VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

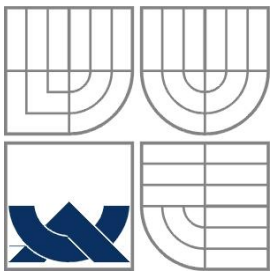
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKCE BUDOUCÍ LOKACE POHYBUJÍCÍHO SE OBJEKTU

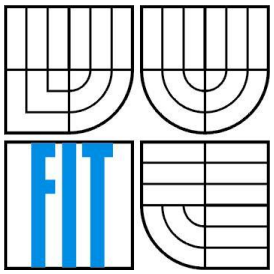
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JÁN KEBÍSEK
AUTHOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PREDIKCE BUDOUCÍ LOKACE POHYBUJÍCÍHO SE OBJEKTU

PREDICTING FUTURE LOCATION OF A MOVING OBJECT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JÁN KEBÍSEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN PEŠEK
SUPERVISOR

BRNO 2013

Abstrakt

Tato práce se věnuje návrhu a implementaci aplikace pro predikci budoucí lokace pohybujícího se objektu. Popisuje metodu predikce založenou na algoritmu WhereNext. Tento algoritmus získá z databáze trajektorií objektů T-Patterny, které představují frekventované vzory pohybu objektů, a ty následně použije k predikci. Algoritmus byl implementovaný v programovacím jazyku Java a jeho funkčnost je odzkoušena na vygenerované datové sadě pohybu aut.

Abstract

This thesis deals with the design and the implementation of the application for predicting future location of a moving object. It describes a method for prediction based on the algorithm WhereNext. This algorithm obtains T-Patterns from a database of trajectories of objects, which represent frequent patterns of movement of objects, and those subsequently uses for prediction. The algorithm was implemented in programming language Java and its functionality was tested on a generated dataset of movement of cars.

Klíčová slova

Dolovaní v datech, predikce budoucí lokace, pohybující se objekt, vzory pohybu, dolovaní vzorů pohybu.

Keywords

Data mining, predicting future location, moving object, trajectory patterns, trajectory patterns mining.

Citace

Kebísek Ján: Predikce budoucí lokace pohybujícího se objektu, bakalářská práce, Brno, FIT VUT v Brně, 2013

Predikce budoucí lokace pohybujícího se objektu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Peška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ján Kebísek
15. května 2013

Poděkování

Chtěl bych poděkovat Ing. Martinu Peškovi za vynikající vedení bakalářské práce, za jeho cenné připomínky, čas a odbornou pomoc k této bakalářské práci.

Obsah

1 Úvod ... 2
2 Získavanie znalostí z dát ... 3
2.1 Proces získavania znalostí z dát ... 3
2.1.1 Dolovanie z dát ... 4
2.2 Využitie ... 4
3 Dolovanie v trajektóriách ... 5
4 Predikcia budúcej lokácie pohybujúceho sa objektu ... 8
5 Algoritmus WhereNext ... 10
5.1 Vyhľadávanie frekventovaných lokalít ... 10
5.2 Získavanie T-Patternov ... 10
5.3 Budovanie T-Pattern stromu ... 11
5.4 Predikcia ... 13
5.4.1 Punctual skóre ... 14
5.4.2 Celkové skóre cesty ... 15
5.4.3 Parametre predikcie ... 17
6 Návrh a implementácia ... 18
6.1 Balík Database ... 18
6.2 Balík GUI ... 19
6.3 Balík WhereNext ... 20
6.3.1 Frekventované lokality ... 20
6.3.2 T-Patterny ... 21
6.3.3 T-Pattern strom ... 22
6.3.4 Predikcia ... 22
6.4 Užívateľské rozhranie a ovládanie aplikácie ... 23
7 Experimentálne overenie ... 25
7.1 Analýza možných problémov ... 25
7.2 Parametre lokalít ... 25
7.3 Parametre predikcie ... 27
8 Záver ... 30
Zoznam príloh ... 32
A. Obsah CD ... 33
B. Návod na použitie ... 34

1 Úvod

Vďaka vynájdeniu zariadení ako GPS lokátory, mobilné zariadenia a rôzne pohybové senzory, máme na internete dostupné veľké množstvo verejne prístupných dát o pohybe najrôznejších objektov ako napríklad zvieratá, ľudia, tornáda, atď. Tie poskytujú priestor na skúmanie a získavanie množstva zaujímavých poznatkov. Jedným z nich je určiť aj predikcia ich pohybu, ktorá je v dnešnej dobe často skloňovaným problémom. Poznatky, pri nej získané, sú neoceniteľnými súčasťami viacerých odvetví, ako je meteorológia, varovanie pred tornádami, smerovacie protokoly na internete, letecký priemysel a mnoho ďalších.

V súčasnej dobe existuje viacero metód predikcie. Na začiatku tejto práce sú vysvetlené princípy najznámejších z nich. Následne sa práca zameria na podrobnejšie vysvetlenie jednej z nich – predikciu pomocou frekventovaných vzorov, konkrétne algoritmus WhereNext.

Tento algoritmus najskôr rozdelí priestor do regiónov a na základe trajektórií objektov uložených v databázy sa vyberú tie regióny, cez ktoré prechádza požadovaný počet objektov. Následne sa získajú takzvané T-Patterny, ktoré predstavujú frekventované vzory pohybu objektov. Z nich sa neskôr vybuduje T-Pattern strom umožňujúci rýchle vyhľadanie potrebného vzoru. T-Pattern strom sa použije k samotnej predikcii s využitím zvolenej funkcie na určenie najvhodnejšieho vzoru pre pohyb objektu.

Posledným krokom je analýza spoľahlivosti predikcie tohto algoritmu a experimentálne overenie funkčnosti pri vopred zvolených parametroch. Experimenty sa uskutočnia na vygenerovanej dátovej sade pohybu áut, ktorá obsahuje trajektórie 65 tisíc objektov zložených z takmer štyroch miliónov bodov. Pre lepšiu prehľadnosť sú všetky body premietnuté do súradnicového systému jednej zemskej pologule.

Na záver zhodnotíme dosiahnuté výsledky a zo získaných vedomostí predostrieme vhodné pokračovanie tejto práce.

2 Získavanie znalostí z dát

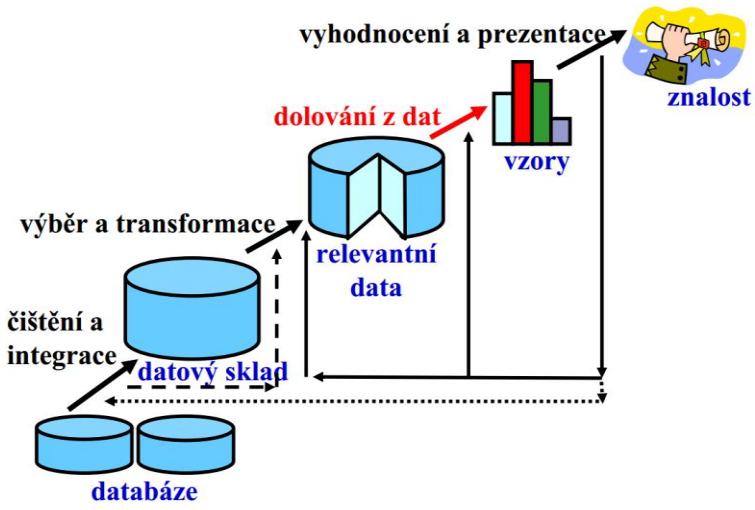
Ľudstvo disponuje obrovským objemom dát z najrôznejších zdrojov. Tie samy o sebe nie sú také cenné ako dôležité vzťahy a znalosti, ktoré obsahujú. Tento stav sa dá opísať ako: „Topíme se v dátach, ale trpíme nedostatkom znalostí.“ [1]. Vďaka týmto okolnostiam sa v súčasnosti získavaniu znalostí z dát venuje čoraz väčšia pozornosť.

Získavanie znalostí z dát (KDD z anglického Knowledge Discovery in Databases) podľa [2] znamená extrakciu zaujímavých informácií, ktoré sú *netriviálne* – nezískame ich napríklad obyčajným SQL dotazom, *skryté* – nie sú na prvý pohľad viditeľné, *skôr neznáme* a *potenciálne užitočné* – majú význam pre ďalšie rozhodovanie.

2.1 Proces získavania znalostí z dát

Pozrime sa teraz na získavanie znalostí z dát ako na proces rozdelený do niekoľkých krokov, ktoré sa môžu opakovať. Na obrázku 2.1 je prehľadne zobrazená chronológia týchto krokov.

Čistenie dát zabezpečí ich konzistenciu odstránením chýbajúcich dát a šumu v nich. *Integráciou* dosiahneme zlúčenie dát z rôznych zdrojov (často heterogénnych). Nie je ničím neobvyklým, keď sa vykonáva spolu s čistením dát, keďže práve rôzne zdroje dát sú častým zdrojom nekonzistencie. Cieľom *výberu dát* je selekcia relevantných údajov potrebných na riešenie daného problému. *Transformácia* zabezpečí konsolidovanú formu dát vhodnú pre dolovanie. Môže sa jednať napríklad o sumarizáciu alebo agregáciu dát. *Dolovanie* nám vytvorí model dát. Pomocou *hodnotenia modelov a vzorov* určíme najdôležitejšie vzory. Nakoniec môžeme *prezentáciou znalostí* ukázať výsledky celého procesu.



Obrázok 2.1: Kroky procesu získavania znalostí z dát (zdroj [2]).

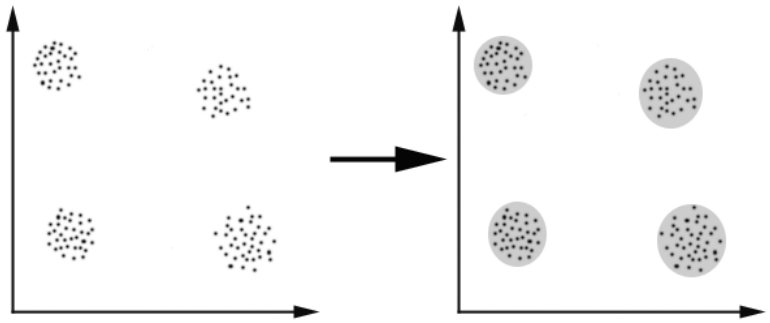
Predspracovanie dát

Reálne dáta sú zvyčajne nekvalitné (neúplnosť, nekonzistencia, ...), preto je potrebné ich okamžite na začiatku spracovať do korektnej podoby. Nízka kvalita dát by mohla následne implikovať nízku kvalitu dosiahnutých výsledkov. Preto predspracovanie môže tvoriť až 60% celkového úsilia [2] a môže sa uskutočňovať dokonca opakovane.

2.1.1 Dolovanie z dát

Dolovanie z dát je jadrom procesu získavania znalostí. Jeho úlohou je aplikácia vybranej metódy nad dátami, ktoré boli podrobené predspracovaniu podľa znalostí a dát. Poznáme štyri základné typy dolovacích úloh [1]:

- *Asociačné pravidlá* – hľadanie zaujímavých vzťahov medzi objektmi. Patrí sem napríklad analýza nákupného košíka a algoritmy ako Apriori, DHP, MaxMiner, atď.
- *Zhlukovanie* – proces hľadania podobností medzi dátami a rozdeľovanie objektov do tried na základe podobnosti, napríklad obrázok 2.2.
- *Klasifikácia* – zaradenie daného objektu do istej triedy na základe jeho vlastností, skladá sa z troch fáz – trénovanie, testovanie a použitie.
- *Predikcia* – predikcia istej hodnoty (zo spojitej funkcie) pre daný objekt.



Obrázok 2.2: Príklad zhlukovania (zdroj [3]).

2.2 Využitie

Uplatnenie získaných znalostí je veľmi široké [1, 2, 4].

Typickou aplikáciou dolovania z dát je *analýza nákupného košíka*. Hlavnou úlohou v tomto prípade je objavenie frekventovaných vzorov, čo predstavuje tovar, ktorý zákazníci často kupujú spoločne. Uskutočňuje sa to analýzou jednotlivých nákupov zákazníkov. Ak si napríklad zákazník kúpi nový LCD monitor, často si k nemu objednáva aj HDMI alebo VGA kábel. Z toho dôvodu je vhodné mu tieto položky pri zobrazení monitoru priamo ponúknuť.

Jednou z ďalších oblastí využitia je *analýza trhu a marketing*. V tomto prípade získané znalosti pochádzajú z dát o uskutočnených transakciách a zákazníkoch, ktorí ich uskutočnili. Výsledkom je objavenie závislostí medzi spôsobom útraty peňazí, výškou príjmu a záujmami rôznych skupín zákazníkov. Významnú úlohu hrá dolovanie v dátach pri *finančnej analýze a riadení rizík*. Využíva sa pri predikcii rizík poskytnutia úverov a vývoji hodnôt akcií v čase. Nezanedbateľný význam má v *bioinformatike a analýze biologických dát*. Tu sa dolovanie z dát používa na dokazovanie a formuláciu nových hypotéz. Odhaľovať poistné podvody pri dopravných nehodách môžeme za pomoci *detekcie podvodov a neobvyklého chovania*. Pomerne netypické požiadavky na dolovanie znalostí (online analýza „nekonečného“ prúdu dát) kladie *získavanie znalostí v prúdoch dát*.

Ďalšie uplatnenie môžeme nájsť v *medicíne* (mapovanie ľudského genómu), *získavaní znalostí z textu na webe, priemysle* (predikcia spotreby) atď.

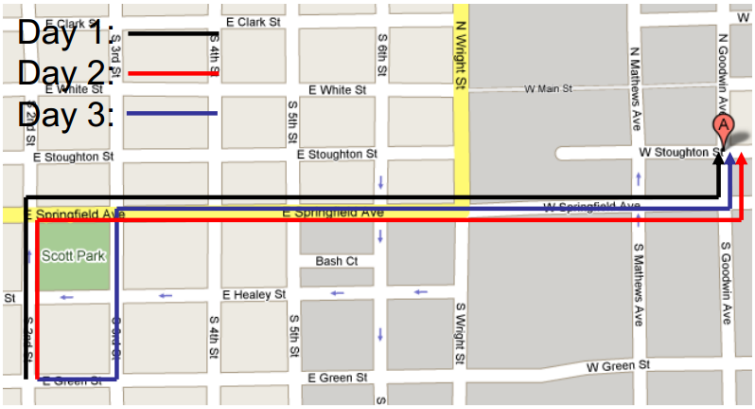
3 Dolovanie v trajektóriách

Trajektória obecne predstavuje *sled polôh*, ktoré hmotný bod počas svojho pohybu, vzhľadom na zvolenú súradnicovú sústavu, zaujíma [5]. Dáta pohybujúceho sa objektu sú tvorené *sekvenciu lokácií* (často GPS súradnice) a *časových značiek* (anglicky timestamp) [6]. Dolovanie v trajektóriách našlo významné uplatnenie v ekologickej analýze, monitorovaní hraníc, kamerovom dohľade, predpovedi počasia a riadení dopravy.

Najdôležitejšie úlohy dolovania v trajektóriách pohybujúcich sa objektov tvorí dolovanie periodických vzorov, frekventovaných vzorov, predikcia, klasifikácia, zhluková analýza a detekcia anomálií.

Dolovanie periodických vzorov

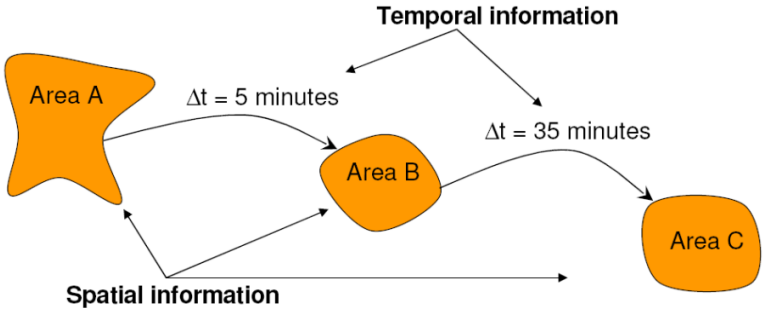
V mnohých prípadoch objekty nasledujú rovnakú trasu (aproximovane) v pravidelných časových intervaloch. Napríklad človek vstáva ráno v rovnaký čas a nasleduje viac-menej rovnakú trasu do práce. Na obrázku 3.1 je zobrazený tento prípad.



Obrázok 3.1: Príklad dolovania periodických vzorov (zdroj [6]).

Dolovanie frekventovaných vzorov

Má princíp sčasti podobný ako dolovanie periodických vzorov. Ide o objavovanie častých vzorov v pohybe všetkých objektov [7]. Je opakom detekcií anomálií. Na obrázku 3.2 je zobrazený príklad tejto úlohy.

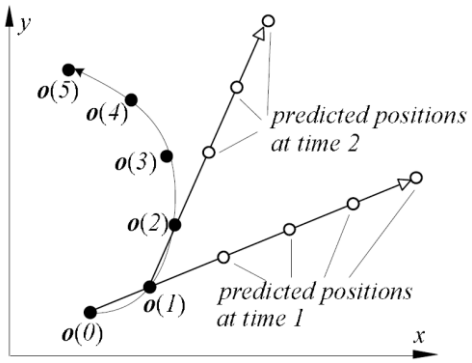


Obrázok 3.2: Príklad dolovania frekventovaných vzorov (zdroj [6]).

Predikcia

Predikcia v tomto kontexte slúži na predpovedanie budúcej lokácie pohybujúceho sa objektu. Môže byť založená na dvoch hlavných faktoch – na vlastnej histórii pohybu objektu alebo na trajektóriách

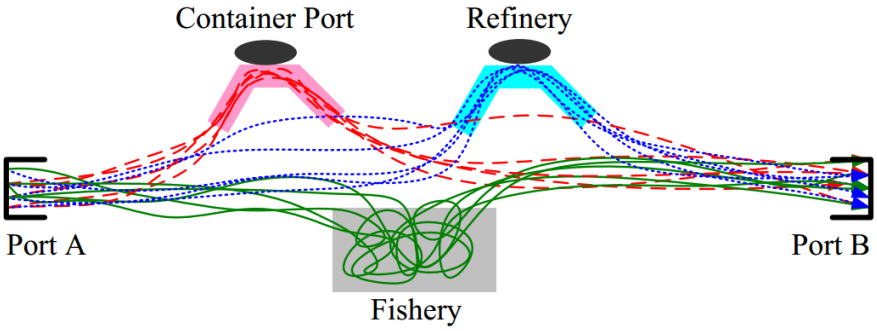
všetkých pohybujúcich sa objektov. Predikciou sa budeme podrobnejšie zaoberať v ďalších častiach tejto práce. Obrázok 3.3 zobrazuje ukážku predikcie.



Obrázok 3.3: Predikcia založená na vlastnej histórii pohybu objektu (zdroj [8]).

Klasifikácia

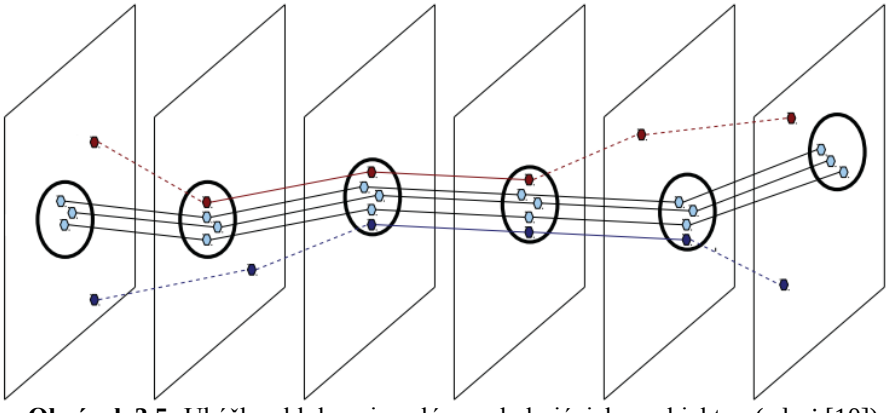
Úlohou klasifikácie pri dolovaní v trajektóriách je roztrieďovanie pohybujúcich sa objektov do tried na základe ich trajektórií a ďalších vlastností. Stretávame sa tu s dvoma hlavnými prístupmi – strojové učenie a klasifikácia založená na trajektóriách (TraClass). Na obrázku 3.4 je uvedený príklad výsledku získaného klasifikáciou založenou na trajektóriách.



Obrázok 3.4: Príklad výsledku klasifikácie (zdroj [6]).

Zhluková analýza

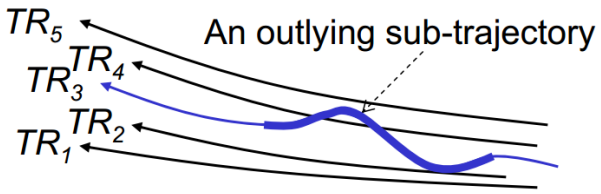
Zhluk je množina objektov pohybujúcich sa spoločne blízko seba dlhší časový interval [9]. Touto množinou môže byť napríklad konvoj áut pohybujúcich sa po meste alebo skupina migrujúcich zvierat. Na obrázku 3.5 môžeme vidieť príklad zhlukovej analýzy.



Obrázok 3.5: Ukážka zhlukovej analýzy pohybujúcich sa objektov (zdroj [10]).

Detekcia anomálií

Slúži na detekciu podozrivého alebo neobvyklého správania pohybujúceho sa objektu, ktorý iné metódy označujú ako šum. Vo väčšine prípadov ide o hľadanie takých pohybov, ktoré sa výrazne odlišujú od väčšiny ostatných [11]. Na obrázku 3.6 smieme vidieť takýto pohyb**.**



Obrázok 3.6: Ukážka detekcie anomálií(zdroj [11])

4 Predikcia budúcej lokácie pohybujúceho sa objektu

Za *pohybujúci sa objekt* považujeme akékoľvek hmotné teleso meniace svoju polohu, vzhľadom k zvolenému súradnicovému systému, v čase. Na predikciu jeho budúcej polohy používame niekoľko metód [6].

Lineárna predikcia

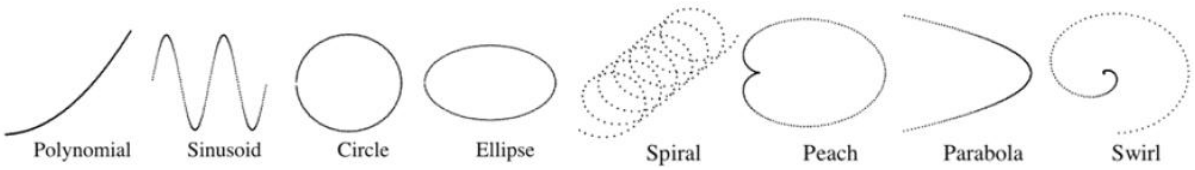
Obrázok 3.3 zobrazuje príklad lineárnej predikcie. Ide o najjednoduchšiu formu predikcie. Už z obrázku je možné vidieť, že táto metóda môže dávať v mnohých prípadoch veľmi nepresné výsledky, čo je jej najväčšia nevýhoda. Na predikciu používa výhradne históriu pohybu daného objektu, ostatné trajektórie sú nepodstatné. Využíva sa na krátkodobú predikciu.

Vektorová predikcia

Táto predikcia berie taktiež do úvahy iba históriu pohybu objektu, ktorého pohyb predikujeme. Využíva rekurzívnu pohybovú funkciu

$$o(t) = C_1.o(t-1) + C_2.o(t-2) + \cdots + C_f.o(t-f) \tag{4.1}$$

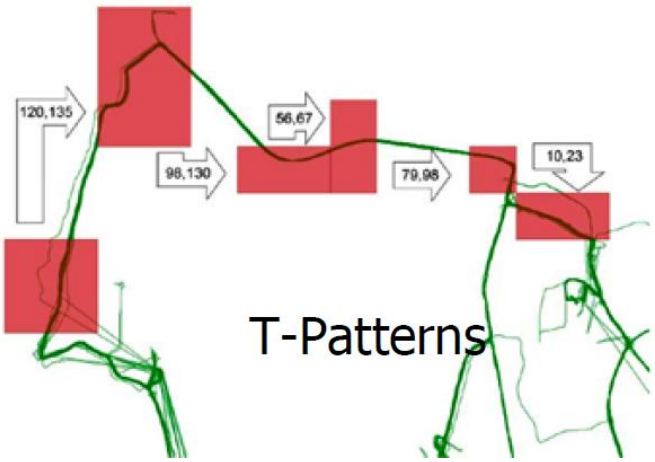
kde C_i predstavuje konštantnú maticu vyjadrujúcu niektorý z komplexných typov pohybu (po kružnici, polynomiálny, po elipse, ...), viď obrázok 4.1. Premenná t zastupuje čas predikcie. Vektorová predikcia je použiteľná pri krátkodobej predikcii [8].



Obrázok 4.1: Niekoľko komplexných pohybových typov (zdroj [8]).

Predikcia založená na frekventovaných vzoroch

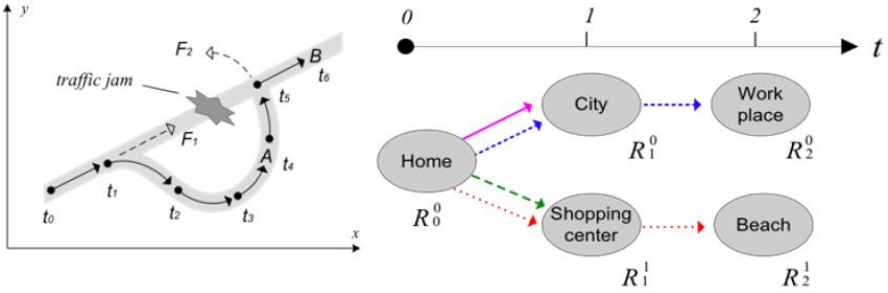
Ak veľké množstvo objektov zdieľa určitú trajektóriu, je pravdepodobné, že daný objekt ju bude zdieľať taktiež. Metódy sem patriace získajú najskôr frekventované vzory pohybu a na nich postavia svoju predikciu. K predikcii sa teda využívajú trajektórie všetkých objektov. Zástupcom tohto druhu predikcie je aj algoritmus *WhereNext*, ktorý bude podrobne vysvetlený v kapitole 5. Výhodou tohto prístupu je možnosť dlhodobej predikcie.



Obrázok 4.2: Predikcia založená na frekventovaných vzoroch (zdroj [12]).

Hybridná predikcia

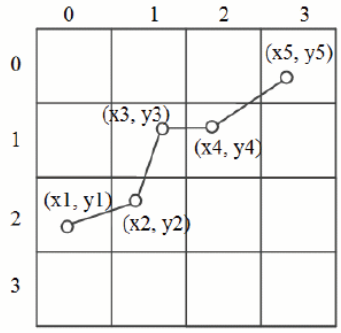
Hybridná predikcia v sebe zahŕňa kombináciu vektorovej predikcie a predikcie založenej na frekventovaných vzoroch. Na predikciu využíva indexovanie vzorov uložených v strome vzorov pohybu. Dokáže predikovať časovo blízku aj vzdialenú lokáciu. Ukážku hybridnej predikcie zobrazuje obrázok 4.3.



Obrázok 4.3: Príklad hybridnej predikcie (zdroj [13]).

Predikcia pomocou Markovských reťazcov

Táto metóda rozdeľuje priestor do buniek a následne sleduje postupnosť pohybu medzi bunkami, čo nakoniec využije na predikciu. Na predikciu sa využívajú trajektórie všetkých objektov. Nevýhodou je možnosť len krátkodobej predikcie a nepresnosť tejto metódy. Ukážku tejto predikcie zobrazuje obrázok 4.4.



Obrázok 4.4: Predikcia pomocou Markovských reťazcov (zdroj [14]).

5 Algoritmus WhereNext

Táto kapitola vychádza z algoritmu WhereNext popísaného v [12]. Metóda predikcie, ktorú využíva tento algoritmus, je založená na využívaní frekventovaných vzorov pohybu. Tie je treba vopred v dátach objaviť. Tieto vzory, ktoré predstavujú často uskutočňované správanie pohybu objektov, WhereNext nazýva *T-Patterny*. Algoritmus sa skladá z niekoľkých hlavných podproblémov, ktoré rieši v presne stanovenom poradí.

5.1 Vyhľadávanie frekventovaných lokalít

Počiatočnou úlohou je objavenie frekventovaných lokalít. Uskutočňuje sa to diskretizáciou priestoru, v ktorom sa pohybujú objekty do štvorcov s veľkosťou strany zadanou užívateľom. Pod pojmom lokalita rozumieme časopriestorové primitíva, ktoré pohybujúce sa objekty pretínajú svojimi trajektóriami. Predstavuje ich ohraničená priestorová plocha a časový interval. Lokalitu považujeme za frekventovanú v tom prípade, že cez ňu prechádza veľké množstvo trajektórií (objektov). Prah určujúci minimálny počet trajektórií prechádzajúcich cez lokalitu na to, aby sa stala frekventovanou, je zadaný ako jeden z parametrov algoritmu. Nazýva sa *minimálna podpora* a označuje sa gréckym písmenom σ. Tento parameter má významný vplyv na neskoršiu predikciu. Ak by bol priveľmi nízky, prípadne vysoký, znamenalo by to často veľmi nepresnú predikciu. V mnohých prípadoch by sa žiadna dokonca nemusela nájsť.

Po diskretizácii priestoru, prejdení trajektórií všetkých objektov, inkrementácii podpory lokalít, cez ktoré tieto trajektórie prechádzajú a aplikácií prahu určeného minimálnou podporou σ, dostávame vyselektovaný priestor obsahujúci frekventované lokality. Zvyšok priestoru sa pre nás stáva bezpredmetný.

5.2 Získavanie T-Patternov

Trajektória pohybu objektu predstavuje sekvenciu lokácií s časovým odtlačkom. Lokácia je daná napríklad globálnym systémom na určenie polohy (GPS), v ktorom je poloha určená zemepisnou šírkou a dĺžkou.

DEFINÍCIA 1: *Trajektória pohybujúceho sa objektu sa dá teda opísať ako sekvencia trojíc*

$$T = < s_0, d_0, t_0 >, \ldots, < s_n, d_n, t_n >$$

kde s_i a d_i predstavujú lokalizáciu tohto objektu v GPS súradniciach v danom čase t_i.

Nasledujúcim krokom algoritmu je hľadanie frekventovaných vzorov pohybu, tzv. T-Patternov (Trajectory Pattern). Tie sú nevyhnutné pre následné budovanie T-Pattern stromu. Môžeme ich definovať ako usporiadané sekvencie lokalít, medzi ktorými sú typické časové intervaly prechodu nasledované s veľkou frekvenciou.

DEFINÍCIA 2: *T-Pattern predstavuje dvojicu (L, T), kde $L = <l_0, \ldots, l_n>$ je sekvencia frekventovaných priestorových lokalít, cez ktoré objekt prechádza a $T = <t_0, \ldots, t_{n-1}>$ je časový popis udávajúci čas presunu medzi lokalitami L. Dvojicu (L, T) reprezentuje taktiež zápis*

$$l_0 \xrightarrow{t_0} l_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{n-1}} l_n$$

Na obrázku 4.2 si pre lepšiu predstavu môžeme priblížiť tento problém. Za príklad T-Patternu pohybu ľudí v meste môžeme považovať nasledujúci vzor:

$$Tesco \xrightarrow{30\ min} McDonald´s \xrightarrow{10\ min} Kino$$

Zisk týchto vzorov zabezpečuje algoritmus 1. T-Patterny podobne ako frekventované lokality obsahujú atribút podpory. Ten nám určuje počet objektov, ktorých pohyb tento vzor opisuje. Ak sa teda tisíc ľudí išlo z Tesca najesť do McDonald´s a potom si išli pozrieť film do kina, znamená to, že podpora tohto vzoru je rovná tisícke.

Algoritmus 1: FindTrajectoryPatterns(*fLocList, trajList*)

Vstup: Zoznam frekventovaných lokalít *fLocSet*, trajektórie pohybu všetkých objektov *trajList*
Výstup: Zoznam T-Patternov *tPatternList*

```
1.  tPatternList = new List() of tPattern
2.  foreach traj in trajList do
3.      tp = new tPattern()
4.      foreach fLoc in fLocList do
5.          if fLoc contains traj then
6.              if tp.isEmpty() then
7.                  tp.add(fLoc, 0)
8.              else
9.                  tp.add(fLoc, transferTime)
10.             end
11.         end
12.     end
13.     tPatternList.add(tp)
14. end
15. return tPatternList
```

5.3 Budovanie T-Pattern stromu

T-Pattern strom predstavuje efektívnu vyhľadávaciu štruktúru v T-Patternoch. Využíva sa pri ňom stromová dátová štruktúra. Táto metóda je založená na princípe budovania klasifikátorov za použitia asociačných pravidiel, ktoré sú v tomto prípade získane z T-Patternov. Každá cesta v strome predstavuje jedno pravidlo.

DEFINÍCIA 3: *T-Pattern strom reprezentuje dvojica (N, Root), kde N predstavuje zoznam uzlov stromu a Root* $\in N$ *je koreň stromu. Nech uzol* $U \in N$ *a zároveň* $U \neq Root$*, potom U tvorí usporiadaná trojica (L, S, I), kde L je frekventovaná lokalita, S predstavuje podporu daného uzla a I je interval tvaru [min, max]. Min určuje minimálny čas prechodu z predchádzajúceho uzla (frekventovanej lokality) do aktuálneho uzla (frekventovanej lokality) a max naopak maximálny čas prechodu.*

Algoritmus 2 ukazuje postup pri vytváraní T-Pattern stromu. Použitá metóda *findChild()* vyhľadáva potomka uzla stromu, ktorý zahŕňa určenú lokalitu. O úpravu intervalu uzla sa stará metóda *updateInterval()*. Tá rozťahuje interval uzla tak, aby interval lokality z T-Patternu do neho patril. Metóda *updateSupport()* nadstaví podporu uzla na hodnotu podpory T-Patternu, ak platí podmienka, že podpora T-Patternu je väčšia ako podpora uzla. Ak podmienka neplatí, podpora sa nezmení.

Algoritmus 2: CreateTPatternTree(*tPatternList*)

Vstup: Zoznam T-Patternov *tPatternList*
Výstup: T-Pattern strom *tpTree*

```
1.  tpTree = new TPatternTree
2.  foreach tPattern in tPatternList do
3.      node = tPatternTree.root()
4.      foreach locality in tPattern.localities do
5.          tmpNode = findChild(node,locality)
6.          if tmpNode == null then
7.              tmpNode = new Node(locality)
8.              tmpNode.support = tPattern.support
9.              node.addChild(tmpNode)
10.             node = tmpNode
11.         else
12.             tmpNode.updateInterval(locality.interval)
13.             tmpNode.updateSupport(tPattern.support)
14.             node = tmpNode
15.         end
16.     end
17. end
18. return tpTree
```

Uzly, ktoré sú priamymi potomkami koreňa stromu, majú minimum aj maximum časového intervalu rovné nule, keďže koreň stromu je iba virtuálnym uzlom a jeho potomkovia teda predstavujú počiatočný uzol cesty. Každý uzol, okrem koreňového uzla, má práve jedného rodiča. Uzol v sebe taktiež nesie informáciu o tom, ktoré uzly sú jeho priamymi potomkami.

Správne vybudovanie T-Pattern stromu je jedným z kľúčových krokov v algoritme WhereNext. Vyhľadávanie v ňom sa uskutočňuje na základe objavovaní prefixov podľa nasledujúcej definície:

DEFINÍCIA 4: *Nech dvojica (L, T)=* $l_0 \xrightarrow{t_0} l_1 \xrightarrow{t_1} \dots \xrightarrow{t_{n-1}} l_n$ *a dvojica (L', T') =* $l_0 \xrightarrow{t_0'} l_1 \xrightarrow{t_1'} \dots \xrightarrow{t_{k-1}'} l_k$ *, potom (L', T') je prefixom (L, T) v tom prípade, že* $k \leq n$ *a* $\forall i = 1 \dots k-1$ *je obsiahnuté* t_i' *v intervale* t_i *.*

Pred vytvorením T-Pattern stromu je potrebné „zgrupovanie“ T-Patternov. Znamená to ich zoskupenie podľa frekventovaných lokalít, ktorými prechádzajú. Pri zoskupovaní berieme do úvahy poradie lokalít. Cieľom tohto kroku je odstránenie duplicít, zistenie podpory daného vzoru a nadstavenie intervalov prechodu medzi lokalitami. Ako príklad majme tri T-Patterny tvaru (*t* predstavuje časovú jednotku):

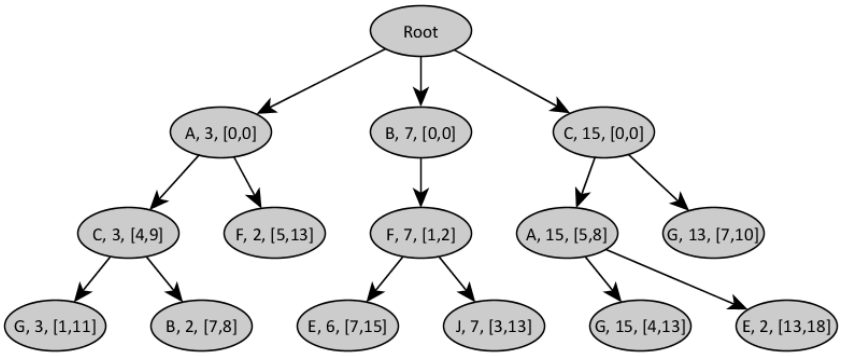
$$A \xrightarrow{4t} C \xrightarrow{8t} G$$
$$A \xrightarrow{7t} C \xrightarrow{11t} G$$
$$A \xrightarrow{9t} C \xrightarrow{1t} G$$

ktoré zoskupíme do jediného vzoru v tvare *<A, [0,0]> <C, [4,9]> <G, [1,11]>, podpora = 3*. Veľké písmeno značí frekventovanú lokalitu a čísla v hranatých zátvorkách určujú interval prechodu medzi danými lokalitami v tvare [minimum, maximum]. Podpora zoskupeného vzoru je daná počtom T-Patternov, ktoré boli použité na jeho vytvorenie.

Ukážme si teda, ako T-Pattern strom vytvárať. Majme množinu zoskupených T-Patternov tvaru:

$$< A,[0,0] > < C,[4,6] > < G,[1,11] > \quad podpora = 3$$
$$< A,[0,0] > < C,[5,9] > < B,[7,8] > \quad podpora = 2$$
$$< A,[0,0] > < F,[5,13] > \quad podpora = 2$$
$$< B,[0,0] > < F,[1,1] > < E,[7,15] > \quad podpora = 6$$
$$< B,[0,0] > < F,[1,2] > < J,[3,13] > \quad podpora = 7$$
$$< C,[0,0] > < A,[5,7] > < G,[4,13] > \quad podpora = 15$$
$$< C,[0,0] > < A,[8,8] > < E,[13,18] > \quad podpora = 2$$
$$< C,[0,0] > < G,[7,10] > \quad podpora = 13$$

z ktorých za pomoci algoritmu 2 vznikne strom zobrazený na obrázku 5.1.



Obrázok 5.1: T-Pattern strom.

5.4 Predikcia

Hlavnou úlohou predikcie v algoritme WhereNext je objavenie vzoru pohybu, ktorý čo možno s najväčšou presnosťou, opisuje trajektóriu pohybujúceho sa telesa a na základe neho uskutočniť samotnú predikciu. Postup predikcie je uvedený v algoritme 3. Vzor pohybu známej časti trajektórie získame zavolaním algoritmu 1 nad tou častou predikovanej trajektórie, ktorú poznáme. Mieru zhodnosti T-Patternu objektu a jednotlivých ciest v T-Pattern strome určuje tzv. *punctual skóre* (pScore). Toto skóre sa počíta pre každý uzol stromu. Ak však istému uzlu je vypočítané pScore s nulovou hodnotou, tak to znamená, že skóre jeho potomkov sa ďalej nebude počítať a ostane nulové. Na ohodnotenie uzlov slúži funkcia *evaluateAllNodes()*. Po jej skončení máme kompletne ohodnotený celý T-Pattern strom a môžeme z neho vyselektovať pre danú trajektóriu najvhodnejší vzor (prípadne vzory) za pomoci funkcie *findBestNodes()*, ktorá dostáva ako parameter typ funkcie, podľa ktorej sa tento vzor vyberie. Z tohto vzoru je kľúčovým uzol, ktorý ma najväčšiu vzdialenosť od koreňa a zároveň ma nenulové pScore. Je to z toho dôvodu, že tento uzol sa stane odrazovým mostíkom predikcie algoritmu WhereNext. Všetci jeho potomkovia sa stávajú eventuálnou predikciou lokácie. Nad týmto uzlom je volaná funkcia *findAllOffsprings()*, ktorá týchto potomkov objaví. Keďže v mnohých prípadoch je potomkov väčšie množstvo, musíme použiť pre predikciu pravdepodobnosť. Za predikciu vyhlásime nakoniec ten vzor, ktorý prechádza našim kľúčovým uzlom a má najväčšiu podporu. Konkrétne tú časť vzoru, ktorá pokračuje za týmto uzlom.

Algoritmus 3: Prediction(*tPTree, objTPattern, config*)

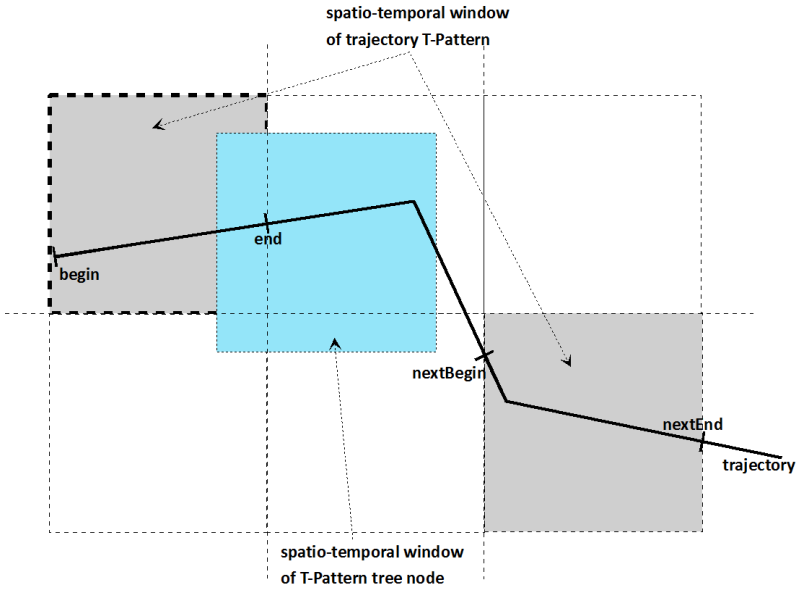
Vstup: Strom *tPTree*, vzor pohybu známej časti trajektórie objektu *objTPattern*, konfigurácia *config*
Výstup: Zoznam možných predikcií *predictList*

1. evaluateAllNodes(tPTree, objTPattern, config)
2. bestNodes = findBestNodes(tPTree, config.scoreFunction)
3. predictList = findAllOffsprings(tPTree, bestNodes)
4. **return** predictList

5.4.1 Punctual skóre

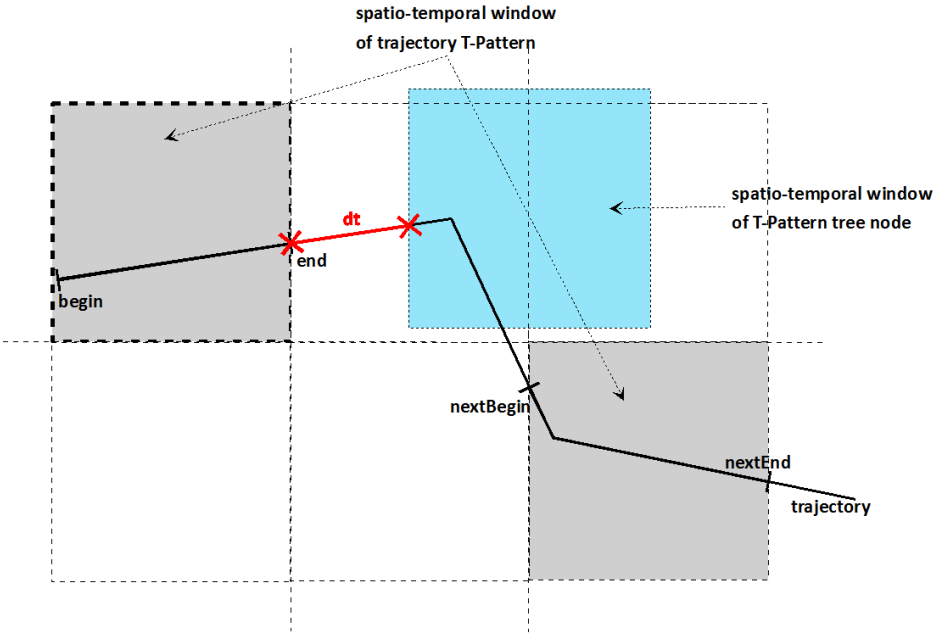
Punctual score určuje presnosť, s akou daný uzol dokáže opísať položku T-Patternu trajektórie pohybujúceho sa objektu, ktorá by k nemu mala prislúchať. Opisuje teda spoľahlivosť uzla *node* relatívne k trajektórii *T*. Pri výpočte pScore uzla sa môžeme stretnúť s tromi rôznymi prípadmi.

Prvým z nich je prípad, kedy časopriestorové okno uzla stromu pretína časopriestorové okno položky T-Patternu. V tomto prípade punctual skóre počítame ako *pScore = node.support*, viď obrázok 5.2.



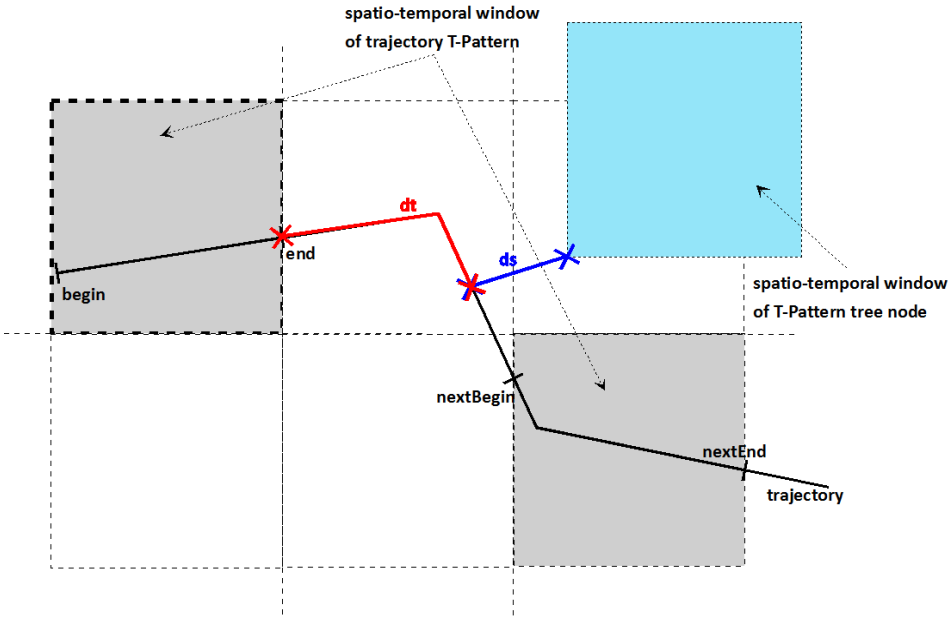
Obrázok 5.2: Priamy prienik časopriestorových okien.

Ďalším prípadom je, keď sa síce časopriestorové okná nepretnú priamo, ale s použitím časovej tolerancie to môžeme dosiahnuť. Okno predstavujúce uzol T-Pattern stromu, pri nepretnutí rozšírime o maximálnu časovú toleranciu th_s a následne skontrolujeme či došlo k prieniku. Ak áno, zistíme veľkosť časovej tolerancie *dt*, o ktorú musíme okno rozšíriť, aby došlo k prieniku. Punctual score tu počítame ako *pScore = node.support / (β * dt)*, viď obrázok 5.3. Parameter *β* nám umožňuje nadstaviť dôležitosť časového faktoru.



Obrázok 5.3: Prienik časopriestorových okien za použitia časovej tolerancie.

Poslednou možnosťou je, že sa časopriestorové okná nepretnú ani po použití časovej tolerancie th_s, viď obrázok 5.4. V tomto prípade treba najskôr nájsť najbližší bod trajektórie T objektu k oknu, ktoré predstavuje uzol. Z tohto bodu vyrátame, aký je časový posun medzi oknom trajektórie a najbližším bodom. Tento princíp je zobrazený na obrázku 5.4. Punctual skóre pri tomto prípade vyrátame podľa rovnice *pScore = node.support / (β * dt + α * ds)*. Parameter *β* rovnako ako v druhom prípade určuje váhu časového faktoru a *α* zase priestorového faktoru. Ak však pri výpočte zistíme, že potrebné *ds* je väčšie ako maximálna priestorová tolerancia th_s, prípadne veľkosť *dt* presiahne maximálnu časovú toleranciu th_t, pScore nadobudne automaticky hodnotu nula.



Obrázok 5.4: Prienik časopriestorových okien za použitia časovej a priestorovej tolerancie.

Použitie časovej tolerancie th_t nám teda umožňuje okno uzla T-Pattern stromu posúvať všetkými smermi bez zväčšovania, čoho konečný výsledkom je ale zväčšenie pôvodného okna. Priestorová tolerancia th_s nám okno uzla priamo iba rozšíri všetkými smermi.

5.4.2 Celkové skóre cesty

Z ohodnotenia jednotlivých uzlov algoritmus WhereNext počíta ohodnotenie jednotlivých ciest v strome. Pre výber najlepšej cesty môžeme použiť jednu z troch funkcií. Každá pritom môže vrátiť odlišný výsledok.

Prvou je funkcia *avgScore*, ktorá vypočíta ohodnotenie celej cesty v strome ako priemer pScore všetkých jej uzlov. Táto metóda sa snaží zovšeobecniť koncept podobnosti ako priemernú vzdialenosť medzi trajektóriou a každým uzlom T-Pattern stromu.

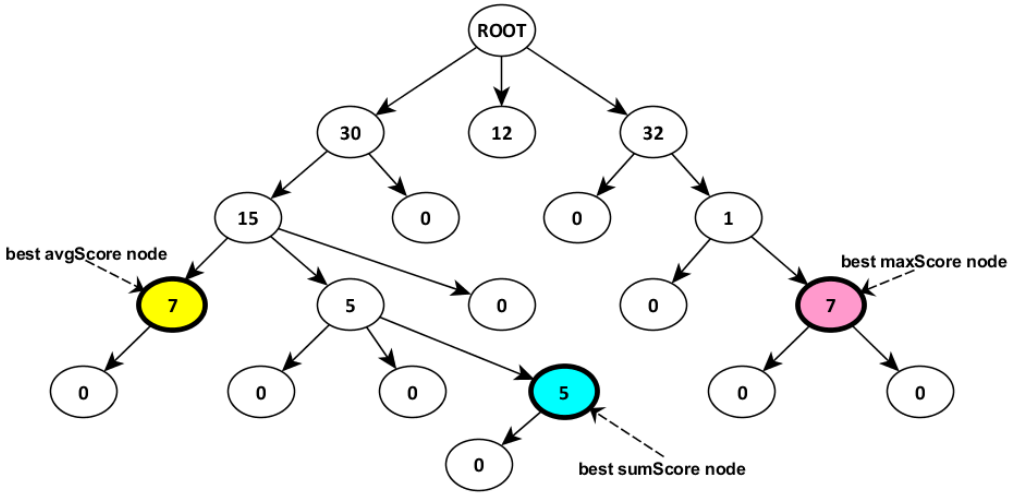
$$avgScore = \frac{\sum_{i=1}^{n} pScore_i}{n} \tag{5.1}$$

Druhou je funkcia *sumScore*, ktorá definuje path score cesty ako súčet pScore jednotlivých uzlov stromu. Je založená na koncepte hĺbky. Z toho vyplýva, že dáva prednosť čo možno najdlhším cestám stromu, ktoré opisujú danú trajektóriu.

$$sumScore = \sum_{i=1}^{n} pScore_i \tag{5.2}$$

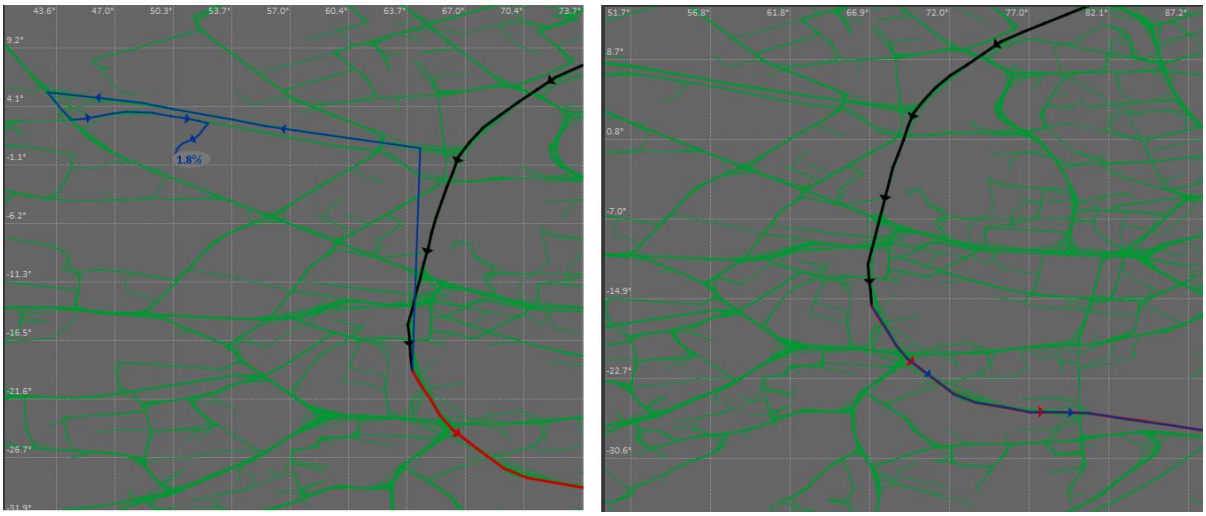
Posledná je funkcia *maxScore*. Je založená na optimistickom princípe. Za celkové skóre cesty určí najvyššie pScore, ktoré v ceste nájde. Za najlepšiu cestu teda určí tú, ktorá najlepšie opíše trajektóriu v akejkoľvek jej časti.

$$maxScore = Max\{pScore_0, \dots, pScore_i\} \tag{5.3}$$



Obrázok 5.5: T-Pattern strom s uzlami ohodnotenými pScore.

Na obrázku 5.5 môžeme vidieť T-Pattern strom s uzlami, ktoré sú ohodnotené pomocou pScore. Je tu prehľadne zobrazený výsledok metódy *findBestNodes()* zavolaný s každou z troch možností na ohodnotenie ciest v strome. Táto metóda vracia vždy až posledný nenulový uzol cesty, ktorú vyhodnotí za najlepšiu. Podľa článku [12] by mala vracať každý uzol najlepšej cesty, ktorý má niektorého z potomkov s nulovým pScore. To však môže spôsobiť značne nepresnú predikciu. Ako príklad uveďme obrázok 5.6, kde bola použitá funkcia maxScore podľa článku [12] a napravo podľa postupu uvedeného v tejto práci. Čierna čiara je známa časť trajektórie, červená je originálna časť trajektórie objektu, ktorú by mala predikcia (modrá čiara) čo najlepšie aproximovať. Vľavo je vidieť zreteľný skok, ktorý ukazuje spomínanú chybu.



Obrázok 5.6: Porovnanie prístupov predikcie. Vľavo nepresný prístup podľa [12] a vpravo môj presnejší.

5.4.3 Parametre predikcie

Algoritmus 3 potrebuje k svojej činnosti niekoľko vstupných parametrov, ktorých hodnoty sú kľúčové pre presnosť získanej predikcie. Nakoľko parametre α a β boli vysvetlené v predchádzajúcom texte, ostávajú nám už len štyri:

- *Agregačná funkcia (th_{agg})* – tento parameter určí, ktorá z metód uvedených v odseku 5.4.2 sa využije na výpočet path score.
- *Prah skóre (th_{score})* – určí minimálnu hodnotu celkového skóre najlepšej cesty, ktorá je považovaná za akceptovateľnú, to znamená, že ak je najlepšie nájdené skóre menšie, predikcia nevráti žiaden výsledok.
- *Priestorová tolerancia (th_s)* – nadstavuje veľkosť, o ktorú sa bude zväčšovať časopriestorové okno predstavujúce uzol stromu. Presnejší popis je uvedený v kapitole 5.4.1.
- *Časová tolerancia (th_t)* – reprezentuje maximálnu veľkosť, o ktorú môžeme posúvať časopriestorové okno uzla. Presnejší popis je uvedený v kapitole 5.4.1.

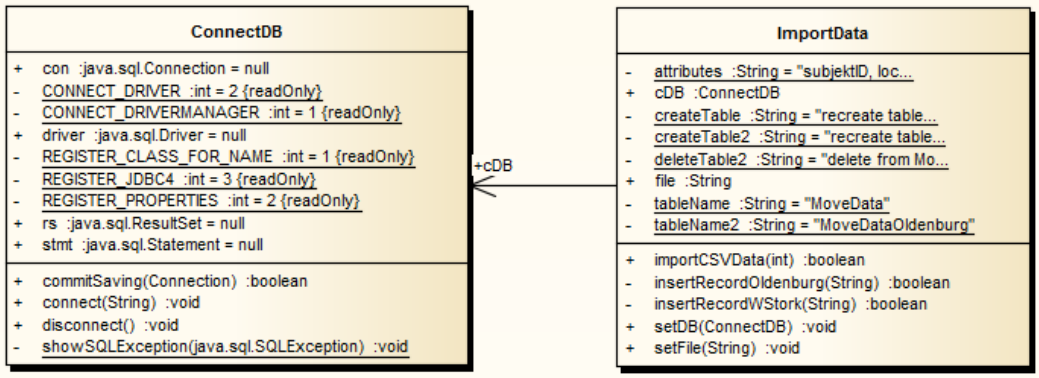
6 Návrh a implementácia

Jazyk na implementáciu aplikácie bola zvolená Java. Dôvodom tohto kroku bola jej prenositeľnosť medzi rôznymi platformami vďaka virtuálnemu stroju, nad ktorým beží v nej implementované aplikácie. Druhým hlavným dôvodom bola jej objektová orientácia, vďaka ktorej poskytuje vysokú prehľadnosť zdrojového kódu a jednoduchú neskoršiu modifikovateľnosť. Vynikajúcim pomocníkom bolo určite aj vývojové prostredie NetBeans, ktoré umožnilo kvalitné debuggovanie a nakoniec automatické vygenerovanie dokumentácie.
Aplikácia bola od začiatku vyvíjaná spolu s grafickým užívateľským rozhraním, ktoré umožňuje jej jednoduché intuitívne ovládanie a užívateľovi prehľadne dokáže zobraziť výsledok predikcie. Celá aplikácia bola rozdelená na tri balíky, z ktorých má každý svoju špecifickú funkciu.

Pre ukladanie a prácu s dátami bola zvolená databáza Firebird verzie 2.5[1]. Táto univerzálna databáza s voľne šíriteľnými zdrojovými kódmi poskytuje na stiahnutie Jaybird driver, ktorý zabezpečí jej spoluprácu s Javou. Veľmi nápomocným nástrojom na administráciu Firebird databáz sa stal FlameRobin, ktorý sa taktiež radí medzi aplikácie s voľne šíriteľnými zdrojovými kódmi.

6.1 Balík Database

Už z názvu balíku môžeme vidieť, že tento balík zabezpečuje prácu s databázou obsahujúcou dáta. Je to najmenší z troch balíkov a tvoria ho len 2 triedy, viď obrázok 6.1.

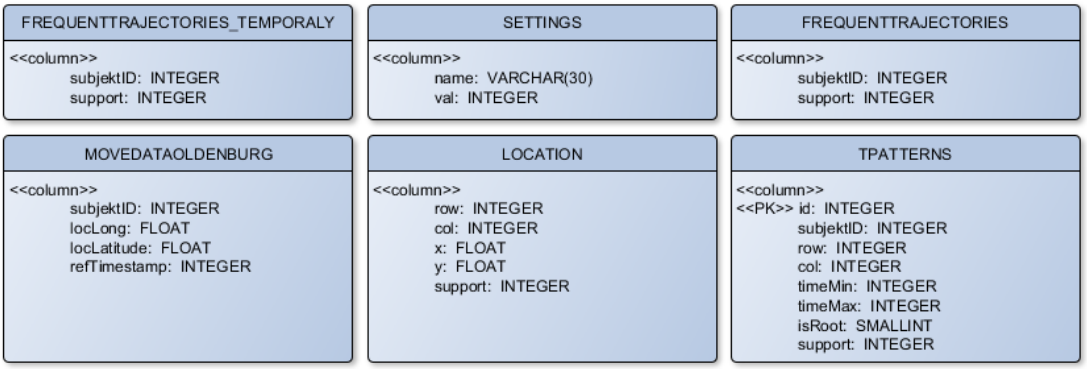


Obrázok 6.1: Diagram tried balíka Database.

Hlavnou úlohou triedy *ConnectDB* je pripojenie sa aplikácie pri jej štarte k zvolenej databázy a pri ukončení aplikácie pripojenie uzavrieť. Je tu taktiež obsiahnutá metóda pre premietnutie uskutočnených transakcií do databázy (commit).

Druhou triedou tohto balíka je *ImportData*. Jej metódy umožňujú import dát, uložených v textovej podobe vo formáte CSV, do databázy. Oddeľovače stĺpcov a čísla stĺpcov, ktoré sa importujú, sú nadstavené pevne na formát, v akom sú uložené dáta v súboroch, ktoré boli využívané. Pri importe súradníc sa tieto automaticky upravujú tak, aby sa trajektórie rozprestreli na celú jednu zemskú pologuľu. Keďže sú použité dáta vygenerované, môžeme si to dovoliť. Hlavným dôvodom prepočtu je ten, aby bola zabezpečená lepšia prehľadnosť na mape.

[1] Voľne dostupná aplikácia na webovej stránke http://www.firebirdsql.org/en/firebird-2-5-2-upd1/.



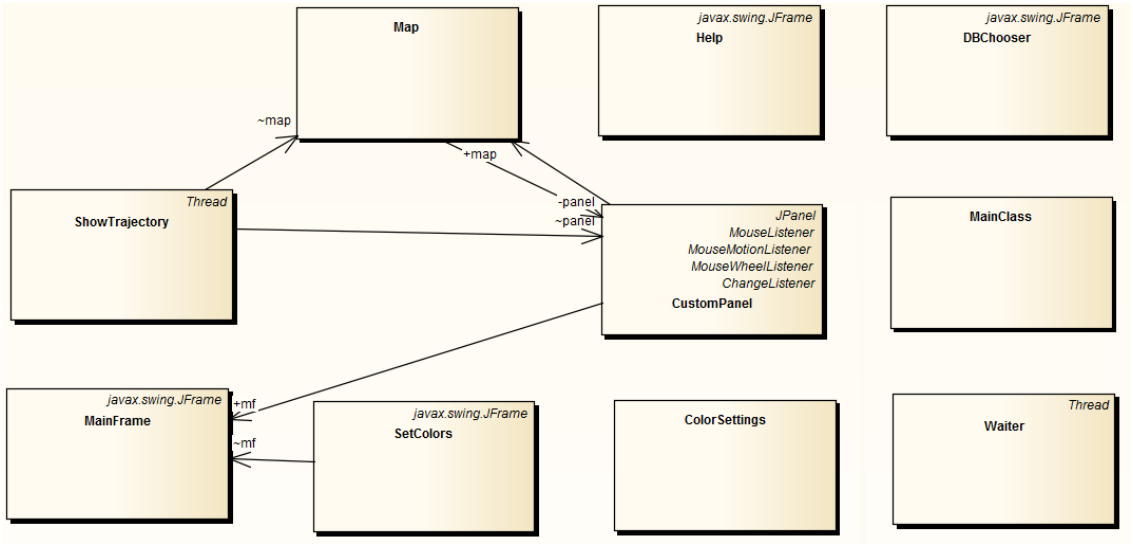
Obrázok 6.2: Tabuľky použitej databázy a ich atribúty.

Na obrázku 6.2 vidíme schému tabuliek v použitej databázy spolu s ich atribútmi. Tabuľky majú nasledovné využitie:

- *frequentTrajectories_temporaly* – dočasné uloženie informácií o tom, ktoré objekty budú ponúkané na predikciu ich trajektórie. Dôvodom iba dočasného uloženia je zachovanie konzistencie dát.
- *settings* – uloženie parametrov, pri ktorých boli získané uložené frekventovane lokality a T-Patterny.
- *frequentTrajectories* – uloženie údajov o tom, ktoré objekty budú ponúkané na predikciu ich trajektórie.
- *moveDataOldenburg* – uloženie všetkých bodov trajektórii objektov.
- *locations* – uloženie údajov o lokalitách.
- *tPatterns* – uloženie vzorov pohybu.

6.2 Balík GUI

Tento balík obsahuje štartovací bod aplikácie – metódu *main()*. Jeho triedy zabezpečujú chod celého grafického užívateľského prostredia (GUI z Grafic User Interface) aplikácie. Obsahuje implementáciu hlavného okna, vedľajších okien a ostatných grafických prvkov. Diagram tried tohto balíka zobrazuje obrázok 6.3.



Obrázok 6.3: Diagram tried balíka GUI.

Trieda *Map* sa stará o prácu s mapou. Umožňuje prepočet stupňov, pohyb súradnicovej siete, priblíženie, oddialenie a ťahanie zobrazovanej mapy.

Najdôležitejšou triedou tohto balíka je trieda *ShowTrajectory*. Uskutočňuje vykresľovanie trajektórií a samotnej predikcie. Táto trieda dedí z triedy *Thread*, čo znamená že vykresľovanie trajektórií beží v samostatnom vlákne. Toto vlákno, ktoré je spustené z panelu obsahujúceho mapu s trajektóriami, je s vláknom panelu synchronizované. Použitie viacerých vláken zabezpečuje svižnejší chod aplikácie. Najvýraznejšie sa to prejavuje pri prekresľovaní trajektórií a súradnicovej siete, ktoré sa deje pri manipulácií s mapou.

Kritickou operáciou pri práci s mapou sa stalo načítavanie trajektórií z databázy. Preto sa v aplikácii po načítaní vzorky trajektórií tieto dáta uložia. Tento krok výrazne urýchlil celkovú prácu s aplikáciou. Spomínanú vzorku tvorí prvých 50 000 bodov trajektórií z databázy (celkový počet je takmer štyri milióny). Pri zobrazení výsledkov predikcie sa uložia do pamäte aj trajektórie, ktoré ju tvoria.

Hľadanie frekventovaných lokalít je taktiež výpočtovo veľmi náročná operácia, preto sú jej výsledky nad danými dátami a parametrami uložené v databázy. Po načítaní z nej ostávajú rovnako uložené v pamäti, čo ale urýchli nie len ich vykresľovanie, ale aj predikciu. Je to vďaka tomu, že sa nemusia vždy nanovo hľadať za predpokladu, že užívateľ nezmení nadstavenia parametrov frekventovaných lokalít pri viacerých predikciách za sebou.

6.3 Balík WhereNext

Balík WhereNext je kľúčovou časťou celej aplikácie. Obsahuje kompletnú implementáciu algoritmu popísaného v kapitole 5. Diagram tried balíka WhereNext môžeme vidieť na obrázku 6.4.

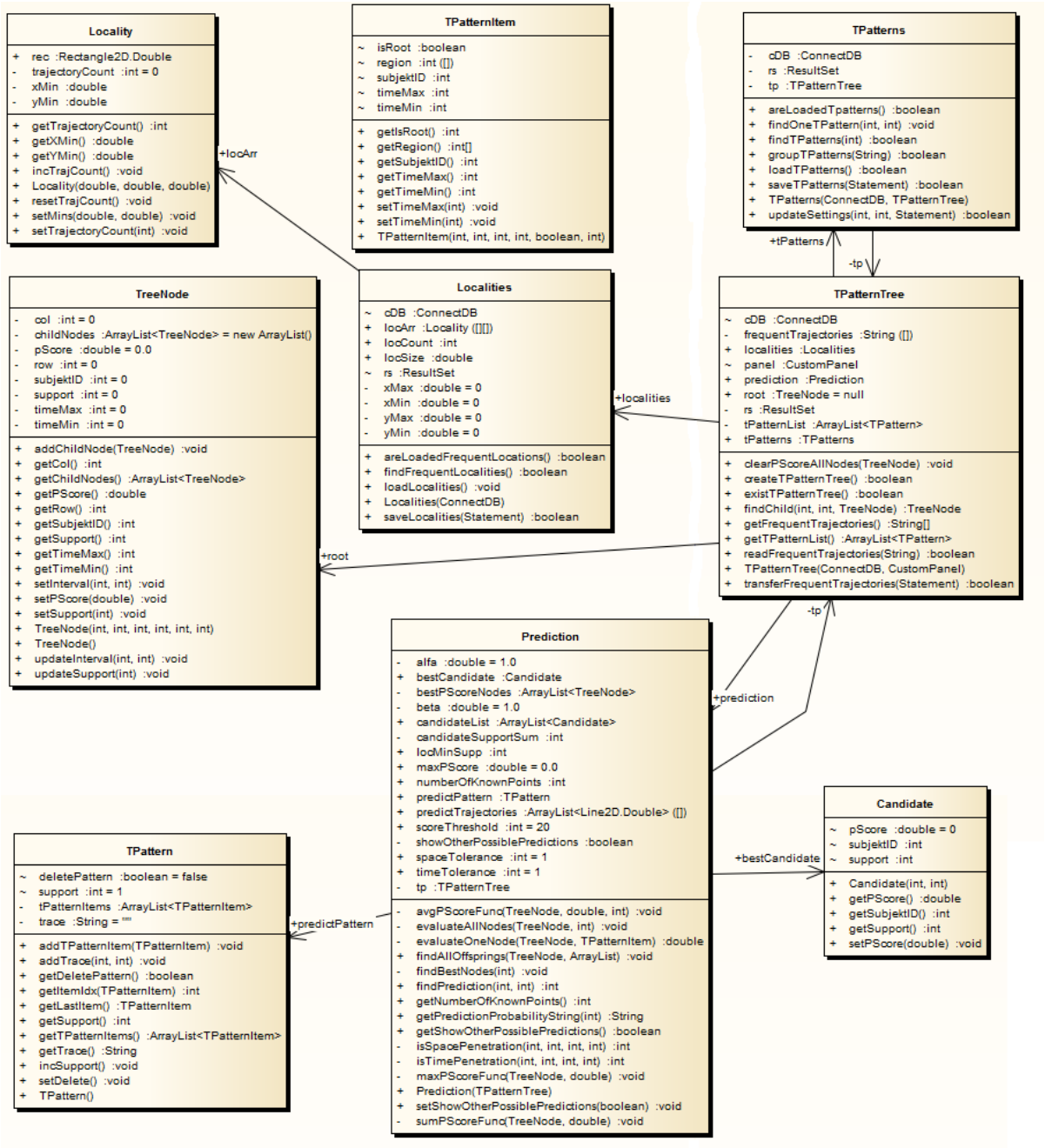
6.3.1 Frekventované lokality

Na objavovanie frekventovaných lokalít sa využijú ponúkané grafické komponenty triedy *java.awt.geom*. Ich metódy poskytujú funkcionalitu, ktorá sa na tento problém presne hodila a nebolo treba zbytočne implementovať niečo, čo nám ponúka sama Java. Ide hlavne o metódy, ktoré zisťujú či došlo k prieniku grafických komponentov, konkrétne prienik úsečky a štvorca.

Vlastné vyhľadávanie frekventovaných lokalít začína rozdelením priestoru, v ktorom sa vyskytujú trajektórie, na štvorce. Tieto štvorce budú reprezentované skutočnými grafickými komponentmi, avšak ich súradnice a veľkosti v pixeloch budú v skutočnosti zastupovať GPS súradnice. Počet štvorcov závisí na nadstavenom parametri *Number of localities*, ktorý udá na koľko častí sa rozdelí horizontálna šírka priestoru. Veľkosť jednej časti určí dĺžku strany štvorcov. Hranice priestoru sa na začiatku určia z extrémov súradníc bodov všetkých trajektórií. Po týchto krokoch prejdeme v cykle všetkými trajektóriami. Pri každej trajektórii prejdeme všetkými lokalitami a popri tom kontrolujeme či trajektória túto lokalitu pretína. Ak áno, inkrementujeme podporu lokality. Po ukončení týchto dvoch vnorených cyklov, dostávame lokality ohodnotené ich podporou. Tie z lokalít, ktorých podpora sa rovná alebo je väčšia ako minimálna podpora, prehlásime za frekventované lokality.

Frekventované lokality sa v aplikácii ukladajú do dátovej štruktúry typu dvojrozmerné pole. Jednotlivé prvky poľa tvoria inštancie triedy *Locality*. Na ich vyhľadávanie a spracovanie je určená trieda *Localities*.

Dáta o frekventovaných lokalitách sú uložené v pripojenej databázy. Pri spustení aplikácie sú automaticky načítané pre rýchlejšou prácu s aplikáciou. V databázy sú spolu s týmito dátami zaznamenané aj informácie, pri akých parametroch boli získané. Aplikácia užívateľovi umožňuje pozmeniť parametre frekventovaných lokalít a spustiť vyhľadávanie podľa týchto parametrov. Nakoľko sa do databázy ukladajú aj T-Patterny, ktoré majú závislosť na frekventovaných lokalitách, spustí sa popritom aj hľadanie frekventovaných vzorov. Tieto nové údaje je možné následne uložiť do databázy.



Obrázok 6.4: Diagram tried balíka WhereNext.

6.3.2 T-Patterny

Operácie, ktoré sú potrebné na vyhľadanie a prácu s T-patternami, zabezpečuje trieda *TPatterns*. Všetky vzory, získané pri načítaní z databázy alebo po vyhľadaní, sa ukladajú do zoznamu s položkami triedy *TPattern*. Podobne ako frekventované lokality aj T-Patterny sa pri spúšťaní aplikácie načítavajú z databázy.

Objavovanie frekventovaných vzorov prebieha v cykle, v ktorom sa prechádza cez všetky body trajektórií uložené v databázy. Body sú usporiadane najskôr podľa identifikátoru objektu a následne podľa časového atribútu. Ak aktuálne spracovávaný bod patrí trajektórii iného objektu ako predchádzajúci bod, vieme, že aktuálny bod je počiatok trajektórie. V tomto prípade vytvoríme nový vzor, ktorý reprezentuje inštancia triedy *TPattern*. Tá obsahuje atribút *tPatternItems*, ktorý je tvorený zoznamom prvkov triedy *TPatternItem*. Tento prvok predstavuje jeden uzol T-Patternu popísaný v definícii 2.

Každá iterácia hlavného cyklu obsahuje vnorený cyklus, v ktorom sa prechádzajú všetky lokality. Ak je lokalita frekventovaná (jej podpora je väčšia alebo rovná minimálne podpore) a bod do tejto lokality patrí, vytvoríme nový uzol vzoru a následne ho vložíme do zoznamu *tPatternItems*. Pri vytváraní uzla počítame čas, aký bol potrebný na prechod z predchádzajúcej frekventovanej lokality do aktuálnej, ako rozdiel aktuálnej časovej známky a predchádzajúcej. Ak je však zoznam ešte prázdny čas prechodu sa automaticky nadstaví na nulovú hodnotu.

Pre rýchle porovnávanie T-patternov potrebnom pri ich grupovaní, trieda *TPattern* obsahuje navyše atribút *trace*, ktorý je typu reťazec. Do neho sa pri vkladaní novej položky pridávajú indexy frekventovanej lokality uloženej v dvojrozmernom poli. Tento atribút umožní porovnaním dvoch reťazcov porovnať celú postupnosť frekventovaných lokalít, ktoré tvoria T-Pattern.

Grupovanie prebieha pomocou dvoch hlavných vnorených cyklov. Vo vonkajšom sa prechádza cez všetky objavené T-Patterny. Ak aktuálny vzor nie je označený ako vymazaný (atribút *deletePattern*), tak sa porovnáva vo vnorenom cykle so všetkými ďalšími vzormi, ktoré nie sú označené za zmazané. Ak platí podmienka, že parametre *trace* sa rovnajú, T-Pattern z vnútorného cyklu je označený ako zmazaný a pristúpime k úprave intervalu položiek vzoru z hlavného cyklu. Tieto upravujeme tým spôsobom, že roztiahneme časový interval, tak aby do neho patril po úprave aj interval súhlasnej položky vzoru z vnútorného cyklu. Nakoniec sa všetky T-Patterny, označené ako zmazané, odstránia zo zoznamu. Pred vymazaním sa uložia do databázy identifikátory objektov, ktorých trajektórie zmazávané T-Patterny reprezentujú. Tieto objekty sú neskôr zobrazené v ponuke objektov, ktorých trajektórie sa budú predikovať.

T-Patterny obsahujú taktiež informácie o tom, ktorého objektu trajektóriu zastupujú. Táto informácia sa neskôr využije pri zobrazení predikcie.

6.3.3 T-Pattern strom

Dáta T-Pattern stromu sa neukladajú do databázy. Je to z toho dôvodu, že parametre predikcie sa stále menia a jeho zostrojenie nie je až tak časovo náročná operácia v porovnaní s hľadaním frekventovaných lokalít a T-Patternov. O vytváranie a prácu s ním sa stará trieda *TPatternTree*. Celý strom sa skladá z jednotlivých uzlov, ktoré zastupuje trieda *TreeNode*. Uzol stromu má podobné atribúty ako uzol T-Patternu. Obsahuje však navyše informáciu o pScore uzla (atribút *pScore*) a o tom, ktoré uzly sú jeho potomkami (atribút *childNodes*). Potomkovia sú v uzle uložený vo forme zoznamu s prvkami triedy *TreeNode*.

T-Pattern strom sa konštruuje zo zgrupovaných vzorov podľa algoritmu 2. Deje sa to v cykle, v ktorom tieto vzory prechádzame. Na začiatku každej iterácie nadstavíme aktuálny uzol na koreň stromu. Následne zahájime vnútorný cyklus, v ktorom budeme iterovať cez jednotlivé položky vzoru. Pre každú položku sa snažíme nájsť ekvivalentného potomka aktuálneho uzla.

Ak takýto potomok existuje, upravíme jeho podporu, interval a tento potomok sa stane aktuálnym uzlom. Podpora sa upraví tak, aby v uzle ostala vždy väčšia hodnota z pôvodnej hodnoty v uzle a z podpory T-Patternu. Interval sa upravuje podobne ako pri zoskupovaní T-patternov. Rozťahuje sa interval v uzle stromu tak, aby do neho spadal aj interval v položke vzoru.

Ak však nastane prípad, že tento potomok neexistuje, musíme vytvárať nový uzol. Jeho atribúty nadstavíme podľa hodnôt atribútov aktuálnej položky T-Patternu a vložíme ho do zoznamu potomkov aktuálneho uzla stromu. Aktuálnym uzlom sa stane práve vytvorený uzol.

6.3.4 Predikcia

Na objavenie predikcie využíva aplikácia triedu *Prediction*. Celá predikcia začína vytvorením T-Patternu, ktorý bude reprezentovať známu časť predikovanej trajektórie pohybujúceho sa objektu.

V poradí druhým krokom je ohodnotenie uzlov stromu za pomoci pScore. Ak už daný strom bol predtým využitý na predikciu, je najskôr vynulované pScore všetkých jeho uzlov. Ohodnotenie zabezpečuje metóda *evaluateAllNodes()*, ktorá pracuje rekurzívne. Metóda zakaždým volá sama seba nad všetkými potomkami aktuálneho uzla. Pri každom prechode z vyššej úrovne stromu do nižšej sa zväčšuje index položky T-Patternu, s ktorou sa bude uzol stromu porovnávať. Na začiatku je metóda volaná nad koreňom stromu a nulovým indexom položky T-Patternu. Každému uzla za pomoci

metódy *evaluateOneNode()* priradí vypočítané pScore. Táto metóda na ohodnotenie jedného konkrétneho uzla porovnáva aktuálny uzol stromu s položkou T-Patternu, ku ktorej by mal byť uzol ekvivalentný s prihliadaním na parametre predikcie. Podrobnejší princíp výpočtu pScore je uvedený v kapitole 5.4.1.

V T-Pattern strome s ohodnotenými uzlami môžeme začať hľadať cesty s najlepším celkovým skóre. O túto činnosť sa stará metóda *findBestNodes()*, ktorá naplní zoznam (atribút *bestPScoreNodes* triedy *Prediction*) posledným uzlom s nenulovým pScore najlepšie ohodnotených ciest. Metóda *findBestNodes()* podľa nadstavenej funkcie na nájdenie najlepšej cesty zavolá jednu z troch metód – *avgPScoreFunc(), sumPScoreFunc()* alebo *maxPScoreFunc()*. Všetky rekurzívne prechádzajú stromom a pracujú na princípe zobrazenom na obrázku 5.5.

Následne je potrebné v cykle iterujúcom cez zoznam najlepších kandidátov (uzlov) zavolať nad každým kandidátom metódu *findAllOffsprings()*. Tá od kandidátneho uzla rekurzívne, smerom ďalej od koreňa stromu, hľadá všetky koncové uzly a vkladá ich do zoznamu (atribút *candidateList* triedy *Prediction*). Každý uzol v sebe nesie informáciu o trajektórii – identifikátore objektu, z ktorého bol uzol cez T-Pattern vytvorený. To znamená, že **zoznam koncových potomkov najlepších kandidátov určuje všetky možné predikcie trajektórie, a teda lokácie pohybu**.

Podľa podpory jednotlivých T-Patternov, z ktorých boli koncové uzly vytvorené, je priradená jednotlivým predikciám pravdepodobnosť správnosti. Predikcia, ktorá má najväčšiu pravdepodobnosť je určená za hlavnú predikciu a je zobrazená na mape hrubou čiarou. Pri predikcii je táto pravdepodobnosť zobrazená v šedej elipse na konci predikovanej časti trajektórie, viď obrázok 6.3.

6.4 Užívateľské rozhranie a ovládanie aplikácie

Jedným z hlavných cieľov už pri návrhu a vytváraní aplikácie bola jednoduchosť a intuitívnosť jej ovládania. Pri vytváraní rozloženia prvkov a výbere farieb sa takisto bral zreteľ na estetickú stránku jej užívateľského prostredia.

Okamžite po štarte sa na tri sekundy otvorí okno s logom aplikácie. Následne je užívateľovi zobrazené okno, v ktorom je treba vybrať v súborovom systéme počítača databázu, nad ktorou bude aplikácia pracovať. Keďže bez nej nie je možné v práci pokračovať, je výber správneho súboru s databázou nevyhnutnou podmienkou. Pri zobrazovaní súborov je použitý filter, ktorý zobrazuje len súbory s príponou „.fdb“, teda Firebird Database File.

Po úspešnom pripojení sa k databázy, sa otvorí hlavné okno aplikácie zložené z ôsmych hlavných častí označených na obrázku 6.5:

1. *Ponuka aplikácie* – užívateľovi umožňuje importovať dáta, ukladať do databázy aktuálne dáta o T-Patternoch a lokalitách, ukončenie aplikácie, meniť farebné nadstavenia a zobrazenie pomocníka.

2. *Spustenie predikcie* – tlačidlo, ktoré spustí samotnú predikciu so zadanými parametrami a pod ním nadstavenie zobrazovaných informácií na mape.

3. *Parametre predikcie* – prvky, vďaka ktorým je možne nadstaviť parametre predikcie.

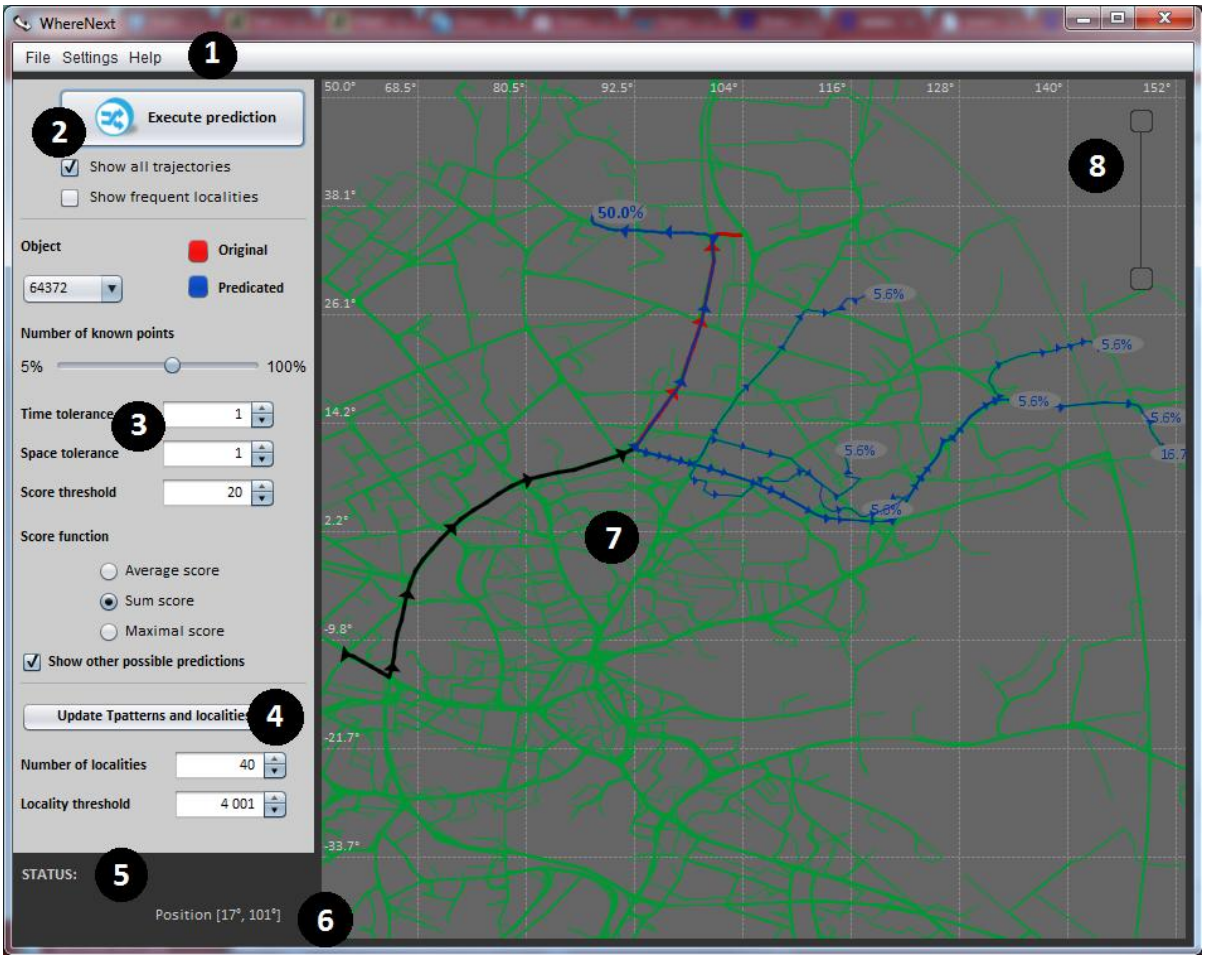
4. *Vyhľadanie frekventovaných lokalít a T-Patternov* – nadstavenie parametrov a spustenie nového vyhľadania frekventovaných lokalít a T-Patternov.

5. *Stav operácie* – užívateľ tu je oboznámený s ukončením časovo náročných operácií.

6. *Ukazovateľ pozície* – zobrazenie aktuálnej polohy, na ktorej sa práve nachádza kurzor.

7. *Mapa* – panel, v ktorom je zobrazená mapa s trajektóriami a po spustení predikcie sa tu zobrazí aj tá. Zelené čiary sú trajektórie ostatných objektov, čierna je známa časť trajektórie, červená je časť, ktorá sa predikuje a modrá čiara zobrazuje predikciu.

8. *Priblíženie a oddialenie mapy* – tlačidlá a posuvník na nadstavenie miery priblíženia mapy, ktoré sa zobrazia, ak sa kurzor dostane do ich blízkosti.



Obrázok 6.5: Celkový vzhľad aplikácie.

Manipulácia s mapou je vyriešená rovnako ako v známej aplikácii Google Earth a jemu podobných aplikáciách. Vďaka tomu si užívateľ na prácu s ňou nemusí takmer zvykať. Aplikácia reaguje aj na dve klávesové skratky – *Ctrl+I* spúšťa importovanie dát a *Ctrl+C* otvára okno s farebnými nadstaveniami.

7 Experimentálne overenie

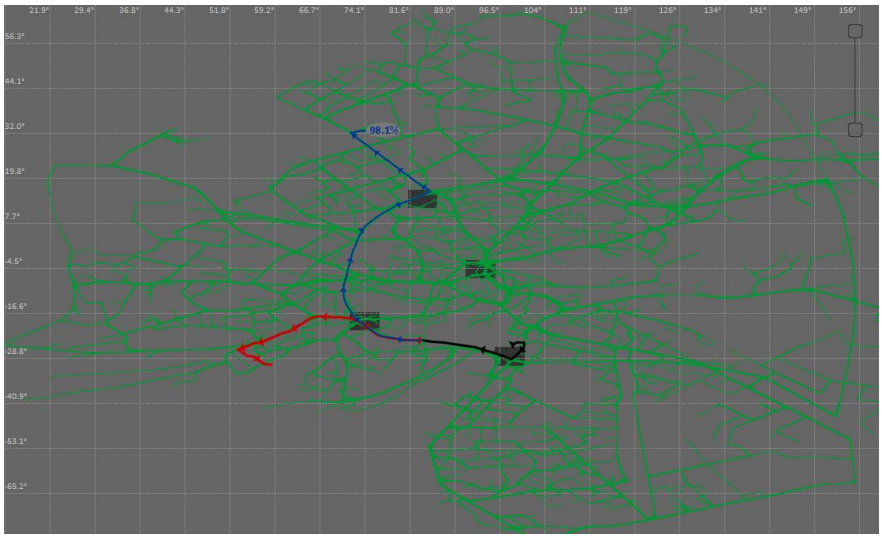
Na overenie správnosti implementácie algoritmu WhereNext sú použité dáta z [15].

7.1 Analýza možných problémov

Algoritmus WhereNext v sebe skrýva niekoľko slabín [12]. Tie môžu spôsobiť situáciu, pri ktorej nie sme schopný nájsť žiadnu možnú predikciu. Prvým takýmto prípadom je, keď známa časť trajektórie, podľa ktorej sa snažíme nájsť najvhodnejší vzor v T-Pattern strome, je dlhšia ako všetky vzory. Keďže predikciu predstavuje pokračovanie najvhodnejšej cesty stromu a v tomto prípade pokračovanie neexistuje, predikcie nie sme schopní. Rovnaká situácia nastane, ak známa časť trajektórie je priestorovo alebo časovo príliš ďaleko od všetkých vzorov stromu. Vtedy nie sme v T-Pattern strome dokonca schopní nájsť žiadnu vhodnú cestu. Ak však takýchto situácií nastáva minimum, predikciu môžeme pri správne zvolených parametroch pokladať za spoľahlivú.

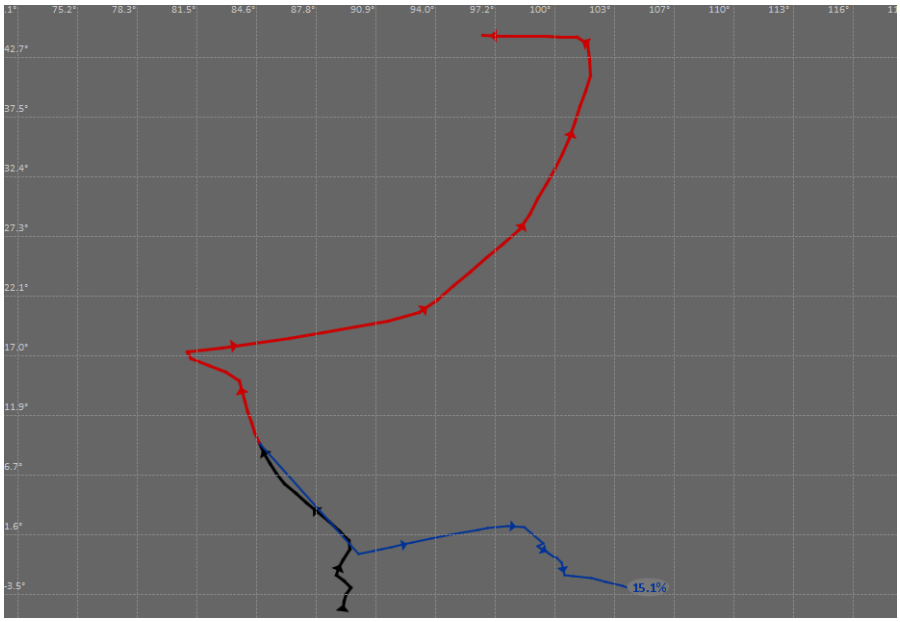
7.2 Parametre lokalít

Na kvalitu výslednej predikcie má veľký význam vhodne zvolená kombinácia parametrov. Zmena parametrov lokalít znamená zmenu veľkosti jednotlivých lokalít, ich počtu a taktiež počtu objavených frekventovaných lokalít. Zmena frekventovaných lokalít implikuje zmenu v počte a tvare T-Patternov, z čoho pramení odlišná stavba T-Pattern stromu, a teda odlišná predikcia. Veľkosť regiónov a ich výskyt v uvažovanom priestore sú jedným z kľúčových krokov kvalitnej predikcie.



Obrázok 7.1: Nepresnosť predikcie spôsobená malým počtom lokalít.

Ak parameter *Number of localities* (určuje na koľko lokalít sa rozdelí priestor vo vodorovnom smere) nadstavíme na príliš veľkú hodnotu alebo minimálnu podporu lokalít nadstavíme vzhľadom k veľkosti lokalít na priveľkú hodnotu, frekventované lokality budú pokrývať len malé percento priestoru. Z toho vyplýva, že T-Patterny budú taktiež pokrývať len malú časť z priestoru, a preto pre mnoho pohybujúcich sa objektov nebude možné nájsť predikciu. Na obrázku 7.1 vidíme tento prípad, keď *Number of localities* ma hodnotu 40 a minimálna podpora je nadstavená na 15 000 pri použití funkcie *sumScore*. Rovnako nežiaduca je situácia, keď tento parameter bude mať primalú hodnotu. Frekventované lokality nadobudnú veľké rozmery a stanú sa nepoužiteľnými pri predikcii, nakoľko by bola veľmi nepresná, viď obrázok 7.2 s parametrami *Number of localities* rovným 10 a minimálnou podporou 5 000 za použitia funkcie *sumScore*. Štatistiky lokalít pri rôznych parametroch zobrazuje tabuľka 7.1 a 7.2.



Obrázok 7.2: Chyba predikcie spôsobená príliš veľkými lokalitami.

Parameter *Number of localities*	**Parameter *Locality threshold***	**Počet frekventovaných lokalít**	**Počet T-Patternov**	**Priemerná podpora T-Patternu**	**Priestor zabraný f. lokalitami [%]**
10	5 000	81	267 124	1,69	90,0
20	5 000	60	123 442	1,41	17,64
30	5 000	81	172 850	2,05	13,38
40	5 000	163	375 827	2,08	11,98
50	5 000	98	214 764	1,90	4,56
60	5 000	109	232 898	1,83	3,56

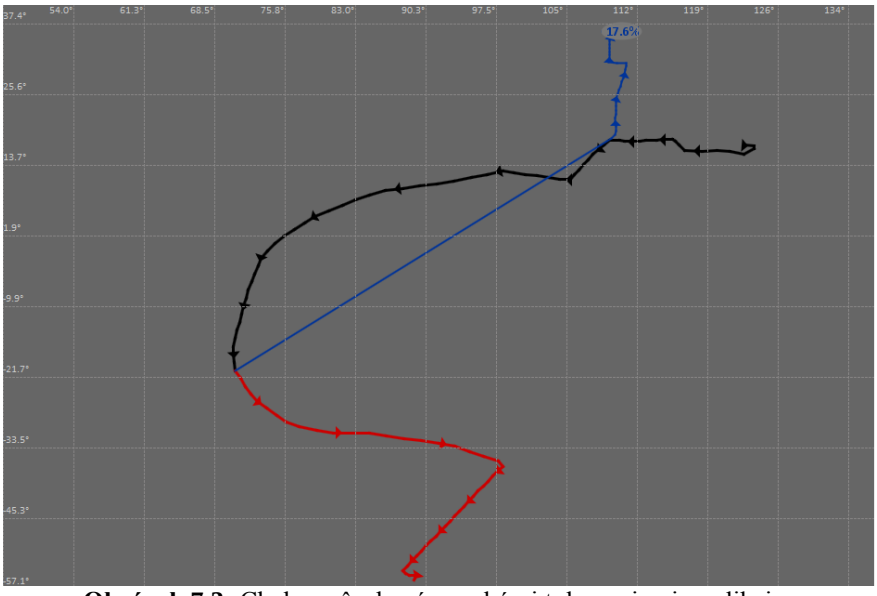
Tabuľka 7.1: Štatistiky lokalít pri rovnakej minimálnej podpore a rôznych hodnotách parametra *Number of localities.*

Parameter *Number of localities*	**Parameter *Locality threshold***	**Počet frekventovaných lokalít**	**Počet T-Patternov**	**Priestor zabraný f. lokalitami [%]**
20	10 000	24	21 183	7,05
30	4 000	101	228 146	15,95
40	100	646	790 926	47,50
40	500	517	764 060	38,01
40	2 000	258	534 493	18,97
40	4 000	120	267 124	8,82
40	6 000	66	116 828	4,85
40	10 000	30	19 796	2,21
40	15 000	4	42	0,29
50	1 000	709	848 845	32,98
50	2 000	310	624 707	14,42
70	500	1134	1 259 032	27,46

Tabuľka 7.2: Štatistiky lokalít pri rôznych parametroch.

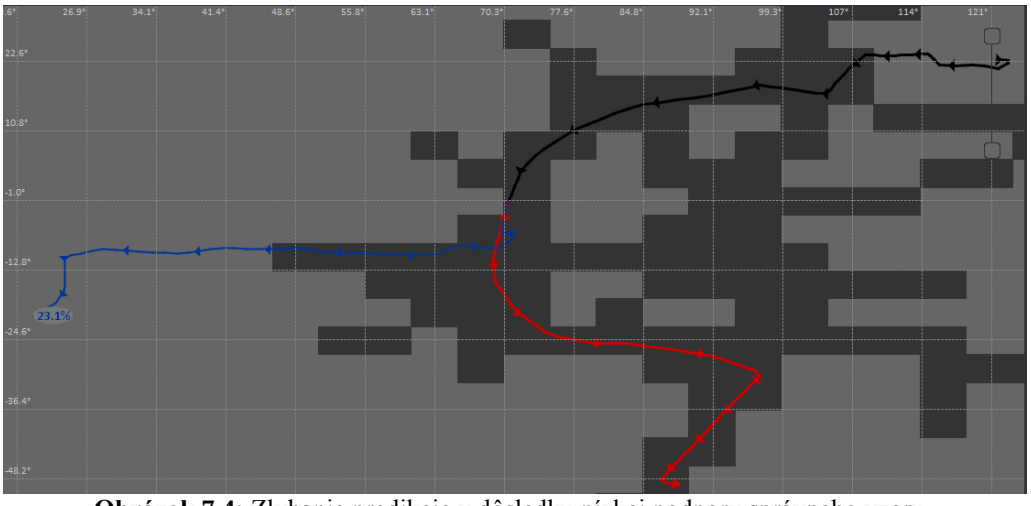
7.3 Parametre predikcie

Predikcia obsahuje tri parametre – minimálne skóre, časovú a priestorovú toleranciu. Nevhodné nadstavenie ich hodnôt môže taktiež viesť k buď veľmi nepresnej alebo dokonca žiadnej predikcii. Nevhodne zvolenými hodnotami sa v tomto prípade myslí primalé minimálne skóre alebo príliš vysoké tolerancie. Na obrázku 7.3 vidíme predikciu s priveľkými hodnotami tolerancií (časová = 30, priestorová = 10) za použitia funkcie *maxScore*.



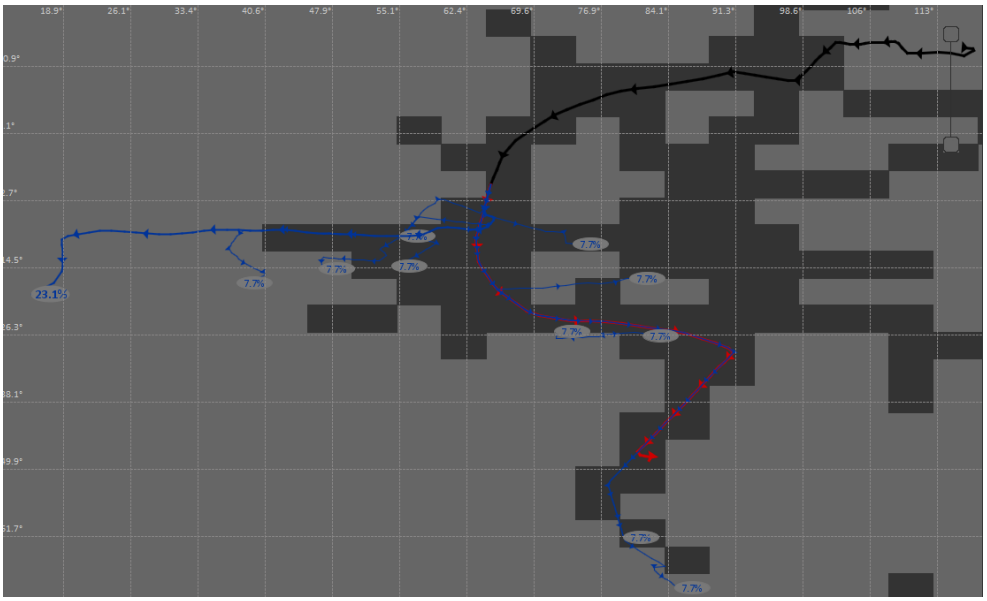
Obrázok 7.3: Chyba spôsobená vysokými toleranciami predikcie.

Bohužiaľ ani správne zvolené hodnoty parametrov nám nezaručujú úplnú spoľahlivosť predikcie, keďže pracujeme s pravdepodobnosťami. Priblížme si túto situáciu na obrázku 7.4, kde vidíme zlyhanie predikcie pri použití funkcie *sumScore*, oboch toleranciách s hodnotou 1 a prahom skóre rovným 20. V strome sa síce našiel správny najvhodnejší uzol, ale vzor, ktorý pokračuje z tohto uzla a má najväčšiu podporu (a teda aj pravdepodobnosť), nepredstavuje správnu predikciu.



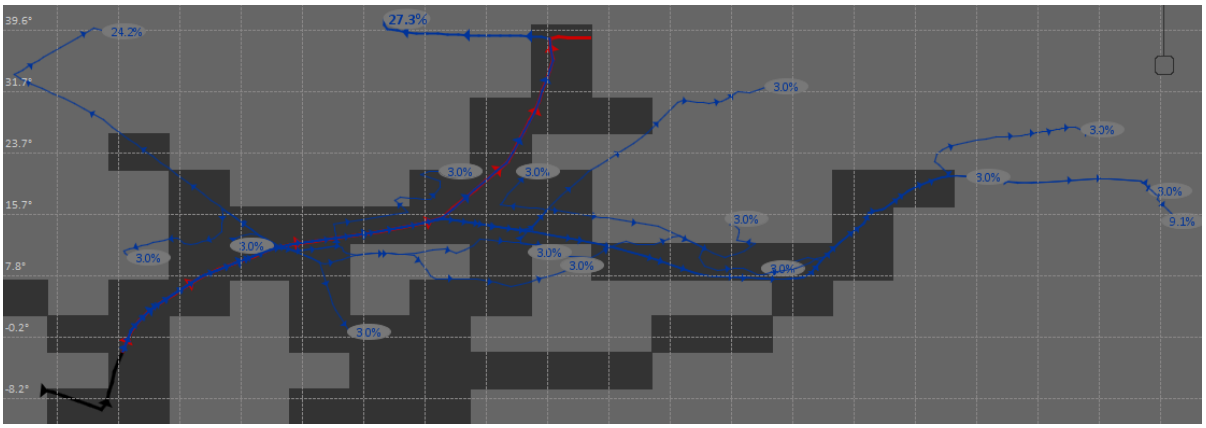
Obrázok 7.4: Zlyhanie predikcie v dôsledku nízkej podpory správneho vzoru.

Túto situáciu sme však schopný vyriešiť tým, že si necháme vyhľadať všetky dostupné predikcie. Obrázok 7.5 ukazuje toto riešenie.



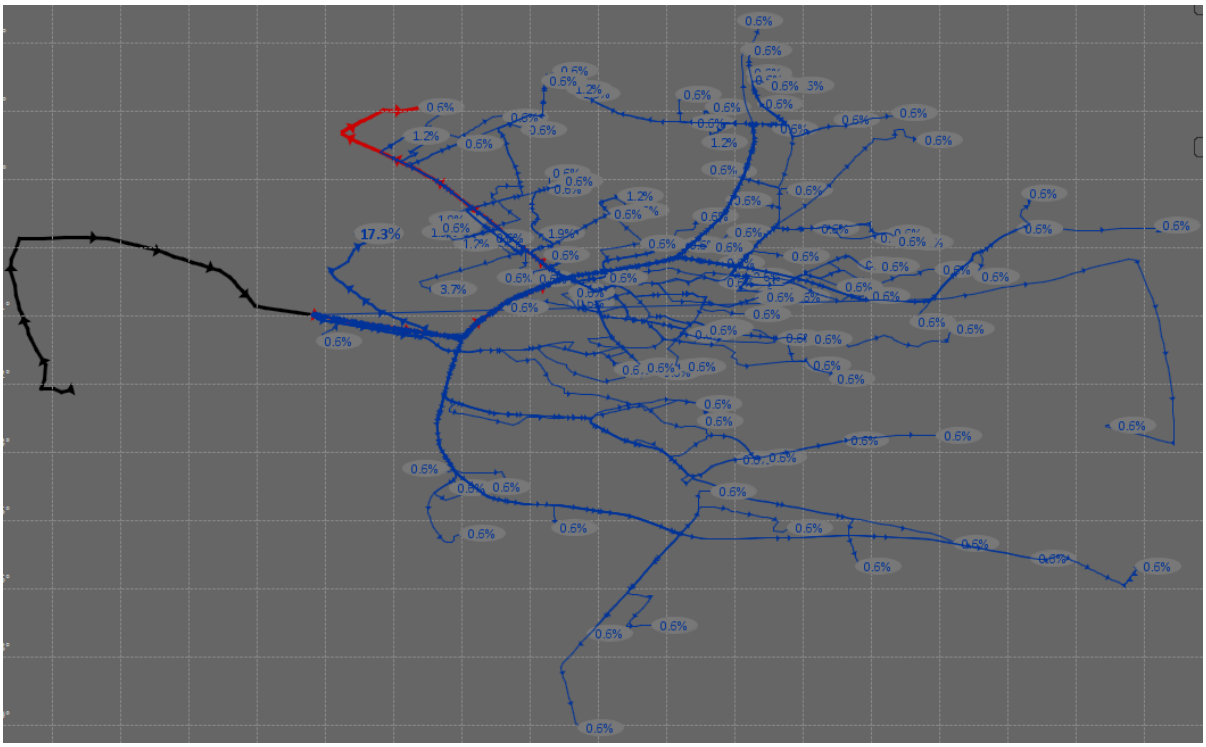
Obrázok 7.5: Riešenie nízkej podpory správneho vzoru.

V prípadoch kedy správny vzor, ktorý by mal byť predikciou má v porovnaní s ostatnými vzormi vysokú podporu, nám stačí veľmi malá známa časť trajektórie na vynikajúcu predikciu, viď obrázok 7.6.



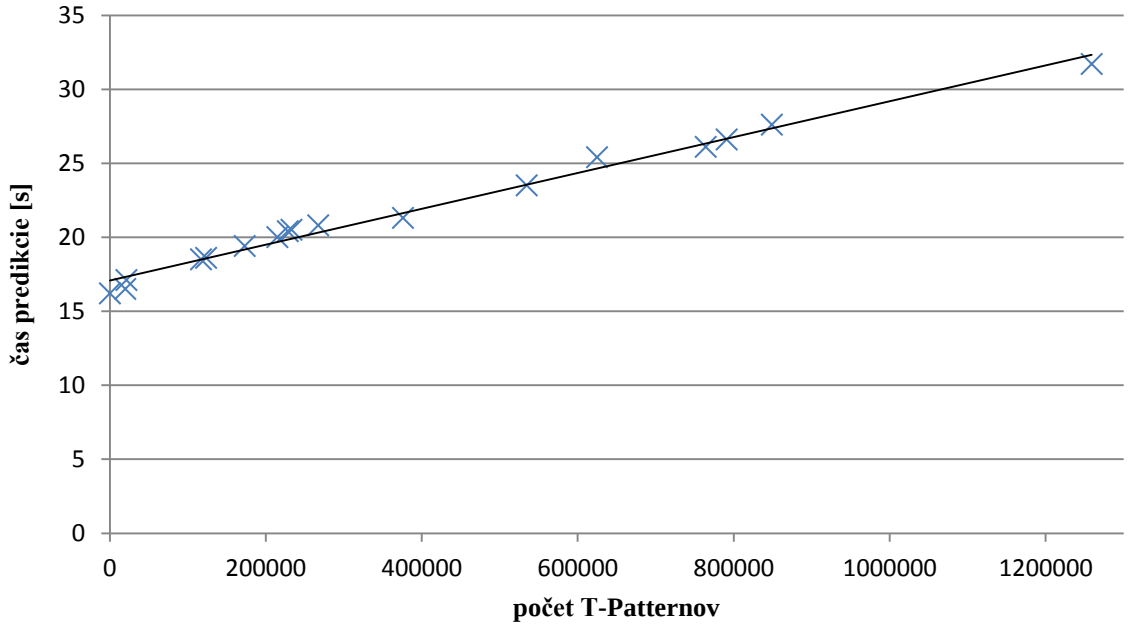
Obrázok 7.6: Správna predikcia pri extrémne krátkej známej časti trajektórie.

Hodnoty parametrov predikcie majú okrem presnosti predikcie vplyv aj na čas, potrebný na jej získanie. Pri krátkej známej časti trajektórie sa najvhodnejší uzol bude nachádzať bližšie pri koreni, a teda bude mať viacej potomkov, ktoré je treba spracovať. Ako príklad uveďme obrázok 7.7, kde známa časť trajektórie tvorí necelých 50 percent a vyžadujeme objavenie všetkých dostupných predikcií. V tomto prípade mál najvhodnejší uzol nezvyčajne veľký počet potomkov. Tento výpočet trval približne 12 minút na stroji so 4 GB operačnej pamäte a dvojjadrovom procesorom Intel Core i3 s taktom 2,26 GHz.



Obrázok 7.7: Veľmi zložitý výpočet predikcie.

Pozrime sa teraz na vývoj dĺžky času potrebného na výpočet predikcie vzhľadom k počtu T-Patternov. V grafe 7.1 môžeme vidieť, že so zvyšujúcim sa počtom T-Patternov, čas výpočtu *lineárne narastá*. Graf bol získaný za pomoci experimentov s použitím funkcie *sumScore* a známa časť trajektórií tvorila 55 percent. Výpočet prebiehal na rovnakom počítači ako predchádzajúci experiment.



Graf 7.1: Zobrazenie lineárneho nárastu času potrebného na výpočet predikcie pri zvyšujúcim sa počtom T-Patternov.

8 Záver

Predmetom tejto práce bolo vytvorenie aplikácie na predikciu lokácie pohybujúceho sa objektu a jej experimentálne overenie na vhodnej dátovej sade.

V prvých fázach bolo najdôležitejšie naštudovanie problematiky dolovania v dátach pohybujúcich sa objektov. Následne bolo nevyhnutné sa zoznámiť s existujúcimi prístupmi a algoritmami predikcie ich budúcej lokácie. Keďže prístupov umožňujúcich predikciu existuje viacero, museli sme s vedúcim bakalárske práce vybrať jeden z nich a potom rozhodnúť, na ktorý algoritmus, do neho patriaci, sa zameriame.

Pre implementáciu som vybral algoritmus WhereNext, ktorý som čiastočne upravil pre dlhodobejšiu predikciu a opravil v ňom objavené nepresnosti. Na overene korektnej činnosti aplikácie som použil vygenerované dáta pohybu áut, ktoré som si vhodne modifikoval.

Pri vývoji aplikácie sme sa stretli s rôznymi problémami. Prvým bol výber vhodných dát nutných na overovanie funkčnosti pri vývoji aplikácie. Pôvodne boli na tento účel využívané dáta o migračnom pohybe bocianov bielych medzi Európou a Afrikou. Tie však neposkytli možnosť objavenia vhodných frekventovaných vzorov pohybu, a preto som ich nahradil už spomínanou dátovou sadou. Najvýraznejším problémom však bola vysoká časová náročnosť výpočtov nad použitým veľkým objemom dát o pohybe objektov a nedostačujúci výkon počítača, na ktorom bola aplikácia vyvíjaná. Vysporiadať sa s touto situáciou som sa snažil čiastočným uložením dát, ktoré boli získané pri výpočte, do databázy. Pri opätovných spúšťaniach predikcie s rovnakými parametrami lokalít, sa tieto predpripravené údaje iba načítavajú z databázy.

Uskutočnené experimenty preukázali, že aplikácia pri vhodne zvolených parametroch pracuje správne. Bolo pri nich taktiež odtestované správanie aplikácie pri nevhodne zvolených hodnotách parametrov. Testy odhalili viacero nedokonalostí algoritmu, z ktorých niektoré boli odstránené a s ostatnými som sa snažil čo najlepšie vysporiadať.

Vhodný priestor na pokračovanie tejto práce vidím vo vytvorení programovej vrstvy, ktorá by dokázala automaticky dosádzať rôzne kombinácie parametrov a vyhodnocovať kvalitu objavenej predikcie. To by umožnilo nájsť optimálne hodnoty parametrov pre danú dátovú sadu. Toto rozšírenie by bolo však nutné vytvárať a testovať na počítači s vysokým výkonom, nakoľko implementovaný algoritmus má vysokú časovú náročnosť. Aj preto je ďalším vhodným pokračovaním optimalizácia aplikácie a tým jej čiastočné urýchlenie.

Literatúra

[1] KUŽELA, Alois. *Získávání znalostí z databází*. [online]. [cit. 2013-04-03]. Dostupné z: http://www.common.cz/attachments/117_alois_kuzela_asociacni_pravidla.pdf

[2] ZENDULKA, Jaroslav, Vladimír BARTÍK, Roman LUKÁŠ a Ivana RUDOLFOVÁ. Získávání znalostí z databází ZZN. Brno, 2009. Studijní opora. VUT v Brně.

[3] POLITECNICO DI MILANO. *A Tutorial on Clustering Algorithms* [online]. [cit. 2013-04-30]. Dostupné z: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html

[4] PARALIČ, Ján. *Objavovanie znalostí* [online]. [cit. 2013-04-03]. Dostupné z: http://people.tuke.sk/jan.paralic/prezentacie/OZ/Proces_OZ.pdf

[5] KÚDELČÍK, Jozef a Peter HOCKICKO. *Základy fyziky*. Žilina: EDIS, 2011, s. 29-30 [cit. 2013-04-03]. ISBN 978-80-554-0341-0.

[6] HAN, Jiawei, LI, Zhenhui, TANG, Lu An. *Mining Moving Object and Traffic Data* [online]. Tsukuba, 2010 [cit. 2013-04-03]. Dostupné z: http://www.cs.uiuc.edu/homes/hanj/pdf/dasfaa10_han_tuto.pdf

[7] GIANNOTTI, Fosca, et al. Trajectory pattern mining. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007. p. 330-339.

[8] TAO, Yufei, et al. Prediction and indexing of moving objects with unknown motion patterns. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 2004. p. 611-622.

[9] KALNIS, Panos; MAMOULIS, Nikos; BAKIRAS, Spiridon. On discovering moving clusters in spatio-temporal data. In: *Advances in spatial and temporal databases*. Springer Berlin Heidelberg, 2005. p. 364-381.

[10] ROMERO, Andres Oswaldo Calderon. *Mining moving flock patterns in large spatio-temporal datasets using a frequent pattern mining approach*. 2011. PhD Thesis. Master Thesis, University of Twente (March 2011).

[11] LEE, Jae-Gil; HAN, Jiawei; LI, Xiaolei. Trajectory outlier detection: A partition-and-detect framework. In: *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008. p. 140-149.

[12] MONREALE, Anna, et al. WhereNext: a location predictor on trajectory pattern mining. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009. p. 637-646.

[13] JEUNG, Hoyoung, et al. A hybrid prediction model for moving objects. In: *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*. IEEE, 2008. p. 70-79.

[14] NIZETIC, I.; FERTALJ, Krešimir; KALPIC, D. A prototype for the short-term prediction of moving object's movement using Markov chains. In: *Information Technology Interfaces, 2009. ITI'09. Proceedings of the ITI 2009 31st International Conference on*. IEEE, 2009. p. 559-564.

[15] ABUL, Osman, Francesco BONCHI a Mirco NANNI. *Time-tolerant Anonymization of Moving Objects Databases* [online]. 2010 [cit. 2013-05-06]. Dostupné z: http://kdd.isti.cnr.it/W4M/

Zoznam príloh

Príloha A. Obsah CD
Príloha B. Návod na použitie

A. Obsah CD

- */src* – zdrojové kódy programu (projekt v NetBeans 7.2)
- */doc* – projektová dokumentácia
- */bin* – spustiteľná aplikácia vo formáte jar
- */data* – databáza k aplikácii
- */manual* – návod na použitie aplikácie
- */bpWord* – technická správa vo formáte docx
- */bpPDF* – technická správa vo formáte pdf

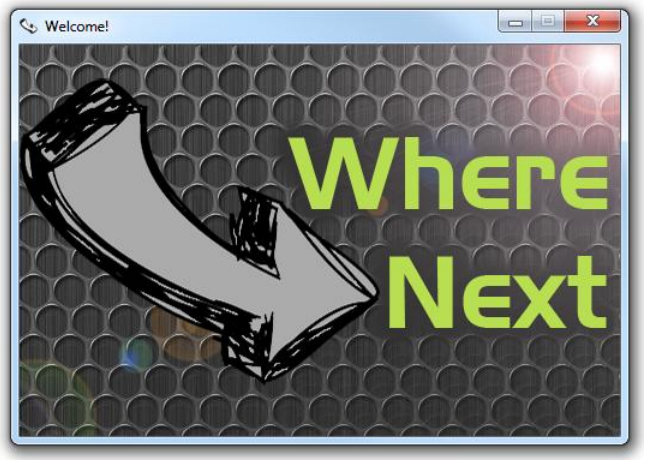
B. Návod na použitie

Spustenie aplikácie

1. Inštalácia servera Firebird 2.5 – aplikácia je voľne dostupná na webovej stránke http://www.firebirdsql.org/en/firebird-2-5-2-upd1/.
2. Spustenie Firebird servera.
3. Spustenie súboru WhereNext.jar alebo otvorenie projektu vo vývojovom prostredí Netbeans 7.2 a následné spustenie. Tento súbor a projekt sú uložené na priloženom CD.
4. Výber cesty k priloženej databázy v úvodnom okne. Databáza je taktiež uložená na priloženom CD.

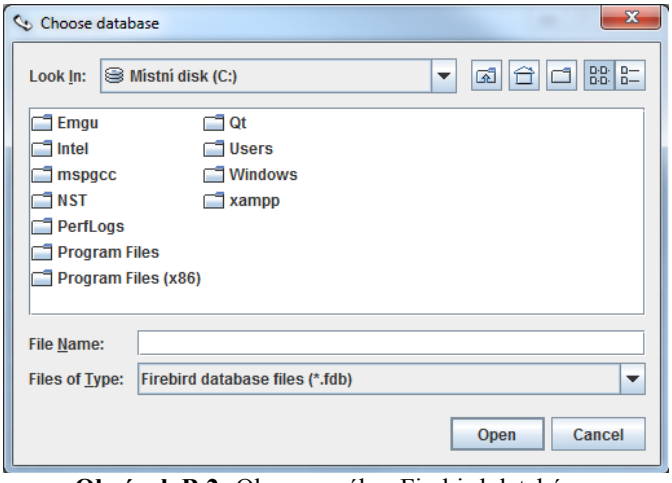
Ovládanie aplikácie

Prvým oknom, ktoré je zobrazené užívateľovi po spustení aplikácie je okno s logom aplikácie, viď obrázok B.1.



Obrázok B.1: Uvítacie okno s logom aplikácie.

Po troch sekundách ho prekryje okno, v ktorom užívateľ musí vybrať Firebird databázu, nad ktorou bude aplikácia pracovať. Toto okno zobrazuje obrázok B.2. Bez výberu správnej databázy nie je možné v práci s aplikáciou pokračovať.



Obrázok B.2: Okno na výber Firebird databázy.

Po úspešnom pripojení sa k databázy sa otvorí hlavné okno aplikácie zložené z ôsmych hlavných častí označených na obrázku B.3:

1. *Ponuka aplikácie* – užívateľovi umožňuje importovať dáta, ukladať do databázy aktuálne dáta o T-Patternoch a lokalitách, ukončiť aplikáciu, meniť farebné nadstavenia a **zobraziť pomocníka**.

2. *Spustenie predikcie* – tlačidlo, ktoré spustí samotnú predikciu so zadanými parametrami a pod ním nadstavenie zobrazovaných informácií na mape.

3. *Parametre predikcie* – prvky, vďaka ktorým je možne nadstaviť parametre predikcie.

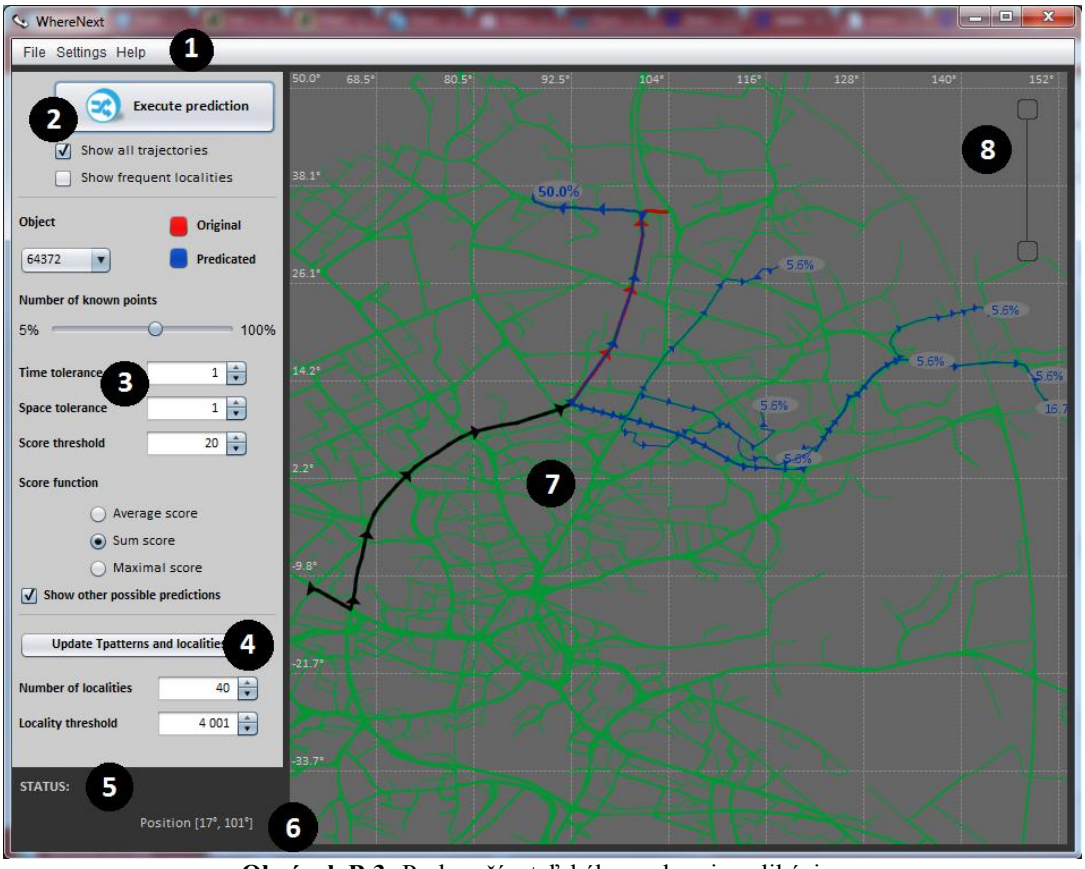
4. *Vyhľadanie frekventovaných lokalít a T-Patternov* – nadstavenie parametrov a spustenie nového vyhľadania frekventovaných lokalít a T-Patternov.

5. *Stav operácie* – užívateľ tu je oboznámený s ukončením časovo náročných operácií.

6. *Ukazovateľ pozície* – zobrazenie aktuálnej polohy, na ktorej sa práve nachádza kurzor.

7. *Mapa* – panel, v ktorom je zobrazená mapa s trajektóriami a po spustení predikcie sa tu zobrazí aj tá. Zelené čiary sú trajektórie ostatných objektov, čierna je známa časť trajektórie, červená je časť, ktorá sa predikuje a modrá čiara zobrazuje predikciu.

8. *Priblíženie a oddialenie mapy* – tlačidlá a posuvník na nadstavenie miery priblíženia mapy, ktoré sa zobrazia, ak sa kurzor dostane do ich blízkosti (na manipuláciu s mapou je vhodnejšie použiť myš).



Obrázok B.3: Prvky užívateľského rozhrania aplikácie.