prezentovat znalosti uživateli pomocí vizualizace a technik reprezentace znalostí.

První čtyři kroky procesu jsou souhrnně označovány jako předzpracování dat. V této fázi se data připravují pro dolování.

2.2 Data pro dolování

Pro dolování je v podstatě možné použít data ze zdroje jakéhokoliv typu. Mezi typické zdroje dat pro dolování patří [11]:

- relační databáze,
- datové sklady,
- transakční databáze,
- objektově-relační databáze,
- temporální databáze, databáze sekvencí a časových řad,
- prostorové a časoprostorové databáze,
- textové databáze,
- multimediální databáze,
- proudy dat,
- web.

2.3 Dolovací úlohy

Typ dolovací úlohy (funkcionalita dolování z dat) určuje druh vzoru, který má být výsledkem dolování. Obecně můžeme dolovací úlohy rozdělit do dvou základních kategorií [11]:

- *deskriptivní* dolovací úlohy, které charakterizují obecné vlastnosti dat v databázi,
- *prediktivní* dolovací úlohy, které provádí předpověď budoucího chování.

Mezi základní typy dolovacích úloh patří popis konceptu nebo třídy, dolování frekventovaných vzorů a asociačních pravidel, klasifikace a predikce, shluková analýza, analýza odlehlých hodnot a evoluční analýza. Popis těchto úloh je obsahem následujících podkapitol, které čerpají z [11].

2.3.1 Popis konceptu nebo třídy

Data mohou být asociována s určitým konceptem nebo třídou. Popis konceptu nebo třídy lze získat jedním ze dvou následujících přístupů:

- *Charakterizace dat* – je sumarizace obecných rysů zkoumané třídy.
- *Diskriminace dat* – je srovnání obecných rysů zkoumané třídy s obecnými rysy jiné třídy nebo množiny tříd.

2.3.2 Dolování frekventovaných vzorů a asociačních pravidel

Frekventované vzory jsou vzory, které se v datech vyskytují často. Frekventované vzory se objevují v mnoha různých podobách, například jako frekventované množiny u transakčních databází, (frekventované) sekvenční vzory, frekventované podgrafy a další. Dolování frekventovaných vzorů odhaluje zajímavé asociace a korelace v datech. Pro definování dalších pojmů bude uvažována transakční databáze.

Definice 2.1. Nechť $I = \{i_1, i_2, \ldots, i_n\}$ je množina položek. Nechť D je databáze transakcí, kde každá transakce T je množina položek taková, že $T \subseteq I$. Každé transakci přísluší unikátní identifikátor. *Asociační pravidlo* je implikace tvaru $A \Rightarrow B$, kde $A \subset I$, $B \subset I$ a $A \cap B = \emptyset$. Podpora (anglicky *support*) a spolehlivost (anglicky *confidence*) pravidla $A \Rightarrow B$ v množině transkací D je pak s využitím pravděpodobnosti definována následovně:

$$support(A \Rightarrow B) = P(A \cup B) \tag{2.1}$$

$$confidence(A \Rightarrow B) = P(B|A) \tag{2.2}$$

Z rovnice 2.2 lze dále odvodit následující rovnici pro výpočet spolehlivosti:

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} \tag{2.3}$$

Uvažujme dále spíše procentuální vyjádření podpory a spolehlivosti. Pravidlo $A \Rightarrow B$ má potom podporu s, pokud $s\,\%$ transakcí v databázi D obsahuje množinu položek $A \cup B$. Spolehlivost c pravidla $A \Rightarrow B$ pak vyjadřuje, že $c\,\%$ transakcí, které obsahují množinu A, obsahují také množinu B. Jako míry pro rozlišení zajímavých asociačních pravidel od nezajímavých se používají hodnoty minimální podpory a minimální spolehlivosti.

Dolování asociačních pravidel je pak tvořeno dvěma kroky:

1. Nalezení všech frekventovaných množin, které splňují podmínku minimální podpory. Pro realizaci tohoto kroku je možné použít například algoritmus *Apriori* či jeho modifikace. Tento algoritmus využívá přístupu generování a testování kandidátů na frekventované množiny s využitím tzv. *Apriori vlastnosti*, která říká, že každá neprázdná podmnožina frekventované množiny musí být také frekventovaná. Hlavními nedostatky tohoto algoritmu je nutnost mnoha průchodů databáze a příliš velký počet generovaných kandidátů. Druhý nedostatek odstraňuje například algoritmus *FP-tree*, který se generování kandidátů vyhýbá. Pro podrobný popis těchto algoritmů viz [11].

2. Generování asociačních pravidel z frekventovaných množin tak, aby získaná pravidla splňovala podmínku minimální podpory a minimální spolehlivosti. Pro realizaci tohoto kroku se nejčastěji využívá rovnice 2.3 pro výpočet spolehlivosti ze známých hodnot podpory.

2.3.3 Klasifikace a predikce

Klasifikace je proces hledání modelu, který přiřazuje data na základě jejich rysů do některé z konečné množiny tříd. Zatímco klasifikace je předpověď kategorického (diskrétního) charakteru, *predikce* umožňuje předpovídat hodnoty obecně spojité funkce.

Klasifikace je proces, který probíhá ve dvou krocích:

1. Během prvního kroku (fáze učení) vytváří příslušný klasifikátor model na základě vybraného vzorku dat, tzv. trénovací množiny. U dat z trénovací množiny musí být známo, do které třídy patří.

2. Ve druhém kroku probíhá testování přesnosti modelu vytvořeného ve fázi učení. Podobně jako v předchozím kroku musí být z databáze vybrán vzorek dat, tzv. testovací množina, u které je známa příslušnost do třídy. Data z testovací množiny by měla být nezávislá na datech z množiny trénovací. Podle výsledku testování klasifikátoru je rozhodnuto, zda může být použit pro klasifikaci dat s neznámou příslušností do některé třídy.

Mezi základní metody používané pro klasifikaci a predikci patří například (pro podrobný popis viz [11]):

- *Rozhodovací strom* – jedná se o graf stromové struktury, kde každý vnitřní uzel reprezentuje test hodnoty některého atributu, každá větev reprezentuje výsledek testu a každý listový uzel reprezentuje třídu, do níž je daný objekt klasifikován.

- *Bayesovská klasifikace* – jedná se o klasifikaci založenou na statistice, konkrétně na výpočtu pravděpodobnosti příslušnosti objektu do jednotlivých tříd.

- *Neuronová síť* – pro klasifikaci se využívá nejčastěji neuronové sítě se zpětným šířením chyby. Jedná se o síť umělých neuronů, přičemž učení zde probíhá postupnou modifikací vah vstupů jednotlivých neuronů.

- *SVM (Support Vector Machines)* – slouží pro klasifikaci lineárních i nelineárních dat. Algoritmus je založen na nelineárním mapování trénovací množiny do vícedimenzionálního prostoru, v němž hledá optimální lineární oddělující rovinu, která maximalizuje vzdálenost mezi jednotlivými třídami.

- *Klasifikace založená na pravidlech* – model je reprezentován jako množina pravidel tvaru IF *podmínka* THEN *třída*, která jsou obvykle extrahována z rozhodnovacího stromu.

- *Klasifikace založená na k-nejbližším sousedství* – klasifikovaný vzorek je přiřazen do třídy, která je nejčetnější v množině k vzorků vybraných z trénovací množiny, které jsou klasifikovanému vzorku nejblíže vzhledem k určité vzdálenostní metrice.

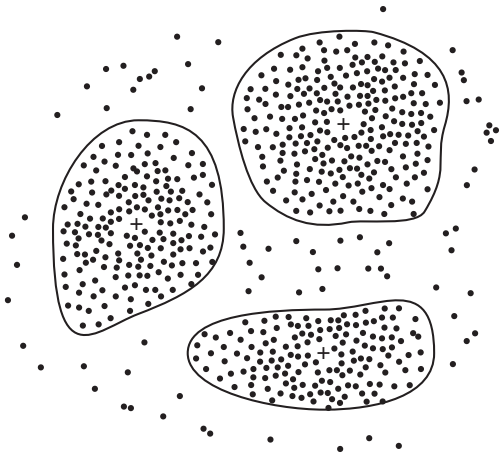
- *Regrese* – jedná se o metodu nejčastěji využívanou pro predikci. Základní variantou je jednoduchá lineární regrese, která je založena na metodě nejmenších čtverců. Zobecněním této metody je násobná lineární regrese. Dalším typem pak je nelineární regrese, kterou však lze většinou transformovat na regresi lineární.

2.3.4 Shluková analýza

Na rozdíl od klasifikace či predikce, kde jsou objekty přiřazovány do předem známých tříd, u shlukové analýzy nejsou cílové třídy předem známy, někdy není dopředu znám ani jejich počet. Zjištění rozdělení analyzovaných dat do tříd je jednou z mnoha možných aplikací shlukové analýzy.

Shlukování (anglicky *clustering*) je proces rozdělování objektů do tříd (shluků) na základě jejich vzájemné podobnosti. Cílem je vytvořit takové shluky, že objekty uvnitř stejného shluku si budou hodně podobné, zatímco objekty z různých shluků si budou podobné

co nejméně. Pro určování podobnosti se nejčastěji využívá vzdálenostní funkce (například Euklidovská vzdálenost). Příklad vytvořených shluků nad dvoudimenzionálními daty je na obrázku 2.1.



Obrázek 2.1: Příklad shluků (převzato z [11])

Základní rozdělení shlukovacích metod je následující (pro podrobný popis viz [11]):

- *Metody založené na rozdělování* – tyto metody rozdělují n objektů do k shluků, přičemž každý shluk musí obsahovat alespoň jeden objekt a každý objekt musí patřit pouze do jednoho shluku. Prvním krokem metody je náhodný výběr k objektů, každý takový objekt reprezentuje právě jeden shluk. Ostatní objekty jsou přiřazeny do shluků podle podobnosti s reprezentativními objekty. Následně dochází k iterativnímu hledání objektů, které nové shluky nejlépe reprezentují, a následnému přesouvání objektů mezi shluky.

- *Hierarchické metody* – metody tohoto typu vytvářejí hierarchický rozklad množiny objektů. Shlukování může probíhat přístupem zdola-nahoru (shlukující metoda), kdy na počátku každý objekt patří do vlastní třídy a postupně dochází ke slučování nejpodobnějších shluků. Opakem je přístup shora-dolů (rozdělující metoda), kdy na počátku patří všechny objekty do jednoho shluku, který je postupně dělen na menší shluky.

- *Metody založené na hustotě* – tyto metody vycházejí z předpokladu, že shluky jsou tvořeny oblastmi s vysokou hustotou objektů v prostoru dat, které jsou navzájem odděleny oblastmi s nízkou hustotou objektů. Metody tohoto typu umí najít shluky libovolného tvaru a lze je použít pro identifikaci odlehlých hodnot.

- *Metody založené na mřížce* – tyto metody rozdělují prostor dat na konečný počet buněk, které tvoří mřížkovou strukturu. Všechny operace jsou pak prováděny nad touto strukturou. Hlavní výhodou těchto metod je rychlá doba zpracování, která typicky není závislá na počtu objektů, ale pouze na počtu buněk mřížky.

- *Metody založené na modelech* – cílem těchto metod je nalézt co nejlepší shodu mezi datovou množinou a daným matematickým modelem.

2.3.5 Analýza odlehlých hodnot

Odlehlé hodnoty (anglicky *outliers*) nebo také anomálie reprezentují takové objekty, které se neshodují s obecným chováním nebo modelem dat. Jsou to tedy takové hodnoty, které se výrazně odlišují od většiny ostatních hodnot v datové sadě. U většiny dolovacích úloh jsou odlehlé hodnoty považovány za nežádoucí šum a bývají identifikovány a odstraněny ve fázi předzpracování dat. Některé úlohy, jako je například detekce neobvyklého chování nebo podvodu, naopak považují odlehlé hodnoty za zajímavé a využívají analýzu odlehlých hodnot ve fázi dolování.

Odlehlé hodnoty lze podle [26] rozdělit do tří následujících kategorií:

- *Ojedinělé (globální) odlehlé hodnoty* – jedná se typicky o izolované instance, které jsou vzdálenené od normálních instancí a odpovídají ojedinělému a extrémnímu chování.
- *Odlehlé shluky (kolektivní odlehlé hodnoty)* – jedná se o shluk instancí, které se společně odlišují od normálních. Jednotlivé instance v odlehlém shluku mohou i nemusí být samy o sobě odlehlé.
- *Okrajové (kontextové) odlehlé hodnoty* – jsou to takové instance s řídkým výskytem v blízkosti shluků normálních instancí, které obvykle odpovídají hranici distribuce normálních instancí.

Na metody pro detekci odlehlých hodnot je možné pohlížet z několika hledisek. Podle toho, zda je pro identifikaci odlehlých hodnot zapotřebí trénovací datové množiny, lze metody klasifikovat následovně:

- *Metody s učením* (anglicky *supervised methods*) – detekce odlehlých hodnot probíhá srovnáním s modelem dat, k jehož vytvoření je nutná trénovací množina. Při volbě konkrétní metody je třeba brát v potaz, zda jsou v trénovací sadě do odpovídajících tříd zařazeny všechny instance, jen normální instance nebo jen odlehlé instance. Typické je zde použití některé z metod klasifikace dat.
- *Metody bez učení* (anglicky *unsupervised methods*) – detekce odlehlých hodnot probíhá výpočtem podobnosti (nebo odlišnosti) analyzované instance se zbytkem datové množiny.

Podle předpokladů o rozdílu mezi normálními a odlehlými hodnotami lze metody pro analýzu odlehlých hodnot kategorizovat následovně [1, 11]:

- *Metody založené na statistické distribuci* – využívají některý ze standardních modelů statistické distribuce dat (například normální rozdělení). Jako odlehlé hodnoty jsou označeny takové, které se od daného modelu liší. Hlavním problémem tohoto přístupu bývá fakt, že většinou je model distribuce vstupních dat neznámý.
- *Metody založené na shlukování* – využívají některou vhodnou shlukovací metodu a jako odlehlé hodnoty identifikují takové, které nejsou zařazeny do žádného shluku.
- *Metody založené na vzdálenosti* – pomocí určité vzdálenostní funkce počítají vzdálenost mezi jednotlivými instancemi. Instance je označena jako odlehlá, jestliže alespoň zadané množství všech instancí leží ve větší vzdálenosti než zadaná minimální vzdálenost.

- *Metody založené na hustotě* – obvykle přiřazují jednotlivým instancím na základě hustoty jejich blízkého okolí hodnotu lokálního faktoru odlehlosti (LOF). Jako odlehlé jsou identifikovány instance s vysokými hodnotami LOF.
- *Metody založené na hloubce* – vytváří nad daty určitou geometrickou strukturu, která je hierarchicky uspořádaná do vrstev. Taková struktura pak splňuje vlastnost, že odlehlé hodnoty jsou velmi pravděpodobně umístěny ve vrstvách s nízkou hloubkou.

2.3.6 Evoluční analýza

Evoluční analýza popisuje a modeluje pravidelnosti nebo trendy objektů, jejichž chování se mění v čase. Mezi typické úlohy tohoto typu patří například analýza trendu či podobnostní hledání v časových řadách a dolovaní sekvenčních a periodických vzorů.

2.4 Dolování v temporálních datech

Temporální (časová) data obsahují kromě vlastních rysů navíc časovou složku. Z pohledu dolování z dat jsou podle [11, 13] dominantní dva typy temporálních dat, a to časové řady a sekvenční data.

Časové řady jsou posloupnosti obvykle reálných čísel nebo událostí měnících se v čase, například ceny akcií, lékařská měření, meteorologická měření a mnoho dalších. Jednotlivé údaje v časových řadách jsou zpravidla zaznamenávány v pravidelných časových intervalech.

Sekvenční data jsou tvořena uspořádanými sekvencemi událostí nebo transakcí, přičemž každá událost může i nemusí být spojena s konkrétním časem. Důležité je zde hlavně uspořádání podle času. Příkladem sekvenčních dat mohou být posloupnosti přístupů na webovou stránku, posloupnosti nákupů zákazníků a další.

Přítomnost časové složky v datech sice zavádí do dolování vzorů z temporálních dat určitou složitost navíc, ale umožňuje získávat nové druhy znalostí, a to zejména znalostí týkajících se chování. Podle [13] lze otázky dolování v temporálních datech klasifikovat podle dvou kritérií:

1. Podle zapojení času do dolovacího procesu:
   (a) Podstatné je pouze pořadí dat, nikoliv konkrétní časový údaj.
   (b) Důležitý je konkrétní časový údaj.
   (c) Kromě časového údaje je podstatná i časová hierarchie.
2. Podle typu vstupních dat:
   (a) Data jsou numerického charakteru (typické pro časové řady).
   (b) Data jsou kategorického charakteru (typické pro sekvenční data).

2.4.1 Dolování v časových řadách

Mezi typické úlohy dolování v časových řadách patří analýza trendu a podobnostní hledání (pro podrobný popis viz [11, 20]).

Pro *analýzu trendu* bývá obvykle časová řada modelovaná dekompozicí jako součin nebo suma čtyř základních charakteristik pohybu časové řady, kterými jsou trend, cyklické, sezónní a nepravidelné pohyby. Trend reprezentuje obecný směr pohybu řady z dlouhodobého

pohledu, vyjádřený přímkou či křivkou trendu. Cyklické pohyby vyjadřují dlouhodobé oscilace řady kolem křivky trendu. Sezónní pohyby se týkají téměř identických vzorů, které se v řadě vyskytují ve stejných úsecích následujících období. Nepravidelné nebo náhodné pohyby popisují ojedinělé pohyby řady způsobené náhodnými událostmi.

Podobnostní hledání v časových řadách hledá sekvence dat, které se nutně nemusí shodovat se zadanou sekvencí, ale mohou se od ní mírně odlišovat, v čemž je rozdíl od běžných databázových dotazů, které vyhledají data zcela vyhovující danému dotazu. V podstatě je možné odlišit dva typy podobnostního hledání. Prvním typem je *porovnání úseků řady*, kde výsledkem dotazu jsou všechny časové řady obsahující úsek, který je podobný časové řadě dotazu. Druhým typem je *porovnání celých řad*, které vyhledá množinu sekvencí podobných navzájem. Mnohem častěji se v praxi objevuje problém porovnání úseků řady. Příkladem častého užití podobnostního hledání v časových řadách je finanční analýza trhu, lékařská diagnostika a další.

2.4.2 Sekvenční vzory

Cílem dolování sekvenčních vzorů je nalezení všech frekventovaných podsekvencí. Pro definování dalších pojmů bude uvažována transakční databáze.

Definice 2.2. Nechť $I = \{i_1, i_2, \ldots, i_n\}$ je množina všech položek. *Sekvence* (viz [11]) $s = \langle e_1 e_2 \cdots e_m \rangle$ je uspořádaný seznam událostí, kde událost e_1 předchází události e_2 a tak dále. Každá událost $e = (x_1 x_2 \cdots x_p)$ je neprázdnou množinou položek, pro kterou platí $e \subseteq I$. Jedna položka se tedy může vyskytovat v rámci jedné události nejvýše jednou, ale v rámci celé sekvence vícekrát.

Sekvence $s_1 = \langle a_1 a_2 \cdots a_k \rangle$ je nazvána *podsekvencí* jiné sekvence $s_2 = \langle b_1 b_2 \cdots b_l \rangle$ a naopak s_2 je *nadsekvencí* s_1, pokud existují celá čísla $1 \leq j_1 < j_2 < \cdots < j_k \leq l$ taková, že $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_k \subseteq b_{j_k}$.

Podpora $support(s)$ sekvence s v databázi sekvencí je počet sekvencí, které obsahují s (jsou její nadsekvencí). Sekvence s je *frekventovaná*, pokud $support(s) \geq minsup$, kde $minsup$ je *minimální podpora*. Frekventovaná sekvence se pak nazývá *sekvenční vzor*.

Základní přístupy k dolování sekvenčních vzorů reprezentují algoritmy GSP, SPADE, a PrefixSpan (viz [11, 13]). Všechny tři přístupy přímo či nepřímo využívají *Apriori vlastnost*, zmíněnou již v kapitole 2.3.2. Tato vlastnost, přizpůsobená pro sekvenční vzory, říká, že každá neprázdná podsekvence sekvenčního vzoru je sekvenční vzor.

Hlavní rozdíl mezi algoritmy GSP a SPADE spočívá v používaném formátu dat. Zatímco GSP uvažuje horizontální reprezentaci databáze, kde každý řádek databáze reprezentuje jednu transakci s odpovídající identifikací a časovým označením, SPADE uvažuje vertikální reprezentaci databáze. Při vertikálním formátu jsou data organizována podle jednotlivých položek. Každé položce (množině položek) pak odpovídá seznam transakcí, které tuto položku obsahují. Užití vertikálního formátu dat má za následek redukci počtu průchodů databáze. Stejně jako GSP však i SPADE využívá prohledávání do šířky, čímž dochází ke generování příliš mnoha kandidátních sekvencí.

PrefixSpan je algoritmus pro dolování sekvenčních vzorů, který je založen na přístupu růstu vzoru a tedy prohledávání do hloubky, čímž se vyhýbá generování kandidátů. V porovnání výkonnosti s algoritmy GSP a SPADE dosahuje PrefixSpan nejlepších výsledků [11].

2.4.3 Temporální asociační pravidla

Temporální asociační pravidla lze chápat jako přirozené rozšíření sémantiky klasických asociačních pravidel o časovou informaci [13, 20].

V [5] je dolování temporálních asociačních pravidel uvažováno nad transakční databází, ve které je každá transakce doplněna o rozsah platnosti ve formě časového intervalu. Temporální asociační pravidla jsou pak dále rozdělena na několik typů vzhledem k možné interpretaci problému, jakou roli v konceptu asociačních pravidel hraje čas.

2.4.4 Periodické vzory

Periodické vzory [13] jsou takové vzory, které se pravidelně opakují v databázi časových řad. Obecně lze odlišit dva druhy periodických vzorů: úplné periodické vzory a částečné periodické vzory.

Pro *úplné periodické vzory* platí, že každý údaj v databázi má podíl na části vzoru. Například každý den v roce se podílí na sezónním cyklu pro každý rok.

U *částečných periodických vzorů* naopak postačuje, když se pouze část všech údajů v databázi podílí na vzoru. Částečná periodicita je volnější formou periodicity než úplná a v praxi je mnohem běžnější. Příkladem může být, že osoba vstává každý pracovní den ve stejný čas, avšak o víkendech nikoliv.

2.5 Dolování v prostorových datech

Prostorová data [5, 11] obsahují kromě vlastních rysů navíc složku s určením pozice v prostoru. Na taková data lze pohlížet jako na data reprezentující objekty, které jsou umístěny ve fyzickém prostoru. Prostorová data bývají uložena v prostorových databázích, které udržují a spravují velké množství těchto dat. O uložených objektech obsahují obvykle jak data prostorového charakteru tak neprostorového. Tyto databáze nesou informaci o topologii a vzdálenostech uložených objektů, organizovanou obvykle v sofistikovaných strukturách prostorových indexů.

Důležitou roli při analýze prostorových dat hraje fakt, že splňují vlastnost nazývanou *prostorová autokorelace* [22], kterou nejlépe vysvětluje Toblerův první zákon geografie [23]:

> *„Všechno souvisí se vším ostatním, ale blízké věci spolu souvisí více než vzdálené věci.“*

Dolování v prostorových datech [13] je proces získávání zajímavých vztahů mezi prostorovými a neprostorovými daty využitím vztahů prostorové blízkosti. Vzhledem k obrovskému množství dat v prostorových databázích a složitosti datových typů a přístupových metod je použití tradičních metod pro dolování dat omezené, a proto je často zapotřebí hledat efektivnější metody [11]. Mezi typické dolovací úlohy zde patří prostorová charakterizace nebo diskriminace, prostorová klasifikace, analýza trendu, prostorové shlukování, detekce odlehlých objektů a dolování prostorových asociačních vzorů. U asociačních vzorů je možné odlišit dva typy: prostorová asociační pravidla a prostorové kolokační vzory. Významnou oblast analýzy prostorových dat dále tvoří prostorové datové sklady a OLAP (viz například [11]).

2.5.1 Prostorová klasifikace a analýza trendu

Prostorová klasifikace [11] analyzuje prostorové objekty za účelem odvození klasifikačního modelu na základě určitých prostorových vlastností, například sousedství s řekou a podobně. Příkladem využití prostorové klasifikace může být odhad ceny pozemku na základě jeho polohy. Popis metod pro klasifikaci a predikci v prostorových datech lze nalézt v [22].

Prostorová analýza trendu [11] se zabývá detekcí trendu a změn podle prostorové dimenze. Jde tedy o zkoumání trendu prostorových nebo neprostorových dat v závislosti na změně polohy. Například jaký je trend změn vegetace v závislosti na rostoucí vzdálenosti od oceánu.

2.5.2 Prostorové shlukování

Prostorové shlukování [5] slouží k nalezení shluků (například hustě osídlených oblastí) na základě vzdálenostních metrik v rozsáhlých multidimenzionálních databázích. Kromě požadavku na efektivní zpracování takových dat patří mezi další důležité požadavky na algoritmy pro shlukování prostorových dat schopnost vytvářet shluky různých tvarů, nezávislost na pořadí při zpracování vstupních dat a odolnost vůči odlehlým objektům. Pro prostorové shlukování je možné použít některé z tradičních algoritmů, existují však také metody cílené přímo na prostorová data. Pro popis shlukovacích algoritmů viz [5, 11].

2.5.3 Prostorová asociační pravidla

Prostorová asociační pravidla [11] jsou přirozeným rozšířením klasických asociačních pravidel v prostorových databázích začleněním prostorových predikátů, které mohou reprezentovat informaci o vzdálenosti (například *blízko_k*), topologické vztahy (například *protíná* nebo *překrývá*) a prostorové operace (například *nalevo_od*). Příkladem použití může být zjištění, které typy objektů se často vyskytují blízko hřiště.

Kvůli typicky velkému množství prostorových dat se při dolování prostorových asociačních pravidel často využívá optimalizace pomocí metody *postupného zjemňování*. Nejprve jsou rychlým algoritmem, který slouží jako filtr, získány přibližné výsledky. Na ty je následně aplikován jiný, méně efektivní algoritmus, který ale dokáže získat oproti prvnímu algoritmu pouze kvalitní výsledky. Na první algoritmus je kladen požadavek, aby zachoval všechny potenciálně zajímavé výsledky.

Hlavním omezením těchto pravidel je jejich závislost na konceptu explicitně uložených transakcí v databázích. Prostorové vztahy typicky nejsou v databázi explicitně zakódovány, ale musí být z uložených dat extrahovány [13].

2.5.4 Prostorové kolokační vzory

Prostorové kolokační vzory [22] reprezentují množiny prostorových rysů těch objektů, které se často vyskytují blízko sebe. Odhalují tedy častý výskyt prostorových rysů v sousedství objektů s jinými prostorovými rysy. K dolování kolokačních vzorů je možné využít různé přístupy, jejichž popis je možné najít například v [22].

Kapitola 3

Dolování v časoprostorových datech

Časoprostorová data jsou data, která kromě vlastních rysů obsahují jak časovou tak prostorovou složku. Typicky se jedná o data pohybujících se objektů v podobě jejich trajektorií, může jít o údaje z meteorologických a oceánografických měření, záznamy přírodních katastrof a podobně. Časoprostorová data lze zjednodušeně kategorizovat následovně [10]:

1. Události v prostoru a čase – například letecké katastrofy, zemětřesení, lesní požáry a podobně.
2. Posloupnosti událostí v prostoru a čase.
3. Trajektorie pohybujících se objektů – může jít o pohyb lidí, zvířat, dopravních prostředků, ale například i ledovců, bouří a podobně.

V poslední době, zejména s rozvojem lokalizačních systémů, dohledových systémů, senzorových sítí a mobilních zařízení, dochází k velkému nárůstu objemu ukládaných časoprostorových dat, obzvláště dat o pohybujících se objektech. Vzhledem k tomu se výrazně zvyšuje potenciál těchto dat a časoprostorové databáze se postupně stávají velmi aktivní oblastí výzkumu. Kromě rozvoje technologií pro modelování, dotazování a indexování časoprostorových dat je velký důraz kladen i na oblast získávání znalostí z těchto dat. I přes existenci řady publikovaných výsledků v této oblasti se dá říci, že dolování v časoprostorových datech je oproti dolování v mnoha jiných typech dat teprve v počátcích.

V období prvních výzkumů v oblasti dolování z časoprostorových dat se nabízela možnost použít existující techniky dolování v temporálních a v prostorových datech pro časoprostorová data. Tento přístup může odhalit zajímavé vzory, ale soustředí se na časovou nebo prostorovou dimenzi zvlášť. Časoprostorová data však obsahují složité vztahy, které nelze odhalit pohledem na časovou a prostorovou dimenzi dat odděleně. Druhým problémem, který se při tomto přístupu často projevuje, je příliš rozsáhlý prohledávací prostor časové i prostorové dimenze. Z těchto důvodů jsou zapotřebí efektivní techniky, které sjednotí časovou a prostorovou informaci dohromady za účelem nalezení zajímavých a užitečných časoprostorových vzorů [13].

Mezi typy úloh dolování v časoprostorových datech patří dolování evolučních vzorů přirozených jevů, časoprostorová klasifikace a predikce, časoprostorové shlukování, detekce anomálií, dolování periodických vzorů, dolování frekventovaných pohybů objektů, a dolování časoprostorových asociačních vzorů. Obsah této kapitoly je zaměřen na popis většiny

z uvedených vzorů či modelů a na přehled existujících technik pro jejich dolování. Pro analýzu dat pohybujících se objektů je možné využít i datové sklady s technologií OLAP přizpůsobené trajektoriím. O této oblasti lze najít více informací například v [8].

Evoluční vzory přirozených jevů se týkají především dat environmentálního charakteru jako jsou údaje o počasí, vědecká data o Zemi a podobně. Přehled některých existujících prací z této oblasti lze nalézt v [13].

Klasifikace časoprostorových dat je proces získávání modelu chování objektů na základě současných dat za účelem určitých předpovědí o nových datech. Příkladem aplikace takového modelu může být simulace městského provozu. Klasifikací a predikcí časoprostorových dat se zabývá podkapitola 3.1.

Shluková analýza dat pohybujících se objektů spočívá v hledání skupin podobných trajektorií společně s jejich souhrnnou charakteristikou. Shluky pak mohou reprezentovat hlavní trasy pohybujících se objektů (například osob či vozidel), jejichž znalost může být významná pro analýzu pohybu. Časoprostorovým shlukováním dat pohybujících se objektů se zabývá podkapitola 3.2.

Problematika dolování frekventovaných vzorů z časoprostorových dat se podle charakteru vstupních dat dělí na dvě oblasti. Data, s nimiž dolovací algoritmus pracuje, mohou být buď ve formě trajektorií pohybujících se objektů, nebo v podobě událostí. Vzory, které lze dolovat z trajektorií, odpovídají nejčastěji frekventovaným pohybům objektů. Tímto typem úlohy se zabývá podkapitola 3.3. Události mohou reprezentovat výskyt určitých rysů či jevů, nebo se může jednat o informace odvozené z dat pohybujících se objektů v podobě časoprostorových predikátů, statických či dynamických agregátů a podobně. Vzory, které lze dolovat z dat tohoto typu, je možné označit jako časoprostorové asociační vzory, kterým se věnuje podkapitola 3.4.

Mezi významné úlohy analýzy časoprostorových dat patří také detekce odlehlých trajektorií nebo anomálií v datech pohybujících se objektů. Touto oblastí se podrobněji zabývá kapitola 4.

3.1 Časoprostorová klasifikace a predikce

Časoprostorová data, zejména data pohybujících se objektů, nabízí širokou oblast pro aplikaci různých úloh typu predikce a klasifikace. Konkrétním příkladem může být předpověď cílové pozice pohybujícího se objektu (například osoby či vozidla) nebo rozpoznání situace jako je například dopravní zácpa. Mezi typické úlohy zde patří predikce polohy a trajektorie, predikce hustoty, predikce dosahu, predikce událostí a klasifikace trajektorií [8].

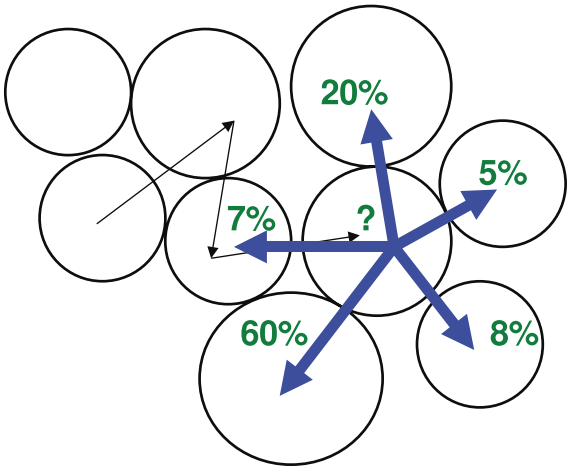
Predikce polohy a trajektorie

Při predikci budoucí polohy pohybujícího se objektu se často využívá současné pozice objektu a vektoru rychlosti. Pro efektivní dotazování nad časoprostorovými daty za účelem predikce budoucí pozice byl navržen například TPR-strom a jeho optimalizovaná verze TPR*. Při tomto přístupu se však vychází z předpokladu, že během uvažovaného časového intervalu objekt nebude měnit směr a rychlost svého pohybu. Tento předpoklad může platit například u pohybu lodí a letadel, nikoliv však v městském provozu.

Dalším možným přístupem k predikci polohy je analýza historických dat trajektorií pohybujících se objektů a odvození převládajících vzorů z nich. Pro trajektorii v dotazu se pak použije vzor, který je s danou trajektorií nejpodobnější. Obdobným způsobem lze provádět předpověď budoucích tras a cílových pozic objektů. Použití je však opět omezeno

na předpoklad, že objekt se pohybuje po některé z častých tras (například osoba cestuje do práce, na nákup a podobně).

Ilustrace problému predikce trajektorie je zobrazena na obrázku 3.1.



Obrázek 3.1: Příklad predikce trajektorie (převzato z [8])

Predikce hustoty

Hustota objektů v určité oblasti je definována jako poměr počtu objektů uvnitř oblasti vzhledem k velikosti oblasti v daném časovém okamžiku. Predikce hustoty objektů může mít uplatnění v různých oblastech. Například v dopravních systémech je takto možné detekovat blížící se překážku a podobně.

Predikce dosahu

Dosah je časově závislá míra, která vyjadřuje publicitu místa u určité populace. Dosah nemusí být omezen pouze na jedno místo, ale může být uvažován také pro síť míst. Pak je definován jako podíl populace, která navštíví alespoň jedno z míst sítě v daném časovém období.

Význam predikce dosahu je zejména v extrapolaci nesouměrného a nekompletního vzorku trajektorií, například pokud jsou data dostupná pro kratší časový interval, než je požadováno, nebo když je zapotřebí rozšířit dostupnou datovou množinu na větší prostor.

Predikce událostí

Příkladem predikce časoprostorových událostí může být určení pravděpodobnosti spáchání nějakého zločinu v dané oblasti a v daném časovém období na základě odpovídajících historických údajů.

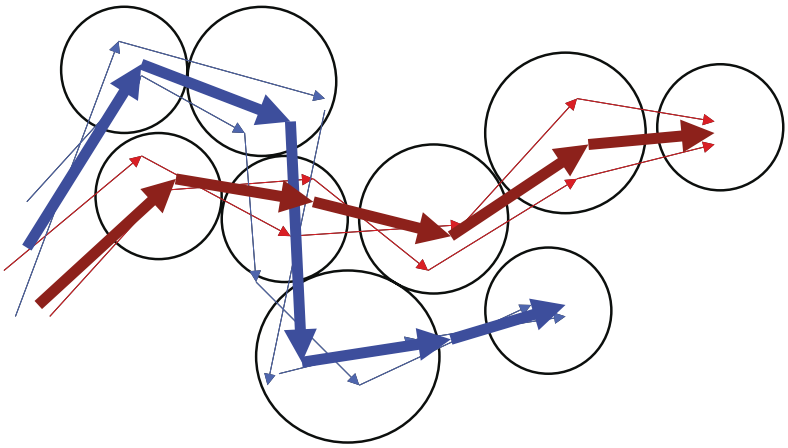
Klasifikace trajektorií

Klasifikace trajektorií umožňuje získávat a přidávat metadata k trajektoriím. Takto lze například přizpůsobit chování navigačního systému na základě odvozeného cestovního záměru osoby. Obecně je možné trajektorie klasifikovat pomocí algoritmů založených na nejbližším

sousedství poskytnutím vhodné vzdálenostní funkce, která ovšem závisí na konkrétní klasifikační úloze.

3.2 Časoprostorové shlukování

Při analýze velkého množství dat je často potřeba rozdělit datovou množinu do více rozdílných skupin (shluků), přičemž data ve stejné skupině jsou si vzájemně blízká (například sdílejí určitou vlastnost), zatímco data z různých skupin nikoliv. Mezi typické úlohy shlukování časoprostorových dat, konkrétně dat pohybujících se objektů, patří nalezení shluků objektů pohybujících se podobnou rychlostí, detekce dopravní zácpy a další. Na obrázku 3.2 je ilustrován problém shlukování podobných trajektorií, kdy každý shluk je reprezentován jednou hlavní trajektorií.



Obrázek 3.2: Příklad shlukování trajektorií (převzato z [8])

Pro shlukování trajektorií je možné využít dva různé přístupy. První je založen na použití existujících obecných shlukovacích algoritmů definováním určité vzdálenostní funkce mezi trajektoriemi. Při tomto přístupu je sémantika časoprostorových dat zcela zapouzdřena ve vzdálenostní funkci. Druhým existujícím přístupem je použití nových algoritmů, které jsou přizpůsobené konkrétnímu typu dat [8].

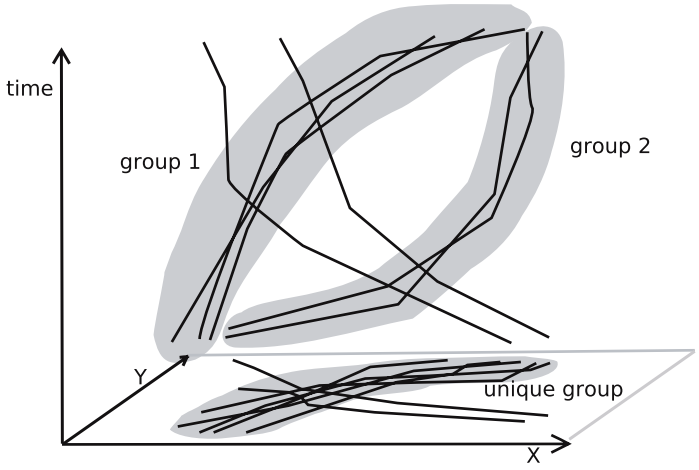
3.2.1 Shlukování trajektorií založené na vzdálenosti

Pro shlukování trajektorií pohybujících se objektů lze vycházet z některého existujícího obecného shlukovacího algoritmu, přičemž vlastnosti získaných shluků závisí na konkrétním zvoleném typu (viz kapitola 2.3.4). Pro určení, které objekty mají patřit do stejného shluku, je nutné definovat vhodnou vzdálenostní funkci. Podle přístupu k časovým a prostorovým omezením při výpočtu vzdálenosti dvou trajektorií lze rozlišit následující tři způsoby shlukování:

1. Jako podobné jsou považovány objekty, jejichž trajektorie jsou přibližně stejné, tedy že v každém čase se nachází na přibližně stejném místě. Jednotlivé takto získané shluky pak reprezentují skupiny objektů, které se pohybují společně. Obrázek 3.3 ukazuje příklad dvou shluků a dvou samostatných trajektorií objektů pohybujících se v čase ve dvourozměrném prostoru.

2. Odstraněním časového omezení z předchozího případu je možné hledat shluky objektů, které se pohybují po stejných trasách, ne však nutně ve stejných časových okamžicích. Tento případ ilustruje spodní část obrázku 3.3, kde jsou trajektorie objektů zobrazeny po promítnutí do dvourozměrného prostoru. Trajektorie, které v předchozím případu tvořily dva shluky, nyní tvoří shluk jediný.

3. Poslední možností je nevyžadovat ani stejné trasy objektů, ale zaměřit se na vytváření shluků objektů, které provádějí podobné pohyby, jako například udržování stejného směru, stejná odbočování ze směru a podobně.



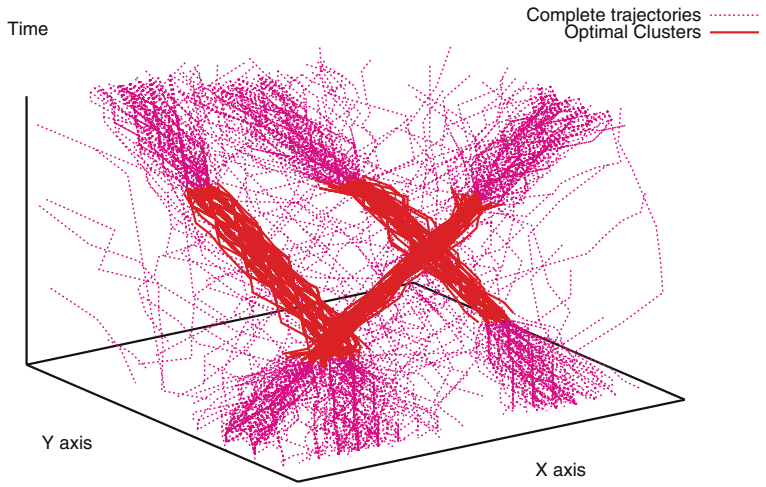
Obrázek 3.3: Shluky trajektorií objektů pohybujících se v čase ve dvourozměrném prostoru (*group 1* a *group 2*) a jediný shluk stejných trajektorií po promítnutí do prostoru (*unique group*) (převzato z [8])

3.2.2 Shlukování přizpůsobené trajektoriím

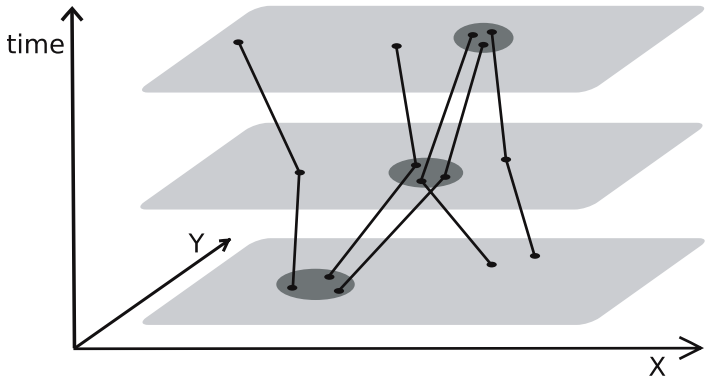
Metody shlukování trajektorií založené na vzdálenosti přinášejí některá omezení, například ve výkonnosti nebo v neschopnosti vyjádřit některé požadavky na shluky vycházející z časoprostorových dat. Kromě dále zmíněných dvou technik existuje řada přístupů, které se touto oblastí zabývají. Jejich stručný přehled je možné nalézt v [8].

Někdy může být potřeba vyhledat shluky objektů, které se pohybují společně jen v nějakém časovém intervalu o určité minimální velikosti. Tento problém lze vyřešit tak, že pro každý časový interval proběhne shlukování segmentů trajektorií spadajících do daného intervalu pomocí některé metody založené na vzdálenosti. Následně jsou vyhledány intervaly, které dávají nejlepší výsledky. Obrázek 3.4 zachycuje množinu trajektorií pohybujících se objektů v čase ve dvourozměrném prostoru, které tvoří tři shluky ve zvýrazněném optimálním časovém intervalu.

Další možný pohled na shlukování trajektorií přináší pojem *pohybujícího se shluku*. Data pohybujících se objektů jsou zde uvažována ve formě prostorových pozic v určitých časových úsecích. V každém časovém úseku jsou nalezeny prostorové shluky pomocí některé metody založené na hustotě. Pohybující se shluky pak přetrvávají v přibližně stejné podobě v několika následujících časových úsecích. Příklad pohybujícího se shluku je ukázán na obrázku 3.5.



Obrázek 3.4: Příklad shluků trajektorií v omezeném časovém intervalu (převzato z [8])



Obrázek 3.5: Příklad pohybujícího se shluku (převzato z [8])

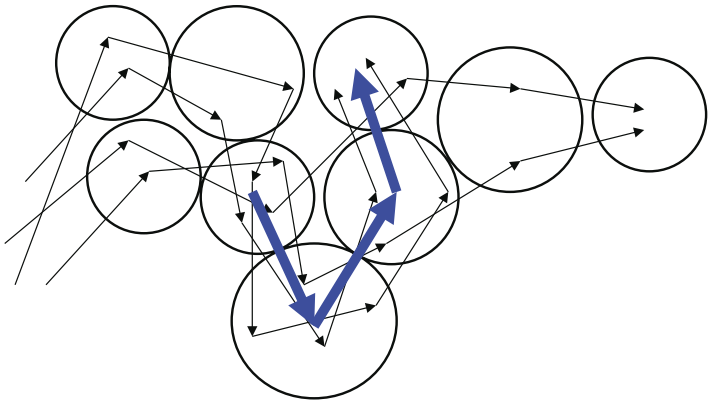
3.3 Frekventované pohyby objektů

Dolování frekventovaných vzorů z trajektorií pohybujících se objektů je závislé na tom, zda dolovací úloha požaduje vyhledat všechny zajímavé frekventované vzory ve vstupní datové množině, nebo zda je úkolem najít všechny výskyty určitého vzoru [8]. První případ odpovídá *přímému vyhledávání*, kdy obvykle hledáme všechny vzory, které odpovídají často sledovaným trajektoriím. Tomuto přístupu se věnuje první část této podkapitoly. Druhý případ odpovídá *inverznímu vyhledávání*, kdy vstupem dolovací úlohy je konkrétní typ vzoru a úkolem je v datové množině nalézt všechny jeho frekventované výskyty. Tento přístup je popsán v druhé části této podkapitoly.

3.3.1 Dolování frekventovaných trajektorií

Účelem dolování frekventovaných trajektorií (případně sub-trajektorií) pohybujících se objektů je nalezení takových tras nebo cest, které jsou pohybujícími se objekty často sledovány. Příklad frekventované sub-trajektorie je ukázán na obrázku 3.6.

Dolováním frekventovaných vzorů z trajektorií se zabývá například [7]. Zavádí nový typ



Obrázek 3.6: Příklad frekventované trajektorie (převzato z [8])

vzoru, tzv. T-vzor, který sjednocuje časovou i prostorou informaci bez předchozí diskretizace časové či prostorové dimenze.

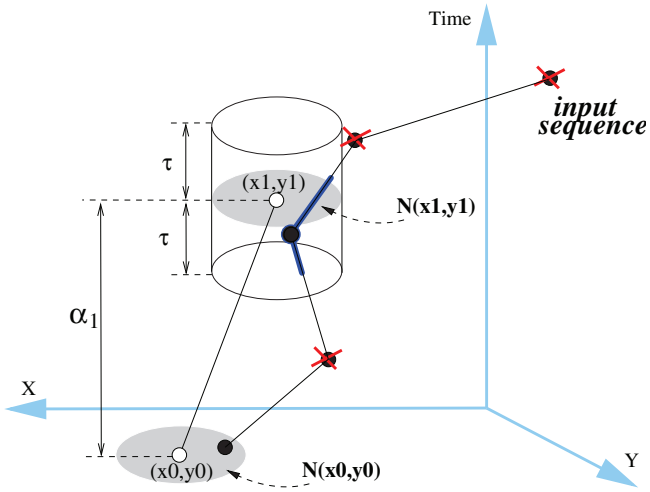
Definice 3.1. *T-vzor* (v angličtině *T-pattern*, *Trajectory pattern*) [7] je dvojice (S, A) taková, že $S = \langle (x_0, y_0), \ldots, (x_k, y_k) \rangle$ je sekvence pozic v prostoru, kde $\forall i \in \{0, \ldots, k\} : (x_i, y_i) \in \mathbb{R}^2$, a $A = \langle \alpha_1, \ldots, \alpha_k \rangle$ je sekvence časů přechodů mezi prostorovými pozicemi, kde $\forall i \in \{1, \ldots, k\} : \alpha_i \in \mathbb{R}^+$.

T-vzor je pak možné reprezentovat jako $(S, A) = (x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1) \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} (x_k, y_k)$.

Čas přechodu mezi dvěma pozicemi vyjadřuje dobu přesunu pohybujícího se objektu z jedné prostorové pozice do druhé. T-vzor obsahující n prostorových pozic reprezentuje ve vstupní databázi trajektorií všechny sub-trajektorie s n prostorovými pozicemi takové, že každá pozice sub-trajektorie je přibližně stejná jako odpovídající pozice ve vzoru a všechny časy přechodů mezi pozicemi sub-trajektorie jsou přibližně stejné jako odpovídající časy přechodů ve vzoru. Kromě parametru minimální podpory vzoru je pro dolování frekventovaných T-vzorů nutné určit hodnotu časové tolerance a funkci prostorové sousednosti. Příklad porovnání konkrétního T-vzoru se vstupní sekvencí (trajektorií) je ilustrován na obrázku 3.7. Z obrázku je názorně vidět, že prostorová pozice ve vstupní trajektorii je ztotožněna s pozicí ve vzoru, pokud se nachází v oblasti vymezené hodnotou časové tolerance τ a funkcí prostorové sousednosti N.

3.3.2 Vyhledávání výskytu vzoru

Na rozdíl od přímého vyhledávání frekventovaných vzorů v trajektoriích pohybujících se objektů je smyslem inverzního vyhledávání najít všechny výskyty vzoru určitého typu. Inverzní dotaz může být podle [8] dvojího typu, a to elementární nebo synoptický. *Elementární dotazy* reprezentují chování jednotlivých pohybujících se objektů. Elementárním inverzním dotazem může být například požadavek na vyhledání všech objektů, které v daném časovém intervalu navštívily dané místo. Takový dotaz lze dále rozšiřovat například o sekvenční informaci, o různé prostorové predikáty a podobně. *Synoptické dotazy* naopak umožňují popisovat kolektivní chování pohybujících se objektů a reprezentují tak skupiny objektů, které vyhovují konkrétnímu chování z pohledu společných pohybů a vzájemných interakcí mezi objekty skupiny.



Obrázek 3.7: Příklad porovnání T-vzoru $(x_0, y_0) \xrightarrow{\alpha_1} (x_1, y_1)$ proti vstupní sekvenci (trajektorii), kde τ je časová tolerance a N funkce prostorové sousednosti (převzato z [7])

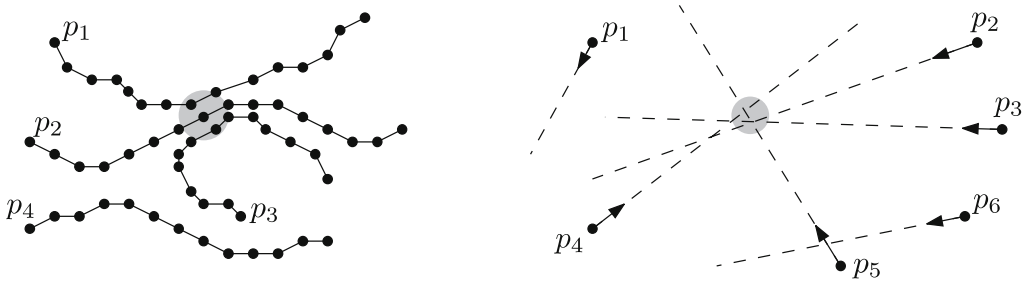
Na konceptu vzájemných pohybů (anglicky *relative motions*) objektů byl vyvinut rámec nazvaný REMO [15, 9], který specifikuje kolekci časoprostorových skupinových vzorů na základě podobnosti směrů a změn směrů pohybu objektů. Jedná se zejména o následující typy vzorů (uvažujme parametry $m > 1$, $r > 0$, případně $\tau > 0$):

- „Stádo“ (*flock*) – vzoru vyhovuje skupina nejméně m objektů, které se nacházejí v kruhové oblasti o poloměru r a pohybují se ve stejném směru. Příklad vzoru typu stádo je na obrázku 3.8 vlevo.
- „Vedení“ (*leadership*) – vzoru vyhovuje skupina nejméně m objektů, které se nacházejí v kruhové oblasti o poloměru r, pohybují se ve stejném směru a alespoň jeden z objektů udržuje tento směr po dobu nejméně τ časových kroků. Příklad vzoru typu vedení je na obrázku 3.8 vlevo.
- „Sbíhavost“ (*convergence*) – vzoru vyhovuje skupina nejméně m objektů, které dosáhnou stejné kruhové oblasti o poloměru r (za předpokladu udržení stejného směru). Příklad vzoru typu sbíhavost je na obrázku 3.8 vpravo.
- „Setkání“ (*encounter*) – vzoru vyhovuje skupina nejméně m objektů, které dosáhnou stejné kruhové oblasti o poloměru r současně (za předpokladu udržení stejného směru a stejné rychlosti).

3.4 Časoprostorové asociační vzory

Zatímco předchozí podkapitola se věnovala dolování frekventovaných vzorů z trajektorií pohybujících se objektů, v této části je pozornost zaměřena na dolování frekventovaných vzorů z časoprostorových dat, která jsou v podobě událostí.

Událost může reprezentovat výskyt nějakého rysu nebo jevu v určitém čase na určité prostorové pozici. Příkladem takové události je mlha, déšť, požár, zemětřesení, dopravní zácpa, havárie a mnoho dalších. Kromě toho, je možné databázi událostí získat odvozením



Obrázek 3.8: Příklad vzoru typu *stádo* a *vedení* s vedoucí trajektorií p_2 (vlevo) a vzoru typu *sbíhavost* pro trajektorie p_2, p_3, p_4, p_5 (vpravo) (převzato z [9])

různých informací z trajektorií pohybujících se objektů. Události pak mohou být ve formě časoprostorových predikátů, statických či dynamických agregátů a podobně. Příkladem takové události je odbočení, zvýšení rychlosti, zastavení a další.

Tato podkapitola obsahuje nejprve popis topologických vzorů, které jsou rozšířením prostorových kolokačních vzorů zavedením časových omezení. Účelem těchto vzorů je nalezení různých prostorových vztahů mezi událostmi v rámci časového intervalu. Dále jsou zde popsány prostorové sekvenční vzory, které lze vyhledávat začleněním prostorové informace do procesu dolování sekvenčních vzorů. Na závěr je nastíněna možnost transformovat časoprostorovou databázi na databázi stromů či grafů a převést tak problém dolování frekventovaných časoprostorových vzorů na problém vyhledávání frekventovaných podstromů či podgrafů.

3.4.1 Topologické vzory

Časoprostorové topologické vzory [13] (anglicky *topological patterns*) jsou rozšířením prostorových kolokačních vzorů (viz kapitola 2.5.4) zavedením temporálních omezení. Dolováním topologických vzorů je možné objevit vztahy jako je například častý výskyt různých událostí blízko sebe ve stejném časovém období. Z vyhledaných topologických vzorů je možné generovat *topologická pravidla*, podobně jako asociační pravidla z frekventovaných množin u transakčních databází.

Definice 3.2. Uvažujme časoprostorovou databázi. Nechť $F = \{f_1, f_2, \ldots, f_n\}$ je množina prostorových rysů a $O = \{o_1, o_2, \ldots, o_m\}$ je množina objektů v časoprostorové databázi. Každému objektu $o \in O$ odpovídá pozice v prostoru, časové razítko a prostorový rys $f \in F$. Uvažujme dále dva parametry: maximální prostorovou vzdálenost R a maximální časovou vzdálenost W. Dva objekty časoprostorové databáze jsou *prostorově blízké*, pokud pro jejich prostorovou vzdálenost *sdist* platí, že $sdist \leq R$. Dva objekty časoprostorové databáze jsou *časově blízké*, pokud pro jejich časovou vzdálenost *tdist* platí, že $tdist \leq W$.

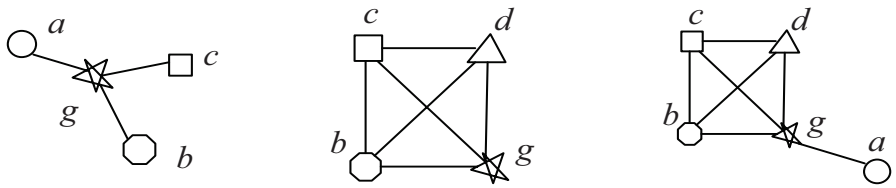
Definice 3.3. *Topologický vzor* $P = \{f_1, f_2, \ldots, f_k\}$ je množina k prostorových rysů. Platná instance topologického vzoru P je množina objektů časoprostorové databáze s odpovídajícími prostorovými rysy, přičemž musí platit, že všechny objekty instance vzoru jsou navzájem časově blízké. Podle prostorových vztahů mezi objekty platné instance vzoru je možné rozlišit tři základní typy topologických vzorů:

1. *Hvězda* – jedná se o vzor, pro který platí, že v platné instanci vzoru existuje objekt, který je prostorově blízký se všemi ostatními objekty instance, zatímco pro ostatní

objekty instance toto platit nemusí. Příklad topologického vzoru typu hvězda je na obrázku 3.9(a).

2. *Klika* – jedná se o vzor, pro který platí, že v platné instanci jsou každé dva objekty navzájem prostorově blízké. Jinými slovy graf vztahů blízkosti všech objektů instance vzoru tvoří kliku. Příklad topologického vzoru typu klika je na obrázku 3.9(b).

3. *Klika s hvězdou* – jedná se o vzor, který je kombinací předchozích vzorů. Takový vzor obsahuje podvzor typu klika, přičemž instance tohoto podvzoru obsahuje objekt, který je prostorově blízký i ke všem objektům instance mimo kliku. Příklad topologického vzoru typu klika s hvězdou je na obrázku 3.9(c).



(*a*) Star-like (b) Clique (c) Star-clique

Obrázek 3.9: Příklady topologických vzorů: (a) hvězda, (b) klika, (c) klika s hvězdou (převzato z [13])

Pro určení síly topologického vzoru jsou v [13] definovány dvě míry: podpora a poměr účasti rysu. *Podpora* vzoru (anglicky *support*) vyjadřuje počet instancí vzoru v databázi. Pro odpovídající topologické pravidlo lze dodefinovat i míru spolehlivosti pravidla. *Poměr účasti* rysu ve vzoru (anglicky *participation ratio*) zachycuje pravděpodobnost, že objekt s daným rysem v databázi je zároveň součástí některé instance vzoru. Poměr účasti rysu f ve vzoru P je definován jako podíl počtu objektů s rysem f v libovolné instanci vzoru P a počtu všech objektů s rysem f v databázi. Zobecněním poměru účasti na úroveň vzoru je pak definována míra *rozšíření* vzoru (anglicky *prevalence*) jako minimální poměr účasti ze všech rysů ve vzoru.

Pro dolování topologických vzorů v časoprostorových databázích je v [13] navržen efektivní a škálovatelný algoritmus *TopologyMiner*. Tento algoritmus využívá metody růstu vzoru, tedy prohledávání prostoru do hloubky, čímž se vyhýbá generování obrovského množství kandidátů. Algoritmus nejprve rozdělí časovou a prostorovou dimenzi do množiny menších disjunktních kostek. Jedním průchodem databáze pak vytvoří tzv. souhrnnou strukturu (anglicky *summary structure*), která zaznamenává informace o počtu instancí rysu v dané kostce, tedy v určité oblasti během jednoho časového okna. V další fázi pak na základě takto vytvořené struktury hledá všechny frekventované topologické vzory.

3.4.2 Prostorové sekvenční vzory

Dolování topologických vzorů, které jsou popsané v předchozí části, sice dokáže vyhledat zajímavé vztahy mezi událostmi během definovaného časového intervalu, nedokáže však odhalit vztahy mezi událostmi v různých časových intervalech. Proto je potřeba zavést nový typ vzoru, a to *prostorové sekvenční vzory* [13]. Pomocí těchto vzorů lze snadno popsat například, jak nějaká událost na jednom místě způsobí výskyt jiné událostí na druhém místě. Prostorové sekvenční vzory se podle přístupu k prostorové pozici ve vzoru dělí na dva typy:

1. Časoprostorové sekvenční vzory (v angličtině *flow patterns*) jsou vzory s absolutními souřadnicemi pozic.

2. Zobecněné časoprostorové vzory (v angličtině *generalized spatio-temporal patterns*) jsou vzory s relativními souřadnicemi pozic.

Pro potřeby prostorových sekvenčních vzorů je nutné nejprve rozdělit čas na disjunktní časová okna určité délky W. Dva libovolné časy t_1 a t_2 jsou blízké, pokud spadají do stejného časového okna, značeno $(t_1, t_2) \in W$. Prostor se rozdělí mřížkou na množinu disjunktních buněk, přičemž každá buňka reprezentuje jednu pozici v prostoru. Nechť je dále nad množinou všech pozic definována relace sousednosti R. Pozice l_1 a l_2 spolu sousedí, pokud $(l_1, l_2) \in R$.

Časoprostorové sekvenční vzory

Definice 3.4. *Událost* označená jako $e(l, t)$ určuje výskyt prostorového rysu e na pozici l v čase t. Dvě události $e_1(l_1, t_1)$ a $e_2(l_2, t_2)$, kde $t_1 \leq t_2$, jsou *blízké* právě tehdy, když $(l_1, l_2) \in R$ a $(t_1, t_2) \in W$. Dvě množiny událostí vyskytujících se ve stejném čase jsou blízké, pokud každá událost z jedné množiny je blízká se všemi událostmi druhé množiny. Množina událostí E_1 v čase t_1 *plyne* k množině událostí E_2 v čase t_2, kde $t_1 \leq t_2$, právě tehdy když E_1 je blízká s E_2 (značíme $E_1 \rightarrow E_2$). Množina událostí se nazývá *reflexivní*, pokud je blízká sama se sebou.

Definice 3.5. *Časoprostorový sekvenční vzor* [13] je podle času seřazená sekvence reflexivních množin událostí taková, že pro každé dvě po sobě jdoucí množiny událostí E_p v čase t_i a E_q v čase t_{i+1} platí, že E_p plyne k E_q ($E_p \rightarrow E_q$).

Časoprostorový sekvenční vzor je frekventovaný, jestliže databáze obsahuje nejméně $minsup$ různých výskytů vzoru, kde $minsup$ je minimální podpora.

Časoprostorové sekvenční vzory podle [13] splňují *Apriori vlastnost*, zmíněnou již několikrát v kapitole 2. Apriori vlastnost říká, že každý podvzor frekventovaného vzoru musí být také frekventovaný.

Pro dolování časoprostorových sekvenčních vzorů je v [13] navržen efektivní a škálovatelný algoritmus *FlowMiner*. Algoritmus nejprve jedním průchodem databáze vyhledá všechny frekventované události a na jejich základě pak všechny frekventované vzory délky 2. Všechny již nalezené frekventované vzory udržuje ve stromové datové struktuře nazvané souhrnný strom (anglicky *summary tree*). V další fázi algoritmus postupně hledá delší vzory kombinací přístupů generování kandidátů a prohledávání do hloubky. Při generování kandidátů na delší vzory se využívá temporálních omezení, která vyplývají z vyhledaných vzorů délky 2, a prostorových omezení, která jsou dána požadavkem na vlastnosti po sobě následujících množin událostí ve vzoru. Aplikací těchto omezení je počet generovaných kandidátů na frekventované prostorové sekvenční vzory výrazně snížen.

Zobecněné časoprostorové vzory

Časoprostorové sekvenční vzory, popsané výše, umožňují zachytit vývoj událostí v sousedících oblastech v čase. Nevýhodou těchto vzorů může být jejich silná závislost na předpokladu, že se takové události opakují na přesně stejných místech. Často však absolutní pozice výskytu události není z pohledu analytické úlohy důležitá a postačující je pozice relativní. V takovém případě je možné odhalit časoprostorové vztahy, které při použití absolutních

pozic mohou zůstat skryté. Prostorové sekvenční vzory s relativními pozicemi se nazývají zobecněné časoprostorové vzory [13].

Definice 3.6. *Událost* označená jako $e(x, y, t)$ popisuje výskyt prostorového rysu e na pozici $l = (x, y)$ v čase t. Dvě události $e_1(x_1, y_1, t_1)$ a $e_2(x_2, y_2, t_2)$ jsou *blízké* právě tehdy, když $((x_1, y_1), (x_2, y_2)) \in R$ a $(t_1, t_2) \in W$.

Zvolme referenční pozici $l_{ref} = (x_{ref}, y_{ref})$. Každá událost $e(x, y, t)$ je pak mapována na odpovídající relativní pozici výskytu vzhledem k referenční pozici jako $e(x-x_{ref}, y-y_{ref}, t)$. Množina takto mapovaných událostí vyskytujících se ve stejném čase t se označí jako $\hat{E} = \langle e_1(x_1 - x_{ref}, y_1 - y_{ref}), \ldots, e_n(x_n - x_{ref}, y_n - y_{ref})\rangle$. Dvě množiny mapovaných událostí $\hat{E}_p$ a $\hat{E}_q$ jsou *blízké*, pokud každá událost v $\hat{E}_p$ je blízká každé události v $\hat{E}_q$.

Definice 3.7. *Zobecněný časoprostorový vzor* $\hat{E}_1 \rightarrow \hat{E}_2 \rightarrow \cdots \rightarrow \hat{E}_m$ je sekvence množin mapovaných událostí taková, že každá množina v sekvenci je blízká se všemi ostatními množinami ze sekvence.

Zobecněný časoprostorový vzor je frekventovaný, pokud počet výskytů vzoru v různých časových oknech je roven nejméně *t-minsup* a v každém takovém časovém okně je počet různých výskytů vzoru roven nejméně *s-minsup*, kde *t-minsup* je minimální časová podpora a *s-minsup* je minimální prostorová podpora.

Pro dolování zobecněných časoprostorových vzorů je v [13] navržen efektivní a škálovatelný algoritmus *GenSTMiner*. Tento algoritmus vychází z přístupu algoritmu PrefixSpan k dolování sekvenčních vzorů. Je založen na metodě růstu vzoru, prohledává tedy prostor všech možných vzorů do hloubky, čímž se vyhýbá nutnosti generovat obrovské množství kandidátů na frekventované vzory.

3.4.3 Dolování časoprostorových vzorů ve stromech a grafech

Prostorové sekvenční vzory mohou být snadno modelovány jako stromy či grafy. Každý vrchol pak reprezentuje událost a každá hrana reprezentuje prostorový vztah, časový vztah nebo oba zároveň.

Pomocí vhodné modelovací techniky je možné každou časoprostorovou databázi převést na databázi stromů nebo grafů, přičemž grafový model umožňuje obecnější reprezentaci časoprostorových vzorů s libovolnými vztahy. Problém dolování frekventovaných časoprostorových vzorů se pak mění na problém hledání frekventovaných podstromů nebo podgrafů [13].

V [13] je pro dolování frekventovaných podstromů v databázi stromů navržen algoritmus *WTIMiner* a pro dolování frekventovaných podgrafů v databázi grafů algoritmus *PartMiner*. Oba tyto algoritmy jsou na základě experimentálního vyhodnocení označeny za efektivní a škálovatelné.

Kapitola 4

Detekce odlehlých trajektorií

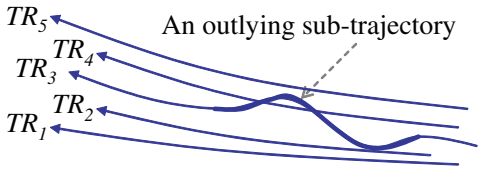
Automatická detekce neobvyklého nebo podezřelého chování pohybujících se objektů je významnou úlohou v oblasti analýzy časoprostorových dat. Obvykle se jedná o hledání *odlehlých trajektorií*, tedy takových pohybů, které se výrazně odlišují od většiny ostatních.

Mezi hlavní přístupy pro detekci odlehlých trajektorií patří metody založené na vzdálenosti a metody založené na klasifikaci. Další přístup pak zastupuje algoritmus TOP-EYE, který je založen na výpočtu míry odlehlosti trajektorie nad vytvořeným pravděpodobnostním modelem sledované oblasti. Popis těchto tří přístupů je obsahem následujících podkapitol.

Kromě zmíněných přístupů lze narazit i na některé další přístupy k dolování odlehlých trajektorií, jež jsou většinou přizpůsobeny konkrétní aplikační oblasti. Například [2] se zaměřuje na efektivní monitorování anomálií v proudu trajektorií pohybujících se objektů. V [18] je navržena metoda pro detekci časově odlehlých objektů v datech dopravního provozu s důrazem na historický trend podobnosti. Pro každý časový okamžik je pro každý segment silniční sítě určena jeho podobnost s ostatními segmenty a zaznamenána v jeho časovém vektoru podobnosti. Odlehlé objekty jsou pak detekovány na základě výrazných změn v těchto vektorech.

4.1 Metody založené na vzdálenosti

Při detekci odlehlých hodnot v datech pohybujících se objektů je často každá trajektorie uvažována jako jeden celek v podobě nějaké sumární charakteristiky (viz například [14]). Tento přístup však nedokáže odhalit neobvyklé chování objektu jen v některé části jeho trajektorie, obzvláště jsou-li trajektorie příliš dlouhé [16]. Z tohoto důvodu je na metody detekce odlehlých trajektorií kladen požadavek, aby uvažovaly každou trajektorii například jako množinu sub-trajektorií. Tímto přístupem lze identifikovat ty části trajektorie, které její odlehlost způsobují. Příklad odlehlé sub-trajektorie je ukázán na obrázku 4.1.



Obrázek 4.1: Příklad odlehlé sub-trajektorie (převzato z [16])

Obecně lze přístup metod pro detekci odlehlých trajektorií založených na vzdálenosti vyjádřit následující definicí [14]:

Definice 4.1. Objekt O v datové sadě T je $DB(p, D)$-*odlehlý* (*distance-based*), jestliže alespoň množství p objektů z T leží ve větší vzdálenosti než D od objektu O.

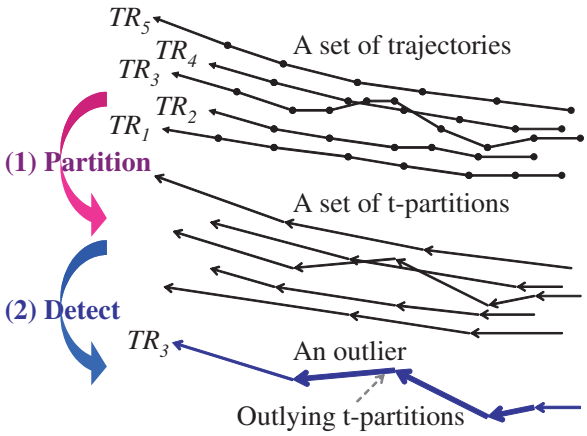
Hlavní rozdíl mezi jednotlivými metodami z této oblasti spočívá v použité vzdálenostní funkci. Například v [19] jsou jednotlivé sub-trajektorie porovnávány pomocí vzdálenostní funkce, která vychází z Hausdorffovy vzdálenosti.

Za účelem detekce odlehlých shluků trajektorií, které mohou reprezentovat například dopravní zácpu, je v [25] navržen algoritmus slučující přístup založený na vzdálenosti se shlukováním. Algoritmus uvažuje trajektorie jako množinu sub-trajektorií. Každá sub-trajektorie je reprezentována pomocí minimálního obalujícího kvádru (MBB), jehož dva rozměry jsou dány prostorovými souřadnicemi a třetí rozměr je určen časovou informací. Vzdálenost dvou trajektorií je pak počítána z objemu průniku jejich MBB. Nad trajektoriemi je následně provedeno shlukování modifikací metody DBSCAN. Pro každý shluk je vypočten stupeň jeho hustoty, který odlišuje normální shluky od odlehlých.

4.1.1 Algoritmus TRAOD

Významným zástupcem metod založených na vzdálenosti je algoritmus **TRAOD** [16]. Algoritmus sestává ze dvou fází:

1. Každá trajektorie je rozdělena na množinu sub-trajektorií.
2. Probíhá vlastní proces detekce odlehlých trajektorií kombinací přístupů založených na vzdálenosti a na hustotě.



Obrázek 4.2: Ilustrace procesu detekce odlehlých trajektorií pomocí algoritmu TRAOD (převzato z [16])

Jednotlivé kroky algoritmu TRAOD jsou znázorněny na obrázku 4.2. Pro dělení na sub-trajektorie je možné použít dvou přístupů. První způsob dělení spočívá v jednoduchém dělení po úsecích pevné délky, například každé dva následující body určují jednu sub-trajektorii. Za účelem zajištění kvality a efektivity je lepší dělení provádět ve dvou krocích:

nejprve hrubým způsobem a následně pak dělení patřičně zjemnit. Dělení podle jemné granularity probíhá jen tehdy, může-li být daný úsek ve výsledku identifikován jako odlehlý.

Při samotném procesu detekce je trajektorie považována za odlehlou, pokud obsahuje alespoň určitý počet odlehlých sub-trajektorií. Sub-trajektorie je odlehlá, pokud je blízká jen s určitým maximálním počtem trajektorií. Množina trajektorií, které jsou blízké ke konkrétní sub-trajektorii, je určena pomocí vzdálenostní funkce, která se skládá ze tří složek: kolmé, paralelní a úhlové vzdálenosti.

Za účelem omezení vlivu hustoty oblasti na posuzování odlehlosti sub-trajektorie je zaveden tzv. upravující koeficient, kterým je násoben zjištěný počet blízkých trajektorií.

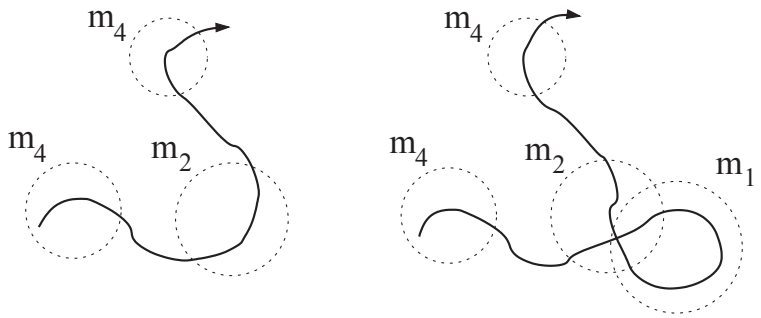
4.2 Metody založené na klasifikaci

Metody založené na klasifikaci jsou zástupcem metod s učením, kdy detekce odlehlých trajektorií probíhá podle modelu vstupních dat, který musí být vytvořen na základě trénovací datové sady. Nevýhodou těchto metod může být silná závislost výsledků na kvalitě trénovací množiny. Výhodou je oproti metodám bez učení časová efektivita procesu detekce.

Přístup kombinující shlukování a klasifikaci pomocí SVM klasifikátoru za účelem nalezení odlehlých trajektori je navržen v [21]. Dalším příkladem využití klasifikace v této oblasti je rámec ROAM, jehož stručný popis je obsahem následující podkapitoly.

4.2.1 Rámec ROAM

Metoda **ROAM**, navržená v [17], zastupuje ucelený rámec, jenž je určený pro detekci anomálií v datech pohybujících se objektů.



Obrázek 4.3: Ukázka dvou trajektorií po extrakci motifů (převzato z [17])

Celý proces prochází postupně třemi moduly:

1. Ze vstupních trajektorií jsou kombinací metody klouzavého okna a shlukování extrahovány společné vzory, tzv. *motify*. Každý motif reprezentuje nějaký typický vzor pohybu, například odbočení doprava, otočení a podobně. Každá trajektorie je pak reprezentována jako sekvence výskytů různých motifů. Příklad dvou trajektorií po extrakci motifů je zobrazen na obrázku 4.3. Trajektorie v pravé části obrázku má oproti levé trajektorii navíc výskyt motifu m_1.

2. Každému výskytu motifu je přiřazena množina odpovídajících atributů, kterými jsou pozice a čas výskytu, případně některé další jako rychlost pohybu a podobně. Množina

všech motifů tvoří hierarchický prostor rysů. Trajektorie se tak stává vektorem rysů v tomto novém prostoru.

3. Jednotlivé trajektorie jsou klasifikovány pomocí klasifikátoru založeného na pravidlech, který pracuje nad vytvořeným hierarchickým vektorem rysů.

4.3 Algoritmus TOP-EYE

Pro detekci odlehlých trajektorií již v průběhu jejich vývoje je v [6] navržena metoda nazvaná **TOP-EYE**. Tato metoda umožňuje, při vhodně nastavených vstupních parametrech, identifikovat odlehlé trajektorie v reálném čase a vyvarovat se velkého množství planých poplachů, které mohou být způsobené například šumem ve vstupních datech.

Algoritmus TOP-EYE pro každou trajektorii průběžně počítá míru odlehlosti, kterou postupně akumuluje s tím, že vliv předchozích měr odlehlosti je snižován prostřednictvím exponenciální funkce.

Sledovaný prostor je diskretizován do množiny malých buněk a nad takto transformovaným prostorem je vytvořen pravděpodobnostní model, který kromě informace o hustotě udržuje informaci o směru trajektorií v rámci každé buňky pomocí vektoru osmi hodnot, který vyjadřuje pravděpodobnosti pohybu touto buňkou v osmi různých směrech.

Pomocí tohoto algoritmu je možné rozlišit dva typy odlehlých trajektorií: podle směru a podle hustoty. Pokud se objekt sledovaným prostorem pohybuje oproti sumarizovaným směrům výrazně odlišnými směry, může být identifikován jako odlehlý podle směru. Pokud se objekt ve sledovaném prostoru pohybuje buňkami s nízkou hodnotou hustoty, kde hustota je reprezentována počtem trajektorií, které buňkou procházely, může být identifikován jako odlehlý podle hustoty.

Následující popis této metody vychází z [6].

4.3.1 Model sledované oblasti

Algoritmus TOP-EYE uvažuje sledovaný prostor jako pravidelnou mřížku malých buněk. Každá buňka je dále členěna do 8 směrových výsečí, kde každá výseč odpovídá úhlu $\pi/4$ (viz obrázek 4.4). Cílem tohoto dělení je sumarizovat směrovou informaci trajektorií procházejících konkrétní buňkou jako vektor osmi hodnot, které vyjadřují pravděpodobnosti pohybu objektu v osmi různých směrech. Buňka g je pak kromě informace o hustotě reprezentována směrovým vektorem:

$$g = (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8) \tag{4.1}$$

kde p_i je frekvence trajektorií, které se pohybovaly skrz buňku g ve směru výseče i.

Tento pravděpodobnostní model musí být před samotnou detekcí odlehlých trajektorií vytvořen, tato metoda tedy vyžaduje trénovací množinu trajektorií. Vzhledem k tomu, že vytvářený model je pravděpodobnostního charakteru, je však možné pro vytvoření modelu použít stejná data, nad kterými bude prováděna detekce odlehlých trajektorií. Analýzou každé trajektorie z této datové množiny jsou navýšeny odpovídající hodnoty směrového vektoru a hodnota hustoty u buněk, kterými daná trajektorie prochází.

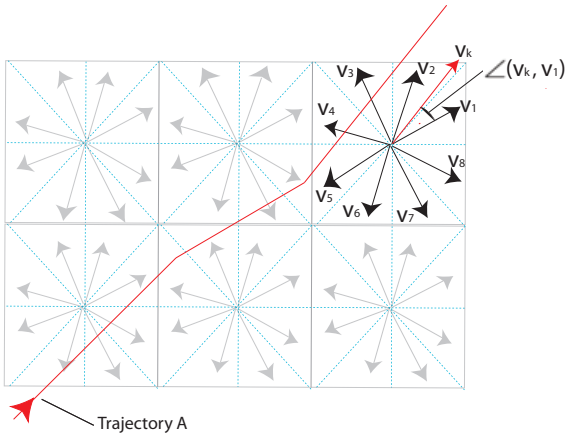
4.3.2 Výpočet míry odlehlosti

Pro každou novou trajektorii pak algoritmus průběžně počítá *okamžitou míru odlehlosti* (anglicky *instant outlying score*). Míru odlehlosti je možné sledovat podle směru trajektorie nebo podle hustoty buněk, kterými prochází.

Definice 4.2. Reprezentujme novou trajektorii směrovým vektorem, například $(q_1, \ldots, q_K)$, kde K je počet směrů pohybu uvnitř sledované buňky a každé $q_k = 1/K$. *Míra odlehlosti podle směru* je pak dána vzdáleností mezi sumarizovaným směrovým vektorem sledované buňky a směrovým vektorem nové trajektorie:

$$OScoreDir = 1 - \sum_{k=1}^{K} q_k \sum_{i=1}^{8} p_i \cdot \cos \angle(v_k, v_i) \tag{4.2}$$

kde $\cos \angle(v_k, v_i)$ je kosinus úhlu mezi směrem pohybu k nové trajektorie a reprezentativním směrem výseče i, viz obrázek 4.4.



Obrázek 4.4: Ilustrace výpočtu směrové míry odlehlosti (převzato z [6])

Definice 4.3. *Míra odlehlosti podle hustoty* je definována následovně:

$$OScoreDen = \begin{cases} s & \text{jestliže } density < \tau \\ 0 & \text{jinak} \end{cases} \tag{4.3}$$

kde *density* je sumarizovaná hustota sledované buňky, s a τ jsou uživatelské parametry.

4.3.3 Vyvíjející se míra odlehlosti

Hlavní myšlenkou algoritmu TOP-EYE je zachytit pomocí *vyvíjející se míry odlehlosti* (anglicky *evolving outlying score*) průběžný vývoj okamžité míry odlehlosti trajektorie v kombinaci s její historickou odlehlostí. Vliv předchozích měr odlehlosti trajektorie je snižován prostřednictvím exponenciální funkce $\exp(-\lambda\Delta t)$ v závislosti na čase. K takto upravené hodnotě historické odlehlosti je vždy přičtena nově vypočítaná okamžitá míra odlehlosti, čímž vzniká nová hodnota vyvíjející se míry odlehlosti.

Definice 4.4. Uvažujme, že S_{t_i} je okamžitá míra odlehlosti získaná pomocí rovnice (4.2) nebo rovnice (4.3). *Vyvíjející se míra odlehlosti* v časovém okamžiku t_i je pak dána jako

$$S_{t_i}^{\sum} = S_{t_i} + S_{t_{i-1}} \cdot \exp(-\lambda\Delta t_{i-1}) + S_{t_{i-2}} \cdot \exp(-\lambda\Delta t_{i-2}) + \cdots + S_{t_0} \cdot \exp(-\lambda\Delta t_0) \tag{4.4}$$

kde Δt_{i-k} je časový rozdíl mezi časy t_i a t_{i-k}. λ je uživatelský parametr, který ovlivňuje vliv historické odlehlosti.

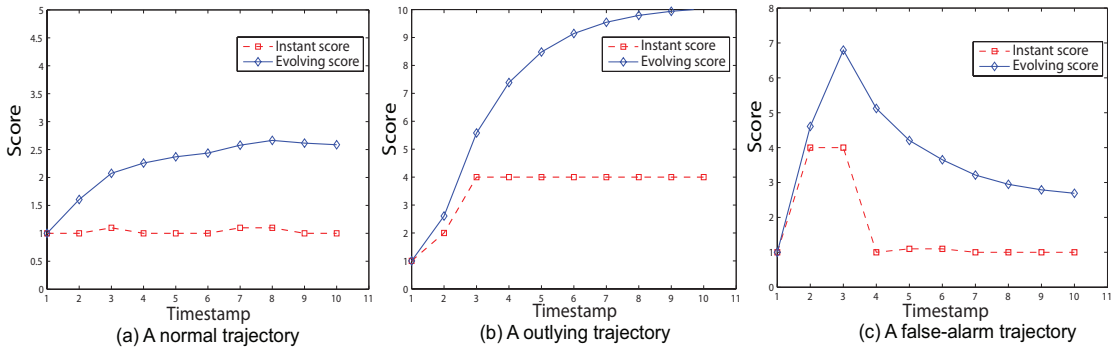
Jestliže je rozdíl mezi každými dvěma následujícími časovými okamžiky roven Δt, může být rovnice (4.4) upravena následovně:

$$S_{t_i}^{\sum} = \sum_{k=0}^{i} S_{t_k} \exp(-(i-k)\lambda\Delta t) \tag{4.5}$$

4.3.4 Detekce odlehlých trajektorií

Takto definovaná vyvíjející se míra odlehlosti umožňuje zachytit odlehlou trajektorii již v rané fázi vývoje. Zároveň může být odolná vůči náhlému nárůstu odlehlosti, který je způsoben například náhodným šumem a mohl by způsobit planý poplach.

Detekce odlehlých trajektorií pomocí algoritmu TOP-EYE probíhá tak, že pro každou vstupní trajektorii je vypočten vývoj míry odlehlosti. Jakmile hodnota vyvíjející se míry odlehlosti vystoupá nad uživatelem definovanou prahovou hodnotu odlehlosti, je trajektorie označena jako odlehlá.



Obrázek 4.5: Ilustrace využití vyvíjející se míry odlehlosti (*evolving score*) ve srovnání s okamžitou mírou odlehlosti (*instant score*): (a) pro normální trajektorii, (b) pro odlehlou trajektorii, (c) pro planý poplach (převzato z [6])

Obrázek 4.5 ilustruje tři různé případy průběhu vyvíjející se míry odlehlosti trajektorie v závislosti na čase ve srovnání s průběhem odpovídající okamžité míry odlehlosti. Zatímco na obrázku 4.5(a) je zobrazen průběh míry odlehlosti pro normální trajektorii, obrázek 4.5(b) zachycuje průběh pro trajektorii, která se vyvíjí jako odlehlá a může reprezentovat neobvyklé chování. Ačkoliv okamžitá míra odlehlosti zůstává konstantní, vyvíjející se míra odlehlosti se postupně zvyšuje. Na obrázku 4.5(c) je zaznamenán příklad, kdy se trajektorie vyvíjí jako odlehlá pouze po krátký časový interval a ve zbytku svého průběhu odpovídá normální trajektorii. V případě vhodně nastavené prahové hodnoty odlehlosti (zde například 7) nestihne vyvíjející se míra odlehlosti převýšit tuto prahovou hodnotu a trajektorie nebude identifikována jako odlehlá. Pokud by k detekci odlehlých trajektorií byla využita pouze okamžitá míra odlehlosti, nebylo by možné vyvarovat se takových planých poplachů.

Kapitola 5

Implementace algoritmu TOP-EYE

Tato kapitola popisuje postup při implementaci algoritmu TOP-EYE, který slouží pro detekci odlehlých trajektorií v datech pohybujících se objektů. Popis tohoto algoritmu lze najít v kapitole 4.3.

Nejprve je popsáno, jakým způsobem je reprezentován přístup ke zdroji dat, následně jsou postupně rozebrány způsoby implementace jednotlivých kroků algoritmu. Pro implementaci tohoto algoritmu byl zvolen programovací jazyk Java.

Na obrázku 5.1 je zobrazen navržený diagram tříd. Třídy jsou rozčleněny do dvou balíčků. Balíček `data` obsahuje rozhraní a třídy, které reprezentují zdrojová data pohybujících se objektů a umožňují přístup k nim. Samotný algoritmus je pak implementován prostřednictvím tříd balíčku `topeye`.

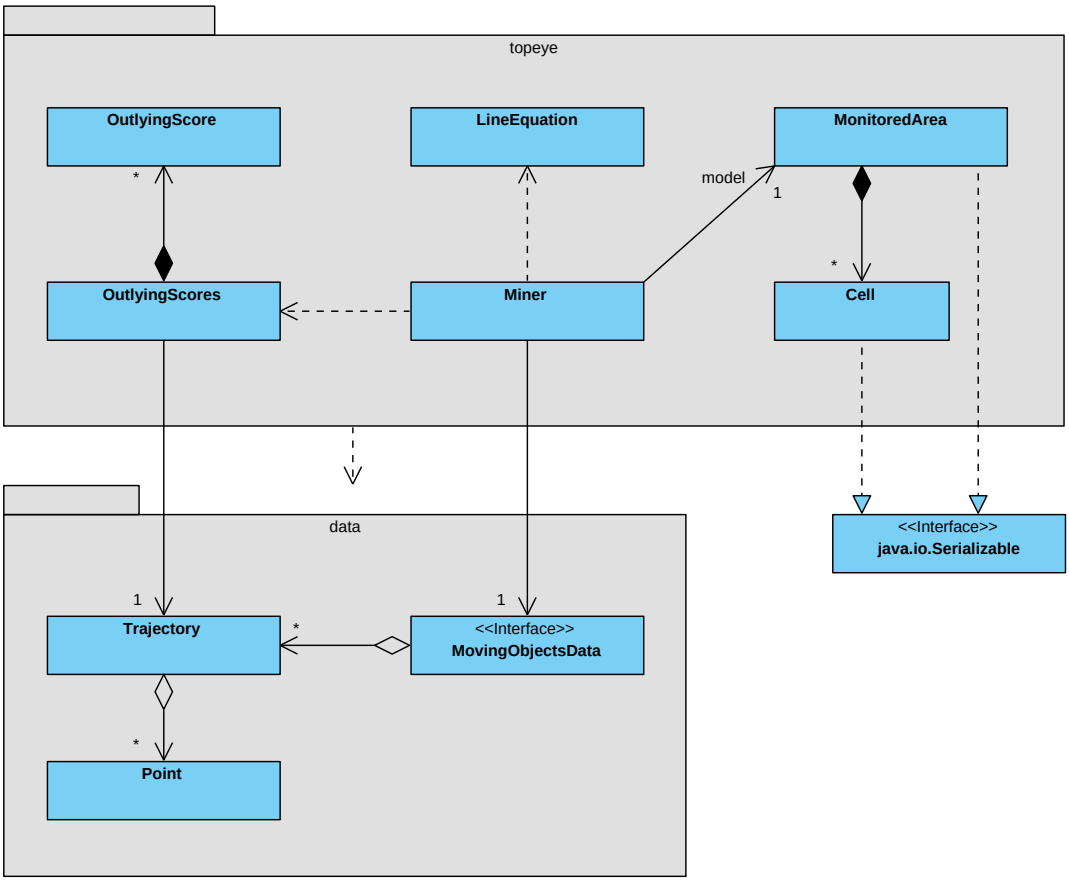
5.1 Reprezentace dat

Při návrhu způsobu přístupu k datům, se kterými pracuje algoritmus TOP-EYE, byla hlavním cílem jejich abstraktní reprezentace. Účelem tohoto požadavku je jednoduše umožnit práci s různými typy dat pohybujících se objektů z různých zdrojů. Definici tříd balíčku `data` a vztahy mezi nimi zachycuje diagram na obrázku 5.2.

Jako přístupový bod k datové sadě není požadován objekt konkrétní třídy, ale objekt libovolné třídy, která implementuje rozhraní `MovingObjectsData`. Toto rozhraní definuje sedm metod. Kromě metod pro zjištění rozsahu prostorových souřadnic sledované oblasti a zjištění celkového počtu trajektorií definuje metodu `readTrajectories()`, která by měla načíst ze zdroje dat všechny trajektorie pohybujících se objektů. Pro přístup k jednotlivým trajektoriím je určena metoda `getNextTrajectory()`, jejímž postupným voláním lze získat všechny načtené trajektorie reprezentované objekty třídy `Trajectory`.

Toto oddělení metod pro načtení dat a přístup k nim bylo zvoleno z důvodu zajištění větší flexibility při implementaci konkrétní třídy. Je tak poskytnuta volba, zda načíst z databáze všechna data včetně průběhů trajektorií najednou, nebo nejprve načíst sekvenci všech trajektorií a informace o průběhu konkrétní trajektorie načíst až při požadavku na ni.

Konkrétní trajektorii reprezentuje třída `Trajectory`. Pro vytvoření nového objektu této třídy je nabízen konstruktor, kterému musí být předán číselný identifikátor trajektorie. Pokud by bylo zapotřebí reprezentovat trajektorie, které jsou identifikovány jiným způsobem (například kombinací více proměnných nebo jiným datovým typem), je vhodné děděním této třídy vytvořit třídu novou a přepsat metodu `idToString()`, která vrací řetězcovou identifikaci trajektorie a je určena především k tomuto účelu. Proto by při prezentaci tra-



Obrázek 5.1: Diagram tříd algoritmu TOP-EYE

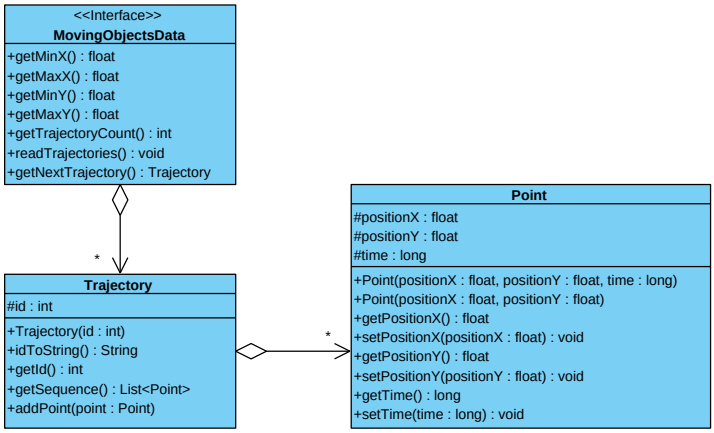
jektorií (například v uživatelském rozhraní) měla být používána právě tato metoda.

Trajektorie je uvažována jako posloupnost prostorových pozic objektu pohybujícího se v čase. Pro reprezentaci jednoho konkrétního výskytu pohybujícího se objektu slouží třída `Point`. Objekt této třídy obsahuje informaci o pozici ve dvoudimenzionálním prostoru a o čase. Pro uložení času byl z důvodu obecnosti zvolen datový typ `long`. Vzhledem k tomu, že časová složka může být ve zdrojových datech určena pouze jejich pořadím, je nabízen i konstruktor, který předání informace o čase nevyžaduje. Čas je v takovém případě implicitně nastaven na zápornou hodnotu.

Pro postupné vložení jednotlivých pozic trajektorie je určena metoda `addPoint()`. Pro získání kompletní posloupnosti pozic slouží metoda `getSequence()`.

5.2 Reprezentace algoritmu

Diagram na obrázku 5.3 zobrazuje definici tříd balíčku `topeye` a vztahy mezi nimi. Ústředním prvkem je třída `Miner`, která slouží jako rozhraní celého algoritmu TOP-EYE a realizuje většinu jeho funkčnosti. Obsahuje metody pro práci s modelem sledované oblasti, pro výpočet měr odlehlosti pro konkrétní trajektorii, pro detekci odlehlých trajektorií a pro nastavení parametrů dolovací úlohy. Pro vytvoření instance této třídy je poskytnut konstruktor, který požaduje předání objektu typu `MovingObjectsData`, jenž reprezentuje zdroj dat, nad nimiž má algoritmus pracovat.



Obrázek 5.2: Diagram tříd balíčku `data`

Algoritmus TOP-EYE požaduje nastavení pěti atributů, které reprezentují uživatelské parametry dolovací úlohy:

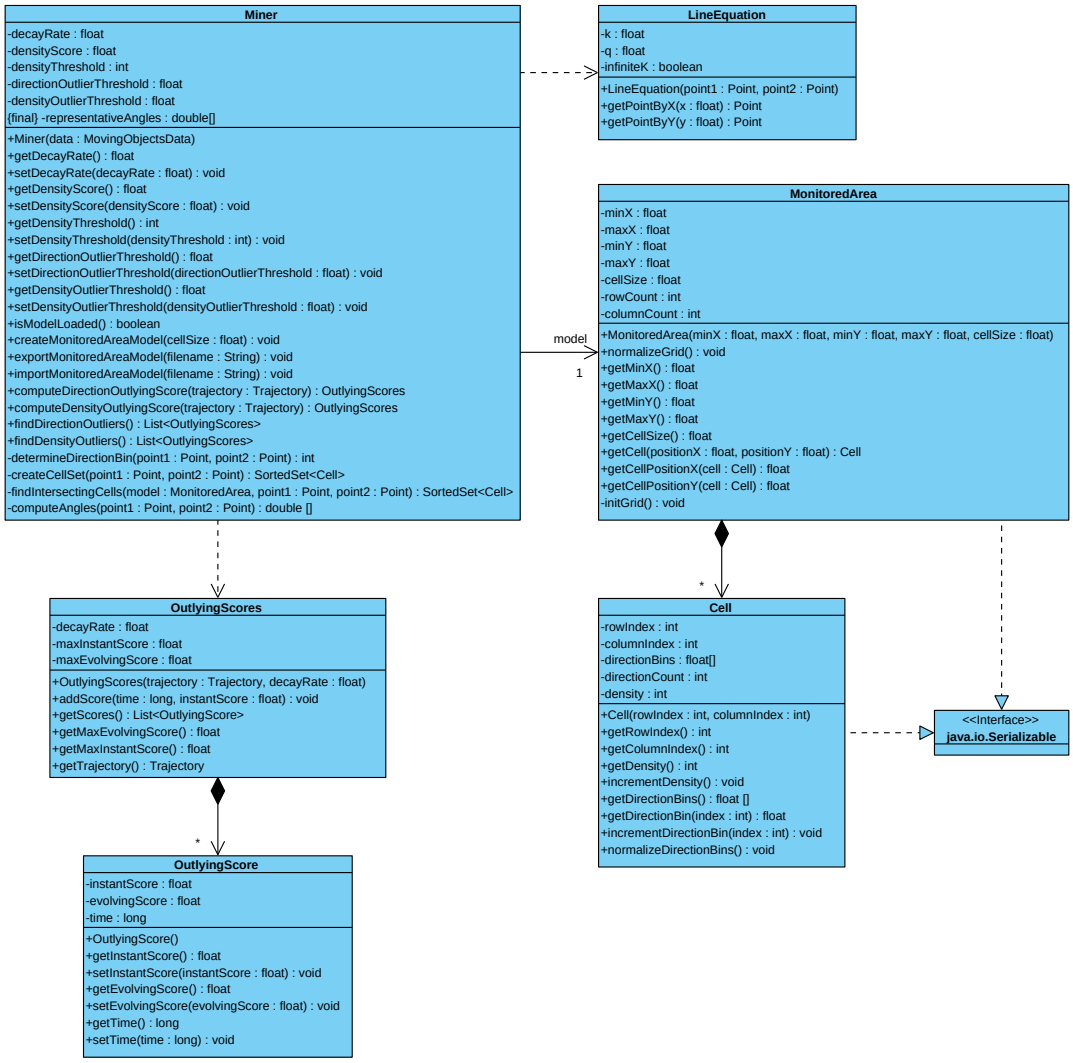
- `decayRate` – odpovídá parametru λ z popisu algoritmu a účastní se výpočtu vyvíjející se míry odlehlosti, kde pomáhá regulovat vliv historické odlehlosti.
- `densityScore` – odpovídá parametru s z popisu algoritmu, viz rovnice (4.3). Vyjadřuje míru odlehlosti na základě hustoty v buňce s nízkou hustotou.
- `densityThreshold` – odpovídá parametru τ z popisu algoritmu, viz rovnice (4.3). Vyjadřuje prahovou hodnotu hustoty buňky pro rozlišení buněk s nízkou a normální hustotou.
- `directionOutlierThreshold` – určuje prahovou hodnotu míry odlehlosti pro rozlišení odlehlých trajektorií od normálních při detekci na základě směru.
- `densityOutlierThreshold` – určuje prahovou hodnotu míry odlehlosti pro rozlišení odlehlých trajektorií od normálních při detekci na základě hustoty.

Třídy `MonitoredArea` a `Cell` reprezentují pravděpodobnostní model sledované oblasti, který slouží jako základ pro výpočet míry odlehlosti jednotlivých trajektorií. Metody třídy `Miner`, které s modelem sledované oblasti pracují, mohou vygenerovat výjimku typu `ModelNotFoundException`, pokud zatím nebyl žádný model vytvořen nebo načten.

Výsledkem výpočtu míry odlehlosti pro jednu konkrétní trajektorii je instance třídy `OutlyingScores`. Obsahuje posloupnost jednotlivých měr odlehlosti reprezentovaných třídou `OutlyingScore` a vyjadřuje tak vývoj okamžité a vyvíjející se míry odlehlosti v průběhu dané trajektorie.

5.3 Vytváření modelu sledované oblasti

Jak je uvedeno v popisu algoritmu, před vlastní detekcí odlehlých trajektorií je nutné vytvořit pravděpodobnostní model sledované oblasti, který reprezentuje třída `MonitoredArea`. Sledovaná oblast je uvažována jako mřížka buněk, přičemž každé buňce odpovídá jeden objekt třídy `Cell`.



Obrázek 5.3: Diagram tříd balíčku `topeye`

Za vytvoření modelu sledované oblasti zodpovídá třída `Miner`, kde je k tomu určena metoda `createMonitoredAreaModel()`. Kromě toho umožňuje již vytvořený model serializovat a exportovat do souboru a uchovat ho tak pro příští použití. K tomuto účelu slouží metoda `exportMonitoredAreaModel()`. Při příštím použití stačí pouze provést import ze souboru a následnou deserializaci, což má za úkol metoda `importMonitoredAreaModel()`. Při importu modelu probíhá ověření korespondence s nastaveným zdrojem dat porovnáním rozsahů prostorových souřadnic. Jestliže některá z ověřovaných souřadnic nesouhlasí, je vygenerována výjimka typu `ModelNotFoundException`.

Algoritmus pro vytvoření modelu sledované oblasti zapsaný v pseudokódu obsahuje Algoritmus 5.1. Výstupem algoritmu je model sledované oblasti, který však odpovídající metoda `createMonitoredAreaModel()` nevrací, ale výsledek ukládá do atributu objektu třídy `Miner`.

Prvním krokem je vytvoření a inicializace objektu třídy `MonitoredArea`. Konstruktor této třídy požaduje předání údajů o rozměru sledovaného prostoru a velikost jedné buňky vytvářené mřížky. Po vytvoření mřížky s odpovídajícím počtem buněk je každá buňka

Algoritmus 5.1: Vytvoření modelu sledované oblasti

Vstup:
databáze trajektorií D
velikost buňky v
rozměry sledované oblasti x_{min}, x_{max}, y_{min}, y_{max}

Výstup:
model sledované oblasti M

M = inicializujModel(x_{min}, x_{max}, y_{min}, y_{max}, v)
foreach trajektorie $t \in D$ **do**
  zvyš hustotu první buňky, kterou trajektorie prochází
  foreach dva po sobě jdoucí body $bod_1 \in t$, $bod_2 \in t$ **do**
    s = určiSměrovouVýseč(bod_1, bod_2)
    zvyš frekvenci směrové výseče s buňky bodu bod_1
    B = najdiProtínajícíBuňky(bod_1, bod_2)
    foreach $b \in B$ **do**
      zvyš hustotu buňky b
      zvyš frekvenci směrové výseče s buňky b
    end
  end
end
normalizujBuňky(M)
return M

inicializována do stavu, kdy její hustota i směrový vektor obsahují nulové hodnoty. Přístup k jednotlivým buňkám mřížky je umožněn pomocí metody `getCell()`, která vrací objekt reprezentující buňku na základě předaných prostorových souřadnic bodu, jenž v dané buňce leží.

Ústřední částí algoritmu je cyklus, který postupně z datového zdroje načítá jednotlivé trajektorie a zkoumá jejich průběh. Nejprve je navýšena hustota buňky, v níž má trajektorie svůj začátek. Poté je postupně analyzován každý úsek vymezený dvěma po sobě následujícími body trajektorie.

Každý takový úsek musí být zařazen do jedné z osmi směrových výsečí, a to podle toho, jaký směr má vektor určený počátečním a koncovým bodem úseku. Tento krok je implementován privátní metodou `determineDirectionBin()`. Při určování směrové výseče je počáteční bod úseku považován za počátek souřadného systému. Vzhledem k tomu, že každá směrová výseč odpovídá úhlu $\pi/4$, jedná se v podstatě o hledání oktantu, v němž daný vektor leží. Porovnáním jednotlivých souřadnic obou bodů je nejprve zjištěno, ve kterém kvadrantu se vektor nachází. Příslušný oktant, a tedy směrová výseč, je pak určen podle toho, zda vektor roste rychleji ve směru vodorovné nebo svislé osy.

Dalším krokem je pak navýšení frekvence odpovídající směrové výseče směrového vektoru u všech buněk, kterými analyzovaný úsek trajektorie prochází. U všech takových buněk, kromě buňky, v níž leží počáteční bod úseku, je zároveň navýšena i hodnota hustoty. Pro navýšení hustoty a frekvence směrové výseče slouží metody `incrementDensity()` a `incrementDirectionBin()` třídy `Cell`.

Posledním krokem při vytváření nového modelu sledované oblasti je normalizace směrových vektorů všech buněk mřížky. Směrový vektor má vyjadřovat pravděpodobnosti pohybu objektu v různých směrech. Po aplikaci výše popsaného postupu ale vyjadřuje počty trajektorií, které se v daném směru pohybovaly. Proto je potřeba provést přepočet těchto frekvencí na hodnoty pravděpodobnosti v rozmezí $\langle 0, 1 \rangle$ tak, aby suma všech hodnot směrového vektoru byla rovna 1 (s výjimkou buněk, kterými neprocházela žádná trajektorie). Pro provedení této normalizace slouží metoda `normalizeGrid()` třídy `MonitoredArea`. Tato metoda pro každou buňku mřížky volá její metodu `normalizeDirectionBins()`, která popsaný přepočet provádí.

5.3.1 Hledání buněk protínaných úsekem trajektorie

Leží-li počáteční a koncový bod úseku trajektorie v různých buňkách mřížky, musí být nalezena množina všech buněk, které úsek trajektorie protíná. Ve třídě `Miner` je tento krok implementován privátní metodou `findIntersectingCells()`. Algoritmus prochází nejprve přes horizontální souřadnice hranic sloupců mřížky, které leží mezi počátečním a koncovým bodem daného úseku trajektorie. Následně prochází přes vertikální souřadnice hranic řádků mřížky, které leží mezi počátečním a koncovým bodem úseku. Při zjišťování těchto souřadnic je využíváno metod `getCellPositionX()` a `getCellPositionY()` třídy `MonitoredArea`, které vrací souřadnice hranic předané buňky. Pro každou takovou souřadnici je vypočten průsečík s úsekem trajektorie a buňka, v níž tento průsečík leží, je přidána do hledané množiny buněk.

Analyzovaný úsek trajektorie je při výpočtu trajektorie reprezentován rovnicí přímky, kterou zastupuje objekt třídy `LineEquation`. Tato třída slouží pro vyjádření přímky zadané dvěma body rovnicí ve směrnicovém tvaru. Kromě toho umožňuje vyjádřit i přímky kolmé na vodorovnou osu, což rovnice ve směrnicovém tvaru neumožňuje.

Při přechodu trajektorie z jedné buňky do druhé dojde ke změně alespoň jedné ze souřadnic buňky v mřížce (změní se buď číslo sloupce mřížky, číslo řádku mřížky, nebo oboje). Případ, kdy dojde ke změně obou souřadnic buňky je ošetřen použitím kolekce typu množina, konkrétně seřazená množina (v Javě kolekce typu `SortedSet`). V množině se každý prvek může nacházet nejvýše jednou a opakované vkládání stejné buňky tedy obsah množiny nezmění. Použití seřazené množiny zde bylo zvoleno z důvodu přidávání buněk do množiny v jiném pořadí, než v jakém jimi trajektorie prochází. Při vytváření modelu oblasti sice není toto uspořádání požadováno, ale důležité je při výpočtu míry odlehlosti, kdy je nalezení množiny všech buněk, kterými úsek trajektorie prochází, také zapotřebí.

5.4 Výpočet míry odlehlosti

Jak je uvedeno již dříve, algoritmus TOP-EYE rozlišuje dva různé typy odlehlých trajektorií podle toho, jaká míra odlehlosti je použita pro jejich detekci. Míru odlehlosti lze počítat buď podle směru trajektorie, nebo podle hustoty buněk, kterými trajektorie prochází (viz kapitola 4.3).

5.4.1 Míra odlehlosti podle směru

Algoritmus pro výpočet míry odlehlosti podle směru trajektorie zapsaný v pseudokódu obsahuje Algoritmus 5.2. Implementován je metodou `computeDirectionOutlyingScore()` třídy `Miner`.

Algoritmus 5.2: Výpočet míry odlehlosti podle směru

Vstup:
trajektorie t
parametr vlivu historické odlehlosti λ
model sledované oblasti M

Výstup:
posloupnost měr odlehlosti O

O = inicializujPosloupnostMěrOdlehlosti(t, λ)
V = vektor reprezentativních úhlů směrových výsečí
$suma = 0$
$pocet = 0$

foreach dva po sobě jdoucí body $bod_1 \in t$, $bod_2 \in t$ **do**
  α = úhel vektoru daného body bod_1 a bod_2
  P = směrový vektor buňky bodu bod_1
  $suma = suma + \sum_{i=1}^{8} P_i \cdot \cos \angle(\alpha, V_i)$
  $pocet = pocet + 1$
  if bod_1 a bod_2 neleží ve stejné buňce **then**
    $o = 1 - suma/pocet$
    přidej o do posloupnosti O
    B = najdiProtínajícíBuňky(bod_1, bod_2)
    foreach $b \in B$ kromě poslední buňky **do**
      P = směrový vektor buňky b
      $o = 1 - \sum_{i=1}^{8} P_i \cdot \cos \angle(\alpha, V_i)$
      přidej o do posloupnosti O
    end
    P = směrový vektor buňky bodu bod_2
    $suma = \sum_{i=1}^{8} P_i \cdot \cos \angle(\alpha, V_i)$
    $pocet = 1$
  end
end

$o = 1 - suma/pocet$
přidej o do posloupnosti O

return O

Postupně je analyzován každý úsek vymezený dvěma po sobě následujícími body trajektorie. Pro každý takový úsek trajektorie je nejprve vypočten úhel, který svírá vektor určený počátečním a koncovým bodem úseku s kladnou částí vodorovné osy. Výpočet tohoto úhlu je prováděn uvnitř privátní metody `computeAngles()`, která kromě toho vypočte rozdíly mezi tímto úhlem a reprezentativními úhly jednotlivých směrových výsečí.

Dokud počáteční i koncový bod úseku leží ve stejné buňce, dochází k průběžnému výpočtu podobnosti trajektorie s modelem sledované oblasti srovnáváním směru právě analyzovaného úseku se sumarizovanou směrovou informací buňky, kterou vyjadřuje její směrový vektor a vektor reprezentativních úhlů jednotlivých směrových výsečí. Jednotlivé hodnoty

směrového vektoru buňky zpřístupňuje metoda `getDirectionBin()` třídy `Cell`.

Jakmile je detekován přechod trajektorie z jedné buňky do druhé, dojde na základě dosud vypočítané míry podobnosti pro poslední buňku k výpočtu okamžité míry odlehlosti v souladu s rovnicí (4.2) z kapitoly 4.3.2. Tato hodnota okamžité míry odlehlosti je přidána do posloupnosti, která odráží vývoj odlehlosti trajektorie (viz podkapitola 5.4.3). Dále je nutné vyhledat všechny buňky, kterými daný úsek trajektorie prochází, vypočítat pro ně jednorázově okamžitou míru odlehlosti a tu přidat do tvořené posloupnosti. Toto se netýká buňky, v níž se nachází koncový bod úseku, protože touto buňkou se trajektorie pohybuje ve více směrech a výpočet míry odlehlosti pro tuto buňku tedy bude pokračovat při analýze následujícího úseku.

5.4.2 Míra odlehlosti podle hustoty

Algoritmus pro výpočet míry odlehlosti podle hustoty buněk, kterými trajektorie prochází, zapsaný v pseudokódu obsahuje Algoritmus 5.3. Implementován je ve třídě `Miner` metodou `computeDensityOutlyingScore()`.

Algoritmus 5.3: Výpočet míry odlehlosti podle hustoty

Vstup:
trajektorie t
míra odlehlosti buňky s nízkou hustotou s
prahová hodnota hustoty τ
parametr vlivu historické odlehlosti λ
model sledované oblasti M

Výstup:
posloupnost měr odlehlosti O

O = inicializujPosloupnostMěrOdlehlosti(t, λ)
h = hustota buňky, ve které se nachází počáteční bod trajektorie t
if $h < \tau$ **then** $o = s$
else $o = 0$
přidej o do posloupnosti O
foreach dva po sobě jdoucí body $bod_1 \in t$, $bod_2 \in t$ **do**
  B = najdiProtínajícíBuňky(bod_1, bod_2)
  foreach $b \in B$ **do**
    h = hustota buňky b
    if $h < \tau$ **then** $o = s$
    else $o = 0$
    přidej o do posloupnosti O
  end
end
return O

Prvním krokem je výpočet okamžité míry odlehlosti pro buňku, v níž se nachází počáteční bod trajektorie, přesně v souladu s rovnicí (4.3) z kapitoly 4.3.2. Takto spočtená hodnota je přidána do posloupnosti zastupující vývoj míry odlehlosti analyzované trajek-

torie (viz podkapitola 5.4.3). Hustotu buňky, jejíž hodnota je potřeba při určování míry odlehlosti, zpřístupňuje metoda `getDensity()` třídy `Cell`.

Následně jsou postupně zkoumány všechny úseky vymezené dvěma po sobě následujícími body trajektorie. Pro každý takový úsek jsou vyhledány všechny buňky, kterými tento úsek prochází. Pro každou buňku je vypočítána okamžitá míra odlehlosti podle hustoty buňky a ta je přidána do tvořené posloupnosti.

5.4.3 Vyvíjející se míra odlehlosti

Vývoj míry odlehlosti trajektorie je reprezentován třídou `OutlyingScores`. Instance této třídy zapouzdřuje posloupnost měr odlehlosti jedné konkrétní trajektorie. Prvky této posloupnosti jsou objekty třídy `OutlyingScore` a obsahují hodnotu okamžité i vyvíjející se míry odlehlosti. Počet prvků této posloupnosti odpovídá počtu buněk, kterými daná trajektorie prochází.

Konstruktor třídy `OutlyingScores` vyžaduje předání trajektorie, jejíž míru odlehlosti má vytvářený objekt reprezentovat, a hodnoty parametru upravujícího vliv předchozích měr odlehlosti při výpočtu vyvíjející se míry. Vytvoření nové instance této třídy odpovídá kroku inicializace posloupnosti měr odlehlosti v algoritmech 5.2 a 5.3. Pro přidávání nových hodnot míry odlehlosti je určena metoda `addScore()`, pro získání přístupu k celé posloupnosti měr odlehlosti slouží metoda `getScores()`.

Výpočet hodnot vyvíjející se míry odlehlosti probíhá průběžně při každém přidávání nové hodnoty okamžité míry, a to v souladu s rovnicí (4.4) z kapitoly 4.3.3. Poslední uložená hodnota vyvíjející se míry odlehlosti je nejprve upravena prostřednictvím exponenciální funkce času a následně přičtena k nové hodnotě okamžité míry odlehlosti. Výsledná hodnota je pak přidána na konec posloupnosti. Kromě toho jsou přidávané hodnoty okamžité i vyvíjející se míry odlehlosti srovnány se zatím nejvýše dosaženými hodnotami, které jsou případně pozměněny.

Vliv historické odlehlosti trajektorie je dán kromě odpovídajícího uživatelského parametru také velikostí rozdílu mezi posledními dvěma časovými okamžiky. Jestliže jsou data trajektorií z konkrétního zdroje uložena bez explicitního určení času (rozhoduje tedy pouze pořadí dat), je rozdíl mezi každými dvěma následujícími časovými okamžiky uvažován jako 1.

5.5 Detekce odlehlých trajektorií

Před detekcí odlehlých trajektorií musí být vytvořen model sledované oblasti. Při vlastní detekci odlehlých trajektorií je pak prvním krokem volba toho, která ze dvou nabízených měr odlehlosti má být počítána. Pro detekci odlehlých trajektorií podle směru trajektorie je ve třídě `Miner` určena metoda `findDirectionOutliers()`. Pro detekci odlehlých trajektorií podle hustoty buněk, kterými trajektorie prochází, slouží metoda `findDensityOutliers()`. Tyto dvě metody se liší v podstatě jen v tom, zda je volána metoda pro výpočet míry odlehlosti podle směru nebo podle hustoty. Druhým rozdílem je oddělení parametrů prahové míry odlehlosti podle směru a podle hustoty. Výsledkem těchto metod je seznam objektů třídy `OutlyingScores`, které jsou asociované s trajektoriemi, jež byly identifikovány jako odlehlé. Algoritmus pro vyhledání odlehlých trajektorií je souhrnně pro oba případy zapsán v pseudokódu Algoritmem 5.4.

Pro každou trajektorii z datového zdroje je nejprve jedním z algoritmů, které jsou uvedeny v předchozí podkapitole, vypočten vývoj míry odlehlosti. Nejvyšší dosažená hodnota

Algoritmus 5.4: Vyhledání odlehlých trajektorií

Vstup:
databáze trajektorií D
prahová hodnota míry odlehlosti δ

Výstup:
seznam odlehlých trajektorií S

foreach trajektorie $t \in D$ **do**
vypočti vývoj míry odlehlosti pro trajektorii t
o_{max} = maximální hodnota vyvíjející se míry
if $o_{max} >= \delta$ **then**
přidej t do seznamu S
end
end
return S

vyvíjející se míry odlehlosti je srovnána s prahovou hodnotou a pokud alespoň této hodnoty dosahuje, je trajektorie označena jako odlehlá. Maximální hodnotu vyvíjející se míry odlehlosti trajektorie zpřístupňuje metoda `getMaxEvolvingScore()` třídy `OutlyingScores`.

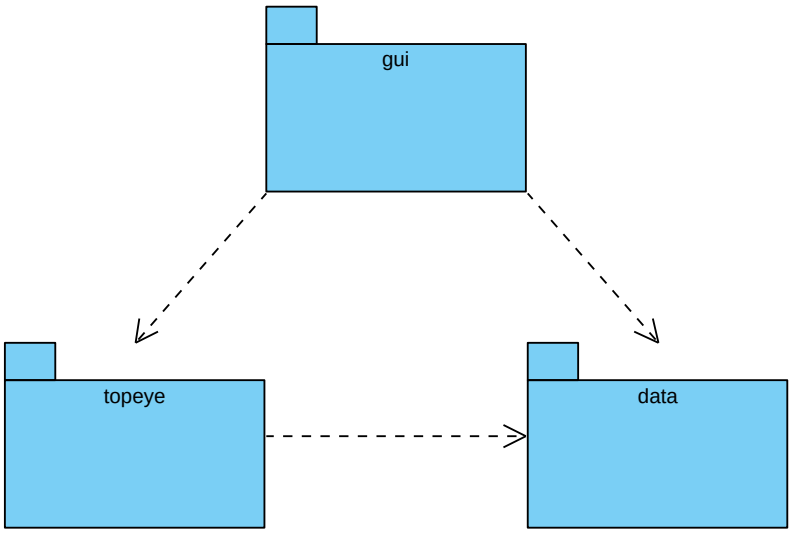
Tato implementace algoritmu TOP-EYE může být využita nejen v aplikacích, které vyžadují detekci odlehlých trajektorií v objemné množině dat, ale i pro aplikace, v nichž je zapotřebí analyzovat jednotlivé trajektorie průběžně, například při napojení na proud dat.

Kapitola 6

Aplikace pro detekci odlehlých trajektorií

Obsahem této kapitoly je popis návrhu a realizace aplikace pro detekci odlehlých trajektorií s využitím implementace algoritmu TOP-EYE, která je popsána v kapitole 5. Aplikace je implementována v programovacím jazyce Java s využitím knihovny Swing pro grafické uživatelské rozhraní a JDBC API pro přístup k databázi dat pohybujících se objektů.

Na obrázku 6.1 je zobrazen diagram balíčků této aplikace. Balíček `topeye` reprezentuje samotný algoritmus TOP-EYE a jeho popisem se zabývala kapitola 5. Balíček `data` zapouzdřuje přístup k databázi trajektorií a kromě tříd, se kterými přímo pracuje algoritmus TOP-EYE, obsahuje implementaci tříd pro přístup ke konkrétním zdrojům dat. Úkolem balíčku `gui` je vytvořit grafické uživatelské rozhraní pro zpřístupnění funkcí aplikace uživateli.



Obrázek 6.1: Diagram balíčků aplikace pro detekci odlehlých trajektorií

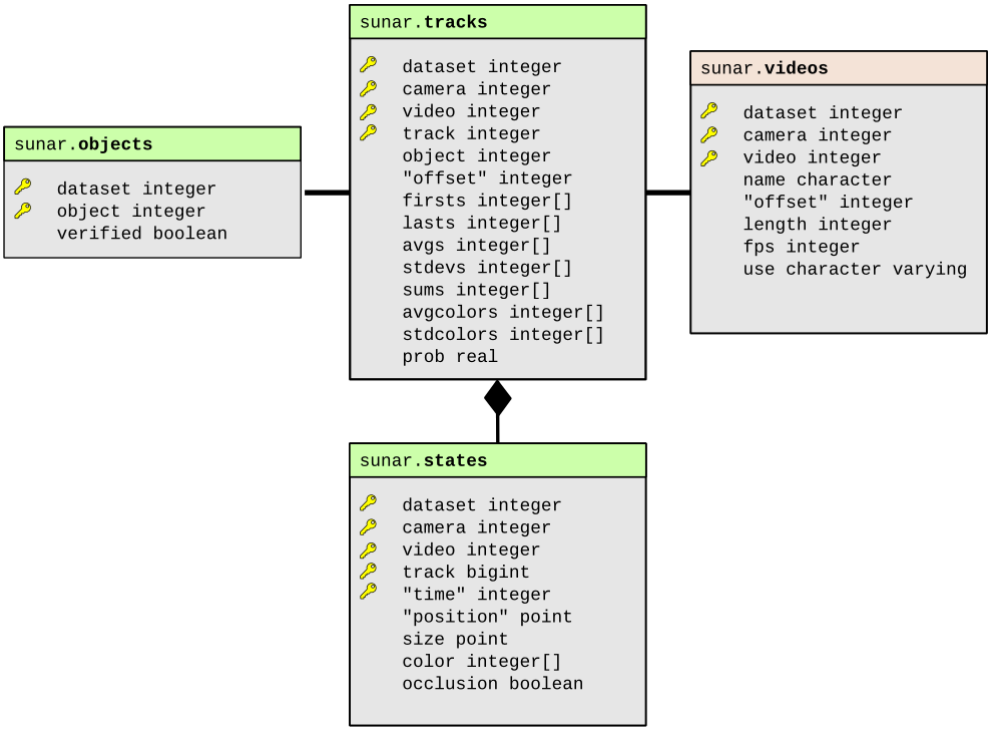
V následujícím textu je nejprve popsána datová vrstva aplikace, konkrétně přístup k datům se systému SUNAR a obecným datům pohybujících se objektů. Zbytek kapitoly se věnuje popisu jednotlivých částí uživatelského rozhraní a postupu při jejich realizaci.

6.1 Datová vrstva

Algoritmus TOP-EYE používá pro přístup k datům pohybujících se objektů rozhraní `MovingObjectsData` z balíčku `data`, což umožňuje jednotné využití algoritmu nad různými datovými zdroji. Přístup ke konkrétním datům tedy musí být realizován implementací tohoto rozhraní. V této aplikaci pro detekci odlehlých trajektorií jsou tímto způsobem zastoupeny dva různé typy dat. V první řadě se jedná o přístup k databázi ze systému SUNAR [4], dále pak je umožněna práce s daty z databáze trajektorií s jednoduchým obecným schématem.

6.1.1 Data ze systému SUNAR

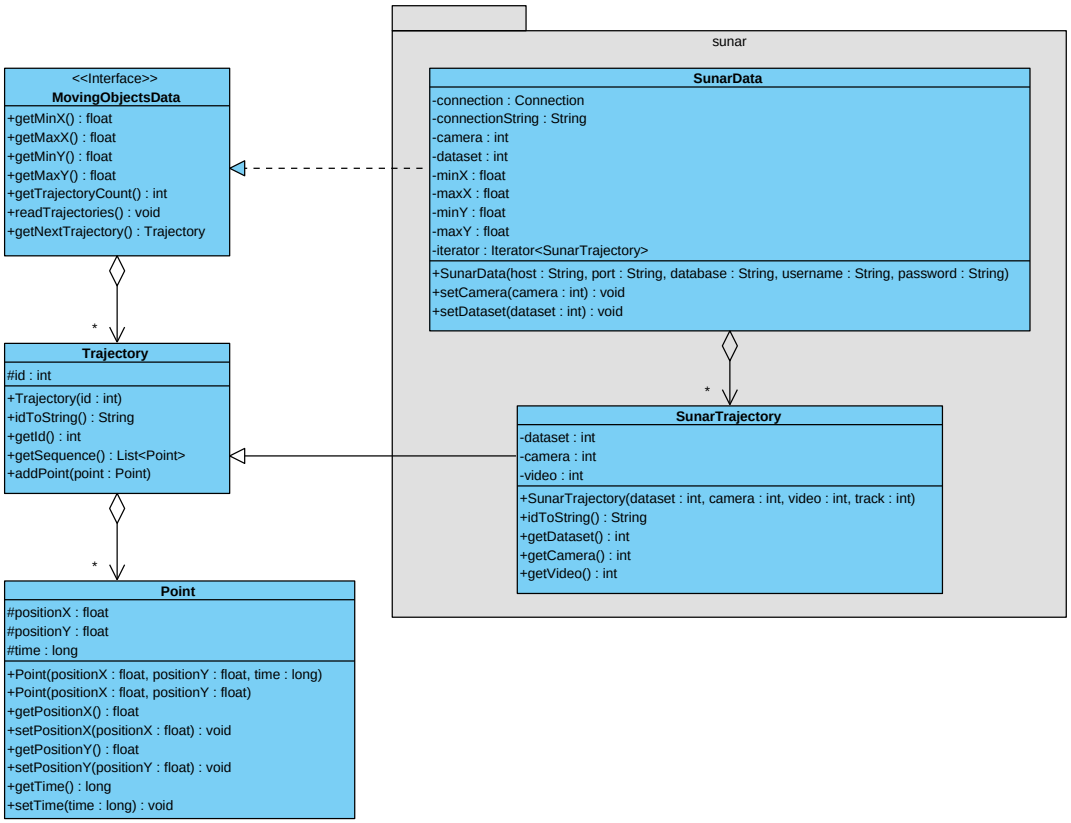
V rámci vývoje systému SUNAR (*Surveillance Network Augmented by Retrieval*) [4] byla použita reálná datová sada trajektorií z multi-kamerového dohledového systému extrahovaných z datové sady z knihovny i-LIDS [12]. Pro uložení těchto dat je využit databázový systém PostgreSQL. Částečné schéma této databáze je zachyceno na obrázku 6.2.



Obrázek 6.2: Částečné schéma databáze systému SUNAR (převzato z [3])

Přístup k datům ze systému SUNAR zajišťuje třída `SunarData`, která implementuje rozhraní `MovingObjectsData`. Objekt této třídy udržuje připojení k databázi, přičemž přístupové a přihlašovací údaje k ní jsou nastaveny přes odpovídající parametry konstruktoru. Diagram tříd balíčku `data`, který je rozšířený o přístup k datům systému SUNAR, je na obrázku 6.3.

Jedna konkrétní instance třídy `SunarData` poskytuje data získaná z jedné určité kamery. Standardně jsou poskytována data ze všech dostupných datových sad pro danou kameru, je však možné rozsah dat omezit na jednu konkrétní datovou sadu.



Obrázek 6.3: Diagram tříd balíčku `data` včetně tříd pro přístup k datům ze systému SUNAR

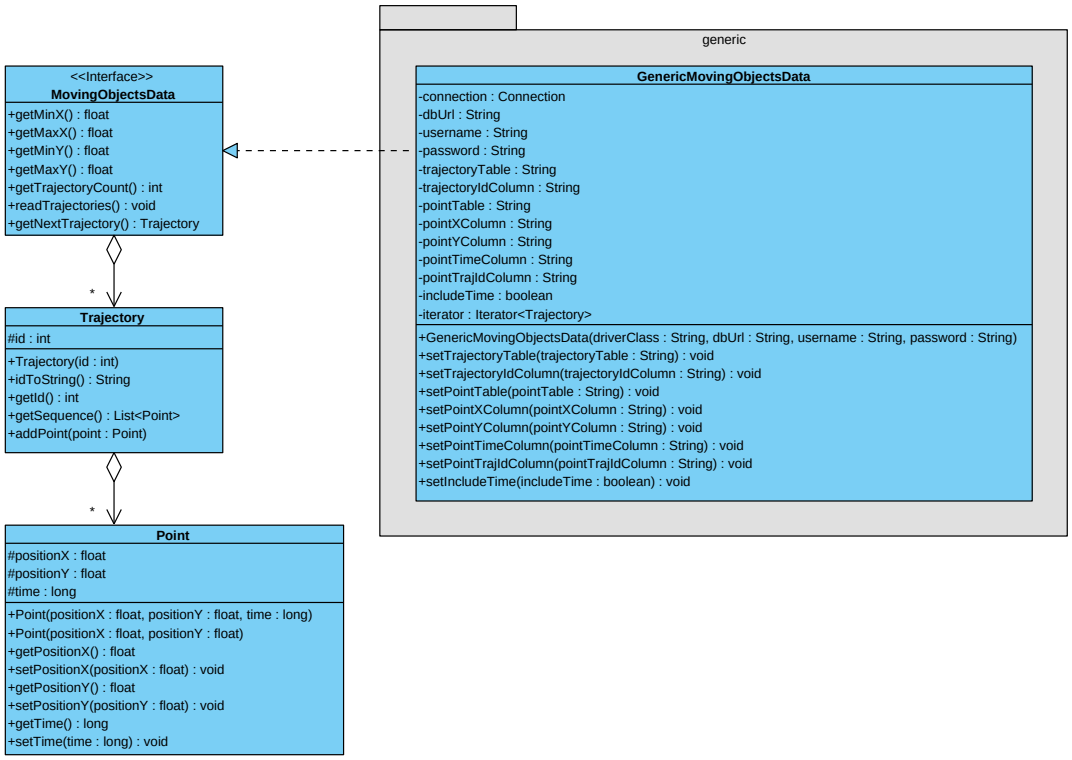
Pro načtení všech požadovaných trajektorií slouží metoda `readTrajectories()`, která podle obsahu databázové tabulky `tracks` vytvoří seznam objektů třídy `SunarTrajectory`. Tato třída je specializací třídy `Trajectory` a byla vytvořena proto, že každá trajektorie je zde identifikovaná nejen svým číslem, ale také identifikací datové sady, kamery a videa, z nichž pochází.

K zpřístupnění jednotlivých trajektorií je určena metoda `getNextTrajectory()`, která na základě aktuální pozice udržovaného iterátoru získá následující trajektorii ze seznamu vytvořeného metodou `readTrajectories()`, kterou sama volá, pokud tento seznam ještě nebyl vytvořen. Pro získanou trajektorii je z databázové tabulky `states` načten její průběh v podobě posloupnosti pozic pohybujícího se objektu.

Aby nedocházelo k opakovanému načítání stejných dat z databáze, kontrolují metody `readTrajectories()` a `getNextTrajectory()`, zda jsou či nejsou požadovaná data uložena v paměti. Pokud ano, nedochází k načítání z databáze, ale místo načtení všech trajektorií je pouze vytvořen nový iterátor pro průchod kolekcí trajektorií. Při požadavku na následující trajektorii dochází k načítání z databáze jen tehdy, neobsahuje-li trajektorie v paměti žádnou pozici.

6.1.2 Obecná data pohybujících se objektů

Třída `GenericMovingObjectsData`, implementující rozhraní `MovingObjectsData`, umožňuje přístup k databázím trajektorií, jež vyhovují níže popsanému jednoduchému schématu. Diagram tříd balíčku `data` rozšířeného o tuto třídu je zobrazen na obrázku 6.4.



Obrázek 6.4: Diagram tříd balíčku `data` včetně tříd pro přístup k datům pohybujících se objektů s jednoduchým obecným schématem

Trajektorie uložené v databázi, ke které je možné přistupovat tímto způsobem, musí být uloženy v jedné tabulce. Každá trajektorie by měla být identifikovaná pomocí jednoho sloupce, jenž obsahuje celočíselné hodnoty. Posloupnost pozic všech trajektorií musí být uložena v tabulce, která obsahuje sloupec s cizím klíčem do tabulky trajektorií. Pro každou pozici trajektorie jsou uvažovány dva sloupce pro uložení souřadnic v prostoru a jeden sloupec pro čas výskytu. Pro uložení prostorových souřadnic je zde uvažován datový typ odpovídající číslu s plovoucí řádovou čárkou, například `float`. Názvy obou tabulek a všech využívaných sloupců mohou být libovolné a je potřeba je nastavit pomocí příslušných metod třídy `GenericMovingObjectsData`.

Kromě názvů tabulek a sloupců lze také zvolit, jestli má být k jednotlivým pozicím trajektorie ukládána informace o čase výskytu, nebo jestli má být časová složka vyjádřená jen pořadím pozic. V případě, že má být čas jednotlivých pozic nastavován, je nutné zohlednit datový typ příslušného sloupce v databázi. Jelikož je čas ve třídě `Point`, která reprezentuje jednu pozici trajektorie, ukládán jako celočíselný datový typ `long`, proběhne nejprve pokus o přetypování časového údaje na tento datový typ. Pokud je tento pokus neúspěšný, dochází k přetypování na datový typ `java.util.Date` a až poté k převedení na hodnotu typu `long`.

Pro úspěšné spojení s konkrétním databázovým systémem musí být dodán jeho JDBC ovladač. Celý název třídy ovladače, která implementuje rozhraní `java.sql.Driver`, musí být předán jako první parametr konstruktoru třídy `GenericMovingObjectsData`. Dalšími parametry konstruktoru jsou pak přístupové a přihlašovací údaje k databázi.

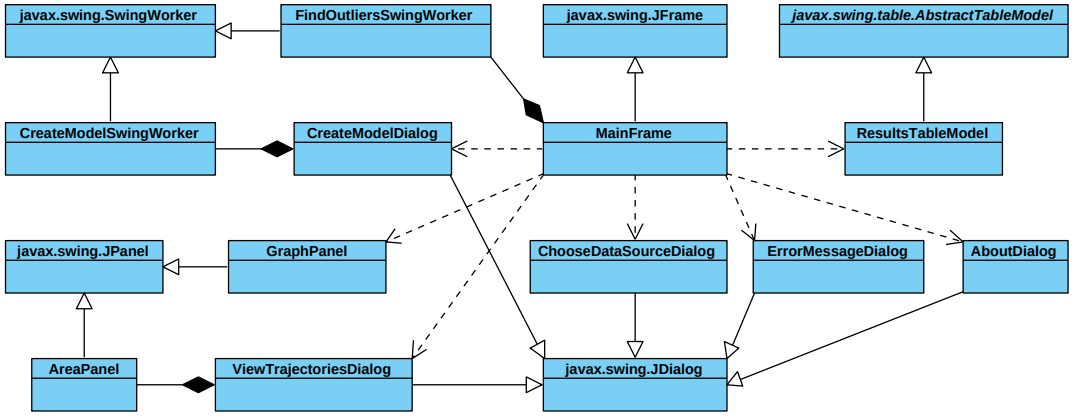
Samotné načítání trajektorií z databáze je zde implementováno obdobným způsobem, jako pro data ze systému SUNAR.

6.2 Uživatelské rozhraní

Pro ovládání aplikace je vytvořeno jednoduché grafické uživatelské rozhraní (GUI), které umožňuje připojení k vybrané databázi trajektorií, vytvoření, import či export modelu sledované oblasti, detekci odlehlých trajektorií podle směru nebo hustoty v souladu se zadanými parametry a prezentaci získaných výsledků.

Obrázek 6.5 ukazuje zjednodušený diagram tříd balíčku `gui`, který je za uživatelské rozhraní této aplikace zodpovědný. Ústředním prvkem je třída `MainFrame`, která reprezentuje hlavní okno aplikace. Vzhled hlavního okna aplikace je zobrazen na obrázku 6.9.

Pro detekci případných chyb, které mohou vzniknout při práci s touto aplikací, je použito systému výjimek. V případě zachycení libovolné výjimky dojde k okamžitému zobrazení dialogového okna reprezentovaného třídou `ErrorMessageDialog`. Obsahem tohoto okna je zpráva, která informuje o vzniklé chybě.



Obrázek 6.5: Zjednodušený diagram tříd uživatelského rozhraní

6.2.1 Výběr datového zdroje

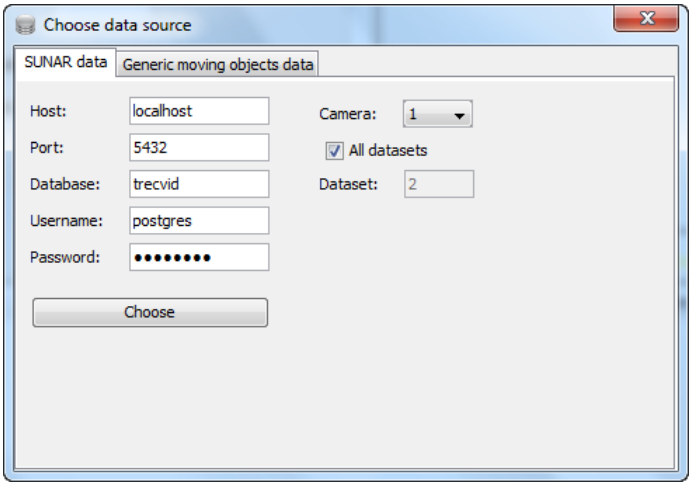
Výběr zdroje dat a připojení k němu umožňuje dialogové okno, které je reprezentované třídou `ChooseDataSourceDialog`. Toto okno je zobrazeno automaticky po spuštění aplikace, ale je možné ho vyvolat i později.

Dialogové okno obsahuje dvě záložky. První záložka slouží pro připojení k databázi systému SUNAR (viz obrázek 6.6) a druhá pro připojení k databázi pohybujících se objektů s jednoduchým obecným schématem (viz obrázek 6.7). V případě potřeby práce s daty jiného typu by bylo zapotřebí aplikaci jednoduše rozšířit vytvořením nové třídy implementující rozhraní `MovingObjectsData` a přidáním nové záložky s příslušnými prvky GUI do tohoto okna.

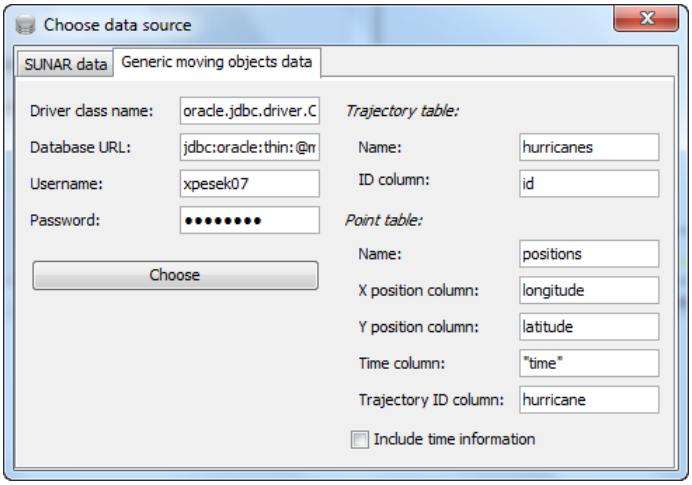
Po zadání a potvrzení přístupových a přihlašovacích údajů požadovaných zvoleným zdrojem dat je vytvořena nová instance třídy reprezentující příslušný datový zdroj. Následně dojde k vytvoření instance třídy `Miner` a její propojení se zvoleným zdrojem dat.

6.2.2 Práce s modelem sledované oblasti

Aby bylo možné provést vlastní detekci odlehlých trajektorií, musí být vytvořen nebo načten model sledované oblasti. Pro vytvoření nového modelu je určeno dialogové okno reprezentované třídou `CreateModelDialog`. Toto okno je zobrazeno na obrázku 6.8.



Obrázek 6.6: Dialogové okno pro připojení k databázi systému SUNAR



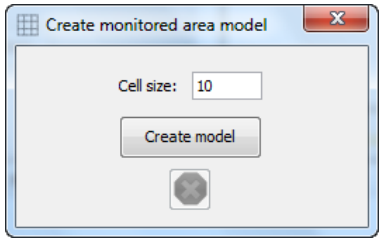
Obrázek 6.7: Dialogové okno pro připojení k databázi s jednoduchým obecným schématem

Po zadání a potvrzení velikosti buňky mřížky, na kterou je sledovaný prostor transformován, je zahájeno vytváření modelu. Protože se jedná o činnost, která může být poměrně časově náročná, je její provádění spuštěno na pozadí ve vlastním vlákně a může být stiskem příslušného tlačítka předčasně zrušeno. Za zajištění běhu na pozadí je zde zodpovědná třída `CreateModelSwingWorker`, která využívá možností třídy `javax.swing.SwingWorker`.

Kromě vytvoření nového modelu sledované oblasti umožňuje aplikace importovat ze zvoleného souboru již dříve vytvořený a exportovaný model, čímž je možné se vyhnout nutnosti vytváření modelu při budoucím použití aplikace.

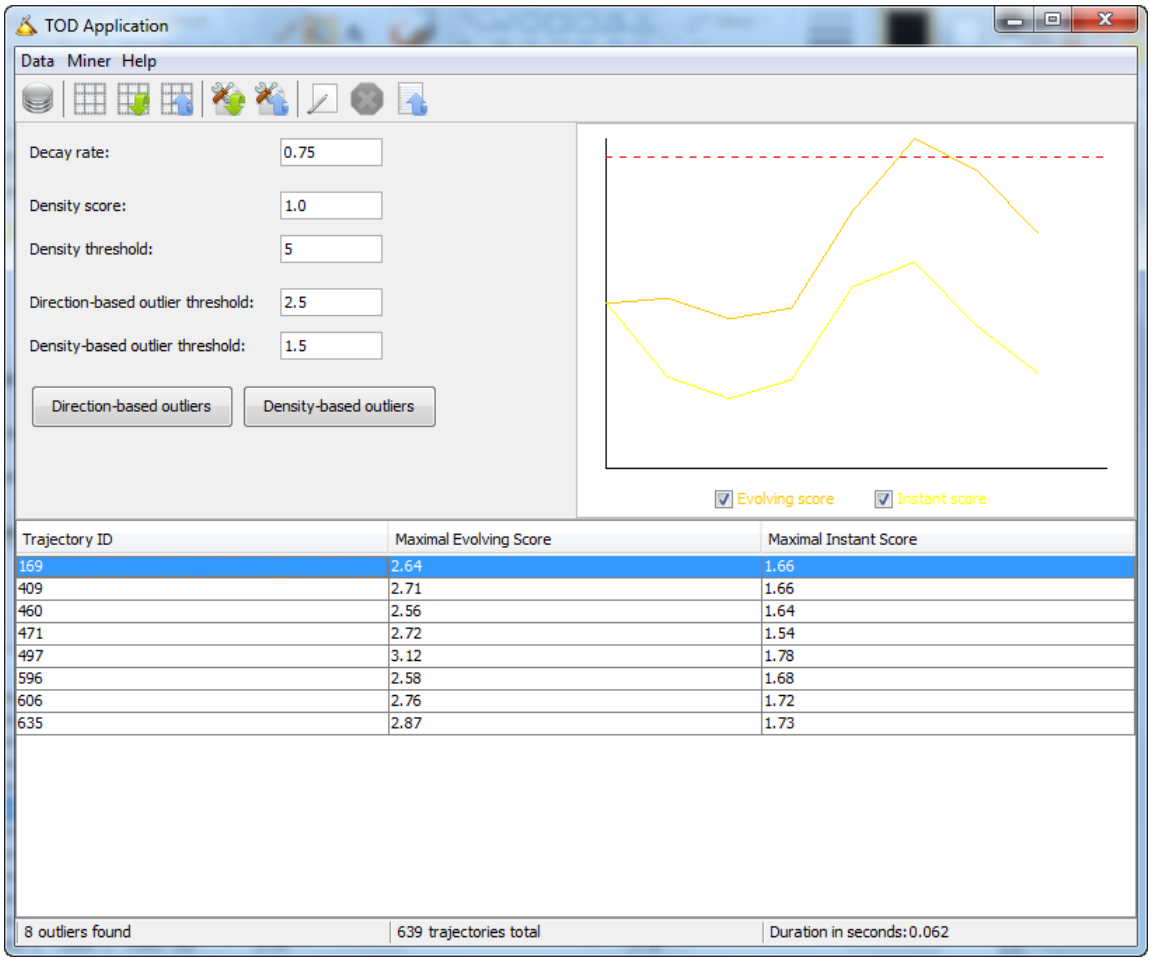
6.2.3 Detekce odlehlých trajektorií

Algoritmus TOP-EYE, který tato aplikace pro detekci odlehlých hodnot využívá, požaduje nastavení pěti uživatelských parametrů, které mají vliv na vlastní proces detekce. Pro zadání těchto parametrů jsou určena příslušná vstupní pole v hlavním okně aplikace (viz obrázek 6.9). Aktuální konfigurace těchto parametrů také může být ve formátu XML uložena do zvoleného souboru a při příštím použití aplikace z tohoto souboru načtena.



Obrázek 6.8: Dialogové okno pro vytvoření modelu sledované oblasti

Pro samotné dolování odlehlých trajektorií je možné zvolit ze dvou přístupů podle toho, jaký typ míry odlehlosti je při detekci počítán. Lze tak vyhledávat odlehlé trajektorie na základě směru, kterým se pohybují, nebo na základě hustoty blízkého okolí trasy jejich pohybu. Vzhledem k tomu, že v případě objemné množiny analyzovaných trajektorií může detekce trvat delší dobu, je tento proces spuštěn na pozadí ve vlastním vlákně provádění. Pro tento účel slouží třída `FindOutliersSwingWorker`, která podobně jako u vytváření modelu využívá možností třídy `javax.swing.SwingWorker`. Proces detekce může být tedy kdykoliv během svého provádění předčasně zrušen.



Obrázek 6.9: Hlavní okno aplikace pro detekci odlehlých trajektorií

6.2.4 Prezentace výsledků

Po dokončení detekce odlehlých trajektorií jsou získané výsledky zobrazeny v tabulce, jejíž obsah řídí objekt třídy `ResultsTableModel`. Řádky tabulky odpovídají trajektoriím, které byly v souladu se zadanými parametry označeny jako odlehlé. Každý řádek obsahuje identifikaci trajektorie a hodnoty maximální dosažené vyvíjející se míry odlehlosti a maximální dosažené okamžité míry odlehlosti. Příklad obsahu hlavního okna aplikace po dokončení procesu detekce je zobrazen na obrázku 6.9.

Kromě seznamu nalezených odlehlých trajektorií lze prohlížet graf vývoje míry odlehlosti vybrané trajektorie, v němž je možné zobrazit buď vyvíjející se míru odlehlosti, okamžitou míru odlehlosti, nebo oboje zároveň. V grafu je zároveň pro názornost zakreslena prahová hodnota míry odlehlosti. Za vykreslování tohoto grafu je zodpovědná třída `GraphPanel`.

Stavový řádek programu obsahuje souhrnné informace jako je počet nalezených odlehlých trajektorií, celkový počet trajektorií a doba trvání procesu.

Pro vykreslení tras odlehlých trajektorií ve sledovaném prostoru je v aplikaci určeno okno reprezentované třídou `ViewTrajectoriesDialog`. Do tohoto dialogového okna jsou vykresleny průběhy všech trajektorií ze zdroje dat. V případě, že jsou v okamžiku otevření okna k dispozici výsledky detekce odlehlých trajektorií, jsou odlehlé trajektorie při vykreslování zvýrazněny. Příklad obsahu tohoto okna lze najít na obrázku 7.4 nebo na obrázku 7.5 v kapitole 7.

Výsledky dolování odlehlých trajektorií pomocí této aplikace je možné v textové podobě exportovat do souboru. Toho může být využito pro další prezentaci výsledků pomocí jiných nástrojů. Například v případě dat ze systému SUNAR je možné výsledky zobrazit přímo ve videu, z něhož byly dané trajektorie extrahovány.

Kapitola 7

Experimentální zhodnocení

Pro experimentální ověření algoritmu TOP-EYE, jehož implementací se zabývala kapitola 5, byla navržena a realizována aplikace s grafickým uživatelským rozhraním, která umožňuje použití jeho funkcí nad daty z různých zdrojů (viz kapitola 6).

Obsahem této kapitoly je zhodnocení výsledné realizace nad dvěma různými reálnými datovými sadami. Nejprve jsou shrnuty výstupy algoritmu nad trajektoriemi extrahovanými z datové sady videí z multi-kamerového dohledového systému v rámci vývoje systému SUNAR. Jako druhá datová množina je zde využita databáze trajektorií hurikánů z oblasti Atlantského oceánu. Na základě výsledků nad druhou datovou sadou je pak provedeno srovnání s algoritmem TRAOD. Následuje ověření vlivu jednotlivých uživatelských parametrů na výsledky detekce. Na závěr je algoritmus TOP-EYE vyhodnocen z hlediska časové složitosti vytvoření modelu sledované oblasti a vlastní detekce odlehlých trajektorií.

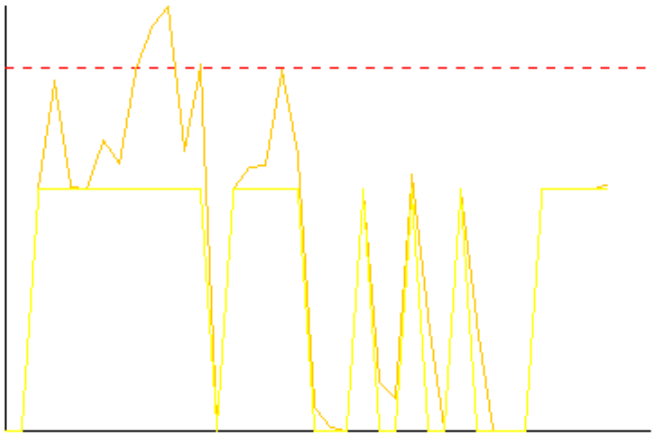
7.1 Výsledky pro data ze systému SUNAR

Databáze ze systému SUNAR [4] obsahuje trajektorie pohybujících se objektů, které jsou extrahovány z datové sady videí pořízených pěti kamerami v prostředí letiště (datová sada z knihovny i-LIDS [12]). Každá trajektorie je kromě čísla kamery identifikována číslem datové sady, číslem videa a číslem trajektorie v rámci tohoto videa. Pro uložení těchto dat je použit databázový systém PostgreSQL.

Při vyhodnocování výsledků experimentů nad těmito daty bylo zjištěno, že většina z trajektorií, které byly identifikovány jako odlehlé, nekorespondují s pohybem jen jediného objektu a odráží tak nekvalitní extrakci ze zdrojového videa. Pro následující příklad odlehlé trajektorie však tento problém neplatí.

Na obrázku 7.1 je znázorněn průběh míry odlehlosti podle hustoty pro trajektorii z druhé datové sady první kamery. Nejvyšší dosažená hodnota vyvíjející se míry odlehlosti zde je 3,5 při nastavení parametrů $\lambda = 0{,}4$, $s = 2$ a $\tau = 10$. Prahová hodnota vyvíjející se míry odlehlosti byla nastavena na 3,0. Velikost jedné buňky mřížky modelu sledované oblasti byla zvolena 50.

Trasa této odlehlé trajektorie v prostoru sledovaném první kamerou je zobrazena na obrázku 7.2. Na tomto obrázku jsou vykresleny všechny trajektorie z druhé datové sady, přičemž detekovaná trajektorie je zvýrazněna červeně. Snímek videa, na němž je odpovídající objekt (ve skutečnosti osoba) identifikován, je na obrázku 7.3. Objekt, jehož pohyb byl označen za neobvyklý, je na videu ohraničen modrým obdélníkem.



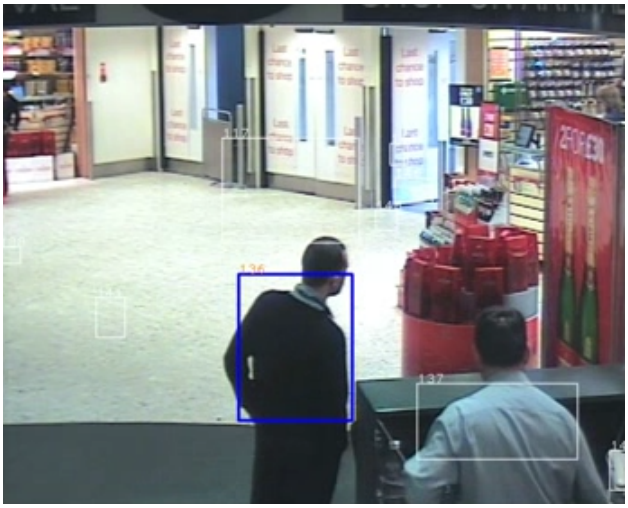
Obrázek 7.1: Graf vývoje míry odlehlosti trajektorie odlehlé podle hustoty (vyvíjející se míra je oranžově a okamžitá míra žlutě)

Obrázek 7.2: Trasa odlehlé trajektorie ve sledovaném prostoru ve srovnání s ostatními trajektoriemi

7.2 Výsledky pro trajektorie hurikánů

Použitá datová sada trajektorií hurikánů z oblasti Atlantského oceánu [24] obsahuje záznamy o hurikánech a tropických bouřích od roku 1851 do současnosti. O každém hurikánu jsou v šestihodinových intervalech zaznamenány údaje o zeměpisné šířce a délce, času, rychlosti větru a tlaku vzduchu.

Pro experimenty s těmito daty byly vybrány trajektorie hurikánů od roku 1950 po rok 2008. Z těchto dat byly extrahovány údaje o prostorové pozici a času a byly uloženy do databáze. Jako databázový systém pro uložení těchto dat byl zvolen Oracle. Schéma databáze bylo navrženo s ohledem na možnost využití již implementovaného přístupu aplikace ke zdroji dat pohybujících se objektů s jednoduchým obecným schématem.



Obrázek 7.3: Snímek videa, z něhož byla extrahována odlehlá trajektorie (video pochází z datové sady knihovny i-LIDS [12])

Výsledky detekce odlehlých trajektorií na základě směru trajektorie jsou znázorněny na obrázku 7.4, který zobrazuje trasy hurikánů ve sledované oblasti. Trajektorie, u kterých byl v souladu se zadanými parametry identifikován neobvyklý vývoj, jsou v obrázku vyznačeny červeně. Obdobným způsobem jsou znázorněny výsledky detekce odlehlých trajektorií na základě hustoty buněk, kterými trajektorie prochází, viz obrázek 7.5. Nastavení parametrů v obou případech bylo následující: $\lambda = 0{,}7$, $s = 1$ a $\tau = 10$. Velikost jedné buňky mřížky modelu sledované oblasti byla zvolena 5°. Prahová hodnota vyvíjející se míry odlehlosti podle směru byla nastavena na 2,5, podle hustoty na 1,8.

Obrázek 7.4: Trajektorie hurikánů odlehlé podle směru

Obrázek 7.5: Trajektorie hurikánů odlehlé podle hustoty

7.3 Srovnání s algoritmem TRAOD

Použitá datová sada hurikánů byla v [16] využita pro experimentální ověření algoritmu TRAOD. Jedná se o algoritmus pro detekci odlehlých trajektorií na základě vzdálenosti. Jeho stručný popis lze najít v kapitole 4.1.1.

Výsledky nad téměř stejnou podobou datové sady trajektorií hurikánů dosažené algoritmem TOP-EYE jsou znázorněny na obrázcích 7.4 a 7.5. Pro srovnání těchto výsledků s výsledky algoritmu TRAOD viz obrázek 7.6. Červeně obarvené trajektorie byly identifikovány jako odlehlé, silně zvýrazněné části pak odkazují na odlehlé sub-trajektorie.

7.4 Vliv parametrů

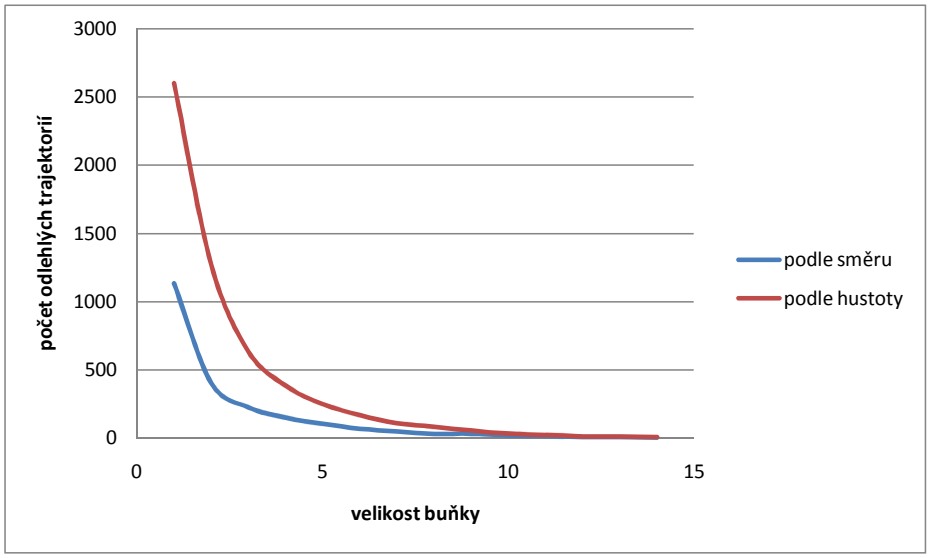
V této podkapitole je zhodnocen vliv jednotlivých uživatelských parametrů dolovací úlohy a velikosti buňky sledovaného prostoru na výsledky.

Hodnota parametru λ (*decay rate*) umožňuje regulovat vliv historické odlehlosti trajektorie na aktuální hodnotu vyvíjející se míry odlehlosti. Zvyšováním hodnoty tohoto parametru je vliv historické odlehlosti trajektorie na aktuální odlehlost snižován a vyvíjející se míra odlehlosti tak nabývá nižších hodnot. S rostoucí hodnotou parametru λ tedy klesá počet nalezených odlehlých trajektorií.

Parametry s (*density score*) a τ (*density threshold*) hrají roli pouze při detekci trajektorií odlehlých podle hustoty. Hodnota s definuje okamžitou míru odlehlosti trajektorie při průchodu buňkami s nízkou hustotou. Zda je hustota buňky nízká nebo normální určuje parametr τ. Parametr s nemá na výpočet vývoje míry odlehlosti žádný vliv a ovlivňuje pouze rozsah hodnot, kterých míra odlehlosti podle hustoty bude nabývat. Zvýšení hodnoty τ způsobí nárůst počtu buněk s nízkou hustotou. S rostoucí hodnotou parametru τ tedy roste počet nalezených trajektorií, které jsou odlehlé podle hustoty.

Obrázek 7.6: Odlehlé trajektorie hurikánů detekované algoritmem TRAOD (převzato z [16])

Prahová hodnota vyvíjející se míry odlehlosti pro rozlišení odlehlých trajektorií od normálních přímo rozděluje datovou množinu na podmnožinu normálních trajektorií a podmnožinu odlehlých trajektorií a nemá tedy smysl se jejím vlivem na výsledky dolování více zabývat.



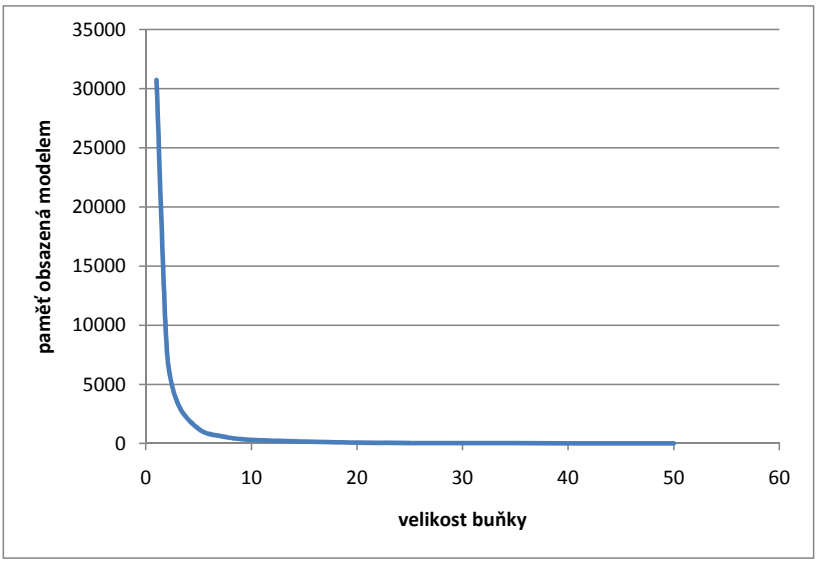
Obrázek 7.7: Graf závislosti počtu detekovaných odlehlých trajektorií na velikosti buňky modelu

Vliv velikosti buňky mřížky modelu sledovaného prostoru na počet výsledků dolování znázorňuje graf na obrázku 7.7. Tento graf byl vytvořen na základě experimentů s druhou datovou sadou druhé kamery z databáze systému SUNAR. Kromě velikosti buňky byly ostatní parametry během zaznamenávání výsledků konstantní, konkrétně: $\lambda = 0{,}75$, $s = 1$, $\tau = 5$ a prahová hodnota vyvíjející se míry odlehlosti podle směru byla nastavena na 2,5,

podle hustoty na 1,5.

S rostoucí velikostí buňky modelu sledovaného prostoru klesá počet výsledků detekce, přičemž prudší pokles byl zaznamenán v případě detekce trajektorií odlehlých podle hustoty.

Se stejnými daty a stejně nastavenými parametry jako v případě předchozího grafu bylo experimentálně ověřeno, jakým způsobem ovlivňuje velikost buňky mřížky množství paměti potřebné pro uložení modelu sledované oblasti. Výsledek ukazuje graf na obrázku 7.8. Zvyšováním velikosti buňky paměťové požadavky na reprezentaci modelu klesají s tím, že nejvýraznější je tento pokles pro velmi malé hodnoty velikosti buňky.



Obrázek 7.8: Graf závislosti množství paměti obsazené modelem v kB na velikosti buňky modelu

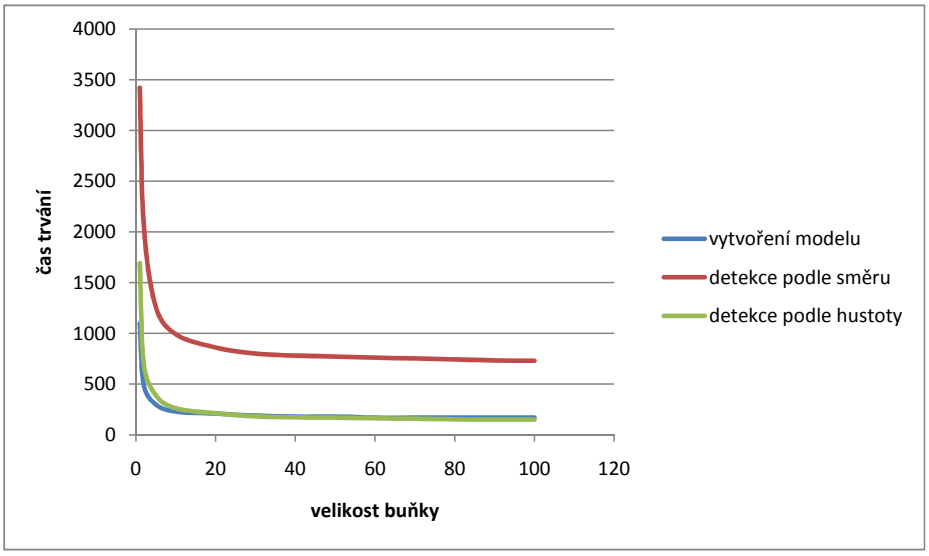
7.5 Zhodnocení časové náročnosti

Nad databází ze systému SUNAR byla experimentálně provedena měření časové složitosti implementace algoritmu TOP-EYE. Jednotlivá měření byla provedena zvlášť pro vytvoření modelu sledované oblasti, pro detekci trajektorií odlehlých podle směru a pro detekci trajektorií odlehlých podle hustoty.

Nastavení uživatelských parametrů bylo ve všech případech konstantní, konkrétně: $\lambda = 0{,}75$, $s = 1$, $\tau = 5$ a prahová hodnota vyvíjející se míry odlehlosti podle směru byla nastavena na 2,5, podle hustoty na 1,5.

Obrázek 7.9 znázorňuje závislost doby trvání výpočtu na velikosti buňky sledované oblasti. Měření bylo provedeno nad druhou datovou sadou druhé kamery. Časová náročnost vytvoření modelu a obou metod detekce zde vykazuje stejný průběh, přičemž doba trvání detekce odlehlých trajektorií podle směru je podle předpokladů větší než u detekce podle hustoty a při vytváření modelu. Průběhy pro vytvoření modelu a detekci podle hustoty téměř splývají. S rostoucí hodnotou velikosti buňky mřížky ve všech případech časové nároky algoritmu klesají. Od určitých hodnot velikosti buňky pak časová náročnost vykazuje konstantní průběh.

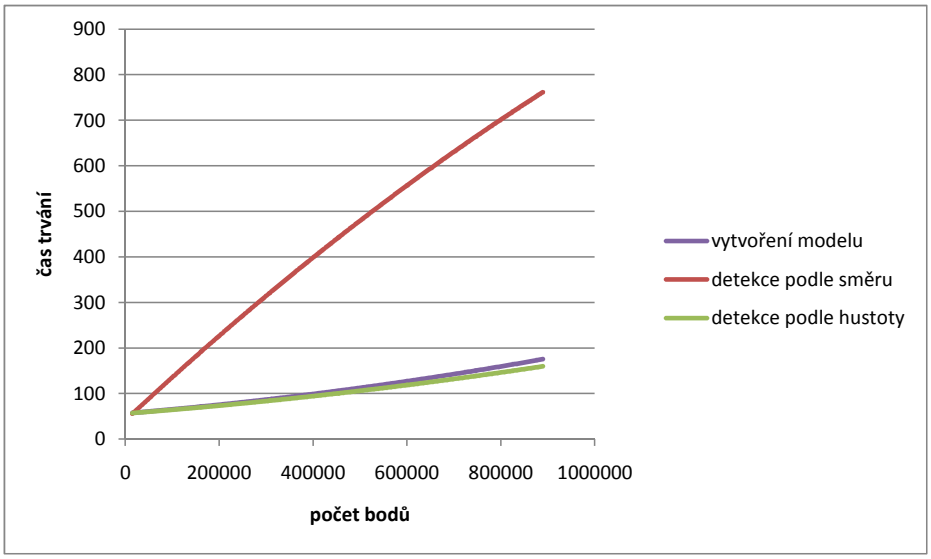
Následující grafy zobrazují experimentálně zjištěné výsledky časové složitosti algoritmu TOP-EYE v závislosti na velikosti vstupní datové množiny. Při těchto experimentech byla



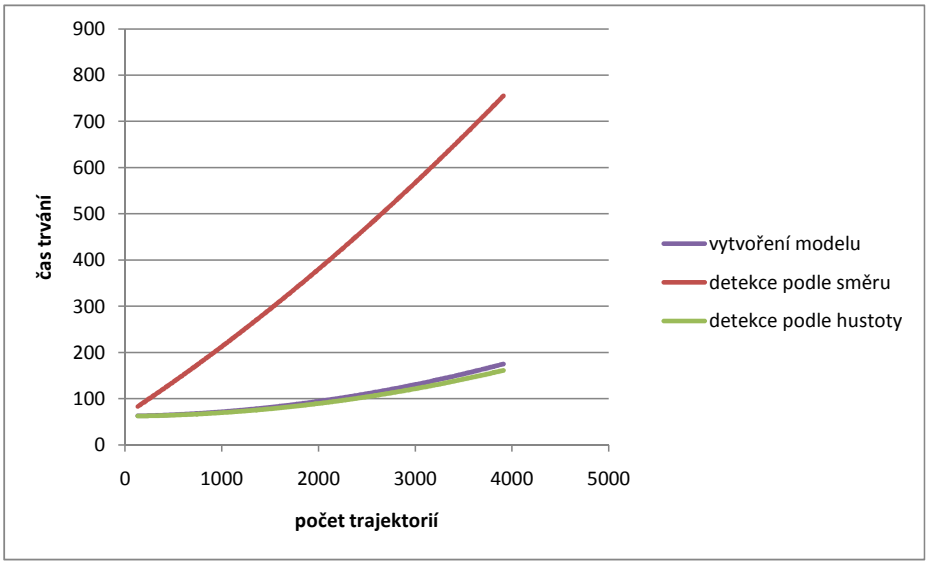
Obrázek 7.9: Graf závislosti doby trvání výpočtu (v milisekundách) na velikosti buňky modelu

nastavena velikost buňky modelu sledovaného prostoru na hodnotu 50.

Obrázek 7.10 znázorňuje závislost doby trvání výpočtu na počtu všech pozic analyzovaných trajektorií, obrázek 7.11 pak ukazuje závislost doby trvání výpočtu na počtu analyzovaných trajektorií. Průběhy zobrazené na obou těchto grafem vykazují obdobný charakter. Časovou složitost algoritmu TOP-EYE lze podle těchto výsledků označit za lineární s tím, že doba trvání detekce trajektorií odlehlých podle směru roste rychleji než doba trvání vytvoření modelu a detekce podle hustoty. Průběhy doby trvání vytvoření modelu a detekce trajektorií odlehlých podle hustoty v obou případech téměř splývají.



Obrázek 7.10: Graf závislosti doby trvání výpočtu (v milisekundách) na celkovém počtu bodů analyzovaných trajektorií



Obrázek 7.11: Graf závislosti doby trvání výpočtu (v milisekundách) na počtu analyzovaných trajektorií

Kapitola 8

Závěr

Časoprostorová data jsou data, která kromě vlastních rysů obsahují jak časovou tak prostorovou složku. Typicky se jedná o data pohybujících se objektů v podobě jejich trajektorií, může jít o údaje z meteorologických a oceánografických měření, záznamy přírodních katastrof a podobně.

Jedním z cílů této práce bylo shrnout základní poznatky z oblasti získávání znalostí z časoprostorových dat a kategorizovat a popsat časoprostorové vzory a existující techniky pro jejich dolování. V počátcích výzkumu v této oblasti se objevovaly zejména pokusy o použití existujících algoritmů pro dolování v časových nebo v prostorových datech pro hledání zajímavých vzorů v datech časoprostorových. Ukázalo se však, že časoprostorová data obsahují složité vztahy, které není možné získat pohledem na časovou a prostorovou dimenzi odděleně. Kromě toho při tomto přístupu často velmi výrazně vzrostl prohledávaný prostor možných vzorů. Z těchto důvodů bylo nutné najít techniky, které by uvažovaly časovou a prostorovou informaci dohromady za účelem nalezení zajímavých a užitečných časoprostorových vzorů.

V současnosti lze oblast dolování v časoprostorových datech klasifikovat podle vzorů, které vyhledávají, do těchto skupin: časoprostorová klasifikace a predikce, časoprostorové shlukování, detekce anomálií či odlehlých trajektorií, dolování frekventovaných pohybů objektů a dolování časoprostorových asociačních vzorů.

Automatická detekce neobvyklého nebo podezřelého chování pohybujících se objektů patří mezi významné úlohy v oblasti analýzy časoprostorových dat. Typicky se zde jedná o identifikaci odlehlých trajektorií, tedy takových pohybů, které se výrazně liší od většiny ostatních. Pro detekci odlehlých trajektorií je možné využít různých přístupů. Jeden z nich reprezentuje metoda nazvaná TOP-EYE. Tato metoda je založena na výpočtu míry odlehlosti trajektorie nad vytvořeným pravděpodobnostním modelem sledované oblasti a umožňuje tak identifikovat odlehlé trajektorie v reálném čase.

Algoritmus TOP-EYE byl v rámci tohoto projektu implementován v programovacím jazyce Java. Aby bylo možné algoritmus použít pro různá data z různých zdrojů, byla zvolena abstraktní reprezentace datového zdroje v podobě rozhraní, které musí být pro přístup ke konkrétním datům implementováno. Pomocí algoritmu TOP-EYE je možné vyhledávat dva typy odlehlých trajektorií: buď podle směru trajektorie, nebo podle hustoty buněk, kterými trajektorie prochází. Pro dolování odlehlých trajektorií musí algoritmus nejprve vytvořit model sledované oblasti. K tomuto účelu je nutné poskytnout trénovací množinu dat. Vzhledem k tomu, že vytvářený model je pravděpodobnostního charakteru, je však možné pro vytvoření modelu použít stejná data, nad kterými bude prováděna detekce odlehlých trajektorií.

Za účelem otestování, ověření a možnosti použití implementovaného algoritmu byla navržena a realizována aplikace s grafickým uživatelským rozhraním. Aplikace poskytuje přístup ke dvěma typům datového zdroje. Prvním je databáze systému SUNAR, která obsahuje trajektorie extrahované z datové sady videí pořízených multi-kamerovým dohledovým systémem. Druhým typem zdroje dat je databáze pohybujících se objektů s jednoduchým obecným schématem. Aplikace zpřístupňuje funkce algoritmu TOP-EYE a umožňuje výsledky detekce graficky prezentovat nebo v textové podobě exportovat do souboru.

Implementovaný algoritmus byl za pomoci vytvořené aplikace experimentálně ověřen a zhodnocen nad dvěma různými datovými sadami. Kromě dat ze systému SUNAR byla využita databáze trajektorií hurikánů z oblasti Atlantského oceánu zaznamenaných v letech 1950 až 2008. Kromě ověření kvality poskytovaných výsledků, byl zhodnocen vliv jednotlivých uživatelských parametrů a časová složitost jednotlivých funkcí algoritmu. Výsledky experimentů mimo jiné ukázaly, že implementovaný algoritmus dokáže detekovat oba typy odlehlých trajektorií s lineární časovou složitostí. Stejně tak doba trvání potřebná pro vytvoření modelu sledované oblasti roste v závislosti na objemu vstupních dat lineárně.

Jako nevýhoda tohoto algoritmu se ukázala neschopnost detekovat případnou odlehlost podle směru u úseků trajektorie, které procházejí oblastmi, jimiž vedlo jen velmi málo trajektorií trénovací datové množiny.

Výsledná implementace algoritmu TOP-EYE může být využita nejen v aplikacích, které vyžadují detekci odlehlých trajektorií v objemné množině dat pohybujících se objektů, ale i v aplikacích, v nichž je požadavkem analyzovat jednotlivé trajektorie průběžně, například při napojení na odpovídající proud dat. Toto je zároveň jeden z možných směrů, kterými by se vývoj této práce mohl ubírat v budoucnosti.

Literatura

[1] Birant, D.; Kut, A.: Spatio-Temporal Outlier Detection in Large Databases. In *Proceedings of the 28th International Conference on Information Technology Interfaces*, 2006, s. 179–184.

[2] Bu, Y.; Chen, L.; Fu, A. W.-C.; aj.: Efficient Anomaly Monitoring Over Moving Object Trajectory Streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, s. 159–167.

[3] Chmelař, P.: SUNAR: Surveillance Network Augmented by Retrieval. URL `http://www.fit.vutbr.cz/~chmelarp/public/AVSS2009%20Brno_UT.pdf`, [Online; cit. 22.5.2011].

[4] Chmelař, P.; Láník, A.; Mlích, J.: SUNAR: Surveillance Network Augmented by Retrieval. In *ACIVS 2010*, s. 155–166.

[5] Dunham, M. H.: *Data Mining: Introductory and Advanced Topics*. New Jersey: Prentice Hall, 2003, ISBN 0-13-088892-3, 315 s.

[6] Ge, Y.; Xiong, H.; Zhou, Z.-H.; aj.: TOP-EYE: Top-k Evolving Trajectory Outlier Detection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010, s. 1733–1736.

[7] Giannotti, F.; Nanni, M.; Pinelli, F.; aj.: Trajectory Pattern Mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007, s. 330–339.

[8] Giannotti, F.; Pedreschi, D. (editoři): *Mobility, Data Mining and Privacy: Geographic Knowledge Discovery*. Springer, 2008, ISBN 978-3-540-75176-2, 410 s.

[9] Gudmundsson, J.; van Kreveld, M.; Speckmann, B.: Efficient Detection of Patterns in 2D Trajectories of Moving Points. *GeoInformatica*, ročník 11, 2007: s. 195–215.

[10] Güting, R. H.; Schneider, M.: *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005, ISBN 978-0-12-088799-6, 416 s.

[11] Han, J.; Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, druhé vydání, 2006, ISBN 978-1-55860-901-3, 770 s.

[12] HOSDB: Home Office Multiple Camera Tracking Scenario data. URL `http://scienceandresearch.homeoffice.gov.uk/hosdb/cctv-imaging-technology/video-based-detection-systems/i-lids`, [Online; cit. 22.5.2011].

[13] Hsu, W.; Lee, M. L.; Wang, J.: *Temporal and Spatio-Temporal Data Mining*. Hershey: IGI Publishing, 2008, ISBN 978-1-59904-387-6, 280 s.

[14] Knorr, E. M.; Ng, R. T.; Tucakov, V.: Distance-based outliers: algorithms and applications. *The VLDB Journal*, ročník 8, 2000: s. 237–253.

[15] Laube, P.; van Kreveld, M.; Imfeld, S.: Finding REMO – Detecting Relative Motion Patterns in Geospatial Lifelines. In *Proceedings of the 11th International Symposium on Spatial Data Handling*, 2004, s. 201–215.

[16] Lee, J.-G.; Han, J.; Li, X.: Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, 2008, s. 140–149.

[17] Li, X.; Han, J.; Kim, S.; aj.: ROAM: Rule- and Motif-Based Anomaly Detection in Massive Moving Object Data Sets. In *Proceedings of the 7th SIAM International Conference on Data Mining*, 2007.

[18] Li, X.; Li, Z.; Han, J.; aj.: Temporal Outlier Detection in Vehicle Traffic Data. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, 2009, s. 1319–1322.

[19] Liu, L.; Fan, J.; Qiao, S.; aj.: Efficiently Mining Outliers From Trajectories of Unrestraint Movement. In *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering*, 2010, s. V2–261–V2–265.

[20] Mitsa, T.: *Temporal Data Mining*. Chapman & Hall/CRC, 2010, ISBN 978-1-4200-8976-9, 395 s.

[21] Piciarelli, C.; Micheloni, C.; Foresti, G. L.: Anomalous Trajectory Patterns Detection. In *Proceedings of the 19th International Conference on Pattern Recognition*, 2008.

[22] Shekhar, S.; Vatsavai, R. R.; Celik, M.: Spatial and Spatiotemporal Data Mining: Recent Advances. In *Next Generation of Data Mining*, kapitola 26, CRC Press, 2009, ISBN 978-1-4200-8586-0, s. 549–584.

[23] Tobler, W. R.: Cellular Geography. In *Philosophy in Geography*, editace S. Gale; G. Olsson, 1979, ISBN 978-9027709486, s. 379–386.

[24] Unisys: Atlantic Tropical Storm Tracking by Year. URL `http://weather.unisys.com/hurricane/atlantic/`, [Online; cit. 22.5.2011].

[25] Ying, X.; Xu, Z.; Yin, W. G.: Cluster-based Congestion Outlier Detection Method on Trajectory Data. In *Proceedings of the 6th International Conference on Fuzzy Systems and Knowledge Discovery*, 2009, s. 243–247.

[26] Yu, X.; Tang, L. A.; Han, J.: Filtering and Refinement: A Two-Stage Approach for Efficient and Effective Anomaly Detection. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, 2009, s. 617–626.

Seznam příloh

Příloha A

Obsah přiloženého CD

Adresářová struktura přiloženého CD:

- **/docs/**
  - **api/** – programová dokumentace implementace algoritmu a aplikace
  - **manual.pdf** – uživatelský manuál k aplikaci
- **/thesis/**
  - **src/** – zdrojové texty technické zprávy
  - **dp-xpesek07.pdf** – technická zpráva
- **/TODApplication/**
  - **dist/** – spustitelná verze aplikace
  - **src/** – zdrojové kódy implementace algoritmu a aplikace