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 DATABÁZÍ POHYBUJÍCÍCH SE OBJEKTŮ

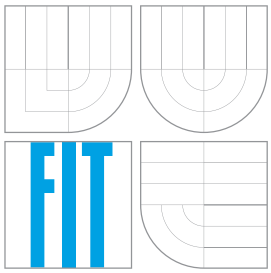
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR CHOVANEC

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 DATABÁZÍ POHYBUJÍCÍCH SE OBJEKTŮ

KNOWLEDGE DISCOVERY IN DATABASES OF MOVING OBJECTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VLADIMÍR CHOVANEC

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR CHMELAŘ

BRNO 2011

Abstrakt

Cílem této diplomové práce je obecně se seznámit s problematikou získávání znalostí a klasifikací. Práce dále navazuje na aplikaci SUNAR, která pak je v praktické části vylepšena o SVM klasifikaci průchodů osob mezi jednotlivými kamerami. V závěru se pak pojednává o způsobech vylepšení klasifikace i rozpoznávání v programu SUNAR.

Abstract

The aim of this master's thesis is to get familiar with problems of data mining and classification. This thesis also continues with application SUNAR, which is upgraded in practical part with SVM classification of persons passing between cameras. In the conclusion, we discuss ways to improve classification and person recognition in application SUNAR.

Klíčová slova

Získavání znalostí, časoprostorová data, identifikace trajektorie, Support Vector Machines (SVM), SUNAR.

Keywords

Knowledge discovery, spatio-temporal data, trajectory identification, Support Vector Machines (SVM), SUNAR.

Citace

Vladimír Chovanec: Získávání znalostí z databází pohybujících se objektů, diplomová práce, Brno, FIT VUT v Brně, 2011

Získávání znalostí z databází pohybujících se objektů

Prohlášení

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

.....
Vladimír Chovanec
25. května 2011

Poděkování

Na tomto mieste by som rád poďakoval vedúcemu práce Ing. Petrovi Chmelařovi za pomoc pri voľbe témy práce, odborné vedenie a za všetku pomoc v priebehu roka.

Obsah

Kapitola 1

Úvod

Letisková hala, metro, centrum mesta, či moderné budovy, majú jednu spoločnú vlastnosť: sledujú každý náš krok svojimi priemyselnými kamerami. Tu sa naskytá otázka nielen efektívneho uchovávania zozbieraných obrazových materiálov, ale aj ich efektívneho spracovania. Je zrejmé, že analýza záznamov zbieraných desiatkami kamier počas dlhého obdobia si vyžaduje tým odborníkov. Údaje sú natoľko cenné, že sa oplatí zaobstarať si takýto tým odborníkov.

Táto diplomová práca nadväzuje na Semestrálny projekt magisterského štúdia. Zo Semestrálneho projektu bola prevzatá a prepracovaná kapitola o databázových systémoch, kapitola Získavanie znalostí z databáz, a základ kapitoly venovanej identifikácií trajektórie. Táto však bola vo veľkej miere ešte prepracovaná a doplnená nielen o nové podkapitoly, ale aj o názorné obrázky.

Súčasťou práce je uvedenie čitateľa do problematiky získavania znalostí z databáz, pričom sa zameriame na databázu pohybujúcich sa objektov a ich trajektórií v priestoroch sledovaných viacerými statickými kamerami, a identifikácií toho istého objektu (osoby) v rôznych kamerách. Pretože problematika získavania znalostí z časopriestorových databáz je príliš obsiahla, niektoré kapitoly nie sú podrobné. Pre podrobné naštudovanie problému je potrebné prejsť si použitú literatúru.

V texte sa najprv zameriame na úvod do rôznych druhov databázových systémov. Následne kapitola venovaná získavaniu znalostí z databáz pojednáva o základných znalostiach tejto disciplíny, vysvetlí niektoré kľúčové pojmy a jednotlivé fázy získavania znalostí. Záver kapitoly je venovaný niektorým príkladom využitia z praxe. V kapitole Klasifikácia je zhrnutých niekoľko poznatkov z daného oboru, a je aj vysvetlený princíp fungovania niektorých základných klasifikačných techník. V závere kapitoly sa opäť posnažíme načrtnúť možné situácie, kedy nám klasifikácia môže byť veľmi nápomocná. Pri identifikácií trajektórie objektu a aj hľadaní totožných objektov v rôznych kamerách sa stretneme s mnohými problémami. Tieto sú opísané v samostatnej kapitole s názvom Identifikácia trajektórie. V tejto kapitole ďalej nájdeme opis techník, ktoré nám aspoň čiastočne pomáhajú pri riešení uvedených problémov.

Prínosnou kapitolou je posledná, ktorá je venovaná výsledkom praktickej časti tejto práce. V úvode sa dostaneme k opisu nástroja SUNAR[1] (a jeho modulov), ktorý bol hojne používaný a vylepšený v priebehu vytvárania tejto práce. Následne sa spoločne pozrieme na výstupy, ktoré budeme ďalej analyzovať. V úplnom závere sú uvedené možnosti vylepšenia do budúcnosti, aby bola identifikácia objektov kvalitnejšia.

[1] Charakteristika a viac informácií o aplikácií SUNAR sa nachádza v kapitole 6: Výsledky práce.

Cieľom práce bolo obohatiť aplikáciu SUNAR o SVM klasifikátor, ktorý na základe charakteristiky typického pohybu ľudí v letiskovej hale (ale teoreticky môže ísť o akýkoľvek priestor sledovaný viacerými statickými kamerami) určí najpravdepodobnejší výskyt osoby v inej z kamier.

Kapitola 2

Databázové systémy

Databázy už od 80-tych rokov patria k veľmi využívanému úložisku dát v určitej logickej štruktúre. Počas svojho vývoja sa dospelo k niekoľkým rôznym typom databázových systémov, ktoré sa navzájom líšia spôsobom ukladania dát a väzieb medzi nimi. Rôzne typy databázových systémov sa v čase skombinovali, čo umožnilo vznik nových, ktoré obsahujú výhody pôvodných databáz, alebo sa snažia eliminovať ich nevýhody.

2.1 Relačné databázy

Relačná databáza je taká databáza, ktorú používateľ vníma ako sústavu v čase sa meniacich normalizovaných tabuliek s usporiadanými stĺpcami. Každá tabuľka reprezentuje určitý typ entity a každý riadok v tejto tabuľke (záznam), jeden výskyt daného typu entity. Stĺpce predstavujú jednotlivé modelované vlastnosti (atribúty) daného typu entity. Príklad schémy je na obrázku 2.1. Viac informácií je možné získať napríklad v [42].

Aby si databáza mohla priradiť prívlastok „relačná“, musí spĺňať 12 Coddových pravidiel. Definoval ich Ted Codd v roku 1985. Ich znenie je nasledovné [10]:

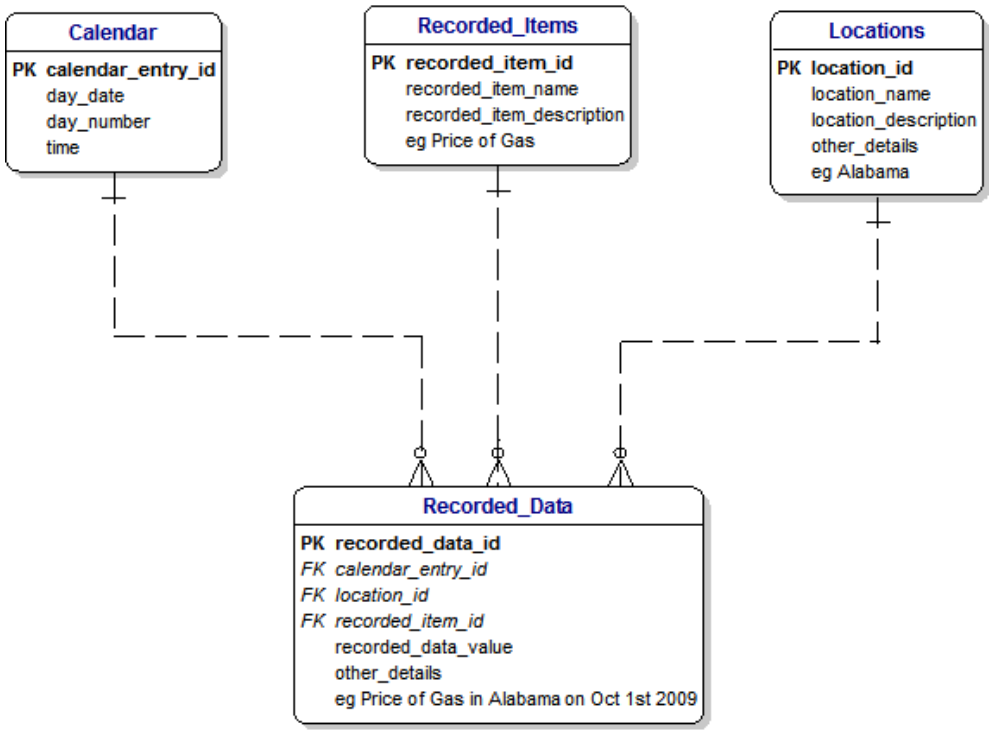
1. **Pravidlo informácie.** Všetky informácie v relačnej databáze sa reprezentujú explicitne na logickej úrovni a jediným spôsobom – hodnotami v tabuľkách.

2. **Pravidlo zaručeného prístupu.** Musí byť zaistené, aby každý údaj v relačnej databáze bol logicky prístupný použitím názvu tabuľky, hodnotou primárneho kľúča a názvu stĺpca.

3. **Systematické ošetrenie prázdnych hodnôt.** Prázdne hodnoty (odlišné od prázdneho znakového reťazca, reťazca medzier a odlišné od nulových alebo akýchkoľvek iných čísel) sú systematicky plne odporované relačným databázovým systémom pre reprezentáciu chýbajúcich informácií a neplatných informácií nezávisle na dátovom type.

4. **Dynamický online katalóg založený na relačnom modeli.** Popis databázy sa reprezentuje na logickej úrovni rovnakým spôsobom ako bežné údaje tak, aby sa oprávnení používatelia mohli dotazovať na tieto údaje pomocou rovnakého relačného jazyka, pomocou ktorého sa dotazujú na normálne údaje.

5. **Pravidlo komplexného dátového podjazyku.** Relačné systémy môžu podporovať niekoľko jazykov a rôzne režimy použitia terminálu (napr. režim doplňovania). Avšak



Obr. 2.1: Príklad schémy relačnej databázy. Prebrané z [41].

musí existovať prinajmenšom jeden jazyk, ktorého príkazy sú vyjadriteľné na nejakú dobre definovanú syntax ako reťazce znakov, a tento komplexný jazyk podporuje všetky nasledujúce položky: definíciu dát, definíciu pohľadu, manipuláciu s dátami, integritné obmedzenia, autorizáciu, vymedzenie transakcie.

6. **Pravidlo aktualizácie pohľadu.** Všetky pohľady, ktoré je teoreticky možné aktualizovať, je rovnako možné aktualizovať systémovo.

7. **Vysokoúrovňové vkladanie, aktualizácia, odstraňovanie.** Schopnosť spracovávať základné relácie alebo odvodené relácie ako jediný operand, sa aplikuje nielen na vyhľadávanie dát, ale rovnako na vloženie, aktualizáciu a odstránenie dát.

8. **Fyzická dátová nezávislosť.** Aplikačné programy a terminálové aktivity zostávajú logicky nedotknuté, kedykoľvek sú prevedené nejaké zmeny buď v reprezentácií úložiska alebo prístupových metódach.

9. **Logická dátová nezávislosť.** Aplikačné programy a terminálové aktivity zostávajú logicky neporušené, pokiaľ sú v základných tabuľkách prevedené zmeny v uchovávaní informácií akéhokoľvek druhu.

10. **Nezávislosť integrity.** Integritné obmedzenia špecifické pre jednotlivé relačné databázy musí byť možné definovať v relačnom dátovom podjazyku a uložiť v katalógu, nie v aplikačných programoch.

11. **Distribučná nezávislosť.** Systém riadenia bázy dát má distribučnú nezávislosť.

12. **Pravidlo nenarušenia.** Ak má relačný systém nízkoúrovňový jazyk (spracovávajúci záznamy jednotlivo), nie je možné pomocou tohto jazyka rušiť alebo obchádzať pravidlá integrity a obmedzenia vyjadrené vo vysokoúrovňovom relačnom jazyku.

Dnes už prakticky žiadna z rozšírených databáz nie je relačná. Postupom času totiž do relačných databáz začali prenikať ďalšie prvky (triggre, nové dátové typy a operácie pre priestorové dáta, jednoduchá podpora dedičnosti, ...). Hovoríme o tzv. *postrelačných databázových systémoch.*

2.2 Objektovo-orientované databázy

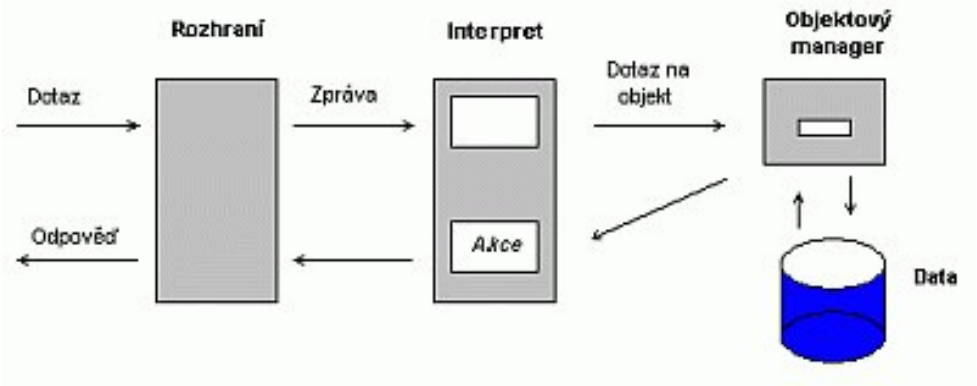
Objektovo-orientované databázové systémy (v anglickom jazyku: object-oriented database management systems – OODBMS) sú objektovým rozšírením relačných databáz. Do relačného modelu teda vkladajú prvky objektovo-orientovaného programovania, a popisuje sa správenie objektov. Prácu s OODBMS popisuje jazyk OQL (Object Query Language).

Vývoj a používanie OODBMS však neznamenal ústup či zánik relačných databáz, ale bol skôr chápaný ako alternatíva a rozšírenie k relačným DBMS. Výhodou OODBMS je možnosť priameho vyjadrenia zložitosti modelovanej reality v databáze. Vďaka tomu, že súčasťou uložených objektov je tiež ich chovanie, zjednodušuje sa štruktúra aplikácie. Odpadajú medzikroky prevodu objektov do normalizovaných tabuliek relačnej databázi pred uložením každého objektu. Takisto je zjednodušený i opačný krok, načítanie objektov z databázi do aplikácie. Pri použití relačných foriem by sme museli jednoduchú štruktúru uloženú v databázi poskladať a vytvoriť objekt, s ktorým už vie aplikácia pracovať. [8] tvrdí, že nevýhodou je zložitý a nákladný prechod od DBMS k OODBMS, a takisto vyššia zložitosť a tažšia pochopiteľnosť. DBMS majú navyše základ v matematickej teórií. Na druhej strane jednoduchosť relačných databáz znamená malú modelovaciu silu. Z toho je zrejmé, že relačné platformy sú výhodné pre správu veľkého objemu jednoduchých dát, pretože obsahujú dobré prostriedky pre selekciu dát, ale komplikovaná je manipulácia s nimi. Objektovo-orientované platformy zase dobre vystihujú zložité vzťahy medzi objektami, umožňujú flexibilnú manipuláciu s dátami, ale dotazovacie možnosti sú pod úrovňou štandardizovaného SQL. Oblasť využitia OODBMS je hlavne v CAD[1] a GIS[2] aplikáciách, CASE[3] systémoch, systémoch pre podporu rozhodovania a pre správu dokumentov. Ide teda o oblasti, kde pracujeme s perzistentnými objektami alebo s hustými grafmi objektov. Architektúra OODBMS je znázornená na obrázku 2.2.

2.3 Objektovo-relačné databázy

V [8] sa uvádza, že objektovo-relačné databázové systémy (ORDBMS- Object-relational database management system) v sebe kombinujú práve tie dobré vlastnosti z DBMS a OODBMS. Nový model vychádza z relačného modelu, ale je otvorený pre rozširovanie. Akonáhle vznikne požiadavka na prechod na objektovo-relačný systém, prechod je jednoduchý, pretože ORDBMS vie ukladať relačné dáta. Rozšírenie umožňuje užívateľom definovať nové zložené dátové typy a metódy pre manipuláciu s týmito dátovými typmi. Tieto metódy zahŕňajú dedičnosť, zapúzdrenie a polymorfizmus. Objektový model v ORDBMS je podobný

[1] Computer-aided design
[2] Geographic information system
[3] Computer-aided software engineering



Obr. 2.2: Architektúra OODBMS. Prebrané z [26].

modelu tried v jazyku C++ či Java. Pre manipuláciu s objektami v databázi je opäť použitý jazyk SQL. Prvý uznaný štandard sa objavil v roku 1999[4] a novšia verzia v roku 2003[5]. Objektovo-relačný model prináša a definuje pojmy ako Štrukturovaný uživateľsky definovaný typ – abstraktné dátové typy (ADT), referenciu, ako aj pojem kolekcia.

Medzi základné výhody ORDBMS patrí to, že objekty môžu jednoduchšie a komplexne modelovať objekty reálneho sveta. Ďalej sem patrí znovupoužiteľnosť, čo uľahčuje vývoj samotných aplikácií. Okrem dát, ORDBMS ukladá typy objektov a to tak, že môžu byť použité rôznymi aplikáciami. Pokiaľ databázová tabuľka obsahuje iba dáta, objekty môžu zahŕňať operácie, ktoré sú potrebné k ich spracovaniu. Čisto relačný prístup obmedzuje modelovanie vzťahov za použitia primárnych a cudzích kľúčov a s tým spojených integritných obmedzení. Objektovo-relačný model rovnako tak dobre ako model čisto objektovo-orientovaný, poskytuje bohaté možnosti pre popis vzťahov.

2.4 Časové databázy

V časových databázach (temporal databases) sú podľa [36] všetky záznamy obohatené o model času. Čas môže byť vyjadrený ako diskrétna veličina, ktorá je lineárne usporiadaná a nemá hraničné body. Je reprezentovaná väčšinou množinou celých čísel. Používa sa na reprezentáciu stavov a na databázu sa da pozerať ako na postupnú sekvenciu „snímkov“. Vhodný príklad je zachytenie stavu pohybujúceho sa objektu na videozázname. Pre každý snímok môže byť poloha objektu iná, a teda je vhodné ukladať pozíciu objektu pre každý snímok zvlášť a nenastavovať im interval platnosti.

Druhý typ uloženia času je uloženie intervalom platnosti. Nastaví sa teda začiatočný a koncový čas platnosti, pričom fakt je platný v ktoromkoľvek časovej bode medzi nimi. Tento spôsob uloženia je častejší ako predchádzajúci.

Na základe toho rozdeľuje [36] tabuľky do skupín:

- snímkové
- platného času
- transakčného času

[4]SQL:1999
[5]SQL:2003

- obojakého času

V snímkových tabuľkách je ku každému záznamu priradený časový okamih (napríklad pracovná pozícia: 2000 pomocný asistent, 2005 marketingový špecialista, 2008 vedúci oddelenia). Snímková tabuľka je bližšie k temporálnej logike. Každá databáza popisuje stav sveta v konkrétnom časovom okamihu. Relácia usporiadania definuje tok času. Uloženie dát v snímkovej tabuľke však nie je vhodné ak potrebujeme konštruovať dotazy, ktoré majú vrátiť všetky okamihy, kedy platil predikát.

Čas platnosti (v tabuľkách platného času) vyjadruje čas, ktorý bol fakt platný (pravdivý) v skutočnom svete, nezávisle na prítomnosť v databázi. Môže byť z minulosti, zo súčasnosti, ale aj z budúcnosti. Napríklad môžeme do databázi ukladať informácie o dovolenkách zamestnancov. Väčšina dovoleniek už prebehla v minulosti a teda čas platnosti má v minulosti. Časť zamestnancov je na dovolenke práve teraz, a niektorí zamestnanci už majú nahlásené dovolenky vopred (čas platnosti faktu v budúcnosti). Ak sa tento informačný systém začal používať pred rokom a potrebujeme do neho vložiť informácie o dovolenkách zamestnancov zo staršieho obdobia (napríklad kvôli štatistikám), je to možné a korektné, pretože platnosť záznamu môže byť vložená aj spätne.

Transakčný čas naopak vyjadruje čas, kedy bol fakt (záznam) uložený v databáze. Ide teda o čas od jeho vloženia až po jeho vymazanie. V tabuľkách obojakého času je uložený nie len čas platnosti, ale aj transakčný čas. Takýto druh tabuliek je vhodný vtedy, keď je potrebné uchovávať si históriu zmien. To znamená, že pri zmene dát sa len pridá nový záznam s opravenými hodnotami a pri vymazaní sa nastaví korektný čas platnosti pôvodného záznamu.

Časové databázy majú aj svoje ďalšie špecifiká. K nim patria nielen nové prístupy k indexovaniu, ale napríklad aj zhlukovanie prekrývajúcich sa (nadväzujúcich) intervalov nesúcich rovnaké dáta kvôli ďalšiemu spracovaniu. Paralelizácia takýchto dotazov takisto nepatrí medzi triviálne záležitosti.

Jazyk pre prácu s časovými databázami doteraz nebol štandardizovaný. V [12] sa z používaných uvádza aspoň TQUEL a TSQL2. TSQL2 je rozšírenie štandardu SQL-92. Podporuje prácu s časom platnosti aj s transakčným časom záznamov bez nutnosti definovať vlastné stĺpce s časovými razítkami. Toto trochu zneprehľadňuje sémantiku TSQL2 a robí ju ťažšie pochopiteľnou. Takisto niektoré špecifické dotazy sú komplikované. Konštrukcie jazyka TSQL2 boli navrhnuté iba pre jazyk SQL, a teda nemôžu byť použité napríklad pre QBE a Datalog.

2.5 Priestorové databázy

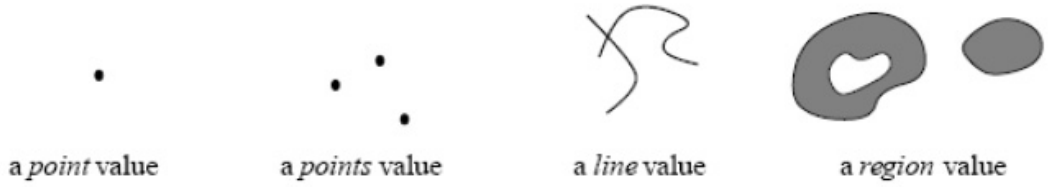
Priestorové databázy (spatial databases) predstavujú rozšírenie relačných databáz o priestorové dátové typy a o operácie nad priestorovými typmi. Majú mnohoraké využitie, pričom častokrát sa spájajú s GIS (Geografický informačný systém). Nové priestorové dátové typy prinášajú nové problémy a komplikácie týkajúce sa uloženia a spracovania dát. Napríklad podľa [29] ukladá NASA terabajt nových priestorových dát denne.

Priestorové databázy sa často mylne zamieňajú s pojmom GIS. Je treba si uvedomiť, že priestorové databázy iba poskytujú podporu pre prácu s N-rozmernými geometrickými útvarmi, zatiaľ čo GIS je už aplikácia, ktorá môže používať priestorovú databázu. Viac informácií o GIS je napríklad v publikácií [16] alebo [37].

Základné dátové typy v priestorových databázach sú podľa [8] nasledovné:

- bod

- body
- línia
- región



Obr. 2.3: Základné priestorové dátové typy definované v [8].

[29] rozdeľuje operácie nad priestorovými dátami na dve základné skupiny: topologické a netopologické. Medzi topologické operácie patria nasledovné:

- **Endpoint (point, arc)** - Bod na konci krivky.
- **Simple-nonself-intersection (arc)** - Nepretínajúca sa krivka.
- **On-boundary (point, region)** - Vancouver je na hranici Kanady a Spojených Štátov.
- **Inside (point, region)** - Minneapolis sa nachádza v štáte Minnesota.
- **Outside (point, region)** - Madison sa nachádza mimo štátu Minnesota.
- **Open (region)** - Vnútrozemie Kanady je otvorená oblasť (nezapočítavaú sa hranice).
- **Close (region)** - Štát Carleton je uzavretá oblasť (vrátane hraníc).
- **Connected (region)** - Švajčiarsko je spojitý región, zatiaľ čo Japonsko nie je (rozprestiera sa na niekoľkých ostrovoch).
- **Inside (point, loop)** - Bod je vnútri slučky.
- **Crosses (arc, region)** - Cesta (krivka) prechádza lesom (oblasť).
- **Touches (arc, region)** - Medzištátna diaľnica IS 90 (krivka) prechádza okolo Michiganského jazera.
- **Overlap (region, region)** - Oblasť sa prekrýva s inou oblasťou.

Základné netopologické operácie sú:

- **Euclidian distance (point, point)** - Vzdialenosť medzi dvoma bodmi.
- **Direction (point, point)** - Madison je východne od Minneapolisu.
- **Length (arc)** - Vzdialenosť jednotky vektora je jedna jednotka.
- **Perimeter (area)** - Obvod jednotkového štvorca sú 4 jednotky.

- **Area (region)** - Obsah jednotkového štvorca je 1 jednotka.

Implementácia uvedených operácií a ich začlenenie do konkrétneho databázového systému už závisí na jeho tvorcovi. Napríklad pre Oracle 11g je k dispozícií SPATIAL rozšírenie [22] a PostgreSQL 9.0 podporuje geometrické dátové typy ihneď po inštalácií [25], prípadne je možné doinštalovať rozšírenie PostGIS [27].

S používaním priestorových databáz sa viaže množstvo ďalších problémov, či už ide o nové metódy indexácie (bodových, plošných, viacrozmerných útvarov) alebo reprezentáciu Euklidovského priestoru v diskrétnom súradnicovom systéme (nie sme schopní uložiť do počítača ľubovoľné reálne číslo, ale iba racionálne a často iba desatinné s rôznou mierou presnosti). Viac sa je možné dočítať v odborných publikáciách zameraných konkrétne na túto tému. Ako príklad môžeme uviesť [29], [11], alebo [15].

2.6 Časovo-priestorové databázy

V predchádzajúcich kapitolách sme sa oboznámili s priestorovými a s časovými databázami. Práve časovo-priestorové databázy sú kľúčové pre spracovávanie dát v praktickej časti tejto práce, a preto sa s nimi zoznámime bližšie.

V [3] sa uvádza, že časovo-priestorové databázy predstavujú kombináciu konceptu časových a priestorových databáz. To znamená, že podporujú nielen priestorové dátové typy a operácie nad nimi (vrátane indexácie), ale každý záznam v databáze môže byť navyše obohatený o model času, ako bolo uvedené v predchádzajúcej kapitole. Časovo-priestorové databázy teda v sebe zahŕňajú všetky výhody spomínaných databáz, ale zároveň od nich aj preberajú všetky problémy (obtiažne zhlukovanie, paralelizácia, a iné).

Časovo-priestorové databázy sa podľa [46] typicky zaoberajú geometrickými objektami meniacimi sa v čase. Často sa v tejto problematike hovorí o abstraktnom dátovom type - pohybujúci sa bod (moving point) a pohybujúca sa oblasť (moving region). Základnou charakteristikou konkrétnych fyzických objektov je pozícia, čas, a prípadne tvar. Tieto charakteristiky môžu byť spojené s takmer ľubovoľným objektom reálneho sveta.

[8] tvrdí, že v súčasnosti môžeme rozlišovať dva spôsoby reprezentácie časovo-priestorových objektov. Sú nimi diskrétne a spojite pohybujúce sa objekty. Zatiaľ čo reprezentácia diskrétne pohybujúcich sa objektov sa rieši napríklad pridaním jedného časového atribútu, je reprezentácia spojite pohybujúcich sa objektov o niečo zložitejšia. Nie je totiž možné oddeľovať časové a priestorové informácie a je komplikované meniť dáta po každej zmene.

2.6.1 Databáza pohybujúcich sa objektov

Cieľom výskumu databáz pohybujúcich sa objektov je rozšíriť databázovú technológiu tak, aby mohla byť akákoľvek pohybujúca sa entita reprezentovaná v databáze. Zároveň je potrebné vytvoriť robustný dotazovací jazyk, pomocou ktorého by bolo možné formulovať dotazy uspokojujúce naše potreby. V tejto podkapitole sa pozrieme bližšie na motiváciu tohto výskumu.

Podľa [12] rozdeľujeme koncepty databáz pohybujúcich sa objektov podľa ich perspektívy do nasledovných skupín:

- správa polohy
- časovo-priestorových dáta

Perspektíva správy polohy

Tento prístup zahŕňa problém správy pozície množiny entít v databáze. [12] uvádza ako príklad polohu taxíkov v meste. Pre konkrétny časový okamih to nie je problém. Môžeme mať tabuľku s ID taxíkov ako kľúč a stĺpce pre uloženie x-ovej a y-ovej súradnice. Avšak taxíky sa pohybujú. Aby sme mali k dispozícií vždy aktuálnu polohu taxíkov, musela by byť poloha často aktualizovaná a to pre každý taxík zvlášť. Tu sa stretávame s nepríjemným kompromisom: ak by taxíky posielali informáciu o svojej polohe priveľmi často, odchýlka v databáze od ich skutočnej polohy by bola síce malá, ale za cenu obrovskej záťaže na databázu. Toto nie je uskutočnitelné pre väčšiu množinu entít (taxíkov). Naopak, ak by aktualizácia polohy nebola posielaná často, odchýlka polohy v databáze od skutočnej polohy taxíkov by sa stala veľkou, čo takisto nie je žiadúce.

Toto viedlo k myšlienke ukladať do databázy pre každý pohybujúci sa objekt nie aktuálnu pozíciu, ale radšej jeho pohybový vektor, ktorý činí polohu funkciou času. Vďaka tomu, ak uchováme pre objekt jeho pozíciu v čase t_0 spolu s jeho rýchlosťou a smerom pohybu v tom čase, možeme odvodiť predpokladanú pozíciu pre akýkoľvek časový okamih po čase t_0. Samozrejme, pohybový vektor musí byť tiež z času na čas aktualizovaný, ale oveľa zriedkavejšie ako keby sme ukladali vždy len aktuálnu pozíciu. Tým teda docielime žiadaný kompromis medzi presnosťou aktuálnej polohy a záťažou servra pri jej aktualizácií.

Preto sa v tejto perspektíve zaujímame o uchovávanie dynamickej polohy každého pohybujúceho sa objektu a môžeme sa dotazovať na aktuálnu polohu, na polohu v blízkej budúcnosti, alebo na akýkoľvek vzťah, ktorý sa po čase môže vyvinúť medzi pohybujúcimi sa entitami.

Všimnime si, že z pohľadu časových databáz tento prístup vôbec nepredstavuje časovú databázu. Je to len snímková databáza uchovávajúca súčasný stav sveta. Neuchováva sa žiadna história pohybu.

Perspektíva časovo-priestorových dát

Táto perspektíva sa podľa [12] snaží narábať s rôznymi druhmi dát, ktoré môžu byť uložené v (statickej) priestorovej databáze a pozorovať, že takéto dáta sa možu zmeniť v čase. V databáze chceme opísať nielen súčasný stav priestorových dát, ale tiež celú históriu ich zmien. Požadujeme preto schopnosť ísť späť v čase do konkrétneho časového okamihu a získať vtedajší stav (hodnotu). Navyše, chceli by sme pochopiť, ako sa záznam menil, analyzovať, kedy boli splnené určité vzťahy, a tak ďalej. Tu sa v [12] nastoľujú dve otázky:

- Aké typy dát sú uložené v priestorovej databáze?
- Aké typy zmien môžu nastať?

Odpoveď na prvú otázku závisí od konkrétneho prípadu, aké entity potrebujeme v databáze uchovávať. Vo všeobecnosti by sme si ale mali vystačiť s dátovými typmi definovanými v kapitole 2.5[6].

Pokiaľ ide o typy zmien, hlavný rozdiel sa týka diskrétnych zmien a spojitých zmien.

Klasický výskum časovo-priestorových databáz sa zameriava na diskrétne zmeny pre všetky spomenuté priestorové dátové typy. Spojitými zmenami sa zaoberá kniha [12], a práve toto rozumie pod pojmom *pohybujúci sa objekt*. Zatiaľ čo sa diskrétne zmeny vyskytujú v ľubovoľnom druhu priestorovej entity, spojité zmeny sú najviac relevantné pre bod a oblasť.

[6] Boli to bod, body, línia a región.

2.6.2 Indexácia v časovo-priestorových databázach

Dizajn a konštrukcia indexov bola vždy intenzívnou a rozsiahlou tematikou v databázovej problematike. Hlavnou úlohou indexovej štruktúry je zaistiť rýchly prístup k jednému alebo viacerým záznamom v databáze na základe vyhľadávacieho kľúča, a tak sa vyhnúť sekvenčnému hľadaniu cez celú tabuľku. Pre tento účel poskytujú indexové štruktúry konkrétne operácie, ktoré pomáhajú pri špecifických typoch požiadaviek. Tieto požiadavky sa dotazujú buď na presnú zhodu hodnoty, na konkrétny rozsah hodnôt, alebo na sekvenčný prístup. Tieto typy požiadaviek hrajú dôležitú úlohu v spracovaní požiadavky. Pre štandardné alfanumerické dáta bolo navrhnutých niekoľko indexových štruktúr, vrátane indexovej sekvenčnej metódy (ISAM), indexy založené na stromovej štruktúre ($B-tree$, $B^{+}-tree$, $B^{*}-tree$), a indexy založené na hashovaní. Podrobnejší opis týchto indexových štruktúr je napríklad v [15].

Vznikajúce neštandardné aplikácie, ako napríklad multimediálne, lekárske, enviromentálne, alebo bioinformatické, sú založené na čoraz komplexnejších dátach a algoritmoch a vyžadujú nové, sofistikovanejšie indexovacie techniky. Je zrejmé, že časovo-priestorové databázy taktiež patria do tejto kategórie. Pre priestorové dáta bolo vymyslených niekoľko indexových štruktúr, ktoré sme si uviedli v predchádzajúcom odstavci. Tieto štruktúry však v sebe nezahŕňajú časové hľadisko. Pre časové dáta boli vymyslené vlastné časové indexové štruktúry, ktoré kladú dôraz na čas platnosti alebo na transakčný čas. Tieto štruktúry však zase nezahŕňajú priestorové hľadisko. Narábajú hlavne s postupným konštantným plynutím času (dáta sa menia diskrétne). Napríklad plat zamestnanca sa z času na čas zmení, ale mení sa len v určitých časových okamihoch, pričom v čase medzi nimi zostáva konštantný. Oba spôsoby indexácie formujú základ pre nový cieľ - vytvorenie časovo-priestorovej indexovej štruktúry. Ich snahou je najmä indexovanie pohybu priestorových objektov v čase. Napredovanie v tejto oblasti je veľmi dynamické, a pre aktuálny stav je potrebné vždy siahnuť po najnovšej publikácií.

Indexácia trajektórie

Vzorkovaním pohybu získame lomenú úsečku reprezentujúcu trajektóriu pohybujúceho sa objektu. S trajektóriou objektu sa môže narábať ako s trojrozmernými priestorovými dátami. Taktiež v prípade analýzy histórie pohybu je hlavnou výzvou dotazovať sa na spojitú zmenu polohy. Indexovanie aj predchádzajúcich polôh pohybu si vyžaduje uchovanie trajektórie. Toto je rozdiel oproti metódam na indexáciu pohybu iba v súčasnom čase, kde minulosť nehrá žiadnu rolu. Dáta sa teda postupne vkladajú do databázy s ohľadom na čas (s pribúdajúcim časom pribúdajú nové dáta).

2.7 Zhrnutie

Táto kapitola bola venovaná predstaveniu rôznych typov databázových systémov. Jej úlohou bolo oboznámiť čitateľa s rôznymi prístupmi k rôznym typom dát. V žiadnom prípade sa však nejedná o vyčerpávajúce informácie. Tie je potrebné hľadať v špecializovanej literatúre. Okrem uvedených databázových systémov ešte existuje množstvo ďalších (XML databázy, multimediálne databázy, deduktívne databázy, ...) a takisto ich kombinácie. Tie sa od spomenutých môžu líšiť v prístupe k dátam, v podpore iných dátavých typov, alebo napríklad môžu obsahovať špecifické indexové štruktúry.

Kapitola 3

Získavanie znalostí z databáz

Doba, v ktorej žijeme, sa často označuje ako *informačná*. Informácie zabezpečujú výhodu oproti konkurencií a to implikuje úspech. Sofistikované zariadenia zbierajú kvantá údajov a čoraz viac organizácií si za niekoľko rokov nazhromaždilo obrovské množstvo informácií, ktoré nie je problém uložiť, ale vydolovať zaujímvé (relevantné) informácie, ktoré sú na prvý pohľad skryté. Dáta sú typicky uložené v dátovom sklade. Obor zaoberajúci sa získavaním znalostí z databáz (KDD – Knowledge Discovery and Data Minig) zaznamenal obrovský rozmach najmä v posledných rokoch.

Internetových predajcov zaujíma, ktoré produkty vkladajú zákazníci najčastejšie spolu do košíka, lekárov, ktorí pacienti sú nachylný na ktoré choroby, banky zaujíma akú vysokú pôžičku môžu poskytnúť klientovi aby nemal problém s jej splácaním alebo sa zaujímajú o neobvyklé (podozrivé) bankové transakcie, a podobne. V týchto obrovských množstvách dát nie je možné jednoducho sa orientovať, a informácie ktoré nás zaujímajú sú častokrát skryté či ťažko vydolovatelné.

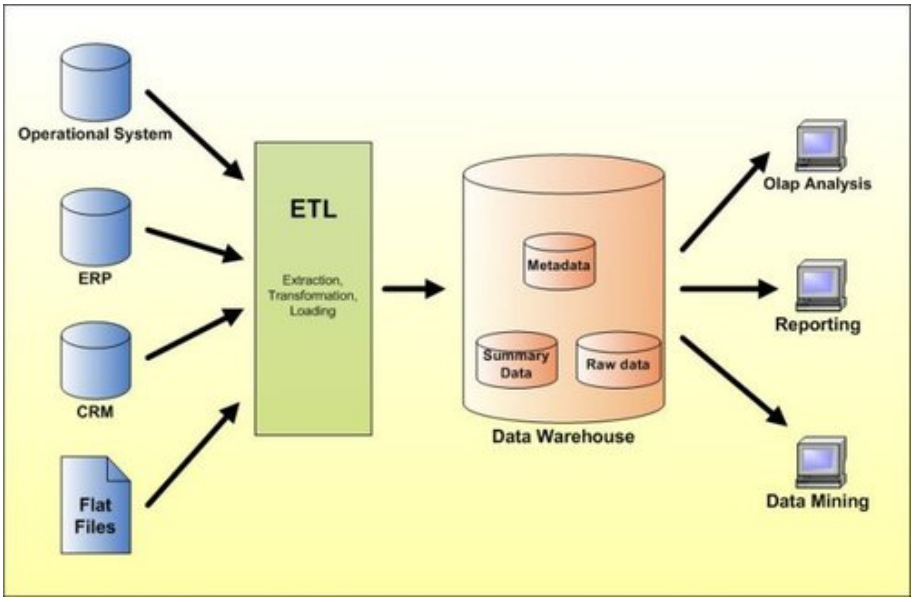
Dnes už máme k dispozícií viac údajov ako dokážeme spracovať. Preto vznikol obor, ktorý má za úlohu skúmať metódy a spôsoby efektívneho získavania znalostí z databáz. Tento obor sa častokrát nepresne označuje pojmom Data mining. Data mining je len jednou zložkou celého procesu získavania znalostí, avšak získal si takú popularitu, že sa používa aj ako súhrnné označenie pre celý proces zahŕňajúci niekoľko fáz.

3.1 Dátový sklad

Dátový sklad je podľa jednej z definícií „podnikovo štrukturované úložisko subjektovo orientovaných, integrovaných, časovo premenlivých, historických dát použitých na získavanie informácií a podporu rozhodovania, obsahuje atomické i sumárne dáta.“ (W.H.Inmon).

Inak povedané, ide o centrálne úložisko obrovského množstva dát slúžiaceho primárne na čítanie. Zápis je vykonávaný len sporadicky (pri dopĺňaní záznamov do skladu), a mazanie starých údajov prebieha taktiež len vo veľmi dlhých časových intervaloch, alebo vôbec. Dátové sklady sú zdrojom dát pri získavaní znalostí. Dáta v nich sú už predspracované (vyčistené) a v prípade viacerých zdrojov aj vhodne integrované do seba. Okrem základných dát zhromaždených z rôznych zdrojov, obsahujú dátové sklady aj predpočítané agregované dáta, ktoré slúžia na urýchlenie dotazov. To však znamená, že je nutné agregované dáta prepočítať pri každej zmene dát. To je jeden z dôvodov, prečo dátové sklady slúžia primárne na čítanie.

Dáta sú v dátovom sklade typicky uložené v tzv. n-rozmernej kocke, aby bola možná



Obr. 3.1: Získavanie znalostí z databáz. Prevzaté z [1].

komplexná analýza a vizualizácia. Dimenzie sú hierarchicky faktorizované podľa úrovne, čo umožňuje agregačné výpočty. Príklad uloženia dát v n-rozmernej kocke je na obrázku 3.2.

3.2 Data mining

Ako sme uviedli na začiatku kapitoly, data mining je len jednou fázou v procese získavania znalostí z databáz. Celý proces je podľa [45] iteratívny a interaktívny, a sa skladá z nasledovných krokov ilustrovaných aj na obrázku 3.3:

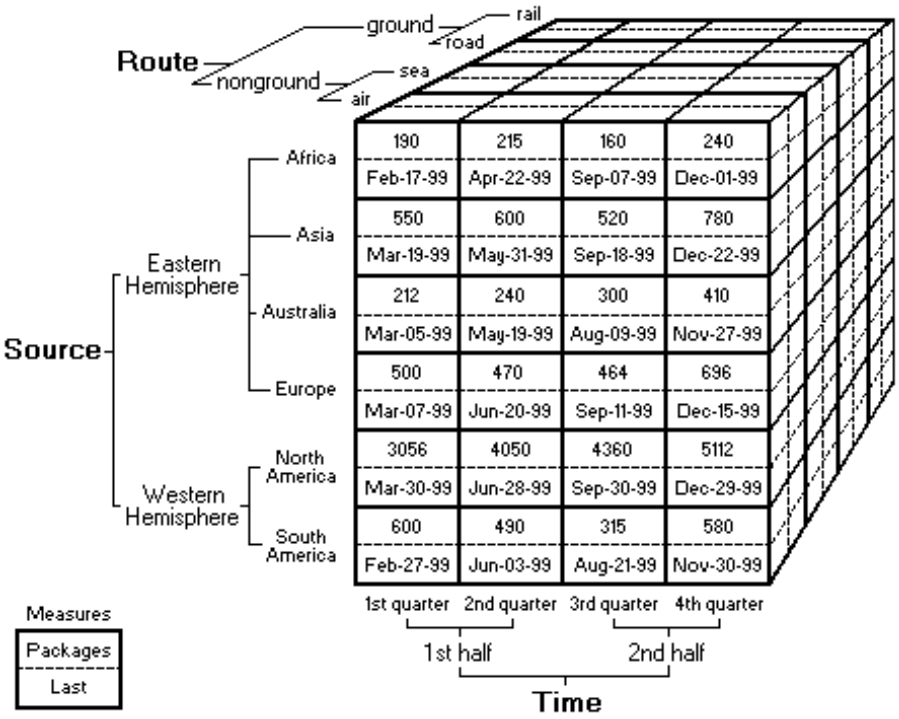
- Čistenie a integrácia dát
- Výber a transformácia dát
- Dolovanie z dát (data mining)
- Vyhodnotenie a prezentácia

Čistenie a integrácia dát

Čistenie a integrácia dát je prvým krokom pri získavaní znalostí z databáz. Je nutný, pretože vstupné dáta sú nekompletné, zašumené a nekonzistentné. Takisto sa v dátach môžu vyskytovať duplicity. Nekompletnosť dát môže spôsobiť napríklad ľudský faktor, zašumenosť chyba meracieho hardvéru, nekonzistentnosť rozporom pri zbieraní dát z rôznych zdrojov (napr. známkovanie 1, 2, 3, 4 a A, B, C, D, E, FX). Vstupné dáta potrebujeme mať „čisté“, pretože iba z kvalitných dát obdržíme kvalitné výsledky.

Výber a transformácia dát

Cieľom tejto fázy je vybrať dáta, ktoré sú pre riešenie danej analytickej úlohy relevantné. Často je to zapríčinené aj veľkosťou dátového skladu, čo by spôsobilo spomalenie dolovania.



Obr. 3.2: Príklad uloženia podnikových dát v dátovom sklade reprezentovanom trojrozmernou kockou. Prevzaté z [30].

Redukované dáta sa následne transformujú do konsolidovanej podoby vhodnej na dolovanie. Môže ísť napríklad o sumarizáciu, agregáciu, vyhladenie či normalizáciu.

Dolovanie dát

Tento krok predstavuje jadro procesu získavania znalostí. Má za úlohu vytvoriť model dát použitím určitej metódy a konkrétneho algoritmu. Bližšie informácie sú uvedené napríklad v [35].

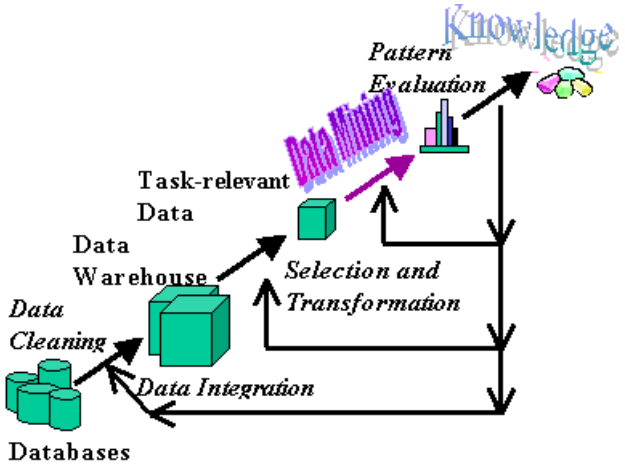
Vyhodnotenie a prezentácia

V tejto záverečnej fáze sa vyhodnotia výsledky získané predchádzajúcimi fázami. Pre relevantné výsledky je potrebné dostatočné množstvo kvalitných dát. Výsledky (znalosti) sú následne prezentované užívateľovi využitím techník vizualizácie a reprezentácie znalostí, aby užívateľ lepšie pochopil získané znalosti.

3.3 Druhy dát na dolovanie

Dolovanie je možné v akomkoľvek druhu dát. Dáta sa v [45] rozdeľujú do 2 základných skupín:

- Kvantitatívne – sú dáta, ktoré sú spojité a môžu nadobúdať nekonečne veľa hodnôt. Je na nich definované usporiadanie. Príkladom je dĺžka strany geometrického útvaru, plat, ...



Obr. 3.3: Celý proces získavania znalostí. Prevzaté z [44].

- Kategorické – sú dáta, ktoré nadobúdajú diskrétne hodnoty, alebo „malé“ množstvo rôznych hodnôt. Usporiadanie môže byť definované (známka, ...), ale nemusí (farba, ...)

Zdroje dát sú najčastejšie uložené v relačných databázach, dátových skladoch, v transakčných databázach, alebo prichádzajú v prúde (obraz z kamery).

3.4 Typy dolovacích úloh

Podľa [45] existujú dve základné typy dolovacích úloh:

- Deskriptívne
- Prediktívne

Deskriptívne úlohy sú charakterizované všeobecnými vlastnosťami analyzovaných dát. Ako príklad môžeme uviesť už spomínané spoločné produkty v nákupnom košíku, zisťované pri analýze nákupov v elektronických obchodoch.

Prediktívne úlohy robia dedukciu na predpoveď budúceho chovania na základe analýzy súčasných dát. Predstaviteľom tohto typu úloh je napríklad klasifikácia. Tá nám umožní napríklad na základe doterajšieho správania zákazníkov pri splácaní poskytnutého úveru predpovedať správanie nového zákazníka, teda aké veľké splátky bude schopný splácať. Tým sa predíde nesolventným klientom a banka nestratí zbytočne peniaze.

Systémy pre dolovanie dát spravidla umožňujú riešenie rôznych typov dolovacích úloh, naviac i rôznymi algoritmami. Je to jednak preto, že musia byť dostatočne univerzálne, ale i preto, že užívatelia často nemajú predstavu o tom, aký model ich dát môže byť užitočný.

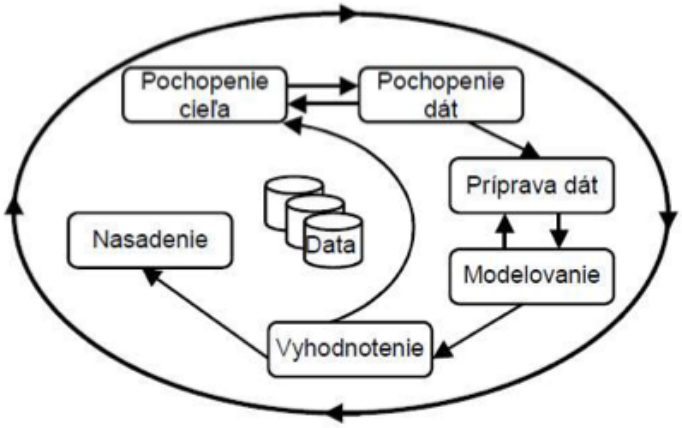
3.5 Zhrnutie

Opísali a vysvetlili sme si pojem Získavanie znalostí z databáz a dátový sklad. Identifikovali sme jednotlivé fázy celého procesu a stručne sme si charakterizovali druhy dát a typy dolovacích úloh. Načrtli sme takisto niektoré významné problémy, s ktorými sa pri získavaní znalostí stretávame. Vo všetkých týchto oblastiach by nebolo možné získať požadované

znalosti manuálne. Mnohé z nich navyše musia byť spracovávané v reálnom čase, alebo v dohľadnej dobe po vykonaní operácie (napríklad pre odhaľovanie podvodov). Získavanie znalostí teda nepatrí medzi jednoduché disciplíny.

V našom prípade môžeme sledovať najčastejšie trasy ľudí na letisku a na základe toho identifikovať miesta, na ktorých sa tvoria zápchy a ľudia nemôžu pokračovať vo svojej ceste obvyklým tempom. Tieto cesty by sa potom mali rozšíriť alebo majiteľ letiska by mal vybudovať nové alternatívne cesty. Nové cesty môžu byť vybudované napríklad na úkor už existujúcich málo využívaných ciest. Nami získané znalosti by takisto mohli byť využité pri architektonickom návrhu nového letiska. Vytvorená aplikácia by však nebola prispôsobená iba na prostredie letiska, ale prakticky na akýkoľvek interiérový či exteriérový priestor čím sa ešte zvýšia možnosti využitia.

Iným využitím získavania znalostí z pohybu osôb po letiskovej hale je identifikácia tej istej osoby v rôznych kamerách. Na to je však potrebná trénovacia množina vzoriek, z ktorej klasifikátor zistí najčastejšie trasy osôb a podľa toho sa bude snažiť klasifikovať nové osoby.



Obr. 3.4: Procesný model získavania znalostí. Prebraté z [23].

Kapitola 4

Klasifikácia

Klasifikácia (alebo presnejšie štatistická klasifikácia) je vedecká disciplína, ktorá sa zaoberá rozdelením objektov do tried na základe podobnosti ich vlastností. Tieto triedy a ani ich počet nemusí byť vopred známy. Úspešnosť klasifikácie sa počíta podľa vzorca 4.1, kde $n_{correct}$ je počet správne klasifikovaných vzoriek a n_{total} počet všetkých vzoriek. p reprezentuje úspešnosť klasifikácie (v percentách).

$$p = \frac{n_{correct}}{n_{total}} * 100\% \tag{4.1}$$

S klasifikáciou úzko súvisí pojem predikcia. Predikcia má za úlohu predpovedať neznámu hodnotu daného objektu na základe jeho znalostí.

Pre spoľahlivú klasifikáciu (predikciu) je nutné transformovať vzorky na kvalitný *vektor príznakov*. Vektor príznakov je súbor vlastností, ktoré sú relevantné pre zaradenie objektu do správnej triedy. Inak povedané, sú to vlastnosti, ktorými sa vzorky z jednotlivých tried líšia. Ak by sme napríklad chceli urobiť klasifikátor, ktorý rozpozná slivky od broskýň, do vektoru príznakov by bolo vhodné zahrnúť farbu objektu. Farba by teda bola príznakom s vysokou rozlišovacou schopnosťou. Ak by sme tam zahrnuli iba jeho váhu, klasifikátor by mal pravdepodobne veľmi nízku mieru spoľahlivosti. Vtedy hovoríme o príznaku s nízkou rozlišovacou schopnosťou. Vektor príznakov samozrejme môže obsahovať viac príznakov (a je to aj žiadúce), ktoré spolu vôbec nesúvisia. V našom príklade so slivkami a broskyňami by sme sa okrem farby mohli rozhodovať aj na základe ich chemického rozboru či tvaru. Kvalitný klasifikačný algoritmus nemá problém spracovať aj veľké množstvo príznakov (rádovo stovky), pričom príznakom s nízkou rozlišovacou schopnosťou priradí nízku váhu alebo ich úplne ignoruje. Priradenie váh dôležitosti sa robí pri učení (čiže algoritmus si to zistí sám, nemusíme mu to explicitne zadať).

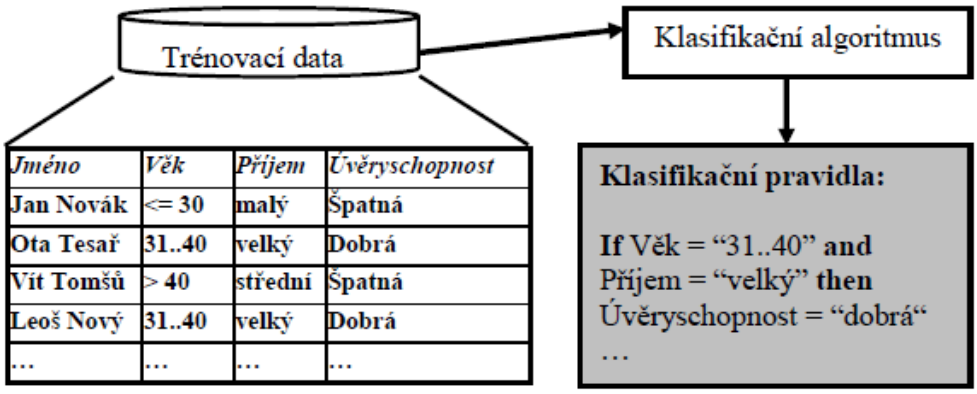
Aby bolo možné vykonávať klasifikáciu a predikciu, potrebujeme najprv natrénovať model z existujúcich dát. Práve na základe tohto modelu budeme ďalej zaraďovať objekty do istých tried.

Pre proces klasifikácie a predikcie potrebujeme rozdeliť vstupné dáta do dvoch množín:

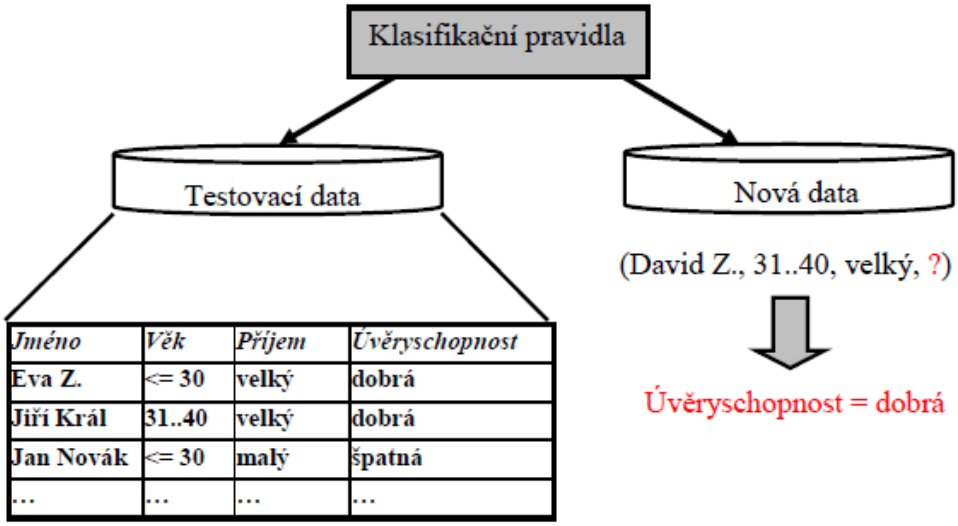
- trénovacia množina
- testovacia množina

Tieto množiny by mali byť disjunktné. V prvej fáze sa natrénuje model z trénovacej množiny, ako je ilustrované na obrázku 4.1. Následne sa vezmú dáta z testovacej množiny

a skúsia sa klasifikovať na základe vytvoreného modelu (obrázok 4.2). Kedže poznáme skutočnú triedu vzoriek z testovacej množiny, porovnáme ju s tou, ktorú určil použitý klasifikačný algoritmus a určíme jeho mieru spoľahlivosti. Tento proces môžeme eventuélne použiť na viac typov algoritmov (prípadne na ten istý algoritmus ale s inými vstupnými parametrami) a na reálne použitie vybrať najlepší z nich.



Obr. 4.1: Znázornenie trénovacej fázy. Prebrané z [45].



Obr. 4.2: Znázornenie testovacej fázy. Prebrané z [45].

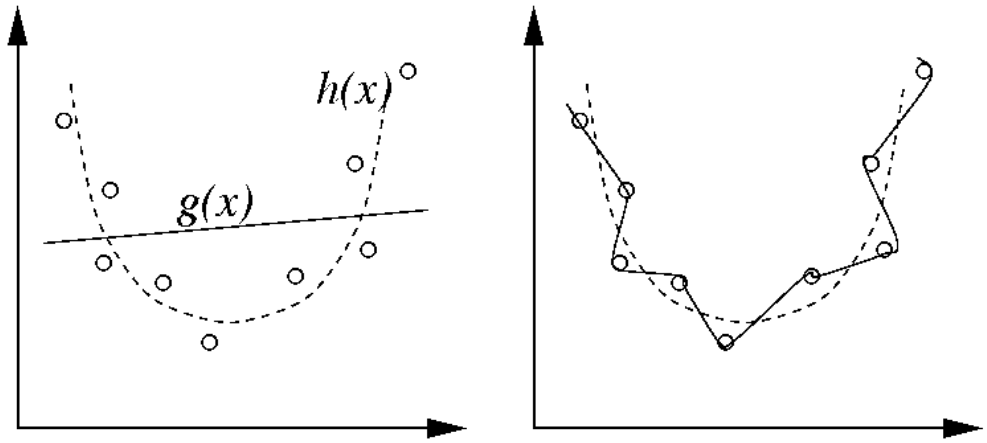
4.1 Trénovanie modelu

Cieľom trénovania je vytvoriť taký model, ktorého miera spoľahlivosti bude čo najvyššia. Podľa toho, či poznáme príslušnosť jednotlivých trénovacích vzoriek do tried, rozlišujeme 2 typy trénovania (učenia) modelu:

- s učiteľom (supervised learning),
- bez učiteľa (unsupervised learning).

Kým pri učení s učiteľom tvoria trénovacie dáta typicky dvojice objekt - trieda, pri učení bez učiteľa znalosť príslušnosti objektu do triedy nepoznáme a teda algoritmus musí byť schopný určiť triedy sám. Ďalším významným problémom je určenie správneho počtu tried. Typicky sa učenie bez učiteľa týka problému zhlukovania (clustering).

Pri trénovaní modelu sa okrem neznámych tried a spomínaného správneho určenia vektoru príznakov stretávame s mnohými ďalšími problémami. Jedným z nich je určenie správneho počtu trénovacích vzoriek. Tých nemôže byť ani málo a ani veľa. Pri nízkom počte sa nevytvorí kvalitný model, a ak ich je veľa, tak nielenže trénovanie zaberie zbytočne veľa drahocenného času, ale takisto hrozí problém pretrénovania. Pri pretrénovaní sa vytvorí veľmi zložitý model, ktorý však funguje kvalitne len pri vzorkách z testovacej množiny (obrázky 4.3 a 4.4) a na reálnych dátach je jeho úspešnosť podstatne nižšia. Závislosť počtu chýb na dĺžke trénovania modelu vyjadruje graf na obrázku 4.5. Vo všeobecnosti je vhodné použiť rádovo tisícky vzoriek na trénovanie, a ak ich je viac tak si (náhodne) určit podmnožinu, na ktorej sa bude trénovať.

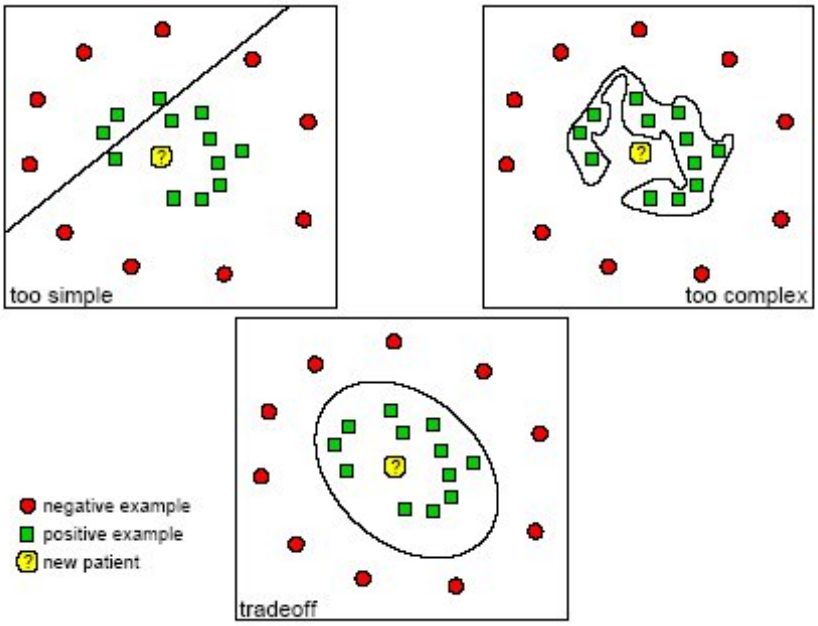


Obr. 4.3: Pretrénovanie. Na obrázku vľavo vidíme príklad slabo natrénovanej funkcie *g(x)* a optimálne natrénovanej funkcie *h(x)*. Na pravom obrázku je vidieť pretrénovaný model, ktorý perfektne sadne na trénovacie dáta, avšak reálne môže podávať horšie výsledky. Prebrané z [39].

V praxi je ale častejší prípad, kedy máme nedostatok vzoriek na fázu vytvorenia modelu. V takom prípade je vhodné použiť metódu *X-validácia (cross validation)*. Postup je nasledovný:

1. Vzorky sa rozdelia do *N* približne rovnako veľkých skupín.
2. Na *N-1* skupinách vzoriek sa natrénuje model a zvyšná je použitá na otestovanie.
3. Bod 2 sa opakuje *N-krát*, aby každá zo skupín bola práve raz použitá ako testovacia sada.

Existuje niekoľko modifikácií horeuvedeného algoritmu. Napríklad je možné pred trénovaním vždy nanovo určiť sady vzoriek. To má výhodu v tom, že je možné vykonať viac iterácií pretrénovania ako *N*, avšak nie je zaručené, že každá zo vzoriek sa dostane do testovacej sady. Viac o X-validácií sa je možné dočítať v odborných publikáciách na tému klasifikácia, napríklad [9].



Obr. 4.4: Pretrénovanie v SVM. Prebrané z [32].

Model trénujeme dovtedy, pokým nedosiahneme požadovanú úspešnosť. Je vhodné sledovať priebežnú úspešnosť a v prípade potreby upraviť vstupné parametre klasifikačného algoritmu. Inou alternatívou je paralelné trénovanie (rôzne algoritmy, rôzne vstupné parametre) na viacerých strojoch, pričom po čase necháme bežať len tie najúspešnejšie a zvyšné zastavíme.

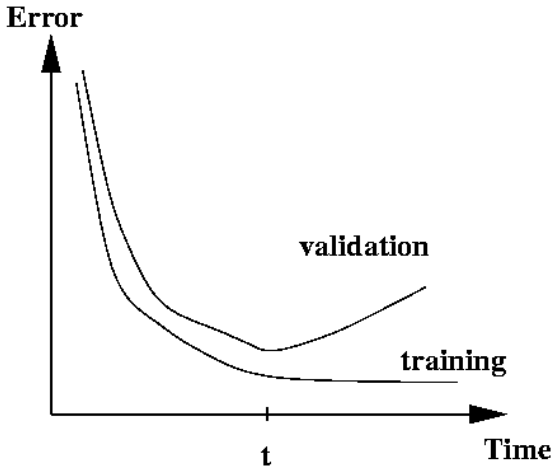
Binárna klasifikácia

Binárna klasifikácia je špeciálnym druhom klasifikácie, kedy existujú iba 2 klasifikačné triedy. V [31] je uvedené, že typicky sa používa keď chceme zistiť, či má objekt požadovanú vlasnosť alebo nie. Bude klient schopný splácať pôžičku? Je objekt na obrázku auto? Má človek s touto DNA vrodenú náchylnosť na rakovinu?

4.2 Spôsoby klasifikácie

V tejto podkapitole sa zemeriame na stručné predstavenie najrozšírenejších klasifikačných algoritmov. [19] navrhuje nasledovné kritériá pre porovnanie klasifikačných metód:

- Presnosť predpovedí - schopnosť dobre triediť neznáme dáta.
- Rýchlosť - výpočetná zložitosť pre vygenerovanie a používanie klasifikačných pravidiel.
- Robustnosť - schopnosť vytvoriť správny model, pokiaľ dané dáta obsahujú šum a chýbajúce hodnoty.
- Stabilita - schopnosť vytvoriť správny model pre veľké množstvo dát.
- Interpretovatelnosť - ako zložitý je model na pochopenie.



Obr. 4.5: Vplyv pretrénovania na počet chýb. Prebrané z [39].

4.2.1 K-nearest neighbour

Tento jednoduchý klasifikátor sa zvykne používať pri učení bez učiteľa. [45] udáva, že pre tento klasifikátor je každá vzorka reprezentovaná n-ticou atribútov číselnej hodnoty. Jednotlivé vzorky ú teda obsiahnuté v n-rozmernom priestore. Medzi dvoma vzorkami $X = (x_1, x_2, ..., x_n)$ a $Y = (y_1, y_2, ..., y_n)$ je definovaná tzv. Euklidovská vzdialenosť, ktorú vypočítame podľa vzorca:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.2}$$

Z trénovacích dát je vybraných práve k vzorov, ktorých vzdialenosť je najmenšia k práve klasifikovanému prvku. Klasifikovaný prvok je potom zaradený do tej triedy, ktorá je najpočetnejšia u týchto k vybraných prvkov. Implementácia je teda opäť relatívne jednoduchá.

4.2.2 Bayesovská klasifikácia

Metódy bayesovskej klasifikácie vychádzajú z Bayesovej vety o podmienených pravdepodobnostiach. Hoci sa teda jedná o pravdepodobnostné metódy, sú intenzívne študované v súvislosti so strojovým učením a uplatňujú sa taktiež v systémoch pre získavanie znalostí.

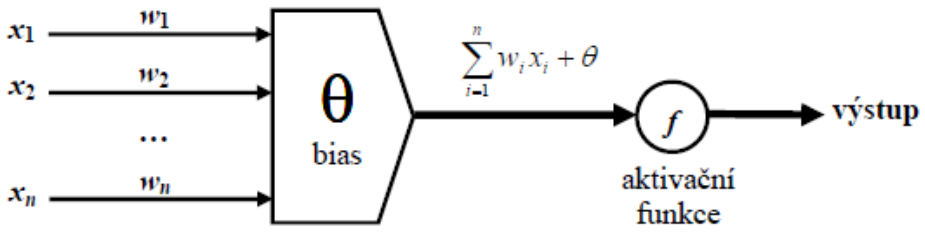
Autor v [2] uvádza Bayesov vzťah pre výpočet podmienenej pravdepodobnosti, že platí hypotéza H, ak pozorujeme evidenciu E, a má tvar:

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)} \tag{4.3}$$

Medzi výhody tejto klasifikačnej metódy patrí jednoznačne jej jednoduchá implementácia. V mnohých prípadoch dáva navyše dobré výsledky. Na druhej strane musí platiť predpoklad, že atribúty vo vektore príznakov musia byť na sebe nezávislé, čo sa v praxi nezvykne vyskytovať často. Možným riešením je použitie bayesovských sietí. Viac informácií o bayesovských sieťach je možné nájsť v publikáciách venovaných tejto tematike, napríklad [13], [21], či [14].

4.2.3 Neurónové siete

Neurónové siete (Neural networks) podľa [24] ako prvý zaviedli neurofyziológ Watten McCulloch a matematik Walter Pitts, ktorí napísali článok o neurónových mechanizmoch a modeloch a popísali jednoduchú neurónovú sieť založenú na elektrických obvodoch. Časovo môžeme zaradiť vznik mechanizmu neurónových sietí do 40-tych rokov minulého storočia.



Obr. 4.6: Schéma umelého neurónu. Prebrané z [45].

Principiálne sa neurónové siete snažia napodobniť fungovanie neurónov v nervovom systéme živočíchov. Opis fungovania neurónov je možné nájsť v [18]. V [45] sa píše, že na základe činnosti biologického neurónu bol zevedený umelý neurón, ktorého schéma je vyobrazená na obrázku 4.6. Na vstup umelého neurónu sú zavedené hodnoty $x_1, x_2, ...x_n$, ktoré sú získané ako výstup iných neurónov alebo zo vstupných dát. Hodnoty $w_1, w_2, ...w_n$ repezentujú synapsie v biologickom neuróne, ktoré sa v umelom neuróne nazývajú váhy. Z týchto hodnôt sa vypočíta suma podľa vzorca 4.4:

$$\sum_{i=1}^{n} w_i x_i + \theta, \tag{4.4}$$

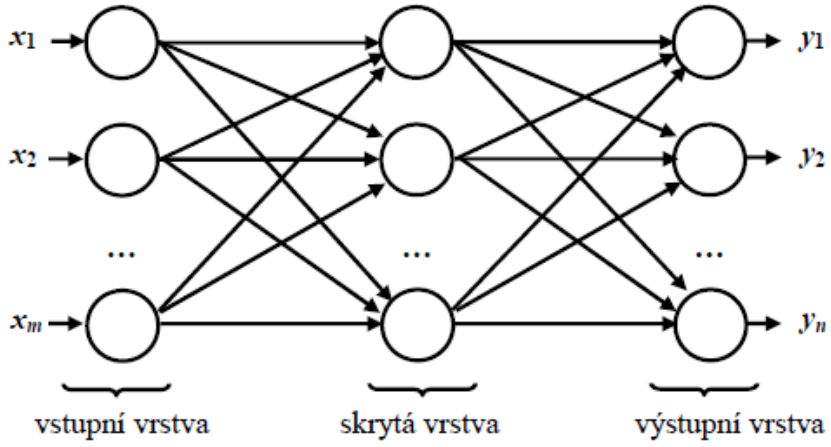
kde θ je vnútorná konštanta daného neurónu, ktorá je k tejto sume vždy pripočítaná. Nazýva sa *bias*. Tento výsledok ešte transformuje istá aktivačná funkcia a výstup môže byť buď vstupom ďalších neurónov, alebo môže tvoriť výstupnú hodnotu, na základe ktorej bude predpovedaná klasifikačná trieda danej vzorky. Z takýchto neurónov je potom zostavená celá neurónová sieť. Učenie siete potom prebieha tak, že sa hľadá správne nastavenie váh $w_1, w_2, ...w_n$ a biasu θ jednotlivo pre každý neurón zvlášť.

Schematicky pozostáva celá neurónová sieť z niekoľkých vrstiev. Prvá vrstva sa nazýva vstupná, za ňou nasleduje niekoľko skrytých vrstiev (ich počet závisí od riešeného problému), a poslednou vrstvou je výstupná vrstva (obrázok 4.7).

Výhodou neurónových sietí je rýchly proces klasifikácie, avšak za cenu pomalého učenia. Vo fáze učenia sú váhy nastavené väčšinou na náhodné hodnoty (blízke 0), a teda výsledná neurónová sieť môže aj pri rovnakých trénovacích vzorkách nadobúdať rôzne úrovne kvality. Preto je vhodné (ak sú na to voľné prostriedky) trénovať paralelne niekoľko neurónových sietí naraz a po čase z nich vyfiltrovať tie, ktoré nepodávajú najlepšie výsledky.

4.2.4 Support Vector Machines

Support Vector Machines (SVM) je klasifikačná technika, ktorá prekvapivo vznikla ešte pred masívnym rozvojom počítačovej techniky už v roku 1909. V tom roku začal podľa [24] anglický matematik James Mercer pracovať na teórií jadra (kernel), ktorá je základným stavebným kameňom tejto techniky. Túto teóriu potom prepracoval v 60-tych rokoch 20.



Obr. 4.7: Vrstvy neurónovej siete. Prebrané z [45].

storočia do algoritmu Vladimir Vapnik zo Sovietskeho Zväzu. SVM nie sú obmedzené len na určitú úzku rodinu problémov, ale je možné ich jednoducho aplikovať na rôzne problémové oblasti. Práve vďaka tomu sa tešia veľkej obľube.

SVM je technikou učenia sa s učiteľom. Typicky sa využíva pre binárnu klasifikáciu. V trénovacích dátach sa snaží nájsť vzor, podľa ktorého by bolo možné zaraďovať nové vzorky do správnej triedy. Každá vzorka má svoj n-rozmerný vektor príznakov. Tento vektor príznakov je možné graficky reprezentovať ako bod v n-rozmernom priestore. Snahou SVM je potom týmto n-rozmerným priestorom preložiť hyperplochu (hyperplane) tak, aby jednoznačne oddelila vzorky patriace do jednotlivých tried. Zároveň platí, že hyperplocha má byť položená tak, aby vzdialenosť najbližších vzoriek oboch tried k nej bola maximálna (obrázok 4.8).

Aby bolo možné umiestnenie oddeľujúcej hyperplochy, dáta musia byť lineárne separovateľné. Vo väčšine prípadov však nie sú. Preto sa algoritmus SVM snaží transformovať n-rozmerný priestor do m-rozmerného ($m > n$), pretože je pravdepodobné, že v ňom už preloženie hyperplochy bude možné (obrázok 4.9). Transformáciu priestoru má na starosti jadrová funkcia. Podľa [34] sa v súčasnosti môžeme stretnúť s nasledovnými typmi jadrových funkcií:

- Polynomiálna (homogénna)

$$k(x_i, x_j) = (x_i * x_j)^d \tag{4.5}$$

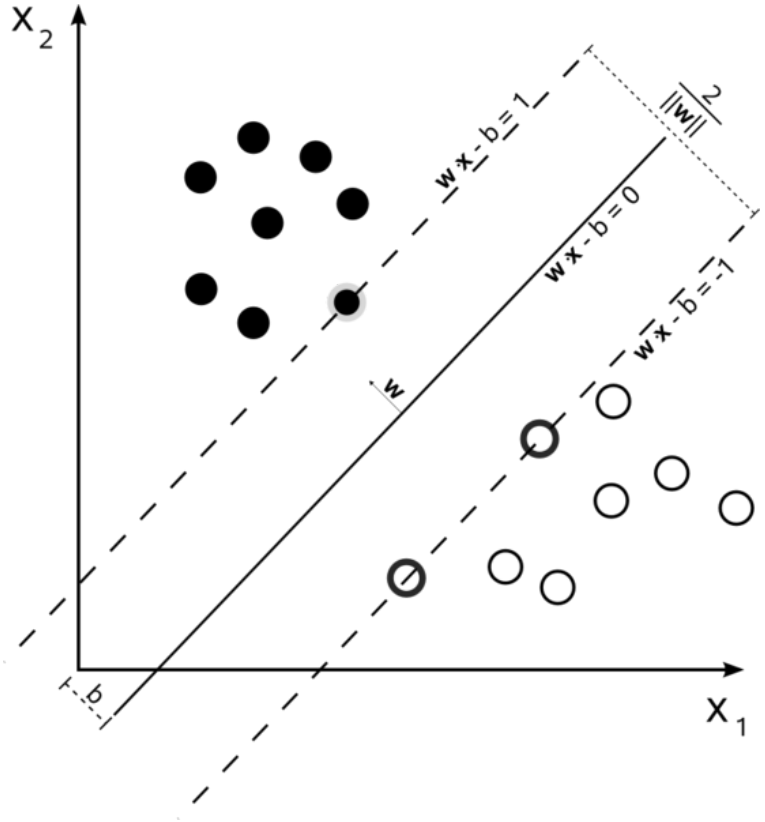
- Polynomiálna (nehomogénna)

$$k(x_i, x_j) = (x_i * x_j + 1)^d \tag{4.6}$$

- Radiálna bázová funkcia (RBF)

$$k(x_i, x_j) = exp(-\gamma||x_i - x_j||^2), \tag{4.7}$$

  pre $\gamma > 0$. Niekedy $\gamma = \dfrac{1}{2\sigma^2}$



Obr. 4.8: Umiestnenie hyperplochy tak, aby vzdialenosť od najbližších vzoriek bola maximálna. Tieto vzorky sa nazývajú *podporné vektory* (support vectors). Prebrané z [34].

- Sigmoid

$$k(x_i, x_j) = tanh(\kappa x_i * x_j + c), \tag{4.8}$$

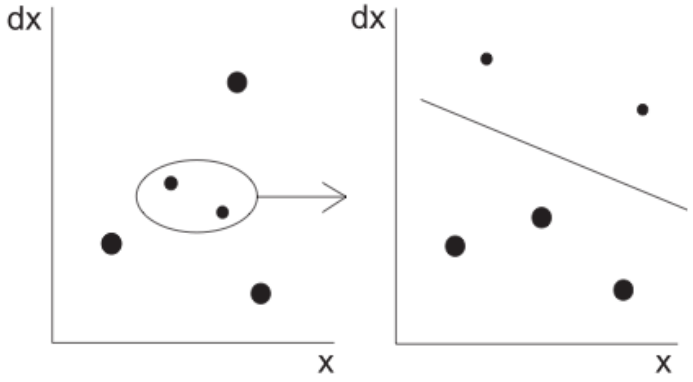
  pre niektoré (nie všetky) $\kappa > 0$ a $c < 0$

Medzi hlavné výhody SVM patrí ich rýchla rozhodovacia schopnosť pre neznáme vzorky, pretože sa vektor príznakov iba prevedie do nového m-rozmerného priestoru a určí sa, na ktorej časti hyperplochy sa nachádza. Avšak natrénovanie kvalitného modelu môže trvať relatívne dlho. Preto je jej nasadenie vhodné opäť najmä v tých prípadoch, keď máme dostatok času na natrénovanie, ale klasifikovať chceme čo najrýchlejšie (napríklad ak máme na vstupe prúd dát z priemyselnej kamery, ktoré musia byť spracované v reálnom čase).

4.3 Zhrnutie

Táto kapitola mala za úlohu predstaviť klasifikáciu ako vednú disciplínu. Boli vysvetlené základné pojmy a stručne opísané niektoré klasifikačné metódy. Okrem nich existuje mnoho ďalších, napríklad klasifikátor založený na genetických algoritmoch, klasifikátor založený na fuzzy množinách, rozhodovacie stromy, diskriminačná analýza, skryté Markovské modely, a iné.

Nie je možné povedať, ktorá z nich by bola všeobecne najlepšia. Výber správnej metódy závisí vždy od aktuálnej situácie a veľkú rolu zohráva aj správny výber parametrov.



Obr. 4.9: Transformácia priestoru tak, aby boli vzorky lineárne separovateľné. Prebrané z [24].

Laik bez znalosti fungovania týchto metód nie je schopný vybrať vhodnú a preto sa táto úloha ponecháva na odborníkov. Tí už na základe znalostí a vlastných skúseností vedia určiť správnu metódu a nastaviť jej vhodné parametre, i keď to niekedy môže byť dosť obtiažne. Ak je napríklad potrebné nájsť zhluky dát, experimentuje sa primárne s metódou hľadania k-najbližších susedov (alebo s nejakou podobnou metódou), ale ak je potrebné rýchlo klasifikovať veľké množstvo vzoriek pravidelne prichádzajúcich v prúde, tak je lepšie siahnuť po neurónových sieťach alebo SVM. Neoceniteľnou výhodou pri tvorbe kvalitného klasifikátoru sú skúsenosti v tomto obore. V praxi sa dokonca niekedy používa niekoľko nezávislých klasifikátorov a je vybraná trieda, na ktorej sa zhodne väčšina z nich.

Kapitola 5

Identifikácia trajektórie

Trajektória je množina bodov alebo krivka v priestore, ktorú opisuje teleso alebo hmotný bod pri svojom pohybe. Môže nadobúdať rôzny, aj nepravidelný tvar. Dĺžka trajektórie sa nazýva dráha.

Pri sledovaní pohybu objektu sa snažíme predpovedať jeho budúcu polohu v nasledujúcom snímku. Predikcia polohy sa robí tzv. Kalmanovým filtrom na základe chovania objektu na posledných snímkoch. Ďalším spôsobom, ako priradiť objekt už existujúcej trajektórií, je napríklad využitie Mahalanobisovej vzdialenosti (viac v [20]).

Objekt identifikovaný na snímke môže byť novým objektom, alebo môže ísť o pokračovanie trajektórie už existujúceho objektu. Výsledná trajektória je teda definovaná ako zoznam objektov z jednotlivých snímkov. Po vyhodnotení je každý objekt priradený jedinej trajektórií. Trajektórií v jednom snímku náleží vždy najviac jeden objekt. Za týchto predpokladov sú jednotlivé objekty identifikované v snímkach priraďované do disjunktných množín.

5.1 Priraďovanie objektov k trajektóriám

Po rozpoznaní objektu v scéne je potrebé rozhodnúť, či ide o nový objekt (jeho prvý výskyt na kamere), či ide o objekt, ktorý bol aj na predchádzajúcom snímku a len sa o niečo pohol, alebo ide o objekt, ktorý je síce na kamere nový, ale bol videný v inej kamere a ide teda o tú istú osobu. Najprv sa hľadá podobnosť s objektami, ktoré boli na scéne zaznamenané v predchádzajúcich snímkoch. To sa robí na základe extrakcie príznakov objektu. [6] píše, že v letiskovej hale (aspoň na Britských ostrovoch) nosí väčšina ľudí tmavé kabáty, a teda nie je vhodné zostaviť príznaky iba z farby objektu. Práve preto sa zisťuje aj tvar osoby, a detekcia totožných osôb sa vo vhodných prípadoch môže opierať aj o detekciu tváre. Zároveň sa pomocou Kalmanovho filtra určí najpravdepodobnejšia pozícia rozpoznaných osôb na novom snímku, čo tiež významne pomáha pri priraďovaní nového objektu k už existujúcej trajektórií. Viac informácií je uvedených v článku [6].

Ak sa rozpoznaný objekt nepodobá na žiadny známy objekt na kamere, ide o novú osobu, ktorá práve vstúpila do scény, alebo sa môže jednať o prvý snímok videa (kedy ešte nemáme k dispozícií informácie o existujúcich objektoch). Každopádne túto osobu prehlásime za novú, a až SVM klasifikátor rozhodne, či táto osoba bola videná aj na inej kamere, alebo nie. Táto problematika bude bližšie rozoberaná v kapitole 6.

5.2 Problémy súvisiace s identifikáciou trajektórie

Pri identifikácií trajektórie pohybujúceho sa objektu v obraze z priemyselnej kamery sa stretávame s viacerými problémami:

- Objavovanie a miznutie objektu, čo je spôsobené jeho príchodom a odchodom z obrazu.
- Prekrývanie objektov medzi sebou. Dochádza k nemu vtedy, keď do kamery vojdú dva rôzne objekty, pričom každému je priradená vlastná trajektória, a po čase sa v nejakom okamihu prekryjú. Pri ich prekrytí môže byť rozpoznaný najviac jeden objekt.
- Prekrývanie objektov s neživými súčasťami scény. Myslené sú napríklad rôzne lavičky, stĺpy, čistiace mechanizmi, a iné. To síce nespôsobí problém s identifikáciou trajektórie, ale takto (hoci aj čiastočne) prekrytá osoba nie je rozpoznaná v obraze a teda trajektórií nemôže byť priradený žiadny z objektov.
- Rozpad objektu na viacero iných objektov. K tomu to dochádza, keď napríklad do kamery vojdú dva prekrývajúce sa objekty, pričom sú vyhodnotené ako jeden samostatný objekt. Pri ich následnom rozdelení (niekde v strede obrazu) musí byť aj trajektória rozdelená na dve.
- Zmena tvaru objektu (napríklad otáčanie osoby, zohnutie sa).
- Podobnosť objektov (väčšina ľudí nosí v zime tmavé kabáty). Toto je problémom hlavne pri vzájomnej interakcií objektov a pri priraďovaní trajektórií objektom z predchádzajúcich snímkov.
- Náhla zmena smeru pohybu objektu.
- Identifikácia toho istého objektu vo viacerých rôznych kamerách.
- Identifikácia trajektórie objektu v pohybujúcej sa kamere.
- Projekcia skutočného 3D priestoru do 2D obrazu. Nielenže sa stráca informácia o hĺbke obrazu, ale obraz je taktiež deformovaný.
- Nesprávne rozpoznanie rôznych objektov ako osôb.

Niektoré z týchto problémov sú ilustrované na konkrétnych videách na obrázkoch 5.1 a 5.2. Ide o výstup programu SUNAR-HMI, s ktorým sa podrobnejšie zoznámime neskôr.

Niektoré z horeuvedených problémov aspoň čiastočne rieši Kalmanov filter. Pomocou Kalmanovho fitra je možné predpovedať polohu objektu na aktuálnom snímku na základe jeho smeru a rýchlosti pohybu na predchádzajúcich snímkoch.

Identifikácia trajektórie funguje najlepšie vtedy, keď je v obraze malý počet pohybujúcich sa objektov, ktoré sú navyše od seba izolované. V opačnom prípade je potrebné vyextrahovať z objektov kvalitný vektor príznakov.

Obr. 5.1: Výstup z programu SUNAR-HMI. Sú na ňom badatelné nasledovné problémy: prekryv veľkého počtu osôb v hale, neidentifikovaná osoba v čiernom obleku v pravej časti, dve osoby identifikované ako jedna na pravom okraji, sediaca osoba identifikovaná ako dve rôzne (anotácie 279 a 276) a množstvo sediacich osôb, ktoré neboli vôbec identifikované.

5.3 Identifikácia tej istej osoby v rôznych kamerách

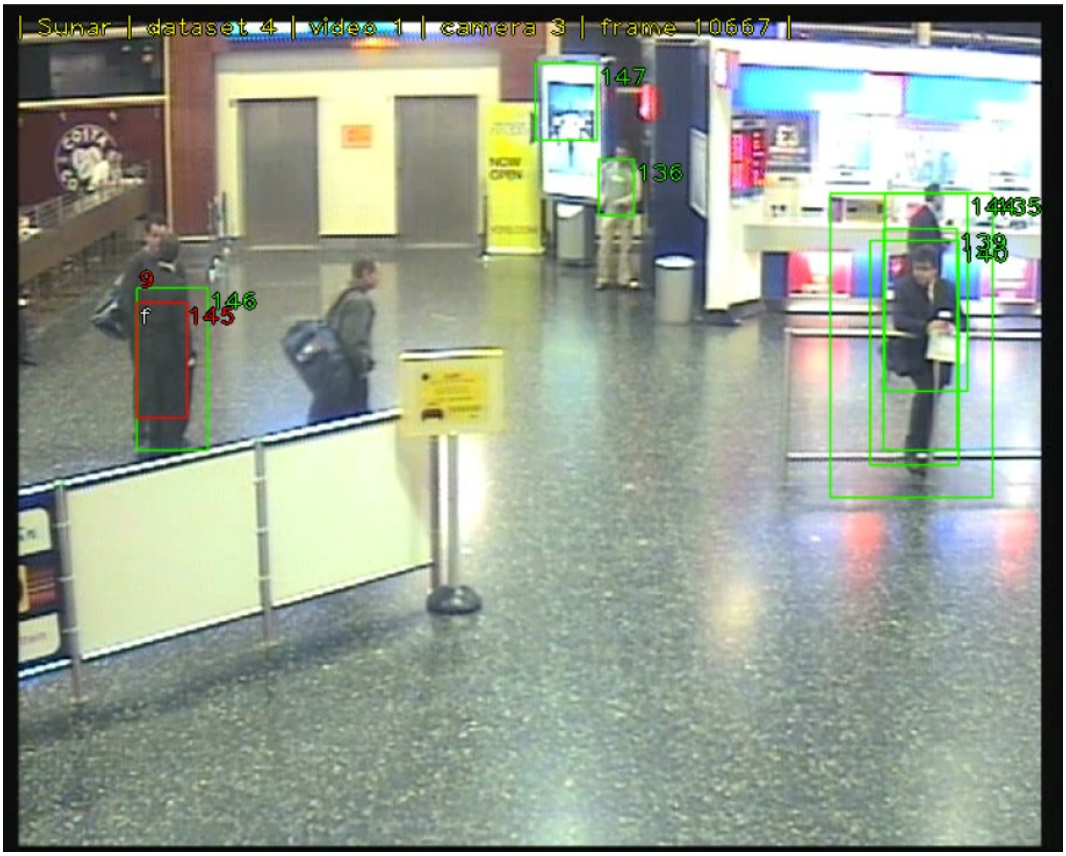
V tejto podkapitole sa zameriame na problém určovania tej istej osoby v rôznych kamerách, pretože je to hlavným cieľom tejto práce. Zároveň treba podotknúť, že ak osoba opustí priestor monitorovaný kamerou, môže sa do neho po čase zase vrátiť. Treba preto počítať aj s touto variantou a takto vrátenú osobu správne identifikovať.

Pre túto identifikáciu nám pomôže aj mapa oblasti sledovanej viacerými kamerami s polohou dohľadových kamier. Na základe týchto znalostí vieme určiť, kade sa asi budú osoby pohybovať a ktoré prechody medzi kamerami sú fyzicky nemožné. Takisto nám to pomôže určiť vstupné a výstupné oblasti priestoru sledovaného jednou konkrétnou kamerou.

Kvalitne fungujúca aplikácia má tú výhodu, že mapu sledovanej oblasti teoreticky nepotrebuje a všetky tieto znalosti si dokáže vyvodiť sama na základe vhodnej množiny trénovacích dát. Každopádne je ale potrebné túto trénovaciu množinu nejako vytvoriť.

Identifikácia totožných osôb sa robí na základe ich vektoru príznakov v kamere. Takto sa nájdu dve osoby, ktoré vyzerajú podobne a zároveň ich výskyt v rôznych kamerách korešponduje s realitou. Vtedy takéto osoby môžeme prehlásiť za totožné.

Celý problém sťažuje fakt, že nie každý bod sledovaného priestoru je monitorovaný práve jednou kamerou. Väčšinou existujú oblasti, ktoré monitorované vôbec nie sú, alebo



Obr. 5.2: Výstup z programu SUNAR-HMI. Sú na ňom badatelné nasledovné problémy: prekryv osôb 145 a 146 (vľavo), neidentifikovaná osoba, ktorá je zčasti prekrytá zábradlím, automat je rozpoznaný ako osoba (147), a identifikácia 3 osôb namiesto 2 v pravej časti.

naopak oblasti, ktoré vidí viacero kamier naraz. Z toho vyplýva, že aj sledovaná osoba sa môže nachádzať na viacerých kamerách naraz, alebo na žiadnej z nich. Priemerná dĺžka času prechodu medzi dvoma konkrétnymi kamerami sa dá zmerať a teda dá sa približne predpokladať, kedy sa osoba objaví v druhej kamere, avšak musíme počítať s tým, že osoba sa môže po ceste zastaviť a tento čas sa aj niekoľkonásobne predĺži. V našom konkrétnom prípade s monitorovanou letiskovou halou trvá prechod medzi kamerami 3 a 5 štandardne okolo 20 sekúnd, avšak môžeme spozorovať veľa osôb, ktoré opustili priestor jednej kamery idúc k druhej z nich ale na nej sa už neobjavili. V nemonitorovanom priestore sa napríklad mohli posadiť na lavičku, ísť si niečo kúpiť do stánku, navštíviť toaletu, alebo mohli úplne zmeniť smer svojho pohybu. Navyše v letiskovej hale je vidieť veľa ľudí, ktorí len niekoho hľadajú a teda sa dá očakávať, že sa pohybujú pomalšie oproti regulárnym cestujúcim alebo že sa často zastavia. Pri prechode kamerou sa však dá určiť správanie osoby, a ak sa pohybuje pomalšie ako ostatní prítomní tak je aj pravdepodobné, že prechod medzi kamerami je bude trvať dlhšie. Opačným prípadom sú cestujúci, ktorí sa ponáhľajú na svoj odlet a musia v hale utekať. Samozrejme aj takýchto pasažierov je možné jednoducho identifikovať a pripraviť sa na ich objavenie v druhej kamere v správnom časovom okamihu.

Pre sledovaný priestor je výhodné, ak každá kamera zaberá len určitý konkrétny malý priestor. V opačnom prípade sú osoby pohybujúce sa ďaleko od kamery veľmi malé, a ich identifikácia nemôže byť kvalitná. Je určite zrejmé, že osoby, ktoré sú blízko kamery, môžu

byť jednoduchšie identifikované ako tie niekde ďaleko v pozadí, s ktorými by mal problém aj človek. V našich videách sú vhodne umiestnené kamery 1, 3, a 4. Naopak, kamery 2 a 5 zaznamenávajú veľký priestor, v ktorom je identifikácia zložitá.

Významný problém je prekrývanie ľudí na kamere. To spôsobuje, že sa osoba môže na kamere stratiť, že sú dve osoby rozpoznané ako jedna pričom keď sa po čase od seba vzdialia tak je problém určiť, ktorá bola identifikovaná ako pôvodná, ale celkovo aj aplikácia má problém s identifikáciou osôb, pretože pri projekcií trojrozmerného priestoru do dvojrozmerného priestoru, aký je v obraze môže mať takýto zhluk prekrývajúcich sa osôb tvar, ktorý nie je pre osoby typický, a teda osoby nie sú vôbec rozpoznané.

Kritický problém je počet osôb na kamere, ale aj v celkovo v hale. Pri nízkom počte ľudí je ich identifikácia pomerne jednoduchá (hoci by aj boli zachytené na obraze ďaleko od kamery), avšak pri vyššom počte sa začnú spolu nielen prekrývať, ale je vysoko pravdepodobné, že niekoľko osôb bude mať podobný vektor príznakov a preto v mnohých prípadoch nebude identifikácia prebiehať správne.

V našom prípade je významným uľahčením to, že kamery sú statické a teda nemusíme riešiť problém ich pohybu.

Ako bolo spomenuté vyššie, problém nastáva aj pri prekryve osôb s neživými súčasťami scény. V našom prípade činí veľký problém rozpoznanie osôb sediacich na lavičkách v popredí kamery 2. Tieto totiž svojim tvarom nepripomínajú osoby (aspoň nie je počítač) a teda nie sú často vôbec rozpoznané. Opačným prípadom je, ak je takáto sediaca osoba mylne rozpoznaná ako viacero osôb, alebo je rozpoznaná iba jej malá časť. V takom prípade je zle určený vektor príznakov (správne je snáď iba poloha osoby v konkrétnom čase) a zo zlého vektoru príznakov nie je možné následne rozpoznať osobu na inej kamere. Toto prekrývanie sa s neživými súčasťami scény sa týka nielen osôb na lavičke ako sme si práve opísali, ale aj napríklad pri prekryvoch so zábradlím.

5.4 Kalmanov filter

Ako sme spomenuli vyššie, Kalmanov filter je matematická metóda a v našom prípade slúži na predpovedanie polohy pohybujúceho sa objektu na základe jeho správania (smer a rýchlosť pohybu) v niekoľkých predchádzajúcich snímkoch. Kalmanov filter je často používaná technika, ktorá sa navyše snaží potlačiť šum vo vstupných dátach.

Svoje uplatnenie našla napríklad v radaroch [43], ale aj v matematickom modeli chemického reaktora pri nelineárnom odhade stavu [17], či inde. V oblasti počítačového videnia sa Kalmanov filter často používa práve na identifikáciu trajektórie pohybujúcich sa objektov.

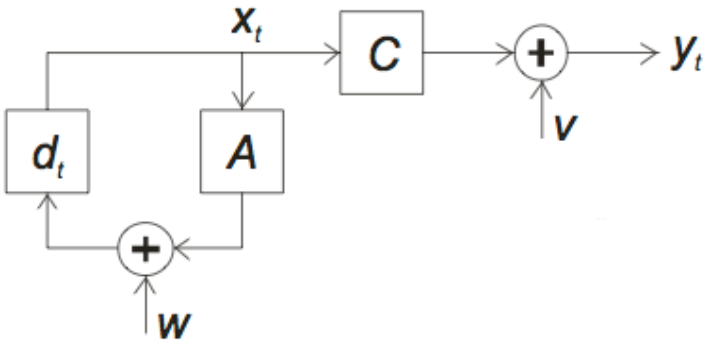
5.4.1 Model systému

[5] uvádza, že Kalmanov filter bol pôvodne navrhnutý na reprezentáciu informácií týkajúcich sa pohybujúcich sa objektov. Pohybujúce sa objekty sú modelované pomocou diskrétneho času dynamického systému. Objekty majú časovo priestorový stav, ktorý môžeme označiť napríklad x_i. Časopriestorový stav je reprezentovaný pomocou údajov o polohe $[x, y, t]$ a rýchlosti $[d_x, d_y]$. $[x, y]$ tu odkazuje na 2D pozíciu v čase t.

V reálnom prostredí nebudú vstupné dáta nikdy ideálne. Dáta sú často zašumené, niektoré stavy môžu chýbať a iné môžu byť zase skreslené. Faktorov, ktoré ovplyňujú kvalitu dát, je mnoho. V prípade, že stav objektu x v čase t obsahuje pozíciu a vektor pohybu $[x, y, d_x, d_y]$, hovoríme o modelovaní trajektórie v dvojrozmernom priestore a to nielen v analyzovanom obraze, ale rovnako aj v reálnom svete.

Ďalej sa v [5] píše, že vstupnú informáciu považujeme zašumenú skytým (Gaussovským) šumom w. Na výstupe systému filtru získavame vektor y, ktorý je jednoduchou lineárnou observáciou (Markovovým procesom prvého rádu) zaťažený šumom v.

Model celého systému v čase t môžeme vidieť na obrázku 5.3. Dynamický model systému je opísaný vzťahom 5.1 a model merania vzťahom 5.2.



Obr. 5.3: Znázornenie dynamického procesu, prebrané z [5].

$$x_t = Ax_{t-1} + Bu_{t-1} + w \qquad w \approx N(0, Q) \tag{5.1}$$

$$y_t = Cx_t + v \qquad v \approx N(0, R) \tag{5.2}$$

Obrázok 5.3 vyjadruje oneskorenie d_t, A je matica prechodu stavu (state transition matrix), B reprezentuje volitelnú riadiacu maticu (control matrix) so vstupom u, C je matica meraných alebo pozorovaných hodnôt normálneho rozloženia a Q a R sú kovariančné matice chýb (šumu) w a v. Založené na Bayesovkej pravdepodobnosti: $P(x_t|y_t) \approx N(\widehat{x}_t, P_t)$ ako bude opísané nižšie.

Efektívny rekurzívny nástroj slúžiaci na odhad stavu procesu, a to spôsobom, ktorý minimalizuje strednú chybu, je Kalmanov filer (napríklad v implementácií [5]).

5.4.2 Spôsob výpočtu

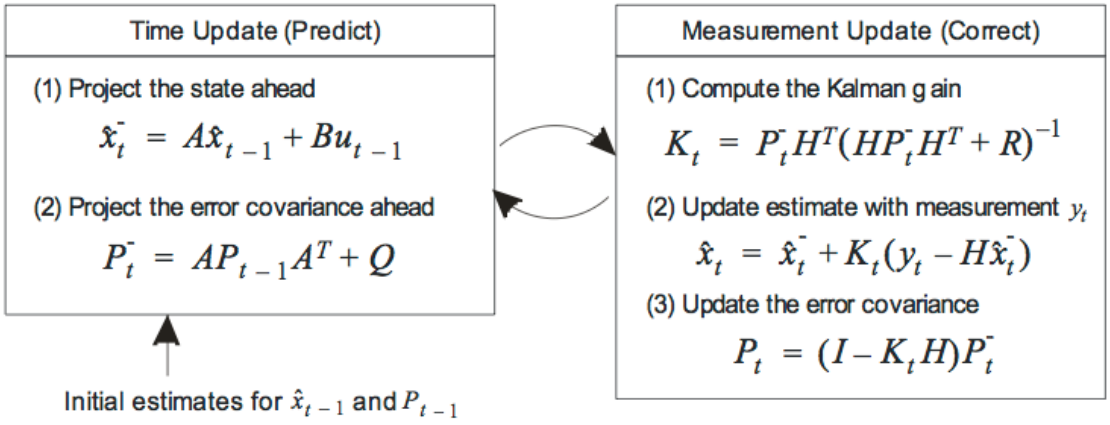
Iteratívny preces Kalmanovho filtru prechádza vždy dvoma stavmi. Sú to stav predikcie a stav korekcie. Prechod stavmi je riadený spätnou väzbou. Keď tento proces prenesieme do roviny pozorovania objektu, môžeme činnosť Kalmanovho filtru tlmočiť v zmysle nasledovného odstavca.

V prípade, že máme k dispozícií nové získané hodnoty ukazujúce polohu objektu v priestore, robíme aktualizáciu filtru. Pri aktualizácií sa vypočítajú nové hodnoty koeficientov s ohľadom na nové meranie. Filter sa tak prispôsobuje novo nameraným hodnotám. V ďalšom kroku potom môžeme previesť predpoveď nového stavu a kovariančnej matice. V tomto okamihu obsahuje stavový vektor predikované hodnoty polohy, rýchlosti, a zrýchlenia. Takýto postup sa opakuje v každom kroku, keď sú k dispozícií údaje z merania. V takýchto prípadoch filter poskytuje vždy nové predikované hodnoty stavového vektoru.

Opačným prípadom je situácia, keď nemáme k dispozícií novo namerané hodnoty, ale v zisťovaní stavového priestoru potrebujeme pokračovať. Kalmanov filter má schopnosť prevádzať extrapoláciu stavového vektoru aj pri nezadaných nových hodnotách. V takom

prípade nemáme nový vstup z ktorého je možné vypočítať zlepšenie y_k, je aktualizácia odhadu stavového vektoru rovná extrapolovanému stavovému vektoru. Postupnou extrapoláciou hodnôt sa znižuje pravdepodobnosť správnej predpovede filtru. Znižovanie kvality extrapolovaných hodnôt sa prejavuje zvyšovaním neurčitosti predikcie, čo odráža zvyšovanie hodnôt kovariančnej matice.

Podrobný popis činnosti Kalmanovho filtru je možné nájsť na [38] alebo tiež [28].



Obr. 5.4: Fungovanie Kalmanovho filtru. Kombinácia modelu a rovníc. Prebrané z [5].

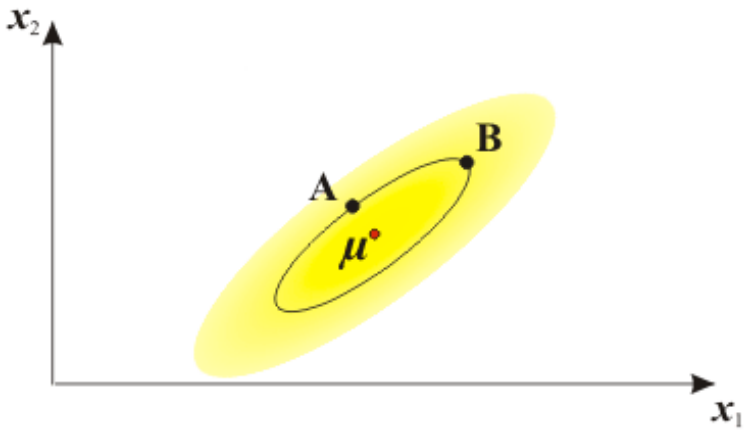
5.5 Iné spôsoby priradenia trajektórie objektu

Ako už bolo spomenuté vyššie, Kalmanov filter nie je jediný spôsob, ako namapovať rozpoznané objekty na trajektórie. Jednou z metrík je aj tzv. Mahalanobisova vzdialenosť. Ide o metriku predstavenú v roku 1936 indským matematikom Prasanta Chandra Mahalanobisom. Je založená na korelácií medzi premennými, v ktorých môžeme identifikovať a analyzovať rôzne vzory. Je to užitočný spôsob nájdenia podobností neznámej vzorky dát k známej. Na rozdiel od Euklidovskej vzdialenosti rešpektuje tvar vzorky dát v priestore (obrázok 5.5). Mahalanobisovu vzdialenosť je možné aplikovať na akýkoľvek n-rozmerný priestor. Práve táto vlastnosť nám dopomôže k správnemu zaradeniu vzorky do požadovanej triedy. Vektor príznakov si totiž môžeme previesť do n-rozmerného priestoru, a v ňom potom hľadať jeho najbližšieho suseda metrikou Mahalanobisovej vzdialenosti.

Mahalanobisova vzdialenosť je široko využívaná napríklad v biológií, kde slúži mimo iného na predikciu štruktúry proteínov (zaradenie do správnej triedy). Viac informácií o tomto konkrétnom probléme sa môžeme dočítať v [7].

5.6 Zhrnutie

Identifikácia trajektórie pohybujúceho sa objektu predstavuje významný problém pri určovaní, či ide stále o ten istý objekt alebo nie. Existujú metódy a postupy, ktoré nám uľahčujú prácu, ale dosiahnuť 100% úspešnosť je takmer nemožné. Tieto metódy sú založené buď na predikcií objektu na novom snímku (Kalmanov filter), alebo majú snahu nájsť vhodný objekt na základe najkratšej vzdialenosti vektoru príznakov (Mahalanobisova vzdialenosť). Všetky metódy však požadujú kvalitný vektor príznakov, na základe ktorého identifikujú



Obr. 5.5: Demonštrácia Mahalanobisovej vzdialenosti. Keďže táto metrika rešpektuje tvar rozloženie vzoriek, body A a B majú rovnakú vzdialenosť od stredu μ. Prebrané z [33].

totožné objekty v rôznych snímkach v čase. Ukázali sme si, že identifikácia trajektórie funguje najlepšie vtedy, keď je na kamere malý počet osôb, ktoré spolu vôbec neinteragujú (neprekrývajú sa a ani sa k sebe nepriblížia).

Kapitola 6

Výsledky práce

V praktickej časti diplomovej práce som nadviazal na projekt SUNAR (Surveillance Network Augmented by Retrieval), vyvýjaný na Fakulte informačných technológií univerzity Vysoké učení technické v Brne. Projekt Sunar sa skladá z troch hlavných modulov, slúžiacich na spracovanie videa a následné monitorovanie:

- SUNAR-VRM
- SUNAR-HMI
- SUNAR-CVM

Moduly opierajúce sa o počítačové videnie sú založené na knižnici OpenCV, ktorú využívajú na identifikáciu a sledovanie objektu na základe vektoru príznakov objektu. Informácie o objektoch a dohľadovej oblasti sú vyčistené, integrované, indexované a uložené v module SUNAR-VRM. Tieto dáta sa uchovávajú v databáze PostgreSQL [1] rozšírenej tak, aby bola vhodná pre uschovanie časopriestorových dát.

Aplikácia spracováva reálny obraz z piatich kamier v letiskovej hale. Ich rozloženie je znázornené na obrázku 6.1. Z toho vyplýva, že najčastejšie budú prechody medzi kamerami 1 - 2, 2 - 3, 2 - 5, 3 - 4, 3 - 5, a samozrejme naopak. Objekt sa po zmiznutí z kamery môže po čase objaviť zase v tej istej kamere (ak sa osoba obráti a začne sa vracať späť). Je vidieť, že osoba sa môže v jednom časovom okamihu nachádzať vo viacerých rôznych kamerách, čo chceme tiež detekovať. Čas od zmiznutia osoby v jednej kamere až po jej objavenie sa v inej, môže zabrať určitý čas. Napríklad prechod medzi kamerami 3 a 5 trvá typicky okolo 20 sekúnd, ale môže aj oveľa viac, ak sa osoba po ceste zastaví.

6.1 SUNAR-VRM

Modul SUNAR-VRM je napísaný v programovacom jazyku Java a má viacero účelov. Po spustení sa pripojí na databázu, z ktorej čerpá údaje. Okrem spracovávania anotácií a experimentov slúži takisto na vygenerovanie trénovacej množiny dát, ktoré zachytávajú prechody osôb medzi kamerami. Toto je robené buď na základe Bayesovského klasifikátoru, alebo metódou SVM. Ďalej sa v práci zameriame práve na SVM klasifikátor, pretože ten bol doplnený v implementačnej časti tejto práce.

Pri spustení SVM klasifikátoru sa najprv načítajú kamery z databázy. Potom sa vytvorí trénovacia množina dát z aktuálnych prechodov uložených v databáze. Tieto prechody boli

[1] PostgreSQL 8.4

identifikované ručne. Aby sme dosiahli zvýšenie počtu trénovacích dát, môžeme pri každom prechode z kamery A do kamery B vytvoriť 2 záznamy, pričom ten druhý bude zachytávať opačný prechod (akoby osoba išla dozadu), ale jej vektor príznakov (rýchlosť, vstupný bod, výstupný bod) bude viac-menej reálny, takže si to môžeme dovoliť. Trénovacia množina dát sa uloží do dočasného súboru. Následne sa vytvorí model volaním knižnice LibSVM, ktorá dostane na vstupe práve textový súbor s prechodmi z databázy. Spoľahlivosť modelu sa určí X-validáciou. Mala by sa postupne zvyšovať so stúpajúcim počtom vzoriek v trénovacej množine. Inou možnosťou je vygenerovanie testovacej množiny, spustiť pre ňu predikciu na základe vygenerovaného modelu a porovnať ju s reálnymi výsledkami.

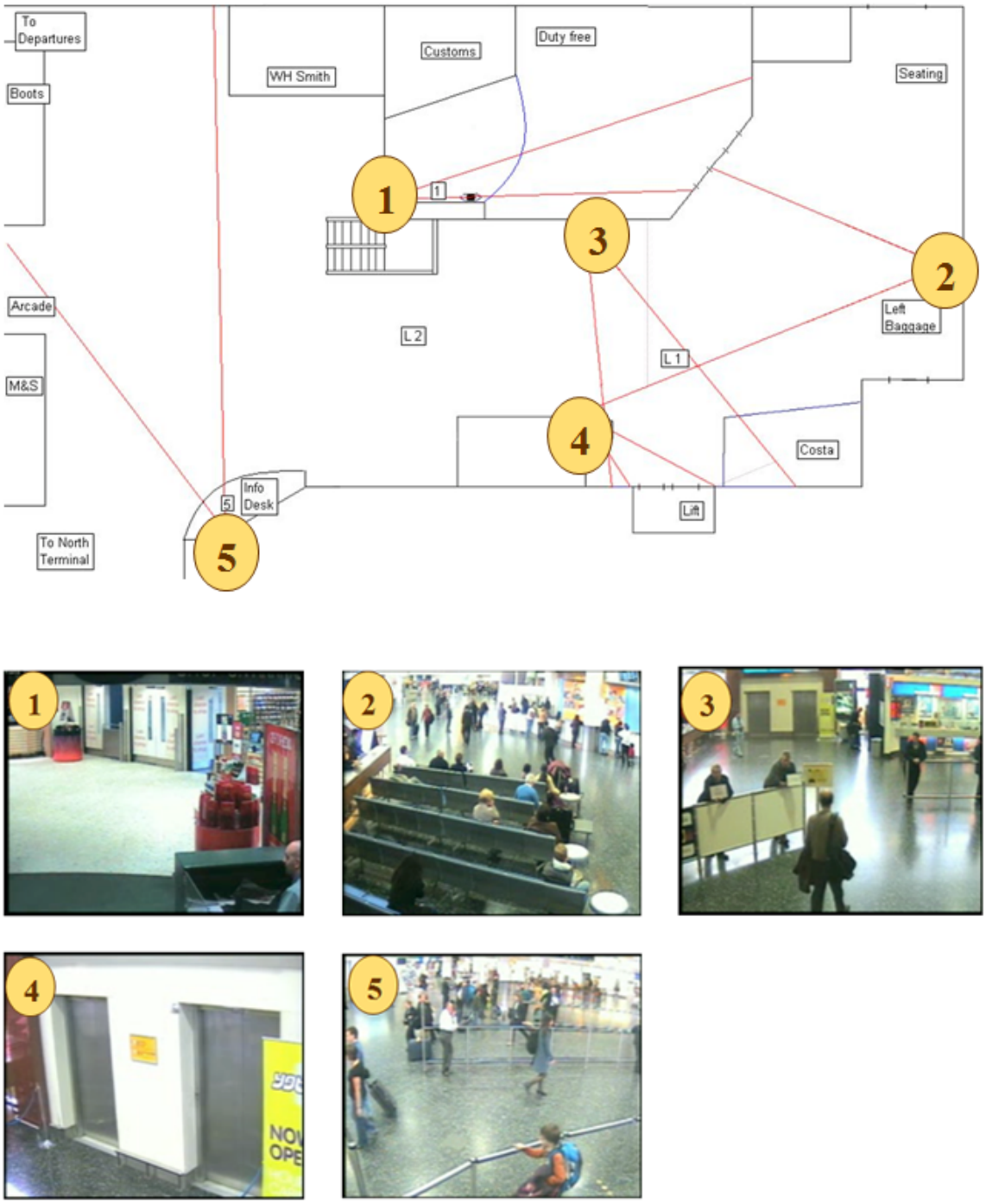
6.2 SUNAR-HMI

Modul SUNAR-HMI je napísaný v programovacom jazyku C++. Umožňuje prehliadanie anotácií, prechodov medzi kamerami, ale najzaujímavejšou súčasťou tohto modulu je funkcia, ktorá spojí obraz zo všetkých piatich kamier do jedného a tak umožní používateľovi vidieť komplexný pohľad na letiskovú halu. Je takisto možné nastaviť rýchlosť prehrávania záznamov.

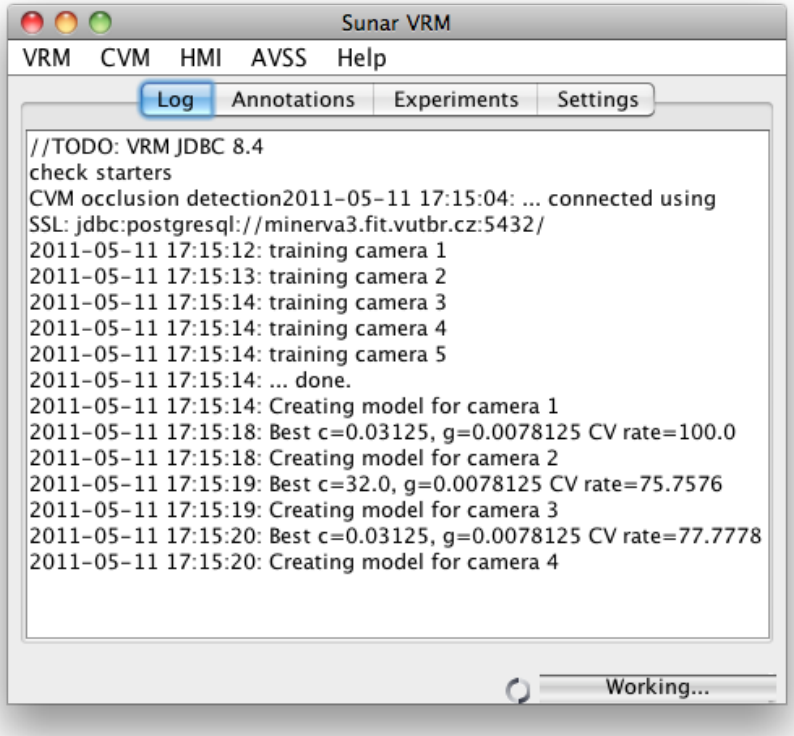
V implementačnej časti tejto diplomovej práce som pridal funkciu, ktorá umožňuje ručne anotovať prechody osôb medzi kamerami a tie sú následne uložené do databázy a použité na ďalšie spracovanie modulom SUNAR-VRM. Súčasťou prínosu v module SUNAR-HMI je aj funkcia napísaná v PL/pgSQL, ktorá má na starosti správne uloženie totožného objektu v databáze.

6.3 Použité externé knižnice

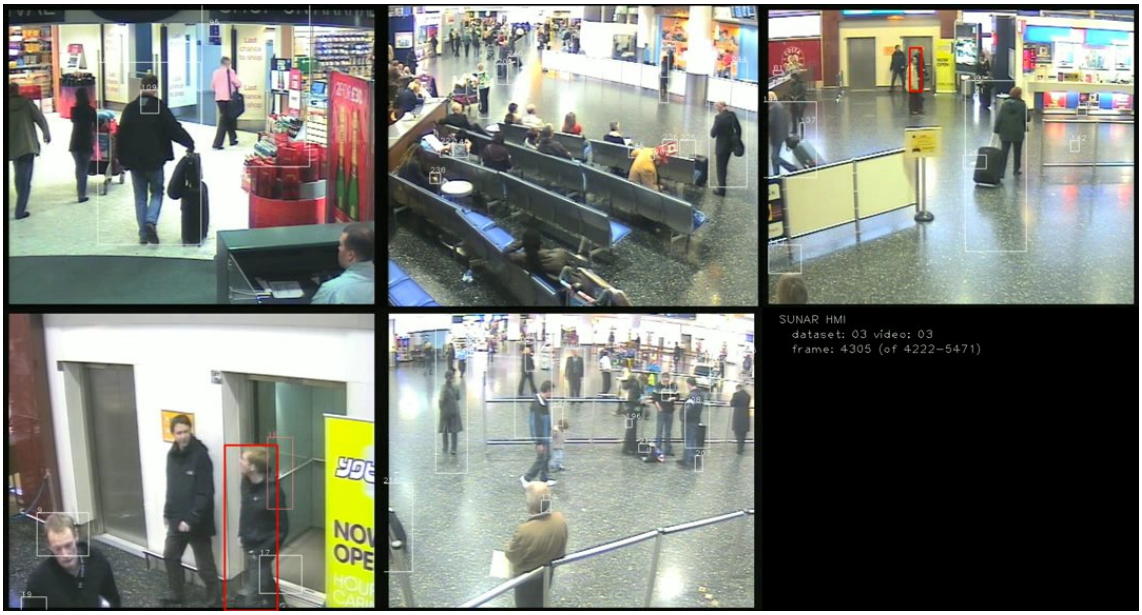
Dnes už je veľmi obtiažne nájsť aplikáciu, ktorá by si vystačila iba so vstavanými a vlastnými knižnicami. SUNAR nie je výnimkou. Pre spracovanie obrazu je v module SUNAR-HMI použitá knižnica OpenCV 2.1, SUNAR-VRM zase používa niekoľko ďalších voľne dostupných knižníc tretích strán (napríklad postgresql-jdbc4 pre pripojenie sa k databáze a iné). Zameriame sa však na knižnicu LibSVM (voľne dostupná na stiahnutie na [4]), ktorá bola priamo používaná v implementačnej časti.



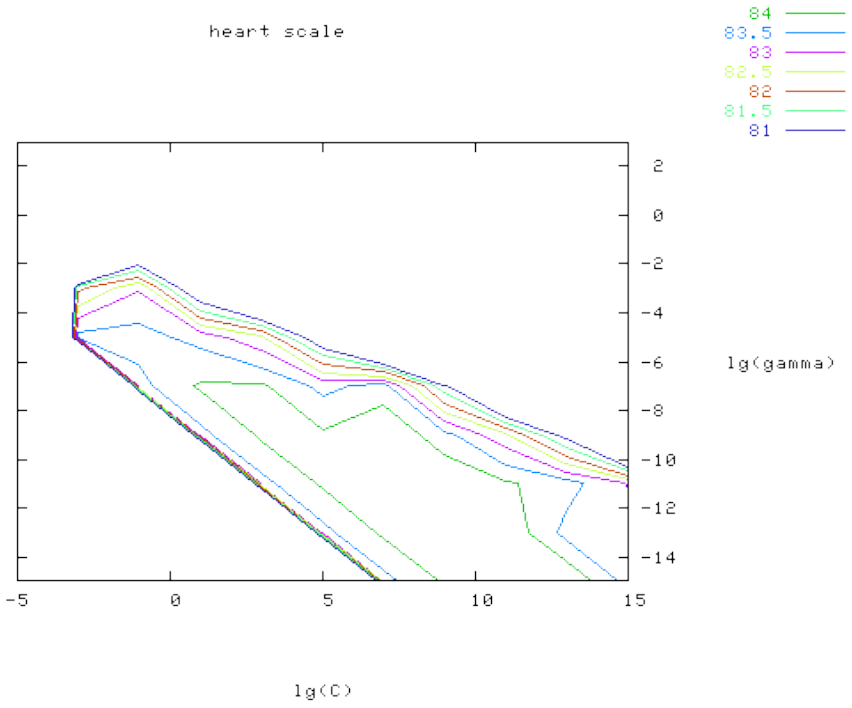
Obr. 6.1: Rozmiestnenie kamier v letiskovej hale.



Obr. 6.2: Grafické užívateľské rozhranie modulu SUNAR-VRM.



Obr. 6.3: Demonštrácia modulu SUNAR-HMI.



Obr. 6.4: Demonštrácia výstupu z knižnice LibSVM, prebraté z [4].

Knižnica LibSVM pochádza od autorov Chih-Chung Chang a Chih-Jen Lin. Pre experimenty bola použitá verzia 3.0, ktorá bola vydaná v októbri 2010. Je napísaná v jazyku C, pričom existujú rozhrania na jej použitie v mnohých ďalších jazykoch (Java, MATLAB, Python, Perl, C#, ...). Knižnica nie je začlenená do projektu a nevolajú sa priamo jej funkcie, ale je preložená do binárneho kódu a z modulu SUNAR-VRM sa volajú tieto binárky. To zaručuje rýchlosť trénovania modelu aj predikovania výsledkov. Ako pomocné sú použité aj Python skripty, ktoré sú súčasťou LibSVM a majú za úlohu uľahčiť volanie binárok tejto knižnice. Pre ich napojenie na SUNAR-VRM však museli byť tieto skripty písané v Python mierne modifikované. Preto sa odporúča dôkladná kontrola funkčnosti aplikácie pri potencionálnom prechode na novšiu verziu tejto externej knižnice. LibSVM ešte pre svoje správne fungovanie vyžaduje aplikáciu Gnuplot (voľne dostupná na stiahnutie na [40]).

6.4 Zhodnotenie výstupov

V praktickej časti tejto práce som urobil analýzu klasifikátoru SVM, pričom som sa zameral na vyhodnotenie jej úspešnosti X-validáciou. Postupne som pridával prechody medzi jednotlivými kamerami a výsledky som zaznačoval do tabuľky. Na konci som vygeneroval graf úspešnosti trénovania.

6.4.1 Proces anotácie

Pred spustením experimentov bolo potrebné anotovať sadu trénovacích dát. Na týchto dátach si SVM klasifikátor vytvoril model, na základe ktorého už je schopný anotovať ďalšie prechody samostatne.

Prechody som anotoval ručne z videí dostupných na fakulte. Išlo o zábery z piatich dohľadových kamier Londýnskeho letiska Gatwick. K tomuto účelu som použil upravený modul SUNAR-HMI, ktorý mi zaznamenané prechody automaticky ukladal do databázy na požadované miesto. Pozitívnou vlastnosťou modulu SUNAR-HMI je jeho automatické ukladanie vygenerovaného videa. Vďaka tomu je možné video spätne zhliadnuť bez nutnosti jeho opätovného znovuvygenerovania. Toto je vhodné z toho dôvodu, že vygenerovanie videa je časovo náročný proces.

Po anotovaní videí prišiel na radu modul SUNAR-VRM, ktorého úlohou bolo vytvorenie modelu rozpoznávania pre metódu SVM, ako aj ohodnotenie tohto modelu. Výsledky sú spracované a diskutované v kapitolách 6.4.2 a 6.4.3.

6.4.2 Výsledky experimentov

Rozloženie trénovacích dát je v tabuľkách 6.1, 6.3, 6.5, 6.7 a 6.9 pre postupne 50, 100, 150, 200 a 250 vzoriek. V riadkoch sú zdrojové kamery (z ktorých objekt odišiel) a v stĺpcoch cieľové (v ktorých sa objekt následne objavil).

V tabuľkách 6.2, 6.4, 6.6, 6.8 a 6.10 je percentuálny podiel úspešnosti. *Úspešnosť 1* značí úspešnosť správnej detekcie objektu v cieľovej kamere, a *Úspešnosť 2* je úspešnosť pre dvojnásobný počet dát, pričom bol braný v úvahu aj "spätný"pohyb. Inak povedané, ak objekt prešiel z kamery 1 do kamery 2, tak v tejto druhej sade bol umelo pridaný prechod tohto objektu z kamery 2 do kamery 1. Celková úspešnosť je počítaná ako váhovaný priemer jednotlivých kamier, počítaná pre *Úspešnosť 1* podľa vzorca 6.1 a pre *Úspešnosť 2* podľa 6.2. k_{ij} značí počet prechodov z kamery i do kamery j, u_i je úspešnosť rozpoznávania v kamere i. n je počet vzoriek v trénovacej množine dát.

	Kamera 1	Kamera 2	Kamera 3	Kamera 4	Kamera 5
Kamera 1	0	15	0	0	0
Kamera 2	1	0	15	0	2
Kamera 3	0	10	1	1	2
Kamera 4	0	0	2	0	0
Kamera 5	0	0	1	0	0

Tabuľka 6.1: Anotované prechody - 50 vzoriek.

Kamera	**Úspešnosť 1**	**Úspešnosť 2**
Kamera 1	100,00 %	100,00 %
Kamera 2	83,33 %	90,70 %
Kamera 3	71,43 %	84,85 %
Kamera 4	100,00 %	100,00 %
Kamera 5	100,00 %	60,00 %
Celkovo	**85,99 %**	**89,00 %**

Tabuľka 6.2: Úspešnosť určenia správneho prechodu - 50 vzoriek.

	Kamera 1	Kamera 2	Kamera 3	Kamera 4	Kamera 5
Kamera 1	0	26	0	0	0
Kamera 2	2	0	33	0	2
Kamera 3	0	15	1	1	10
Kamera 4	0	0	2	0	0
Kamera 5	0	1	7	0	0

Tabuľka 6.3: Anotované prechody - 100 vzoriek.

Kamera	**Úspešnosť 1**	**Úspešnosť 2**
Kamera 1	100,00 %	100 %
Kamera 2	89,19 %	91,14 %
Kamera 3	85,19 %	77,14 %
Kamera 4	100,00 %	100,00 %
Kamera 5	87,50 %	85,00 %
Celkovo	**91,00 %**	**87,00 %**

Tabuľka 6.4: Úspešnosť určenia správneho prechodu - 100 vzoriek.

	Kamera 1	Kamera 2	Kamera 3	Kamera 4	Kamera 5
Kamera 1	1	32	0	0	0
Kamera 2	4	0	52	0	2
Kamera 3	0	24	2	1	17
Kamera 4	0	0	3	0	0
Kamera 5	0	2	10	0	0

Tabuľka 6.5: Anotované prechody - 150 vzoriek

Kamera	**Úspešnosť 1**	**Úspešnosť 2**
Kamera 1	96,97 %	94,74 %
Kamera 2	89,66 %	88,79 %
Kamera 3	88,64 %	75,68 %
Kamera 4	100,00 %	100,00 %
Kamera 5	83,33 %	87,10 %
Celkovo	**90,67 %**	**84,67 %**

Tabuľka 6.6: Úspešnosť určenia správneho prechodu - 150 vzoriek.

	Kamera 1	Kamera 2	Kamera 3	Kamera 4	Kamera 5
Kamera 1	3	43	0	0	0
Kamera 2	5	0	67	0	2
Kamera 3	0	32	3	1	27
Kamera 4	0	0	3	0	0
Kamera 5	0	2	12	0	0

Tabuľka 6.7: Anotované prechody - 200 vzoriek

Kamera	**Úspešnosť 1**	**Úspešnosť 2**
Kamera 1	93,48 %	94,44 %
Kamera 2	91,89 %	90,73 %
Kamera 3	85,71 %	72,30 %
Kamera 4	100,00 %	100,00 %
Kamera 5	85,71 %	90,70 %
Celkovo	**91,38 %**	**84,50 %**

Tabuľka 6.8: Úspešnosť určenia správneho prechodu - 200 vzoriek.

	Kamera 1	Kamera 2	Kamera 3	Kamera 4	Kamera 5
Kamera 1	3	64	0	0	0
Kamera 2	5	0	81	0	3
Kamera 3	0	41	3	1	30
Kamera 4	0	0	3	0	0
Kamera 5	0	2	14	0	0

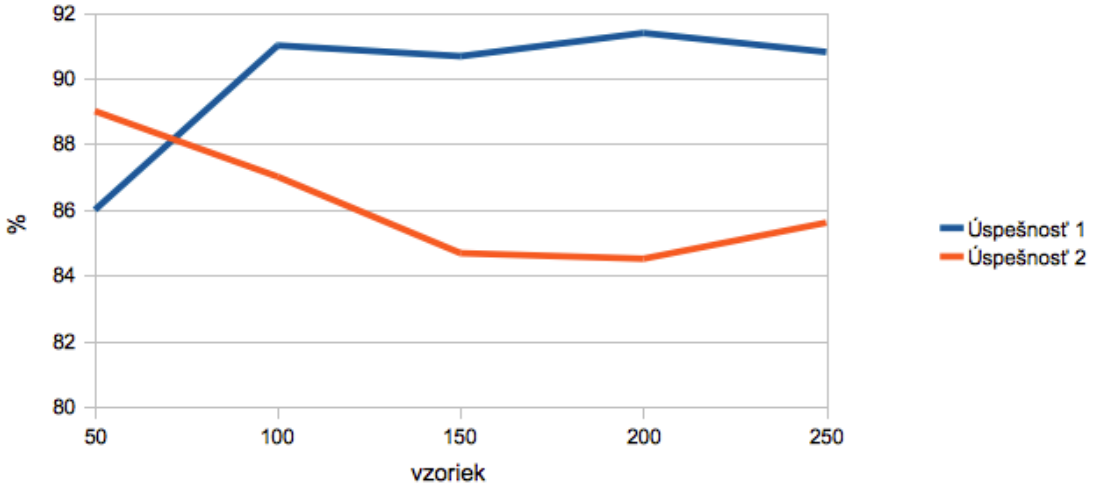
Tabuľka 6.9: Anotované prechody - 250 vzoriek.

Kamera	**Úspešnosť 1**	**Úspešnosť 2**
Kamera 1	95,52 %	96,00 %
Kamera 2	92,13 %	90,82 %
Kamera 3	85,33 %	73,86 %
Kamera 4	100,00 %	100,00 %
Kamera 5	87,50 %	89,80 %
Celkovo	**90,80 %**	**85,60 %**

Tabuľka 6.10: Úspešnosť určenia správneho prechodu - 250 vzoriek.

$$uspesnost1 = \frac{\sum_{i=1}^{5}(\sum_{j=1}^{5} k_{ij}) * u_i)}{n} * 100\% \tag{6.1}$$

$$uspesnost2 = \frac{\sum_{i=1}^{5}(\sum_{j=1}^{5} k_{ij} + k_{ji}) * u_i}{n * 2} * 100\% \tag{6.2}$$



Obr. 6.5: Úspešnosť rozpoznávania správnych prechodov objektov v závislosti od počtu vzoriek v trénovacej množine dát.

Už na prvý pohľad je vidieť relatívne vysokú úspešnosť metódy SVM. Istý podiel na tom má aj fakt, že nie je možný prechod objektu medzi ľubovoľnými dvoma kamerami, a objekty sa často pohybujú podobných trasách. Málo rôznych prechodov sa prejavuje najmä v experimentoch s nižím počtom vzoriek. Pre *Úspešnosť 1* je vidieť pomalý postupný rast od počtu 100 vzoriek. Naopak, *Úspešnosť 2* najprv rýchlo klesá a až následne sa stabilizuje a začne postupne narastať. V tomto prípade nízky počet trénovacích vzoriek ovplyvňuje úspešnosť rozpoznávania pre menej ako 150 vzoriek, ale s ich postupným pridávaním sa situácia stabilizuje a viac pripomína realitu. Niektoré z dôvodov, prečo nebola dosiahnutá vyššia úspešnosť rozpoznávania, sú diskutované v kapitole 6.4.2.

6.4.3 Faktory negatívne ovplyvňujúce rozpoznávanie

Odhalil som nasledovné nedostatky identifikácie objektov aplikáciou SUNAR:

- Pri príchode osoby na kameru nie je táto osoba rozpoznaná ihneď, ale niekedy až po prejdení niekoľkých metrov. Kvôli tomu má klasifikátor mylné informácie nielen o vstupnom a výstupnom bode medzi kamerami, ale aj o čase potrebnom na prekonanie vzdialenosti medzi nimi.
- Ak je v obraze veľa osôb, SUNAR ich nedokáže jednoznačne identifikovať a obdĺžnik reprezentujúci rozpoznaný objekt "skáče"medzi nimi. Toto je vidieť hlavne pri zhluku osôb ďalej od kamery, pretože sú opticky menšie (obrázky A.1 a A.2).

- Ak na jednom mieste osoby odchádzajú a zároveň prichádzajú do kamery, často sa stane, že prichádzajúci objekt je identifikovaný ako odchádzajúci, čiže ako by odchádzajúca osoba neodišla z kamery ale prudko zmenila svoj smer pohybu. V prípade osôb stojacich pri kraji obrazu akoby odchádzajúca osoba iba zastala. Tento problém sa týka osôb s podobným vektorom príznakov (obrázok A.5).
- Problém, kedy je pod rovnakým ID identifikovaných viac objektov, sa vyskytuje nielen v prípadoch opísaných v predchádzajúcich dvoch bodoch, ale často aj pri ich prekryvoch. Vidieť to môžeme mimo iného aj na kamere 1 (obrázok A.1).
- Niektoré osoby nie sú vôbec rozpoznané počas celého ich výskytu na kamere.
- Problém spôsobuje aj to, ak je ako osoba rozpoznaná iba jej tvár alebo batožina. Vtedy je získaný zlý vektor príznakov, ktorý zťažuje identifikáciu tejto osoby na inej kamere, ak tam už bola rozpoznaná celá (obrázok A.4).
- Na rozdiel od predchádzajúceho bodu sa stretneme aj s prípadmi, kedy je ich v jednej osobe rozpoznaných niekoľko (obrázok A.5).
- Veľmi často sa opakujúca situácia je, keď sa v obraze prekrýva obrovské množstvo ľudí a ani človek ich nedokáže identifikovať.
- Na kamere 3 nie sú identifikované osoby, ktoré prejdú v jej tesnej blízkosti. Je to preto, že na obraze nie sú zachytené celé, ale len ich časť (obrázok A.3).
- Osoby sa často prekrývajú s inými súčasťami scény, čo spôsobuje ich nerozpoznanie. Vidieť to nielen pri prekryve so zábradlím na kamere 3 alebo stĺpmi na kamere 2, ale aj pri sediacich osobách opäť v druhej kamere.
- Ako osoba je rozpoznaný neživý objekt (výťah na obrázku A.2 alebo reklama na obrázku A.3).

6.5 Možnosti vylepšenia

Pre spoľahlivejšiu identifikáciu objektov by bolo vhodné nasledovné:

- Kvalitnejší vektor príznakov, aby neboli rôzne osoby rozpoznané ako jeden objekt.
- Lepšie rozmiestnenie kamier tak, aby neboli slepé miesta.
- Kamera by nemala zachytávať rozľahlý priestor (ako kamery 2 a 5), ale skôr konkrétnu časť chodby (kamery 1 a 3).
- Zaujímavým vylepšením by bola aj konštrukcia 3D scény. Tým by sa odstránil problém prekrývania osôb. Vyžadovalo by to však oveľa väčšie množstvo kamier, bol by potrebný ďalší modul na konštrukciu 3D scény (ktorý by bol samozrejme výpočetne náročný), a neposlednom rade by to aj zvýšilo čas vykonávania rozpoznávania objektov.

Videá v module SUNAR-HMI je síce možné prehrávať smerom dopredu rôznymi rýchlosťami či ich zastaviť a krokovať, avšak pri analýze by sa zišiel aj pohyb dozadu, prípadne jazdec, ktorým by bolo možné skočit na ktorýkoľvek časový okamih záznamu (podobne ako to majú programy na prehrávanie videa). SUNAR má však slúžiť hlavne na spracovanie a analýzu videa, takže táto vymoženosť zatiaľ nie je implementovaná.

6.6 Zhrnutie

Opísali sme si aplikáciu SUNAR, ktorá bola využitá na praktickú časť tejto práce. Takisto sme sa bližšie pozreli na tie súčasti tejto aplikácie, ktoré boli vytvorené, doplnené, alebo využité pri našich experimentoch s klasifikátorom SVM. Vykonali sme niekoľko experimentov s rôznym počtom anotovaných vzoriek v trénovacích dátach a výsledky sme zaznačili do prehľadných tabuliek a grafov. V závere kapitoly sme sa venovali identifikácií problémov, ktoré nás vzďaľujú od rozumnej miery identifikácie objektov. Navrhli sme aj možné vylepšenia, ktoré sú však viac-menej iba v teoretickej rovine (videá už boli vytvorené a teda nie je možné manipulovať s kamerami).

Na jednej strane je vhodné, keď sú videá zaznamenávané vo vysokej kvalite, na druhej strane to však spotrebúva viac výpočtového výkonu počítača, na ktorom sú videá analyzované a spracovávané. Určite by ale bolo vhodné zamyslieť sa nad možnosťou paralelizácie niektorých modulov, a tým využiť potenciál súčasných viacjadrových procesorov. Ako príklad môžeme uviesť funkciu modulu SUNAR-HMI, ktorá spracováva 5 videí naraz a zobrazuje ich na svoj výstup. Ak by bolo každé video spracovávané jedným jadrom, výrazne by to urýchlilo celkové spracovanie záznamov.

Kapitola 7

Záver

Táto práca sa venovala stručnému zhrnutiu teoretických znalostí, ktoré boli potrebné pre následné pochopenie účelu práce. Jednotlivé pojmy a metódy boli demonštrované na rôznych príkladoch z praxe, aby si čitateľ uvedomil ich dôležitosť v širšom okruhu. Dôraz bol takisto kladený aj na používanie obrázkov, ktoré graficky pomáhajú pri pochopení opisovaných znalostí. V žiadnom prípade však nešlo o detailné rozoberanie tematiky a pre bližšie naštudovanie je potrebné siahnuť po odbornej literatúre venujúcej sa opisovanej téme.

V druhej časti práce sa nadviazalo na teoretické poznatky z predchádzajúcich kapitol a boli prakticky demonštrované na programe SUNAR. Program SUNAR slúži na spracovanie videí z dohľadových kamier, umiestnených na rôznych miestach letiskovej haly. V rámci spracovania videí sa identifikujú osoby v obraze, zisťuje sa trajektória ich pohybu, a snažia sa identifikovať totožné osoby na rôznych kamerách, či už v tom istom časovom okamihu, alebo po určitom čase, ktorý potrebovala osoba na prechod z miesta monitorovaného jednou kamerou na miesto monitorované inou. Sú detekované aj situácie, keď osoba opustí monitorované územie, ale následne sa do neho vráti.

Práve identifikácia tej istej osoby v rôznych kamerách bola hlavnou súčasťou tejto práce. To sa určí na základe vektoru príznakov rozpoznaných osôb, ktorý je predaný klasifikátoru SVM. Pre správnu funkčnosť sa potrebuje klasifikátor najprv naučiť, ako vyzerá prechod medzi kamerami, na čo mu slúži anotovaná sada 250 vzoriek prechodov medzi kamerami. Vytvorenie tejto sady bolo tiež súčasťou tejto práce. Kvôli tomuto účelu bol upravený modul SUNAR-HMI tak, aby bolo možné anotovať prechody z jeho grafického užívateľského rozhrania.

Záver práce bol venovaný vyhodnoteniu dosiahnutých výsledkov, pričom sa identifikovali jednotlivé problémy rozpoznávania a boli aj demonštrované na obrázkoch. Následne sa uviedlo niekoľko spôsobov, ktoré by pomohli kvalitnejšiemu rozpoznávaniu objektov.

Na prácu je možné nadviazať napríklad skvalitnením vektoru príznakov (skvalitnenie sa pozná na lepšej klasifikačnej schopnosti SVM klasifikátoru), skvalitnením rozpoznávania osôb (napríklad tých čo sa objavujú na okraji kamery), alebo paralelizáciou výpočtov.

Literatúra

[1] Ari: Systémy podnikové inteligence.
http://ari.wikidot.com/systemy-podnikove-inteligence, citované [2011-05-20].

[2] Berka, P.: Bayesovská klasifikace. [online].
http://sorry.vse.cz/ berka/docs/izi456/kap_5.6.pdf, citované [2011-05-20].

[3] Chen, C. X.: *Data Models and Query Languages of Spatio-Temporal Information.* 2001, 124 s.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.5328&rep=rep1&type=pdf

[4] Chih-Chung Chang; Chih-Jen Lin: LIBSVM – A Library for Support Vector Machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm/.

[5] Chmelař, P.: JKalman, Java Kalman Filter.
http://sourceforge.net/projects/jkalman/, voľne dostupná knižnica implementujúca Kalmanov filter v Jave.

[6] Chmelař, P., Láník, A.: Information Retrieval Based Wide Area Surveillance System.

[7] Chou, K.-C.: *Proteins: Structure, Function & Genetics.* Proceedings of the National Institute of Sciences of India, 1995, 319-344 s., a novel approach to predicting protein structural classes in a (20–1)-D amino acid composition space.

[8] Chytka, K.: *Získávání znalostí z objektově relačních databází. [Diplomová práca].* Vysoké učení technické v Brně., 2008, 65 s., fakulta informačních technologií. Vysoké učení technické v Brně. Ústav informačních systémů. Vedoucí diplomové práce Ing. Petr Chmelař.

[9] Duda, R. O.; Hart, P. E.; Stork, D. G.: *Pattern Classification.* Wiley-Interscience, 2000, ISBN 9780471056690, 654 s.

[10] Groff, J. R.; Weinberg, P. N.: *SQL: The Complete Reference.* Osborne/McGraw-Hill, 1999, ISBN 0072118458, 2008 s.

[11] Güting, R. H.; Böhlen, M. H.; Erwig, M.; aj.: *Spatio-temporal Models and Languages: An Approach Based on Data Types.* 2003, ISBN 978-3-540-40552-8, 117-176 s.

[12] Güting, R. H.; Schneider, M.: *Moving Objects Databases.* Morgan Kaufmann, 2005, ISBN 0120887991, 389 s.

[13] Jensen, F. V.: *Introduction to Bayesian Networks.* Springer, 1997, ISBN 0387915028, 178 s.

[14] Jensen, F. V.: *Bayesian Networks and Decision Graphs*. Springer, 2002, ISBN 0387952594, 284 s.

[15] Kolář, D.: *Pokročilé databázové systémy : prostorové databáze*. Vysoké učení technické v Brně., 2006, 63 s.

[16] Korte, G.: *The GIS Book*. OnWord Press, 2000, ISBN 0766828204, 400 s.

[17] Krajmer, M.: *Matematický model chemického reaktora a nelineárny odhad stavu [Diplomová práca]*. Slovenská technická univerzita v Bratislave, 2004, slovenská technická univerzita v Bratislave, Fakulta chemickej a potravinárskej technológie, Katedra informatizácie a riadenia procesov. Školiteľ: Prof. Ing. Ján Mikleš, DrSc.

[18] Levitan, I. B.; Kaczmarek, L. K.: *The Neuron: Cell and Molecular Biology*. Oxford University Press, USA, 2001, ISBN 0195145232, 632 s.

[19] Lukáš, R.: Klasifikace a predikce. [online]. `https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/ZZN-IT/lectures/05_klasi` citované [2011-05-20].

[20] Mahalanobis, P.C.: *On the generalized distance in statistics*. Proceedings of the National Institute of Science of India, 1936, s. 49-55, 1936.

[21] Neapolitan, R. E.: *Learning Bayesian Networks*. Prentice Hall, 2003, ISBN 9780130125347, 674 s.

[22] Oracle Corporation: Oracle Spatial [online]. `http://www.oracle.com/technetwork/database/options/spatial/index.html`, citované [2011-05-20].

[23] Paralič, J.: *Objavovanie znalostí v databázach*. 2003, 80 s.

[24] Patlevič, P.: *Použitie techník Support Vector Machines pri učení neurónových sietí. [Diplomová práca]*. Technická univerzita v Košiciach., 2005, fakulta elektrotechniky a informatiky. Technická univerzita v Košiciach. Katedra kybernetiky a umelej inteligencie. Vedúci diplomovej práce Ing. Rudolf Jakša, PhD.

[25] PgFoundry: PostgreSQL Geometric Types [online]. `http://www.postgresql.org/docs/9.0/static/datatype-geometric.html`, citované [2011-05-20].

[26] Procházka, J.: Objektově orientované databáze [online]. `http://www.dbsvet.cz/view.php?cisloclanku=2004030301`, citované [2011-05-20].

[27] Refractions: PostGIS [online]. `http://postgis.refractions.net`, citované [2011-05-20].

[28] Roweis, S.; Ghahramani, Z.: *An Unifying Review of Linear Gaussian Models*. Neural Computation, 1999.

[29] Shekhar, S.; Chawla, S.: *Spatial Databases: A Tour*. Prentice Hall, 2003, ISBN 9780130174802, 262 s.

[30] [S.N.]: Accessing OLAP using ASP.NET.
http://www.aspfree.com/c/a/MS-SQL-Server/Accessing-OLAP-using-ASP-dot-NET/,
citované [2011-05-20].

[31] [S.N.]: Binary classification [online].
http://en.wikipedia.org/wiki/Binary_classification, citované [2011-05-20].

[32] [S.N.]: Introduction to Support Vector Machine (SVM) Models [online].
http://www.dtreg.com/svm.htm, citované [2011-05-20].

[33] [S.N.]: Mahalanobis distance [online].
http://www.aiaccess.net/English/Glossaries/GlosMod/e_gm_mahalanobis.htm,
citované [2011-05-20].

[34] [S.N.]: Support vector machine [online].
http://en.wikipedia.org/wiki/Support_vector_machine, citované [2011-05-20].

[35] Tan, P.; Steinbach, M.; Kumar, V.: *Introduction to Data Mining.* Addison Wesley, 2005, ISBN 0321321367, 769 s.

[36] Tansel, A. U.; Clifford, J.; Gadia, S.: *Temporal Databases: Theory, Design, and Implementation.* Benjamin-Cummings Pub Co, 1993, ISBN 0805324135, 656 s.

[37] Tuček, J.: *Geografické informační systémy.* Computer Press, 1998, 438 s.

[38] Welch, G.; Bishop, G. : An Introduction to the Kalman Filter.
http://www.cs.unc.edu/ welch/kalman/, 2006.

[39] Willamette: Overfitting [online].
http://www.willamette.edu/ gorr/classes/cs449/overfitting.html, citované
[2011-05-20].

[40] Williams, C., T.; Kelley: Gnuplot download.
http://www.gnuplot.info/download.html.

[41] Williams, B.: Data model [online].
http://www.databaseanswers.org/data_models/spatial_temporal_databases/index.htm,
citované [2011-05-20].

[42] WWW stránky: Databázové systémy [online].
http://hornad.fei.tuke.sk/predmety/ivpp/5datab.htm, citované [2011-05-20].

[43] WWW stránky: Kalmanov filter [online].
http://en.wikipedia.org/wiki/Kalman_filter, citované [2011-05-20].

[44] Zaiane, O. R. : Introduction to Data Mining [online].
http://webdocs.cs.ualberta.ca/ zaiane/courses/cmput690/notes/Chapter1/index.html,
citované [2011-05-20].

[45] Zendulka, J.; kolektív: *Získávání znalostí z databází : Studijní opora.* Vysoké učení technické v Brně., 2006, 160 s.

[46] Zhang, J.: *Spatio-Temporal Database.* 2005, 18 s.

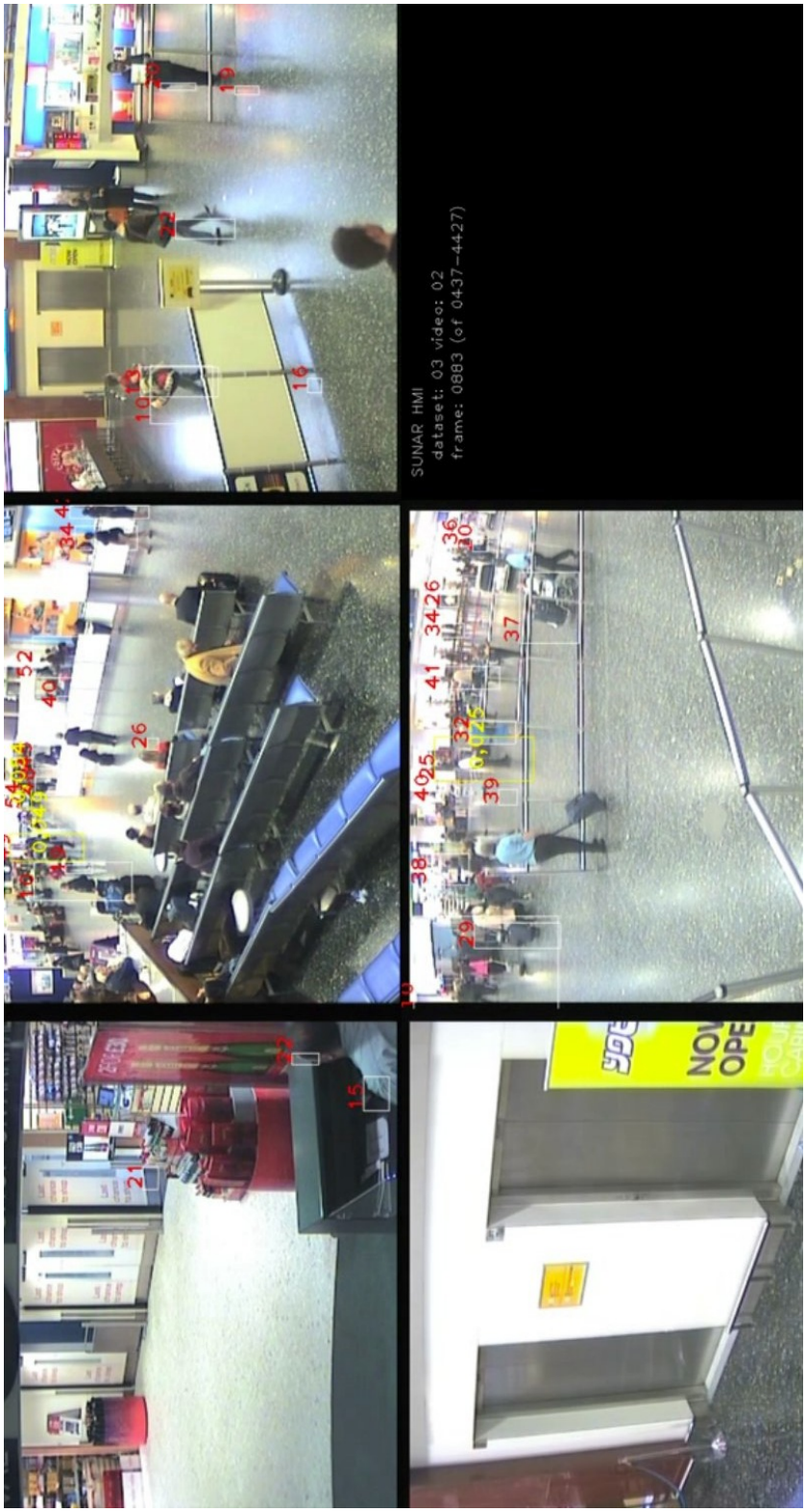
Dodatok A

Demonštrácia problémov pri rozpoznávaní osôb

Obr. A.1: Na kamere 1 je vidieť zhluk ľudí, ktorí boli rozpoznaní ako jediný objekt. Na kamerách 2, 3, a 5 je vidieť prehustená letisková hala.

Obr. A.2: Všimnime si (mimo iného) nerozpoznanú telefonujúcu osobu v bledom na kamere 3, či výťah otvorený rozpoznaný ako 4 osoby na kamere 4.

Obr. A.3: Všimnime si zle zamerané osoby na kamere 1, osoby na kamere 3 stojace za zábradlím, alebo osobu prechádzajúcu popredím kamery 3, pričom na kamere ani nie je zachytená celá. Častou chybnou identifikáciou bola aj reklama (53) na kamere 3.



Obr. A.4: Tento obrázok je ukážkou rozpoznania iba časti osoby, či jej batožiny. Je ich vidieť viacero na kamere 3, ale aj na kamere 5 je napríklad rozpoznaný iba vozík (37).

Obr. A.5: Všimnime si detailne pani stojacu vľavo pri zábradlí. Najprv je chybne rozpoznaná ako 3 osoby, následne do scény vchádzajú 3 ďalšie osoby (ktoré sú taktiež zle rozpoznané), a jedna z anotácií (71) následne skočí na pohybuúcich sa ľudí. Tento prípad sa opakoval pomerne často.

Dodatok B

Popis zdrojových kódov

V tejto prílohe sú stručne popísané zdrojové kódy vytvorené v praktickej časti práce. Sú súčasťou CD nosiča priloženého k práci.

SUNAR-VRM

Do modulu SUNAR-VRM bola pridaná trieda *Svm*, ktorá obsahuje nasledovné verejné metódy:

public Svm(Commons commons); - konštruktor, ktorý má ako svoj jediný argument štruktúru obsahujúcu nastavenia.
public boolean trainSVM(); - metóda vytvorí trénovaciu sadu dát.
public void createModel(); - má za úlohu zavolať LibSVM a vytvoriť model z trénovacích dát.
public void createTestDataset(); - vytvorí testovaciu sadu dát. Používa sa len v prípade, ak sa netestuje úspešnosť X-validáciou.
public void predict(); - predikuje kameru, v ktorej sa má zobraziť objekt.

Zároveň bola upravená trieda *Commons* tak, aby v sebe mohla mať uložené nastavenia potrebné pre Svm. Ide hlavne o názvy tabuliek a názvy k dočasných súborov. Poslednou zmenou bolo pridanie odkazu do grafického uživateľského rozhrania, aby túto funkciu bolo možné spustiť manuálne.

K zdrojovým kódom modulu SUNAR-VRM bola do adresára *./external* pridaná upravená externá knižnica LibSVM 3.0, ktorá je volaná z tohto modulu.

SUNAR-HMI

Do modulu SUNAR-HMI bola pridaná funkcia *int processInputKey (char dc)*, ktorá má na starosti pokročilé spracovanie vstupu od užívateľa. Po stlačení klávesy *h* sa prepne do módu, kedy očakáva číslo anotácie objektu v obraze. Z toho dôvodu nie je v tomto móde možné meniť rýchlosť prehrávania záznamu (napríklad anotácia číslo 185 by zároveň spôsobila zmenu rýchlosti, a to je nežiadúce). Objekty sa oddeľujú znakom *čiarka*, a po zadaní druhého objektu je potrebné potvrdiť to klávesou *enter*. Klávesy *q* a *w* si ponechávajú svoj pôvodný

význam[1]. Všetky ostatné vstupy sú ignorované.

Handover

Do databázy bola pridaná doleuvedená funkcia, napísaná v jazyku PL/pgSQL. Má za úlohu označiť dva totožné objekty rovnakým ID.

```
CREATE OR REPLACE FUNCTION sunar.handover(dataset integer, video integer, camera1
 integer, track1 integer, camera2 integer, track2 integer) RETURNS integer AS $$
DECLARE
    t1 RECORD;
    t2 RECORD;
    object INTEGER;
BEGIN
    object := 0;

    EXECUTE 'SELECT object FROM sunar.tracks WHERE dataset = $1 AND video = $2
     AND camera = $3 AND track = $4'
    INTO t1
    USING dataset, video,camera1, track1;

    EXECUTE 'SELECT object FROM sunar.tracks WHERE dataset = $1 AND video = $2
    AND camera = $3 AND track = $4'
    INTO t2
    USING dataset, video,camera2, track2;

    IF t1.object > 0 THEN
    object := t1.object;
    EXECUTE 'UPDATE sunar.tracks SET object = $1 WHERE dataset = $2 AND video = $3
    AND camera = $4 AND track = $5' USING object, dataset, video, camera2, track2;
    ELSEIF t2.object > 0 THEN
    object := t2.object;
    EXECUTE 'UPDATE sunar.tracks SET object = $1 WHERE dataset = $2 AND video = $3
    AND camera = $4 AND track = $5' USING object, dataset, video, camera1, track1;
    ELSE
    EXECUTE 'INSERT INTO sunar.objects (dataset, verified) VALUES ($1, true)
    RETURNING object' INTO object USING dataset;
    EXECUTE 'UPDATE sunar.tracks SET object = $1 WHERE dataset = $2 AND video = $3
    AND ((camera = $4 AND track = $5) OR (camera = $6 AND track = $7))'
    USING object, dataset, video,camera1, track1, camera2, track2;

    END IF;

    RETURN object;
END;
$$ LANGUAGE plpgsql;
```

[1] q - ukončenie prehrávania celej sady videí, w - preskočenie aktuálneho videa.