VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

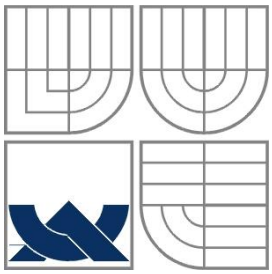
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

# ZÍSKÁVÁNÍ ZNALOSTÍ Z MULTIMEDIÁLNÍCH DATABÁZÍ

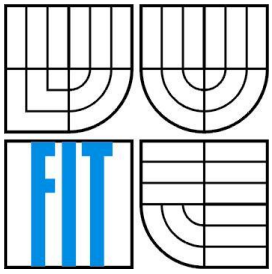
DIPLOMOVÁ PRÁCE
MASTERS'S THESIS

AUTOR PRÁCE Bc. Peter Málik
AUTHOR

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 MULTIMEDIÁLNÍCH DATABÁZÍ

KNOWLEDGE DISCOVERY IN MULTIMEDIA DATABASES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Peter Málik
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Chmelař
SUPERVISOR

BRNO 2011

# Abstrakt

Tato diplomová práce se zabývá problematikou získávání znalostí z multimediálních databází. Obsahuje obecný princip získávání znalostí z databází. Důraz je kladen na metody shlukové analýzy pro dolování dat v rozsáhlých a multidimenzionálních databázích. Dále tahle práce obsahuje úvod do multimediálních databází se zaměřením se na extrakci nízkourovňových vizuálních rysů z obrázků a video dat. Praktickou částí práce je potom implementace metod BIRCH, DBSCAN a k-means určených pro shlukovou analýzu. Závěr je věnován experimentům nad datovou sadou TRECVid 2008 a popisu dosažených výsledků.



# Abstract

This master's thesis deals with the knowledge discovery in multimedia databases. It contains general principles of knowledge discovery in databases, especially methods of cluster analysis used for data mining in large and multidimensional databases are described here. The next chapter contains introduction to multimedia databases, focusing on the extraction of low level features from images and video data. The practical part is then an implementation of the methods BIRCH, DBSCAN and k-means for cluster analysis. Final part is dedicated to experiments above TRECVid 2008 dataset and description of achievements.



# Klíčová slova

Získávání znalostí z databází, dolování dat, shlukování, shluková analýza, multimediální databáze, nízkoúrovňové vizuální rysy, BIRCH, DBSCAN, k-means, RapidMiner, TRECVid, Trecvid Search



# Keywords

Knowledge discovery in databases, data mining, clustering, cluster analysis, multimedia databases, low level visual features, BIRCH, DBSCAN, k-means, RapidMiner, TRECVid, Trecvid Search



# Citace

Málik Peter: Získávání znalostí z multimediálních databází, semestrální projekt, Brno, FIT VUT v Brně, 2011

# Získávání znalostí z multimediálních databází

## Prehlásenie

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne pod vedením Ing. Petra Chmelaře. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

……

Peter Málik

25.05.2011

## Poďakovanie

Ďakujem týmto svojmu vedúcemu pánovi Ing. Petrovi Chmelařovi za jeho odborné vedenie, rady a pomoc pri riešení tejto diplomovej práce.

# Obsah

# 1 Úvod

V dnešnej dobe sme vďaka technologickému pokroku a globalizácii sveta doslova zaplavený obrovským objemom informácií. Tieto informácie sú zaznamenávané v prevažnej väčšine prípadov ako digitálne dáta. Aj vďaka tomu požadujeme stále dokonalejšie techniky zhromažďovania a ukladania získaných dát. Vzniká tým potreba sofistikovaných nástrojov a metód, ktoré by boli schopné automaticky získať z uložených dát užitočné informácie. Preto je odbor získavania znalostí z databáz jeden z najrýchlejšie sa rozvíjajúcich odborov informačných technológií. V súčasnosti je kladený veľký dôraz na efektívnu prácu s multimediálnymi dátami a to predovšetkým na nástroje umožňujúce spracovávať multimediálne dáta na základe ich obsahu.

Táto práca sa zameriava na problematiku získavania znalostí z multimediálnych databáz. Zaoberá sa hlavne zhlukovou analýzou a spracovaním nízkoúrovňového popisu video dát a obrázkov. Cieľom je vytvoriť rozšírenie aplikácie Trecvid Search, ktorá pracuje s vývojovou podmnožinou dátovej sady Sound and Vision zo súťaže TRECVid. Toto rozšírenie musí ponúkať viacero možností zhlukovej analýzy nad nízkoúrovňovými rysmi multimediálnych dát.

Prvé štyri kapitoly tejto práce boli prevzaté zo semestrálneho projektu, ktorý mal za úlohu vypracovať teoretickú prípravu k diplomovej práci.

Druhá kapitola obsahuje uvedenie do problému získavania znalostí z databáz, popisuje celkový priebeh tohto procesu, rôzne druhy databáz a jednotlivé typy dolovacích úloh.

Tretia kapitola podrobne popisuje zhlukovú analýzu, rôzne metódy a algoritmy. Veľký dôraz je kladený na algoritmy, ktoré majú za úlohu spracovávať viacdimenzionálne dáta, medzi ktoré patria aj multimediálne dáta.

Vo štvrtej kapitole je uvedené nízkoúrovňové spracovanie multimediálnych dát. Sú tu popísané techniky na získavanie a extrakciu lokálnych a globálnych rysov týchto dát.

Piata kapitola je venovaná samotnej implementácii rozšírenia programu Trecvid Search. Sú tu popísané použité nástroje, rozširujúce knižnice programovacieho jazyka Java a databázový systém PostgreSQL, spolu s dátovou sadou TRECVid. V rámci tejto kapitoly je popísaný aj návrh a implementačný postup praktickej časti diplomovej práce.

V šiestej kapitole sa nachádzajú, popis a výsledky experimentov vykonávaných nad dátovou sadou TRECVid. Porovnávajú sa tu výstupy rôznych metód zhlukovej analýzy z rôznych programov. Je ponúknutý náhľad na výsledky porovnávania obrázkov na základe nízkoúrovňových rysov.

Poslednou kapitolou je záver, ktorý obsahuje zhodnotenie všetkých dosiahnutých výsledkov, celkový prínos práce a návrhy pre možné rozšírenie a pokračovanie tejto práce.

# 2 Získavanie znalostí z databáz

Definícia získavania znalostí znie podľa [2] : "*Je to netriviálne získavanie implicitných, predtým neznámych a potenciálne užitočných informácií z dát* ". Týmto pojmom v angličtine nazývanom Knowledge Discovery in Databases (KDD) sa označuje smer v oblasti informačných technológií, ktorý sa začal výraznejšie presadzovať v 90. rokoch minulého storočia. Nasledujúci odsek je parafrázovaním [27].

Získavanie znalostí z databáz vzniklo ako reakcia na obrovské množstvo dostupných dát a ako naliehavá potreba premeniť tieto dáta na užitočnú informáciu a znalosť. Metódy analýzy založené na štatistike nie sú schopné odhaliť požadované informácie v takomto obrovskom objeme dát. Samotné získavanie znalostí z databáz je extrakcia zaujímavých modelov dát a vzorov z veľkého množstva dát. Extrahované modely dát sú prevažne netriviálne, skryté, predtým neznáme a potenciálne užitočné. To znamená, že tieto informácie sa nedajú získať jednoduchým použitím SQL príkazu a často sa vďaka nim robia dôležité rozhodnutia (napr. pôžička v banke). Ako synonymá sa používajú aj pojmy dolovanie z dát, alebo zjednodušene dolovanie dát. Tieto pojmy sú však mierne zavádzajúce, pretože ide iba o jednu časť celého procesu získavania znalostí z databáz. Tento vedný odbor sa dá označiť ako multidisciplinárny. Využíva napr. štatistiku, strojové učenie, algoritmy, databázové technológie, vizualizáciu a pod.

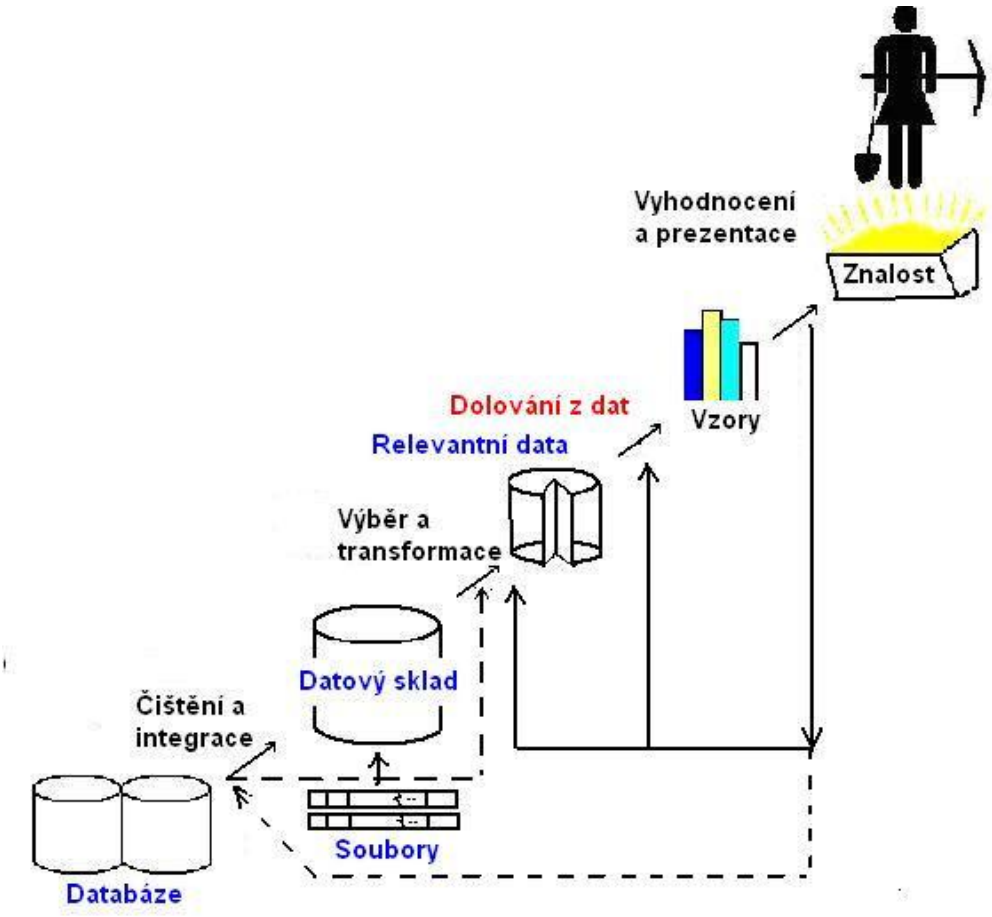
## 2.1 Proces získavania znalostí

Samotný proces získavania znalostí z databáz je netriviálny. Skladá sa z niekoľkých iteratívnych, navzájom oddelených častí, ktoré na seba nadväzujú. Niektoré z týchto častí sa môžu opakovať, aby sa dosiahlo získanie požadovaného výsledku. Celý proces získavania znalostí je zobrazený na obrázku 2.1.

Získavanie znalostí z databáz sa skladá z nasledujúcich krokov. Prvé štyri kroky sú označované ako predspracovanie dát a ich výsledkom sú dáta vhodné na dolovanie. Nasledujúce kroky a ich popisy sú prevzaté z [5] a [27].

1. **Čistenie dát:** Dáta prevzaté z reálneho sveta majú tendenciu byť nekompletné, zašumené a nekonzistentné. Takéto dáta nie je možné podrobiť dolovaciemu algoritmu, pretože informácie, ktoré by sme ním získali by boli vo väčšine prípadov nedostačujúce, alebo chybné. Preto je cieľom tohto kroku dáta predspracovať, teda doplniť chýbajúce dáta, odstrániť šum, či vyriešiť ich nekonzistentnosť.
2. **Dátová integrácia:** Zdrojové dáta môžu byť uložené v rôznych zdrojoch (napr. dátové kocky, obyčajné súbory). Cieľom dátovej integrácie je spojiť takéto dáta do jedného zdroja. Tento krok býva spojený s čistením dát, pretože spolu úzko súvisia. Vyčistené dáta je potrebné niekam uložiť a naopak pri integrácii vzniká nekonzistentnosť, ktorú rieši čistenie dát.
3. **Výber dát:** V tomto kroku sa vyberú dáta, ktoré sú relevantné pre pripravovanú dolovaciu úlohu. Pokiaľ sú dáta uložené v relačnej databáze, výber zahŕňa vybratie relevantných stĺpcov. Pri dátových skladoch predstavujú výber dimenzie.

4. **Transformácia dát:** Cieľom je transformovať alebo skonsolidovať dáta do formy umožňujúcej vykonanie dolovacej úlohy nad nimi. Transformácia zahrňuje obvykle operácie ako normalizácia, vyhladenie, agregovanie, generalizácia alebo konštrukcia atribútov. Normalizácia je vyžadovaná, pretože nenormalizované dáta môžu nepriaznivo ovplyvniť výsledok dolovacieho algoritmu. Vyhladenie sa používa pre odstránenie zašumených dát. Pri agregovaní sú detailné hodnoty dát agregované. Generalizácia poskytuje nahradenie zdrojových dát konceptmi z konceptuálnej hierarchie. Konštrukcia atribútov je používaná pri vytváraní nových atribútov, ktoré sú odvodené od iných atribútov a môžu priaznivo ovplyvniť výsledok dolovacej úlohy.
5. **Dolovanie dát:** Dolovanie dát je jadrom celého procesu získavania znalostí z databáz. Jeho cieľom je aplikácia určitej metódy, alebo vhodného algoritmu, extrahovať z dát vzory, prípadne vytvoriť model dát. Tieto techniky sa dajú rozdeliť do niekoľkých základných skupín, ktoré budú popísané v kapitole 2.3.
6. **Hodnotenie modelov a vzorov:** Cieľom je identifikovať skutočne zaujímavé vzory, resp. určiť, ktoré z vydolovaných dát sú zaujímavé a užitočné. Zaujímavé vzory musia byť pre ľudí pomerne ľahko pochopiteľné, platné na analyzovanej množine dát, potenciálne použiteľné, nové, alebo skrývať to čo je nezaujímavé [10].
7. **Prezentácia znalostí:** Jej cieľom je prezentovať výsledky dolovania koncovému užívateľovi pomocou dostupných techník vizualizácie a reprezentácie znalostí, teda zrozumiteľnou formou. Tento krok je zároveň posledným krokom celého procesu získavania znalostí z databáz.




**Obrázok 2.1 – Proces získavania znalostí z databáz (prevzaté zo [27])**

## 2.2 Druhy dát pre dolovanie

Získavať znalosti je možné takmer z akejkoľvek databáze či úložiska dát. Zdroje dát pre dolovanie môžeme rozdeliť podľa spôsobu uloženia dát podľa [5] na nasledujúce:

- **Relačná databáza:** Je to databázový systém, ktorý vychádza z relačného modelu dát a je založený na relačnej algebre. Relačná databáza pozostáva z tabuliek, z ktorých každá má unikátne meno. Každá tabuľka sa skladá z atribútov a obvykle uchováva veľké množstvo záznamov (riadkov). Každý záznam reprezentuje identifikovateľný objekt popísaný množinou atribútov. Atribúty obsahujú informácie o reláciách medzi jednotlivými záznamami v matematickom slova zmysle.
- **Dátový sklad:** Dátový sklad je väčšinou modelovaný multidimenzionálnou databázovou štruktúrou, kde každá dimenzia zodpovedá atribútu, alebo množine atribútov v schéme, a kde každá bunka uchováva agregované údaje.
- **Transakčná databáza:** Pozostáva zo súboru, kde každý záznam reprezentuje transakciu. Transakcia obvykle zahŕňa unikátne identifikačné číslo a zoznam položiek, ktoré tvoria transakciu. Transakčná databáza môže byť asociovaná s prídavnými tabuľkami, ktoré obsahujú doplňujúce údaje o transakciách.

Dáta pre dolovanie môžeme tiež rozdeliť podľa toho, aké dáta chceme vyhľadávať. Nasledujúce rozdelenie podľa [14] a [10].

- **Temporálne dáta:** Dáta sú uchovávané v klasickej relačnej alebo transakčnej databáze. Prevládajú v nich štruktúrované dáta, teda text a časová informácia s časovým razítkom.
- **Priestorové dáta:** Sú to dáta, ktoré sa vzťahujú k určitému priestoru, a ktoré tento priestor reprezentujú. Typickým príklad priestorových dát sú napr. satelitné alebo medicínske snímky a geografické mapy. Tie môžu byť uložené v rastrovej alebo vektorovej podobe, pričom k vektorovej informácii musí existovať popis a lokalizácia.
- **Multimediálne dáta:** Tieto dáta rozdeľujeme na 6 základných typov médií: text/dokument, obraz, video, audio, náčrty/poznámky a klasické štruktúrované dáta. Patria sem tiež rôzne heterogénne dáta. Viac o tomto druhu dát sa nachádza v kapitole 4.
- **Textové dáta:** Patria sem dokumenty typu XML, HTML a ďalšie potenciálne rôznorodé informácie a aj samotný Web.

## 2.3 Typy dolovacích úloh

Dolovanie dát tvorí základ celého procesu získavania znalostí z databáz. V predchádzajúcej kapitole sme si ukázali rôzne druhy dát a databáz. Zo všetkých spomínaných dátových zdrojov je možné dolovať dáta. Dokonca aj z webu, ktorý v sebe skrýva rozličné druhy dát a nespočetné množstvo informácií. Vďaka tejto rôznorodosti vzniklo postupom času viacero metód pre dolovanie. Existujú dve hlavné kategórie dolovacích úloh. Prvou z nich sú **deskriptívne** dolovacie úlohy, ktoré charakterizujú obecné vlastnosti analyzovaných dát v databáze. Typickým príkladom takejto úlohy je určenie spoločných nákupov pri analýze nákupného košíka. **Prediktívne** dolovacie úlohy, naproti tomu zo súčasných dát a ich vlastností, predpovedajú budúce chovanie [5] a [27].

Medzi konkrétne dolovacie úlohy patrí popis konceptu/triedy, asociačná analýza, klasifikácia a predikcia, zhluková analýza, analýza odľahlých objektov a analýza evolúcie. V nasledujúcom texte budú jednotlivé metódy stručne popísané podľa [5] a [27].

- **Popis konceptu/triedy**
  Princíp tejto metódy spočíva v asociovaní dát s určitou triedou alebo konceptom. Na základe týchto asociácií je možné získať stručný, presný a súhrnný popis daných tried, prípadne konceptov. Tieto popisy sa dajú získať dvoma spôsobmi. Prvým z nich je *charakterizácia*, ktorá sumarizuje obecné charakteristiky alebo črty cieľovej triedy. Druhým spôsobom získania popisu konceptu/triedy je *diskriminácia*. Tá porovnáva obecné charakteristiky cieľovej triedy s obecnými charakteristikami objektov jednej, prípadne viacerých odlišných tried.
- **Asociačná analýza**
  Asociačná metóda sa používa na odhaľovanie frekventovaných vzorov v dátach. Tieto vzory zobrazujú vzťahy medzi atribútmi a ich hodnotami, ktoré sa vyskytujú často spoločne. Asociačné pravidlo je implikáciou $X \rightarrow Y$, ktorá vraví, že ak prvok tabuľky vyhovuje podmienke X, bude s najväčšou pravdepodobnosťou vyhovovať aj podmienke Y. Asociačná analýza sa najčastejšie využíva pri dolovaní dát z transakčných databáz.
- **Klasifikácia a predikcia**
  Klasifikácia je proces hľadania modelu, ktorý popisuje a súčasne rozlišuje triedy dát za účelom jeho budúceho využitia pri predikcii triedy objektu, ktorého zaradenie nepoznáme. Celý tento proces pozostáva z troch krokov. Na základe *trénovania* je vytvorený klasifikačný model. *Testovanie* slúži k ohodnoteniu kvality vytvoreného modelu. Napokon *aplikácia* použije model pri klasifikácii nezaradeného objektu. Klasifikácia sa využíva pri predikcii diskrétnych hodnôt. Naproti tomu predikcia predpovedá spojité, teda numerické, hodnoty.
- **Zhluková analýza**
  Na rozdiel od klasifikácie a predikcie, ktoré analyzujú dátové objekty, ktorých priradenie do tried poznáme, zhlukovanie analyzuje objekty bez priradených tried. Klasifikácia je niekedy označovaná aj ako klasifikácia s učiteľom, teda hľadá koncept na základe príkladov. Naproti tomu zhlukovanie je možné označiť ako klasifikáciu bez učiteľa, pretože rozdelenie do skupín vykonáva len podľa vzájomne podobných vlastností. Cieľom zhlukovej analýzy je nájsť dáta, ktoré sú si čo najviac podobné vo vnútri zhluku, ale zároveň čo najrozdielnejšie medzi zhlukmi. Objekty sú zhlukované na základe maximalizácie podobností tej istej triedy a minimalizácie podobností medzi triedami. Zhluková analýza bude podrobne popísaná v kapitole 3.
- **Analýza odľahlých hodnôt**
  Ako odľahlé môžeme definovať tie hodnoty, ktoré sa výrazne odlišujú od ostatných často sa vyskytujúcich, alebo podobných hodnôt. Dáta nadobúdajúce tieto hodnoty sú vo väčšine prípadov považované za šum, ktorý je potrebné odstrániť. Avšak v niektorých prípadoch obsahujú tieto dáta cenné alebo ojedinelé informácie. Analýza odľahlých hodnôt sa zaoberá dolovaním týchto odľahlých dát.
- **Analýza evolúcie**
  Tento typ dolovacej úlohy popisuje a modeluje pravidelnosti a trendy pri objektoch, ktorých správanie sa mení v čase.

# 3 Zhluková analýza

Zhluková analýza alebo skrátene zhlukovanie, je jednou z dolovacích metód a je označovaná aj ako klasifikácia bez učiteľa. Cieľom je zaradiť neznáme dáta do tried, ktoré však nepoznáme. Na základe podobností a rozdielov sa hľadá optimálne rozdelenie dát do tried, ktoré sa nazývajú aj *zhluky*. Jednotlivé zhluky obsahujú dáta, ktoré sú si veľmi podobné, ale sú čo najviac odlišné od dát obsiahnutých v iných zhlukoch. Zhlukovanie sa využíva typicky podľa [17] ako samostatný nástroj pre získanie náhľadu do distribúcie dát, alebo ako jeden z krokov predspracovania pre iné algoritmy. V dnešnej dobe je vývoj zhlukovacích algoritmov zameraný najmä na škálovateľnosť, efektivitu, multidimenzionálne dáta a zmiešané numerické a kategorické dáta v rozsiahlych databázových systémoch.

Na zhlukovacie metódy sú kladené nasledovné požiadavky, prevzaté z [27].

- **Škálovateľnosť:** Vybratá metóda musí byť schopná analyzovať a roztriediť aj rozsiahle databáze.
- **Schopnosť spracovať rôzne druhy atribútov:** Numerické atribúty dokáže spracovať takmer každý algoritmus, avšak je potrebné zhlukovať aj dáta iných typov.
- **Tvorba zhlukov rôznych tvarov:** Väčšina metód vytvára zhluky guľovitého tvaru, ktoré spravidla nezodpovedajú ich skutočnému tvaru.
- **Minimálne požiadavky na znalosť problému pri určovaní parametrov:** Algoritmus spĺňujúci túto požiadavku by mal byť schopný zhlukovať dáta bez zadávania vstupných parametrov.
- **Odolnosť voči šumu:** Schopnosť zhlukovať dáta obsahujúce chybné, neznáme alebo chýbajúce dáta.
- **Necitlivosť voči poradiu vstupných záznamov:** Poradie uložených záznamov v databáze by nemalo mať vplyv na výsledok zhlukovacieho algoritmu.
- **Schopnosť spracovať vysokodimenzionálne dáta:** Bežné zhlukovacie metódy dokážu spracovať dátové položky s maximálne dvomi alebo tromi atribútmi. Zaujímavejšie sú metódy schopné spracovať položky s veľkým počtom atribútov.
- **Schopnosť zhlukovať na základe obmedzení:** Aplikácie často vyžadujú zhlukovanie na základe rôznych obmedzení. Cieľom je nájsť zhluky na základe týchto obmedzení.
- **Interpretovateľnosť a použiteľnosť zhlukov:** Výsledky zhlukovej analýzy musia byť zrozumiteľne vizualizované užívateľovi.

## 3.1 Typy dát pri zhlukovaní

Predpokladajme, že zhlukovanie bude prebiehať nad množinou dát, ktorá obsahuje $n$ objektov a tie budú popísané pomocou $p$ atribútov. Najbežnejšie zhlukovacie algoritmy využívajú jednu z nasledujúcich dátových štruktúr. Prevzaté z [5].

- **Dátová matica:** Reprezentuje dátové množiny. Jej rozmery sú $n \times p$, kde $n$ je počet objektov a $p$ je počet atribútov, teda dimenzií.

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix} \tag{3.1}$$

- **Podobnostná matica:** Uchováva vzdialenosti pre všetky dvojice objektov. Jej rozmery sú $n \times n$, kde $n$ udáva počet objektov.

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix} \tag{3.2}$$

  V podobnostnej matici $d(i,j)$ značí vzdialenosť medzi objektmi $i$ a $j$. Ak je $d(i,j) = d(j,i)$ a $d(i,i) = 0$, dostávame trojuholníkovú maticu (3.2).

Vzdialenosť jednotlivých objektov je závislá na type dát atribútov. V nasledujúcich podkapitolách budú popísané jednotlivé typy atribútov podľa [5] a [27].

### 3.1.1 Intervalové premenné

Intervalové premenné sú spojité hodnoty s približne lineárnym rozložením. Typickým príkladom je napr. hmotnosť alebo výška osôb. Pri určovaní vzdialeností vzniká problém s použitými jednotkami. Hmotnosť môže byť daná v kilogramoch ale aj v librách. Tieto nezrovnalosti vedú k odlišným výsledkom zhlukovania. Preto zavádzame pojem *štandardizácia*. Tá prevedie hodnoty na bezjednotkové premenné. Štandardizáciu je možné vykonať nasledujúcim spôsobom [17].

1. Výpočet strednej odchýlky

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|) \tag{3.3}$$

   kde $x_{1f}, \dots, x_{nf}$ sú hodnoty premennej $f$ a $m_f$ je stredná hodnota $f$.

2. Výpočet štandardizovanej hodnoty (z-score)

$$z_{if} = \frac{x_{if} - m_f}{s_f} \tag{3.4}$$

Po štandardizácii je potrebné vypočítať podobnosť alebo odlišnosť objektov, ktorá je v prípade intervalových premenných vyjadrená pomocou vzdialenosti. Každá vzdialenostná funkcia musí splňovať nasledujúce kritéria.

1. $d(i,j) \geq 0$ : Vzdialenosť je vždy nezáporné číslo.
2. $d(i,i) = 0$ : Vzdialenosť objektu od seba samého je 0.
3. $d(i,j) = d(j,i)$ : Vzdialenosť je symetrická funkcia.
4. $d(i,j) \leq d(i,h) + d(h,j)$ : Trojuholníková nerovnosť. Vzdialenosť medzi objektmi $i$ a $j$ nemôže byť väčšia ako obchádzka cez iný objekt $h$.

Najpopulárnejšie sú nasledujúce vzdialenostné funkcie.

- **Euklidovská vzdialenosť:**

$$d(i,j) = \sqrt{(|x_{i_1} - x_{j_1}|^2 + |x_{i_2} - x_{j_2}|^2 + \dots + |x_{i_p} - x_{j_p}|^2)} \tag{3.5}$$

  kde $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ a $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ sú dva p-dimenzionálne objekty.

- **Manhattonská vzdialenosť:**

$$d(i,j) = |x_{i_1} - x_{j_1}| + |x_{i_2} - x_{j_2}| + \dots + |x_{i_p} - x_{j_p}| \tag{3.6}$$

- **Minkowského vzdialenosť:**

$$d(i,j) = \sqrt[q]{(|x_{i_1} - x_{j_1}|^q + |x_{i_2} - x_{j_2}|^q + \dots + |x_{i_p} - x_{j_p}|^q)} \tag{3.7}$$

### 3.1.2 Binárne premenné

Binárne premenné sú premenné, ktoré môžu nadobúdať len dve hodnoty, 1 alebo 0. 1 značí, že daný objekt má vybranú hodnotu a 0 značí, že objekt túto hodnotu nemá. Typickým príkladom je napr. potreba dioptrií.

K výpočtu koeficientu zhody binárnych premenných sa využívajú súčty, ktoré značia počet premenných, v ktorých sa dva objekty zhodujú, prípadne líšia. Tieto súčty sa zadávajú do tzv. kontingenčnej tabuľky. Tabuľka 3.1 ukazuje príklad takejto tabuľky, kde napr. $a$ je počet premenných rovných 1 pre oba objekty $i$ a $j$, $c$ je počet premenných rovných 0 pre objekt $i$ a rovných 1 pre objekt $j$ a celkový počet premenných je $p$.

| | Objekt i | | |
|---|---|---|---|
| Objekt j | 1 | 0 | sum |
| 1 | a | b | a+b |
| 0 | c | d | c+d |
| sum | a+c | b+d | p |

Tabuľka 3.1: Kontingenčná tabuľka pre binárne premenné (prevzaté z [5])

Binárne premenné sa delia na *symetrické* a *asymetrické*.

- **Symetrické:** Obidva stavy premennej majú rovnakú váhu. Príkladom je pohlavie (muž, žena). Pre symetrické binárne premenné sa používa *jednoduchý koeficient zhody*:

$$d(i,j) = \frac{b+c}{a+b+c+d} \tag{3.8}$$

- **Asymetrické:** Jeden zo stavov premennej má väčšiu váhu ako druhý. Príkladom je zasiahnutie povodňami, kde hľadáme obce, ktoré boli povodňami postihnuté. Pre výpočet takýchto premenných sa používa *Jaccardov koeficient zhody*:

$$d(i,j) = \frac{b+c}{a+b+c} \tag{3.9}$$

### 3.1.3 Nominálne, ordinálne a pomerové premenné

- **Nominálne premenné:** Sú zovšeobecnením binárnych premenných. Môžu nadobúdať viac ako dve hodnoty, napr. farba. Počet stavov môžeme označiť ako M. Vzdialenosť sa dá opäť vypočítať na základe *jednoduchého koeficientu zhody*:

$$d(i,j) = \frac{p-m}{p} \tag{3.10}$$

  kde $m$ je počet premenných, v ktorých majú objekty $i$ a $j$ rovnakú hodnotu a $p$ je celkový počet premenných oboch objektov.

- **Ordinálne premenné:** Sú to vlastne nominálne hodnoty, ktorých M stavov je zoradených do určitej postupnosti. Typickým príkladom je vek, ktorý nadobúda hodnoty nízky, stredný a vysoký. Tieto hodnoty môžeme nahradiť hodnotami $1 \dots M_f$. Objekt $x_i$ má určitú hodnotu premennej na pozícii $f$ a túto hodnotu nazývame $r_{if}$. Hodnoty $r_{if}$ potom prepočítame do intervalu <0,1> pomocou nasledujúcej transformácie:

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{3.11}$$

Podobnosť môže byť následne vypočítaná podľa jednej z vyššie uvedených vzdialenostných funkcií, pretože s premennými vyjadrenými pomocou $z_{if}$ môžeme zachádzať ako s intervalovými premennými.

- **Pomerové premenné:** Vyjadrujú rozloženie hodnôt na inej ako lineárnej stupnici, napr. na exponenciálnej. Existujú tri metódy ako s týmito premennými zaobchádzať. Môžeme ich spracovávať ako intervalové premenné, avšak tento prístup je nevhodný, pretože ich stupnica môže byť skreslená. Druhým spôsobom je pristupovať k nim ako k ordinálnym hodnotám a ich rozsah spracovávať ako intervalovú premennú. Poslednou metódou je aplikácia určitej transformácie. V prípade, že sú premenné $x_{if}$ v exponenciálnej stupnici, môžeme použiť logaritmickú transformáciu, čím následne dostaneme $y_{if}$, ktorý spracovávame ako intervalovú premennú. Túto transformáciu popisuje rovnica 3.12.

$$y_{if} = \log x_{if} \tag{3.12}$$

### 3.1.4 Premenné rôzneho typu

V reálnych databázach sú objekty popísané premennými rôznych typov. Vo všeobecnosti môže databáza obsahovať všetky vyššie popísané typy premenných. Analýza môže byť v takomto prípade vykonaná vždy len nad jedným z typov, avšak toto riešenie je v praxi nepoužiteľné, pretože výsledné zhluky budú pre každý typ premennej rôzne. Preto sa využíva metóda, kedy sú všetky atribúty spracované naraz pomocou jednej vzdialenostnej funkcie v tvare:

$$d(i,j) = \frac{\sum_{f=1}^{p} \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^{p} \delta_{ij}^{(f)}} \tag{3.13}$$

kde $p$ je počet premenných a indikátor $\delta_{ij}^{f} = 0$ ak $x_{if}$ alebo $x_{jf}$ chýbajú, prípadne ak $x_{if} = x_{jf} = 0$ a premenná $f$ je asymetricky binárna. V ostatných prípadoch je indikátor rovný 1.

Prírastok $d_{ij}^{f}$ premennej $f$ ku vzdialenosti je následne spočítaný podľa typu premennej $f$. Pri všetkých typoch premenných sa uplatňujú princípy vysvetlené v predchádzajúcich podkapitolách. Pri intervalových premenných sa však prírastok počíta podľa nasledujúceho vzťahu:

$$d_{ij}^{f} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}} \tag{3.14}$$

kde $h$ prechádza cez všetky neprázdne dátové objekty premennej $f$. To znamená, že sa spočíta minimálna a maximálna hodnota danej premennej $f$.

## 3.2 Zhlukovacie metódy

V súčasnosti sa využíva veľké množstvo rôznych zhlukovacích algoritmov. Každá dolovacia úloha však vyžaduje rozdielny prístup a tým aj iný druh zhlukovacej metódy. Pri získavaní znalostí z multimediálnych databáz sa využívajú najmä metódy BIRCH, DBSCAN alebo DENCLUE., ktoré budú popísané v ďalších kapitolách. Medzi najzákladnejší algoritmus zhlukovania patrí k-means, ktorý je taktiež popísaný nižšie.

### 3.2.1 Rozdelenie zhlukovacích metód

Existuje viacero metód zhlukovej analýzy. V nasledujúcom texte budú tieto metódy rozdelené do základných skupín podľa [5] a [27]. Iné rozdelenie je možné nájsť v [26].

- **Metódy založené na rozdeľovaní:** Vstupné dáta, ktoré obsahujú $n$ objektov, sú rozdeľované do $k$ zhlukov. Parameter $k$ je vopred známy a platí $k \leq n$. Musí platiť, že každý objekt patrí do práve jedného zhluku a každý zhluk obsahuje aspoň jeden objekt. Na začiatku sa vytvorí $k$ zluhkov a do nich sa zatriedia objekty. Toto zaradenie sa potom, technikou nazývanou *iteratívna relokácia*, vylepšuje presúvaním objektov medzi zhlukmi. Kritérií na posúdenie kvality rozdelenia do zhlukov je viacero. Hlavným z nich je, že objekty v rovnakom zhluku sú vzdialenostne blízko a zároveň objekty rôznych zhlukov sú od seba ďaleko. Táto heuristika funguje veľmi dobre pri vytváraní zhlukov guľovitého tvaru v malých až stredne veľkých databázach. Najvýznamnejšími metódami sú *k-means*, popísaný v kapitole 3.2.2 a *k-medoids*, ktorý je mu veľmi podobný.
- **Hierarchické metódy:** Vytvárajú hierarchickú dekompozíciu danej množiny vstupných dát. V závislosti na vytváraní tejto dekompozície sa delia na *aglomeratívne* a *rozdeľovacie*. Aglomeratívny prístup pracuje na princípe *zdola-nahor*. Na začiatku každý objekt tvorí zvlášť zhluk a tie sa postupne spájajú do väčších zhlukov, až pokým nie je splnená ukončujúca podmienka. Rozdeľovací prístup sa nazýva aj prístup *zhora-nadol*. Tento prístup naopak, vytvorí na počiatku jediný zhluk, do ktorého patria všetky objekty a ten potom delí na menšie zhluky. Nevýhodou týchto metód je neschopnosť vrátiť sa na predchádzajúcu úroveň hierarchie. Napriek tomu sú hojne používané, vďaka svojej nízkej výpočtovej cene. Známymi predstaviteľmi týchto algoritmov sú AGNES, ROCK, CHAMELEON a najmä BIRCH popísaný v kapitole 3.2.3.
- **Metódy založené na hustote:** Väčšina metód používa na určovanie zhlukov vzdialenosť medzi objektmi. To má za následok vytváranie zhlukov výlučne guľovitého tvaru. Tento problém riešia práve metódy založené na hustote, ktoré sú založené na zväčšovaní zhlukov až kým nie je dosiahnutá dopredu daná hranica hustoty okolia. Hustota v tomto zmysle znamená počet bodov v užívateľom zadanom okolí vybraného bodu. Tieto metódy sa využívajú pri odstraňovaní šumu a vytváraní zhlukov nepravidelných tvarov. Ich predstaviteľmi sú DBSCAN, DENCLUE, OPTICS atď. Prvé dva z menovaných sú popísané v kapitolách 3.2.4 resp. 3.2.5.
- **Metódy založené na mriežke:** Tieto algoritmy rozdeľujú priestor objektov na konečný počet buniek, vytvárajúcich mriežku. Všetky zhlukovacie výpočty potom prebiehajú na tejto mriežke, nazývanej tiež *kvantizovaný priestor*. Hlavnou výhodou tohto prístupu je rýchlosť výpočtov, pretože algoritmy nie sú závislé na počte objektov ale len na počte buniek v jednotlivých dimenziách mriežky. Hlavnými algoritmami používajúcimi mriežku sú STING, ktorý využíva statické informácie uložené v mriežke a WaveCluster, ktorý používa tzv. vlnovú transformáciu.
- **Metódy založené na modeli:** Tieto metódy sa snažia vytvárať zhluky, ktoré by čo najvernejšie zodpovedali určitému matematickému modelu. Predpokladá sa, že dáta boli generované na základe zloženej pravdepodobnostnej distribučnej funkcie. Obvykle berú do úvahy aj šum a odľahlé hodnoty, a teda bývajú pomerne robustné. EM algoritmus vykonáva analýzu založenú na štatistickom modelovaní. COBWEB je algoritmus s učením, ktorý využíva pravdepodobnostnú analýzu a konceptuálne zhlukovanie. SOM je metóda neurónových sietí, ktorá mapuje viacdimenzionálne dáta do 2-D alebo 3-D funkčných máp, vhodných aj pre vizualizáciu dát.

- **Zhlukovanie viacdimenzionálnych dát:** Viacdimenzionálne dáta sú charakterizované objektmi s obrovským počtom atribútov. Typickým príkladom je napr. DNA. Zhlukovanie takýchto dát býva neľahkou úlohou, pretože so zvyšujúcim sa počtom dimenzií strácajú niektoré z nich svoju relevantnosť. Tým sa zmenšuje hustota dát a určovanie vzdialeností medzi nimi stráca svoj význam. Preto je potrebné vytvoriť algoritmy, ktoré dokážu vstupné dáta špeciálne pripraviť na zhlukovanie. Metódy CLIQUE a PROCLUS patria medzi takzvané *metódy zhlukovania podpriestoru*. Odlišné podpriestory totiž obvykle obsahujú iné zhluky. Problémom však býva nájdenie toho správneho podpriestoru.
- **Metódy založené na obmedzení:** Tento prístup umožňuje užívateľovi zadať požadovaný výsledok zhlukovej analýzy, a tým ovplyvniť samotný výsledok zhlukovania. Užívateľ môže zadávať rôzne typy takýchto obmedzení.

### 3.2.2 K-means

K-means patrí medzi najznámejšie a najpopulárnejšie algoritmy zhlukovej analýzy. Patrí do skupiny algoritmov založených na rozdeľovaní. K vytvoreniu popisu tohto algoritmu bola využitá kniha [26]. K-means hľadá, využíva iteratívnu optimalizačnú funkciu, spadajúcu do kategórie *„hill-climbing"* algoritmov. Pomocou nej sa objekty rozdeľujú do tried v jednotlivých krokoch iterácie, pokiaľ nie je splnená funkcia slúžiaca ako kritérium. Kritériom býva určené tzv. štvorcom chyby popísanom ako:

(3.15)

$$J_s(\Gamma, \mathrm{M}) = \sum_{i=1}^{K}\sum_{j=1}^{N}\gamma_{ij}\left\|x_j - m_i\right\|^2 = \sum_{i=1}^{K}\sum_{j=1}^{N}\gamma_{ij}\left(x_j - m_i\right)^T\left(x_j - m_i\right)$$

kde $\Gamma = \{\gamma_{ij}\}$ je podobnostná matica, $\gamma_{ij} = 1$, ak $x_j \in$ zhluku $i$, inak $\gamma_{ij} = 0$, pričom musí platiť $\sum_{i=1}^{K}\gamma_{ij} = 1\ \forall j$. $M = [m_1, \dots, m_k]$ je prototypová, alebo priemerná matica a $m_i = \frac{1}{N_i}\sum_{j=1}^{N}\gamma_{ij}x_j$ je jednoduchý priemer zhluku $i$, ktorý obsahuje $N_i$ objektov.

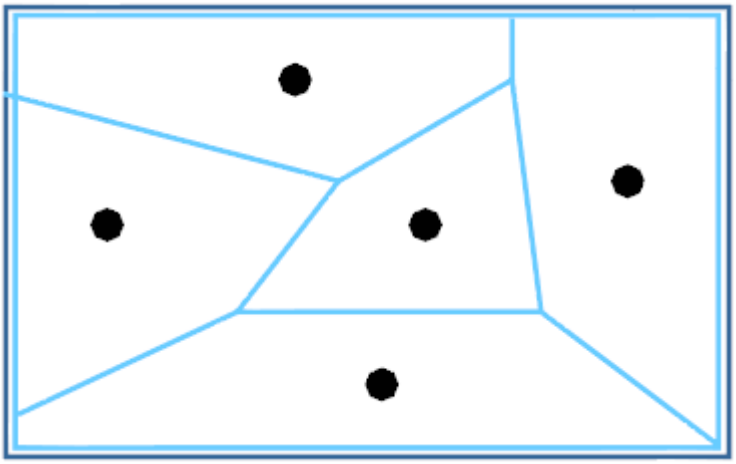
Samotný algoritmus *k-means* sa dá popísať nasledovným spôsobom podľa [26]:

1. Inicializuj $k$ zhlukov náhodne, alebo na základe nejakej funkcie a vypočítaj prototypovú zhlukovú maticu $M = [m_1, \dots, m_k]$.
2. Priraď každý objekt zo vstupných dát do najbližšieho zhluku $C_l$, napr.:
   $x_j \in C_l$, if $\left\|x_j - m_l\right\| < \left\|x_j - m_i\right\|$, for $j = 1, \dots, N$, $i \neq l$, and $i = 1, \dots, K$ (3.16)
3. Prepočítaj prototypovú zhlukovú maticu podľa súčasných zhlukov:

   (3.17)

   $$m_i = \frac{1}{N_i}\sum_{x_j \in C_i} x_j$$
4. Opakuj kroky 2 a 3 pokiaľ nedôjde ku zhode výsledných zhlukov po dvoch po sebe idúcich iteráciách.

V kroku 2 sa využíva pravidlo najbližšieho suseda. Vstupný priestor je rozdelený na *voronoi*, ktorého ukážka je na obrázku 3.1. Popísaný algoritmus sa dá označiť ako dávkový režim učenia, pretože k aktualizácii prototypov dôjde až po spracovaní všetkých objektov.



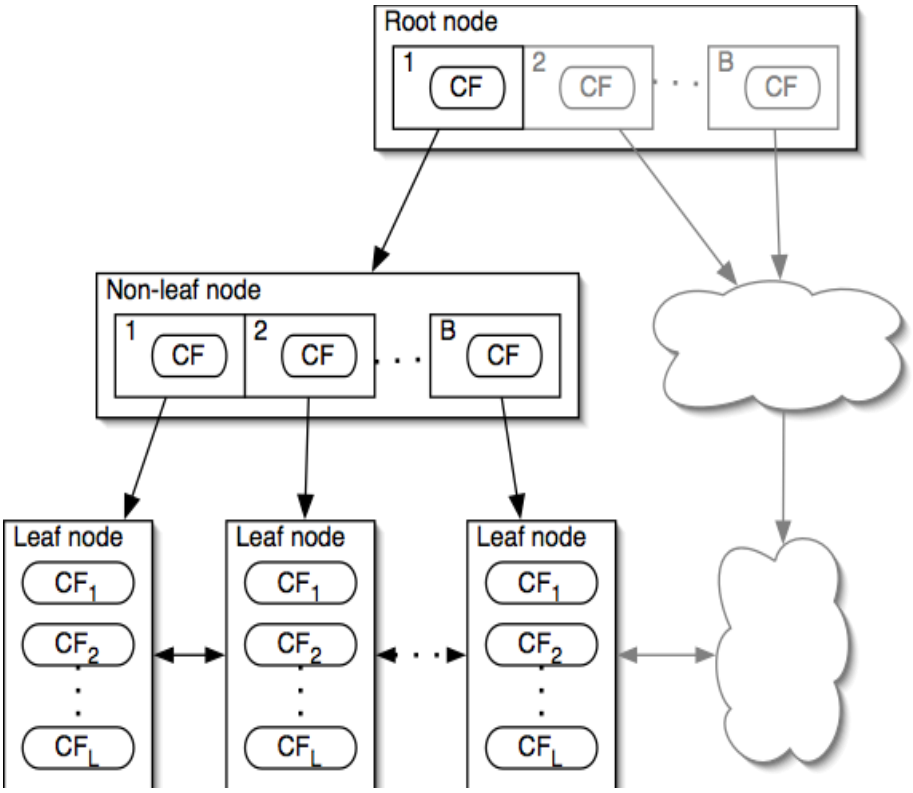
**Obrázok 3.1 – Ukážka voronoi (prevzaté z [14])**

*K-means* sa vďaka svojej jednoduchosti a rýchlosti, ktorá je pri veľkom počte objektov lineárna, využíva na riešenie mnohých praktických problémov. Má však aj svoje nevýhody. Najvýraznejšou je, že algoritmus nemusí nájsť optimálny výsledok, ale môže uviaznuť v lokálnom extréme. Ďalšou nevýhodou je nutnosť počiatočného udania počtu požadovaných zhlukov.

### 3.2.3 BIRCH

Algoritmus *BIRCH* (Balanced Iterative Reducing and Clustering using Hierarchies), popísaný podľa [28], sa používa pri zhlukovej analýze veľkých databázových systémov, vďaka svojej rýchlosti a schopnosti efektívne využívať pamäťový priestor. Súhrne môžeme povedať, že *BIRCH* je lineárne škálovateľný. Patrí medzi hierarchické algoritmy. Medzi jeho ďalšie prednosti patrí schopnosť odstrániť zašumené dáta a získať kvalitné výsledky zhlukovania už po prvom spracovaní množiny dát. Často je označovaný aj ako lokálny, pretože zhlukuje objekty bez toho aby mal prístup k ostaným objektom.

Významnú úlohu v tomto algoritme zohráva takzvaná *„Clustering feature“* (CF) a *CF-Tree* (CF-Strom). *Clustering feature* je trojica, ktorá sumarizuje informácie o zhluku. Je efektívna, pretože neobsahuje všetky body daného zhluku a zároveň má dostatok informácií k vykonaniu zhlukovania. Môžeme ju popísať ako: $CF_i = (N_i, LS_i, SS_i)$, kde $N$ je počet objektov v zhluku, $LS$ je suma $N$ dátových bodov, $SS$ je kvadratická suma $N$ dátových bodov a $i = 1, 2, \dots, M$ ($M$ je celkový počet objektov). *CF-Tree* je výškovo vyvážený strom s dvomi parametrami: $B$ - faktor vetvenia a $T$ - prah. Ukážka takéhoto stromu je na obrázku 3.2. Veľkosť uzla určuje dimenzionalita dát a vstupný parameter $P$, predstavujúci veľkosť stránky.




**Obrázok 3.2 – Ukážka CF-Tree (prevzaté z [7])**

*CF-Tree* je vytváraný dynamicky, podľa dát, ktoré sú do neho vkladané. Vkladanie sa dá podľa [28] popísať nasledovným algoritmom.

1. **Identifikácia vhodného listu:** Zostupne a rekurzívne prehľadávaj strom a vyberaj najbližšieho potomka na základe vybranej vzdialenostnej funkcie.
2. **Modifikácia listu:** Vyskúšaj či môže list absorbovať uzol bez toho aby porušil nastavený prah. Ak už nie je v liste miesto, rozdeľ ho na dve časti.
3. **Modifikácia cesty:** Ak bol pridaný nový objekt, uprav *CF* všetkých predkov.

Algoritmus *BIRCH* sa dá podľa [28] rozdeliť na 4 fázy:

1. **Fáza 1:** Zadaj počiatočný prah a vlož dátové objekty do stromu. Ak je vyčerpaná pamäť, zvýš prah a vytvor nové, menšie stromy preukladaním hodnôt zo starého stromu a pridaním nových. Pri vytváraní menších stromov sa odstráni časť odľahlých objektov. Dôležitým faktorom je určenie správnej hodnoty počiatočného prahu.
2. **Fáza 2 (voliteľná):** Fáza 3 potrebuje určitý počet zhlukov, ktorý nemusí byť zhodný s výsledkom Fáze 1. Preto zmenši strom na požadovanú veľkosť odstránením ďalších odľahlých premenných, prípadne zoskupením zhlukov.
3. **Fáza 3:** Použi listové uzly stromu ako vstup štandardného zhlukovacieho algoritmu (napr. *k-means*). Fáza 1 zredukovala veľkosť vstupných dát dostatočne na to, aby mohol celý výpočet štandardného algoritmu prebehnúť v pamäti. Výsledok tejto fáze závisí od poradia vstupných dát.

4. **Fáza 4 (voliteľná):** Znova priraď každý objekt zhluku s najbližším ťažiskom. Táto fáza prerozdelí objekty medzi zhluky omnoho presnejšie ako pôvodne. Opakuj pre spresnenie zhlukov. Nezáleží na počte iterácií, vždy konverguje.

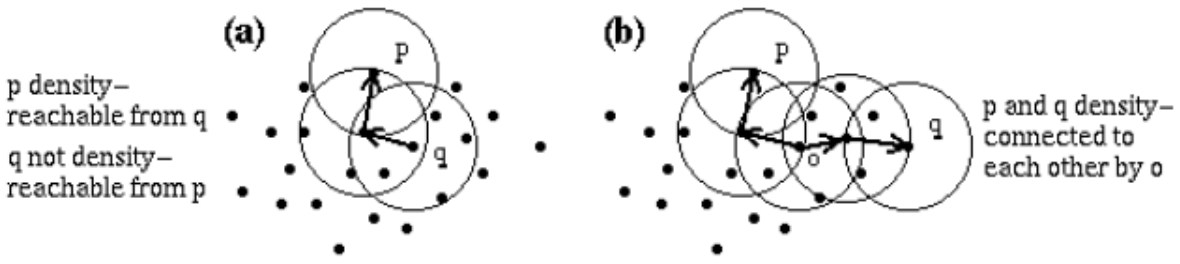
*BIRCH* poskytuje rýchlejšie zhlukovanie na veľkom objeme dát ako väčšina algoritmov. Jeho hlavnou výhodou je, že pre postačujúci výsledok stačí spracovať dáta len raz a efektívne sa zbavuje odľahlých hodnôt. Je vhodný pre analýzu obrázkov, obchodných praktík alebo bioinformatických údajov.

### 3.2.4 DBSCAN

*DBSCAN* (Density-Based Spatial Clustering of Applications with Noise) je typickým zástupcom metód založených na hustote. Už z názvu je možné vyčítať jeho hlavné ciele a prednosti, ktorými sú zhlukovanie priestorových a zašumených dát. *Hustota* značí v zhlukovej analýze počet bodov v rámci určitej oblasti celkového priestoru. *DBSCAN* pracuje na princípe postupného pridávania bodov s dostatočnou hustotou do zhlukov. Nasledujúci popis tejto metódy vychádza z [1].

V ďalšom texte budeme používať databázu $D$ s $k$ dimenziami. Hlavnou ideou je, že okolie každého bodu musí dosiahnuť určitý prah. Tvar okolia určuje vybratá vzdialenostná funkcia pre dva body, $p$ a $q$, označovaná $dist(p,q)$. Pre správne pochopenie algoritmu je potrebné si najskôr definovať nasledujúce pojmy.

- $\varepsilon$-okolie bodu $p$ je definované ako $N_\varepsilon(p) = \{q \in D | dist(p,q) \leq \varepsilon\}$. Ak okolie obsahuje určité minimum počtu bodov, označované ako $MinPts$, potom sa jedná o okolie jadrového objektu. V opačnom prípade je bod $p$ označený ako hraničný objekt.
- Bod $p$ je priamo dosiahnuteľný na základe hustoty z bodu $q$, ak $p \in N_\varepsilon(q)$ a zároveň $|N_\varepsilon(q)| \geq MinPts$. Táto podmienka nie je symetrická, ako vidieť na obrázku 3.3, pretože neplatí pre hraničné objekty.
- Bod $p$ je dosiahnuteľný na základe hustoty z bodu $q$, ak existuje postupnosť bodov $p_1, \ldots, p_n$, $p_1 = q$ a $p_n = p$ taká, že bod $p_{i+1}$ je priamo dosiahnuteľný z bodu $p_i$, pre $i = \{1, \ldots, n\}$.
- Bod $p$ je spojený na základe hustoty s bodom $q$, ak existuje bod $o$, z ktorého sú oba body dosiahnuteľné na základe hustoty. Príklad na obrázku 3.3.
- Zhluk $C$ je neprázdna podmnožina vstupnej databáze $D$ spĺňajúca nasledovné podmienky:
  1. $\forall\, p,q$: if $p \in C$ and $q$ je dosiahnuteľné na základe hustoty z $p$, then $q \in C$
  2. $\forall\, p,q \in C$: $p$ je spojený na základe hustoty s $q$, s ohľadom na $\varepsilon$ a $MinPts$
- Nech $C_1, \ldots, C_k$ sú zhluky vstupnej databáze $D$ s parametrami $\varepsilon_i$ a $MinPts_i, i = 1, \ldots, k$. Šum je potom definovaný ako $noise = \{p \in D | \forall i: p \notin C_i\}$.




**Obrázok 3.3 – Dosiahnuteľnosť a spojenie na základe hustoty (prevzaté z [1])**

Zjednodušený algoritmus *DBSCAN* popísaný podľa [5], pričom detailnejší popis algoritmu spolu so pseudokódom je možné nájsť v [1]:

1. Najskôr urči hodnoty $MinPts$ a $\varepsilon$.
2. Následne vyhľadávaj zhluky kontrolovaním $\varepsilon$-okolí všetkých bodov vstupnej databáze $D$. Ak $\varepsilon$-okolie bodu $p$ obsahuje viac ako $MinPts$ bodov, vytvor nový zhluk s bodom $p$ ako jadrovým objektom.
3. Iteratívne zoskupuj objekty do zhluku, ak sú priamo dosiahnuteľné na základe hustoty z jadrového objektu. Tento proces môže zahŕňať zlúčenie niekoľkých zhlukov, ktorých jadrové objekty sú dosiahnuteľné na základe hustoty.
4. Ukonči zhlukovanie ak už nie je možné pridať žiaden objekt do žiadneho zhluku.

Určovanie parametrov $\varepsilon$ a $MinPts$ je obvykle veľmi náročné a je potrebné skúšať viacero možností. Ak zmenšíme parameter $\varepsilon$, musíme zmenšiť aj $MinPts$. Algoritmus má pri použití $R^* - Stromu$ podľa [1] logaritmickú časovú zložitosť a je efektívnejší ako algoritmus CLARANS. Rovnako vykazuje dobré hodnoty pri odstraňovaní odľahlých hodnôt a pri zhlukovaní veľkých objemov dát. Obmedzujúcim faktorom tohto algoritmu je jeho nepoužiteľnosť na viacdimenzionálne dáta.

### 3.2.5 DENCLUE

Hlavnou myšlienkou *DENCLUE* (DENsity based CLUstEring) je analytické modelovanie hustoty všetkých bodov ako súčtu vplyvu na ostatné dané body. Vznikol pre potreby zhlukovania zašumených dát nachádzajúcich sa vo veľkých databázach. Príkladom je databáza obsahujúca 2D alebo 3D multimediálne dáta. *DENCLUE* zovšeobecňuje rôzne zhlukovacie metódy a pri použití rôznych parametrov je jeho výsledok rovnaký ako výsledky týchto metód.

Metóda využíva tzv. *funkcie vplyvu a hustoty*, ktoré matematicky popisujú vplyv bodov na svoje okolie. Pre ďalšie pochopenie algoritmu je potrebné si definovať nasledujúce pojmy podľa [6]. Pri definovaní pojmov sa využíva $k$-dimenzionálna databáza $D$.

- **Funkcia vplyvu:** Funkcia udávajúca vplyv bodu $y \in D$ na bod $x$ sa dá vyjadriť ako: $f_B^y(x) = f_B(x, y)$, kde $f_B$ musí byť symetrická a reflexívna. Pre jednoduchosť sa používa Euklidovská vzdialenostná funkcia, avšak často sa môžeme stretnúť aj s nasledovnými funkciami:
  - *Obdĺžniková funkcia vplyvu*

$$f_{Square}(x, y) = \begin{cases} 0, & d(x, y) > \sigma \\ 1, & d(x, y) < \sigma \end{cases} \tag{3.18}$$

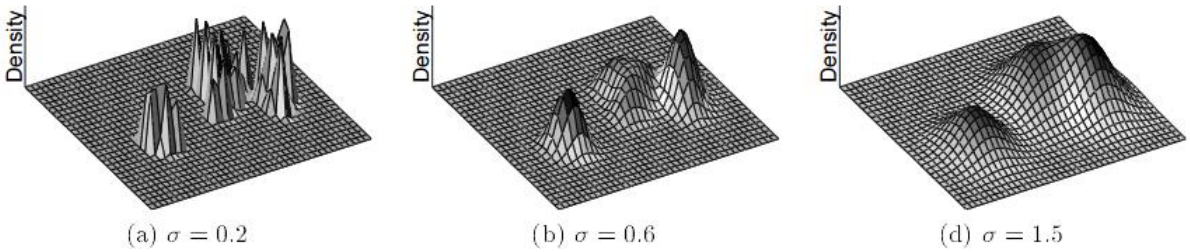
  - *Gaussovská funkcia vplyvu*

$$f_{Gauss}(x, y) = e^{-\frac{d(x, x_i)^2}{2\sigma^2}} \tag{3.19}$$

- **Funkcia hustoty:** Je definovaná ako súčet vplyvov všetkých bodov z množiny $D$ na daný bod. Z Gaussovskej funkcie vplyvu vyplýva funkcia hustoty:

$$f_{Gauss}^D(x) = \sum_{i=1}^{N} e^{-\frac{d(x, x_i)^2}{2\sigma^2}} \tag{3.20}$$

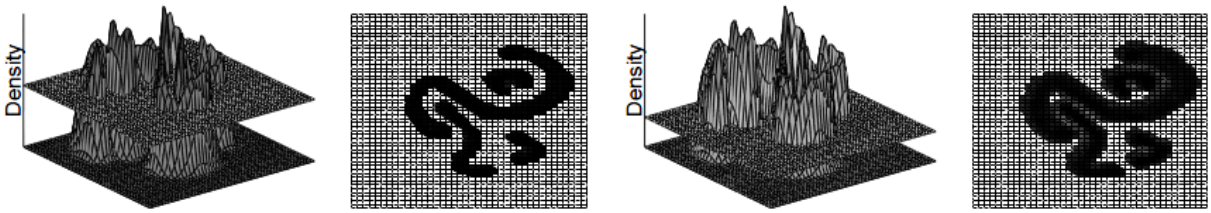
- **Atraktor hustoty:** Bod $x^* \in F^d$ je nazývaný *atraktor hustoty* pre danú funkciu vplyvu, ak je lokálnym maximom tejto funkcie. Bod $x$ je priťahovaný k atraktoru hustoty, ak $\exists k \in N:$ $d(x^k, x^*) \leq \epsilon$. Zároveň musí platiť, že pre každé $x^{i-1}$ je sklon funkcie hustoty ku $x^i$ v rozmedzí $\langle 0, \ldots, k \rangle$
- **Zhluk definovaný centrálnym bodom:** Aby sme mohli adekvátne rozdeľovať body do zhlukov, potrebujeme definovať parameter $\xi$. Tento parameter určuje počet bodov potrebných k vylúčeniu šumu. Zhluk definovaný centrálnym bodom pre atraktor $x^*$ je podmnožina $C \subseteq D$, kde všetky $x \in C$ sú priťahované ku $x^*$ a $f_B^D(x^*) \geq \xi$. Príklady takýchto zhlukov sú na obrázku 3.4.




**Obrázok 3.4 – Príklady zhlukov definovaných centrálnym bodom pre rôzne $\sigma$ (prevzaté z [6])**

- **Zhluk ľubovoľného tvaru:** Pre množinu atraktorov hustoty $X$ je zhluk ľubovoľného tvaru podmnožinou $C \subseteq D$, kde platí:
  - $\forall x \in C\ \exists x^* \in X:\ f_B^D(x^*) \geq \xi$, $x$ je priťahované ku $x^*$ a
  - $\forall x_1^*, x_2^* \in X:\ \exists$ cesta $P \subset D$ z $x_1^*$ do $x_2^*$, kde $\forall p \in P:\ f_B^D(p) \geq \xi$.

  Na obrázku 3.5b a 3.5d je možné nájsť príklady zhlukov rôznych tvarov pre rozdielne $\xi$.




**Obrázok 3.5 – Príklady zhlukov ľubovoľného tvaru pre rôzne $\xi$ (prevzaté z [6])**

Samotný algoritmus sa skladá z dvoch krokov.

1. **Predzhlukovací krok:** V prvom kroku vytvor mapu relevantných častí celého priestoru. Táto mapa sa využíva pre zrýchlenie výpočtu funkcie hustoty, ktorá vyžaduje efektívny prístup k okolitým častiam priestoru. Vytvor mapu rozdelením priestoru na $d$-dimenzionálne hyperkocky s veľkosťou hrany $2\sigma$. Ďalej využívaj len kocky obsahujúce relevantné dáta, teda body. Týmto spôsobom sa efektívne zbavíme šumu. Kocky reprezentujú jednodimenzionálne kľúče, ktoré sa najčastejšie uchovávajú v $B^+$-Strome. Vytvorené dátové štruktúry majú logaritmický prístupový čas, a čas potrebný k prístupu ku relevantným dátam v okolí kocky je konštantný.

2. **Zhlukovací krok:** Do úvahy ber len kocky s väčším počtom bodov, prípadne kocky s nimi susediace. S použitím dátovej štruktúry, vytvorenej v prvom kroku, vypočítaj lokálnu funkciu hustoty a lokálny sklon. Pomocou nich potom vyhľadaj atraktory, ku ktorým sú dané body priťahované. Body priradené k jednotlivým atraktorom tvoria zhluky.

Výhodou algoritmu je jeho matematický základ. Vďaka tomu je schopný rýchlo a efektívne spracovávať veľký objem dát s výraznom podielom odľahlých hodnôt. Umožňuje popísať zhluky ľubovoľného tvaru a podľa autorov [6] je rýchlejší ako ktorýkoľvek iný algoritmus spracovávajúci podobné dáta. Údajne až 45-krát rýchlejší ako *DBSCAN*. Má však aj nevýhody v podobe veľkého množstva parametrov. Tie musia byť experimentálne určené, aby bola dosiahnutá čo najväčšia efektivita. Táto skutočnosť obmedzuje využívanie tohto algoritmu v praxi, aj keď len minimálne.

# 4 Multimediálne databáze

V dnešnej dobe sme doslova zaplavený obrázkami, videami či zvukovými nahrávkami, jedným slovom multimediálnymi dátami. Podľa štandardu MPEG-7 [15], ktorý je normou pre popis obsahu multimediálnych dát, sa multimediálne dáta delia na štyri typy: audio dáta (zvuk, reč, hudba), obrazové dáta (čiernobiele a farebné obrázky), video dáta (časová postupnosť obrazov) a digitálny atrament (postupnosť súradníc s časovými razítkami, ktoré sú generované senzormi).

Je stále problematickejšie sa v takýchto dátach orientovať a vyhľadávať v nich. Ako vhodný kandidát na ukladanie, spravovanie a vyhľadávanie dát sa ukázali byť databázové systémy. Zaisťujú totiž základné podmienky, ktorými sú konzistencia, súbežnosť, integrita, bezpečnosť a dostupnosť. Poskytujú tiež funkcie pre pohodlnú manipuláciu a získavanie užitočných informácií z obrovského množstva dát. Multimediálne dáta však nemôžu byť uložené v klasických relačných databázach, ktoré sa používajú na ukladanie obyčajných štruktúrovaných dát. Sú totiž už vo svojej podstate neštruktúrované, heterogénne a neobsahujú žiadne presne definované atribúty s jednoznačným významom. Preto musia byť databázové systémy, slúžiace pre ukladanie multimediálnych dát, rozšírené o postrelačné (objektovo orientované) vlastnosti. Viac o postrelačných databázach v [14]. Tieto systémy by mali poskytovať informácie o uložených dátach, ale aj samotné dáta. Vyhľadávanie je v multimediálnej databáze založené na tzv. *metadátach*, ktoré uchovávajú informácie o obsahu alebo obsahujú popis dát. Existujú dva hlavné prístupy k popisu multimediálnych dát, prebraté zo [10].

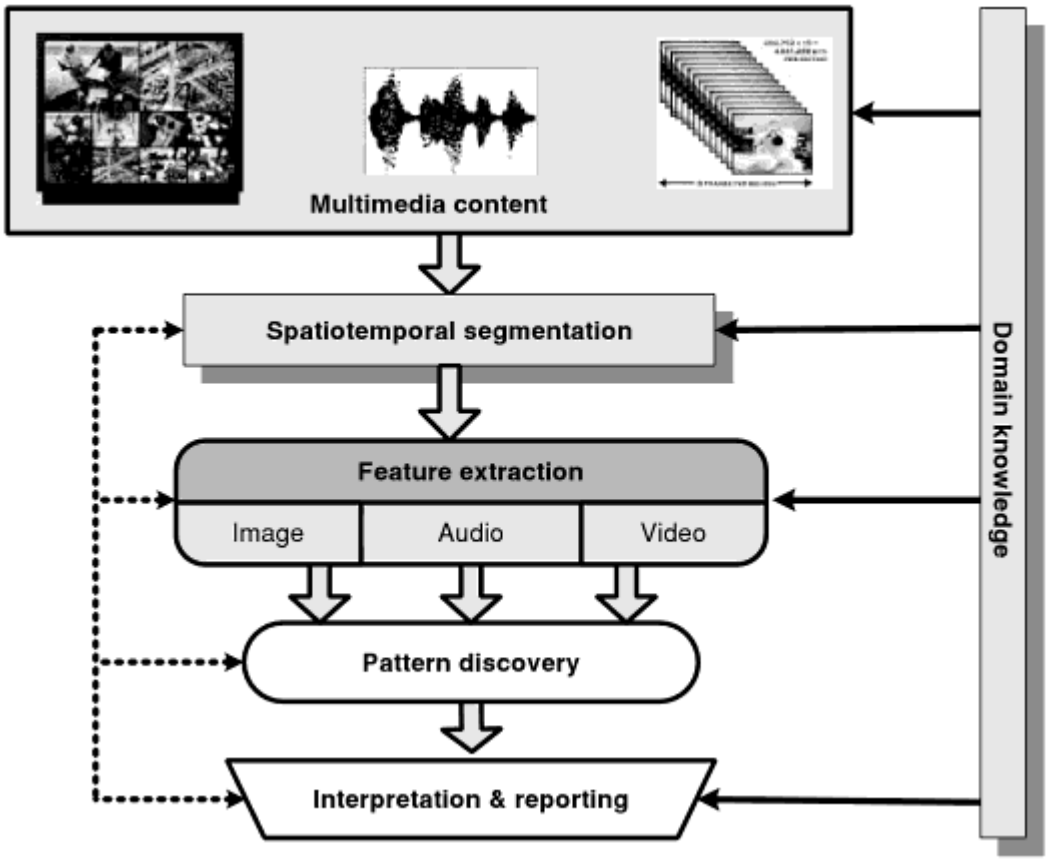
- **Podľa popisu dát:** Dáta je nutné určitým spôsobom sémanticky popísať. K tomu slúži vytvorenie názvu, kľúčových slov, pomenovanie osôb, objektov, obrázkov a akcií. Tento popis nie je možné vytvoriť automatizovaným systémom, a preto ho musíme väčšinou vytvoriť sami pri ukladaní dát do databáze alebo ich triedení. Ručné vytváranie popisu je zložité, pomalé a neefektívne, pretože vždy bude hrať určitú úlohu subjektívny prístup autora.
- **Automatický popis obsahu:** Jedná sa o, z veľkej časti automatizovaný, popis syntaxe multimediálnych dát. Táto metóda, aj napriek nesporným výhodám automatizácie, nedosahuje kvalitu ručného popisu dát. Pri zvuku je možné rozpoznať kľúčové slová, hudobné nástroje, noty, žáner a pod. Obrazové dáta je možné popísať histogramom farieb, hranami, tvarmi, objektmi, textúrami či pohybom.
- **Kombinácia oboch metód:** Používa sa ak poznáme napr. názov obrázku, dátum vytvorenia, prípadne zopár kľúčových slov. Ďalšie vlastnosti sa dajú potom extrahovať z jeho obsahu. Typickým príkladom sú fotografie.

Ak už sú dáta popísané, je rovnako netriviálne vyhľadávanie v nich. Využívajú sa dva typy dotazov, popísaných podľa [10].

- **Predloženie vzorového obrazu:** Využíva porovnanie deskriptorov (napr. vektorov, či popisujúcich vlastností). Deskriptory ku každému obrázku sú extrahované z dát a uložené spolu s nimi do databáze. Výsledkom vyhľadávania sú obrázky, ktorých deskriptory sú si najviac podobné.
- **Špecifikácia vlastností:** Vytvoríme náčrtok (zadáme farbu, tvar, textúru, melódiu a pod.), z ktorého je následne vyextrahovaný deskriptor. Ten je následne porovnaný s deskriptormi uloženými v databáze.

## 4.1 Získavanie znalostí z multimediálnych dát

Získavanie znalostí z databáz vo všeobecnosti, je popísané v kapitole 2. Proces získavania znalostí z multimediálnych databázových systémov je približne rovnaký, avšak s menšími odchýlkami spresnenými v tejto kapitole podľa [13] a [17].




**Obrázok 4.1 – Architektúra získavania znalostí z multimediálnych dát (prevzaté zo [17])**

Obrázok 4.1 zachytáva architektúru a jednotlivé kroky získavania znalostí z multimédií. Prerušované šípky v ľavej časti obrázku znázorňujú a potvrdzujú, že celý proces je interaktívny. Šípky vychádzajúce z *Domain Knowledge* (doménovej znalosti) v pravej časti, indikujú jej dôležitosť v jednotlivých krokoch dolovania. Celý proces sa dá rozdeliť na štyri základné kroky.

1. **Časopriestorová segmentácia:** V tomto kroku sú multimediálne dáta rozdelené na jednotlivé časti, ktoré už môžu byť charakterizované určitými atribútmi alebo rysmi. Pri obrázkoch sa vykonáva obrazová segmentácia, pri videu dochádza k jeho členeniu na súvislé kolekcie snímok a audio dáta sú delené podľa slov, fenoménov alebo na časti pevnej veľkosti. Tento krok je dôležitý vzhľadom k neštruktúrovanej povahe dát.
2. **Extrakcia rysov:** Spolu s časopriestorovou segmentáciou zodpovedajú predspracovaniu dát vo všeobecnom procese získavania znalostí z databáz. Extrakcia rysov sa zameriava hlavne na extrakciu nízkoúrovňových rysov multimediálnych dát. Podrobne bude tento krok popísaný v nasledujúcej kapitole.

3. **Zisťovanie vzorov:** Tento krok je samotným dolovaním dát. Takmer vôbec sa neodlišuje od kroku dolovania dát vo všeobecnosti. Na získanie cieľových znalostí sa používajú klasické metódy pre získavanie znalostí (napr. klasifikácia a predikcia, zhlukovanie, asociácia, atd.). Tie však musia byť patrične upravené pre dolovanie dát z multimédií. Používajú sa tiež automatické metódy pre popis konceptu a dolovanie rysov a udalostí.
4. **Vyhodnotenie a prezentácia:** Posledný krok je totožný s posledným krokom všeobecného procesu dolovania dát. Ide o vyhodnotenie získaných informácií, definíciu užitočných znalostí a prezentáciu výsledkov koncovému užívateľovi.

## 4.2 Extrakcia rysov

Automatický popis vytvárania deskriptorov sa nazýva *extrakcia rysov*. Rysy sú určité charakteristické vlastnosti obsahu média. Cieľom extrakcie rysov je predspracovanie, teda redukcia dimenzionality dát. Vytvárajú sa rysy, ktoré nesú najviac informácií [5]. Z obrázkov a videí sú získavané globálne a lokálne rysy. Globálne rysy popisujú vlastnosti celého obrázku, prípadne pri video dátach popisujú vlastnosti jednej alebo viacerých snímok. Lokálne rysy naproti tomu popisujú vlastnosti vybranej časti obrázku alebo pri videu časopriestorové časti snímky.

Rozlišujeme tri základné úrovne rysov. Podľa [8] znie ich popis nasledovne:

- **Nízka úroveň popisu:** Predstavuje metódu extrakcie rysov, ktorá extrahuje základné fyzikálne vlastnosti. Z týchto rysov sa nedajú zistiť podrobnosti o obsahu, teda nezaoberajú sa sémantikou. V prípade zvukových dát popisujú noty, základný tón a jeho zafarbenie a pod. Z obrazových dát sa dajú získať rysy ako textúra, farba alebo farebný histogram.
- **Stredná úroveň rysov:** Niekedy sa nazýva aj štatistické modely rysov dát. Jedná sa o štatistické výpočty nad nízkoúrovňovými rysmi, ktoré slúžia pre podobnostné vyhľadávanie. Stredná úroveň reprezentuje tiež informácie o objektoch obsiahnutých v dátach.
- **Vysoká úroveň rysov:** Obvykle je získavaná z rysov nižšej úrovne. Jedná sa o sémantickú anotáciu niekoľkých vzorov v databáze, z ktorých sú extrahované rysy, analyzované nejakým dolovacím algoritmom a nakoniec rozdelené do tried.

## 4.3 Nízkoúrovňové vizuálne rysy

Nízkoúrovňové rysy sú automaticky extrahované z digitálnej reprezentácie multimediálnych dát, uložených v databáze a nemajú žiadnu súvislosť s ľudským vnímaním týchto dát. Výsledkom extrakcie sú metadáta, slúžiace k podobnostnému vyhľadávaniu. Nízkoúrovňové rysy obrazových dát a videí sú relatívne jednoduché a popisujú vlastnosti, ako napr. farba, textúra, tvar, pohyb a umiestnenie [8]. V ďalších kapitolách však budú rysy popísané podľa štandardu MPEG-7 [15] a podľa [21].

### 4.3.1 Lokálne rysy

Lokálne nízkoúrovňové rysy sú objektmi z pohľadu ľudí, avšak dokážu ich pochopiť aj počítače. Extrakcia prebieha podľa [9] v dvoch krokoch. Prvým je detekcia *regiónov záujmu* (ROI) v obraze. Môžu to byť hrany, rohy alebo farebne spojité oblasti. Druhý krok reprezentuje samotnú extrakciu lokálnych obrazových rysov, vrátane ich okolia. Najdôležitejšou vlastnosťou je opakovateľnosť, teda

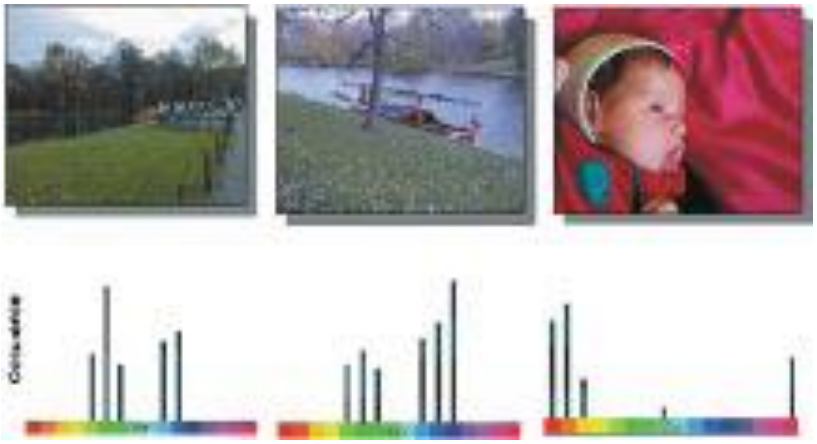
schopnosť rovnakých detekcií pri rôznych geometrických podmienkach a v prípade šumu. Existujú tri základné metódy popísané v [9].

- **MSER:** Maximaly stable extremal regions je metóda určujúca spojité komponenty obrazu s vhodne nastaveným prahom tak, aby boli maximálne stabilné. Všetky body vo vnútri regiónu majú nižšiu alebo vyššiu intenzitu ako body na okraji toho istého regiónu.
- **SIFT:** Scale Invariant feature transform sa využíva pre popis regiónov určených pomocou MSER. Pomocou histogramu, lokálne orientovaných sklonov, zachytáva určitú informáciu o strednom bode regiónu a ukladá ho ako 128 bitový vektor.
- **SURF:** Speed up robust features je metóda využívaná na detekciu mnohých zaujímavých bodov. Je založený na výpočte determinantu Hessovej matice z integrálneho obvodu. Popisuje regióny pomocou Haarovej vlnkovej transformácie.

### 4.3.2 Farba

Farba je jedným z najpoužívanejších rysov pri získavaní znalostí z multimediálnych dát. Rysy farby sú pomerne robustné a odolné voči zmene pozadia, veľkosti, či orientácie obrázku. Deskriptory môžu byť použité na popis obrázkov, ale aj videosekvencií. MPEG-7 štandardizoval širokú škálu deskriptorov farby, pričom každý má svoje špecifické využitie. Nasleduje stručný prehľad podľa [21].

- **Farebné modely:** Aby bola umožnená vzájomná kompatibilita medzi deskriptormi, musia byť farebné priestory obmedzené na tzv. *HSV* (hue-saturation-value) a na *HMMD* (hue-min-max-diff). *HSV* je známy a rozšírený farebný model. *HMMD* je nový model definovaný MPEG a využívaný výlučne v deskriptore farebnej štruktúry (CSD).
- **Deskriptor farebnej škály:** Popisuje rozloženie farieb v obrazových dátach. Ak meria rozloženie nad celým obrázkom, môže popisovať aj globálne vlastnosti obrazu. Obrázok 4.2 zobrazuje príklady farebných obrázkov a rozloženie ich farieb vo forme histogramu farieb. Deskriptor farebnej škály je vlastne histogramom farieb, ktorý je zakódovaný Haarovou transformáciou. Využíva rovnomerne rozdelené *HSV* na 255 košov. Hodnoty v týchto košoch sú rozdelené v rozsahu od 16 po 1000 bitov na histogram. Od tejto hodnoty závisí kvalita reprezentácie dát pomocou farebnej škály.



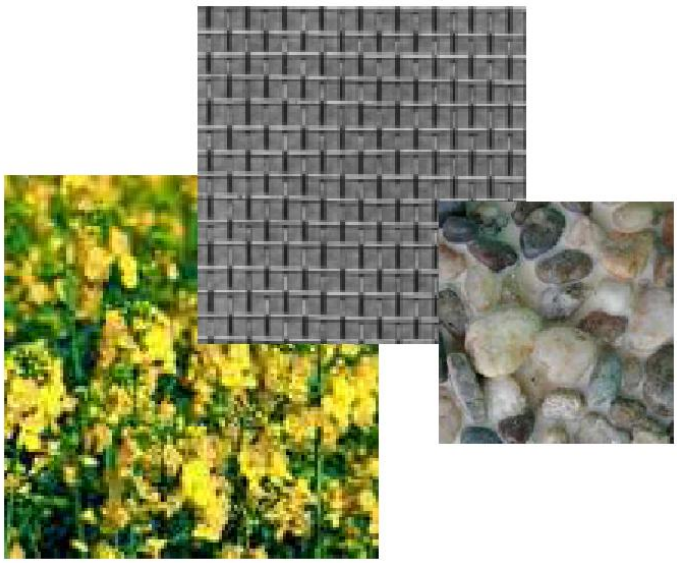
**Obrázok 4.2 – Ukážka histogramu farieb (prevzaté z [21])**

- **Deskriptor dominantnej farby:** Popisuje globálne, ale aj lokálne rozloženie farieb, pre rýchlejšie vyhľadávanie a prehliadanie obrázkov. Je kompaktnejší ako farebná škála, vďaka tomu, že farby sú v jednotlivých regiónoch zhlukované do menšieho počtu reprezentatívnych farieb. Pozostáva z reprezentatívnej farby, jej percentuálneho zastúpenia v regióne a odchýlky.
- **Deskriptor farebnej štruktúry:** Účelom tohto deskriptoru je určovať farebné vlastnosti obrázku za behu. Využíva posuvné okno, ktoré postupne skenuje obrázok. Po každom jeho posuve sa zaznamenávajú farby a histogram farieb je nanovo konštruovaný.
- **Skupina snímok/Skupina obrázkov:** Tento deskriptor definuje štruktúru, potrebnú pri popise farebných vlastností podobných obrázkov alebo snímok videa.

### 4.3.3 Textúra

Textúra je vlastnosť všetkých reálnych objektov a reprezentuje projekciu 3D povrchu reálneho objektu do 2D priestoru. Príklady textúr sa nachádzajú na obrázku 4.3. Ak je obrázok popísaný správnym deskriptorom textúry, ten potom predstavuje bezkonkurenčne najlepší spôsob pre vyhľadávanie v multimediálnych dátach. Nasledujúce deskriptory môžu byť použité nezávisle alebo spolu s inými deskriptormi [21].

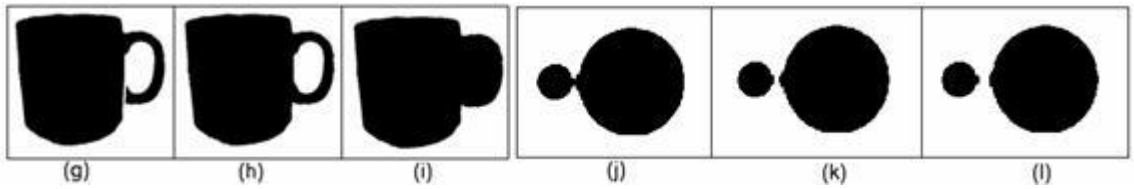
- **Deskriptor homogénnej textúry:** Popisuje smer, hrubosť a pravidelnosť vzorov v obrázku a je najvhodnejšia pre kvantitatívnu charakterizáciu textúry s homogénnymi vlastnosťami. Používa sa pri porovnávaní textúr dvoch obrázkov v multimediálnych databázach. Štruktúra textúry a jej odchýlka sú získané z frekvencie jej opakovania. Je možné využiť Fourierovú transformáciu pre efektívne výpočty menej náročných aplikácií. Obrázok sa väčšinou rozdelí na menšie časti a tie sa potom filtrujú pomocou 2D Gaborovej funkcie.
- **Deskriptor nehomogénnej textúry:** Označovaný niekedy aj ako histogram okrajov. Predstavuje rozdelenie priestoru podľa piatich typov okrajov. Dokáže vyhľadávať obrázky s podobnou sémantikou, avšak jej primárnou úlohou je porovnávanie obrázkov. Je jedným z najlepších deskriptorov pre určovanie zhody medzi dvomi fotografiami [15].



**Obrázok 4.3 – Ukážky textúr (prevzaté z [8])**

### 4.3.4 Tvar

V mnohých databázových systémoch je tvar hlavnou charakteristikou daného obrázku (napr. binárne obrázky). Deskriptor tvaru by mal byť nemenný voči rotácii, posuvu a zmene veľkosti a mal by poskytovať reprezentáciu objektu v 2D aj 3D formáte. Za objekt určitého tvaru sú považované oblasti s rovnakou textúrou, farbou alebo hranami. Jednotlivé regióny je možné popísať ako plochu alebo hranicu. Existujú tri typy tvarových deskriptorov [15].




**Obrázok 4.4 – Ukážky rôznych tvarov (prevzaté z [15])**

- **Deskriptor tvaru na základe plochy:** Môže sa skladať z jedného, ale aj viacerých plôch. Tým, že pozostáva zo všetkých bodov tvoriacich objekt, dokáže popísať aj zložité útvary (napr. uško hrnčeka na obrázku 4.4). Je odolný voči deformáciám odohrávajúcim sa za hranicami objektu. Navyše je charakteristický aj svojou rýchlosťou a presnosťou porovnávania. Má fixnú veľkosť 17,5 bajtu na objekt, teda je aj celkom malý.
- **Deskriptor tvaru na základe hraníc:** Zachytáva charakteristické tvarové rysy objektu na základe jeho hraníc. Lepšie zodpovedá ľudskému vnímaniu tvarov. Je odolný voči pohybu, zmene kamery a je kompaktný.
- **3D deskriptor:** Ak zoberieme do úvahy súčasný trend vývoja 3D vytvárajúcich a zobrazovacích zariadení, musíme sa pripraviť na možnosť ukladať 3D dáta do databáze. 3D deskriptory popisujú polygonálnu sieť, pomocou ktorej sú 3D dáta reprezentované.

### 4.3.5 Pohyb a umiestnenie

Pod pojmom pohyb sa rozumie nielen zmena umiestnenia objektu v rámci po sebe idúcich snímok, ale aj ostatné transformácie objektu (rotácia, deformácia, atd.). Opäť existuje viacero deskriptorov a niektoré z nich sú popísané podľa [15].

- **Deskriptor pohybu kamery:** Je založený na 3D pohybe kamery, ktorý môže byť automaticky prenášaný zo vstupného zariadenia, ktoré sa pohybuje dopredu, dozadu, rotuje, približuje a vzďaľuje sa. Tieto pohyby kamery je možné odhadnúť pomocou zhody niekoľkých náhodných bodov v po sebe idúcich snímkach.
- **Deskriptor pohybovej aktivity:** Snaží sa zachytiť intuitívny zmysel pre intenzitu akcie (napr. spomalený pohyb, akčná scéna a pod.). To sa dá využiť pri určovaní príliš veľkej, prípadne príliš malej aktivity. Následne môžeme prijať adekvátne opatrenia.

Umiestnenie sa používa pre identifikáciu pohyblivých objektov [8]. Tieto objekty je potom možné obaliť konvexnou obálkou, čím získame *priestorový deskriptor*. Ten obsahuje okrem iného aj súradnice pohybujúcich sa objektov. Ak pridáme aj časový údaj, získame **časopriestorový deskriptor**.

# 5 Implementácia

Predchádzajúce kapitoly mali za úlohu podrobne popísať teóriu potrebnú pre nasledujúcu praktickú časť diplomovej práce. Jej cieľom bolo vybrať vhodné metódy, používané pri zhlukovaní viacdimenzionálnych dát a vhodne ich implementovať. Samotná implementácia mala byť rozšírením už existujúcej aplikácie TRECVid Search. Následne bolo potrebné ukázať funkčnosť zvolených metód na dátovej sade TRECVid a porovnanie ich výsledkov s výsledkami voľne dostupných programov, umožňujúcich získavanie znalostí z databáz na základe zhlukovania. K tomuto účelu boli vybrané programy RapidMiner a Weka.

V tejto kapitole budú popísané všetky techniky, nástroje a dáta použité pri riešení tejto diplomovej práce. Taktiež bude popísaný samotný postup tvorby programu a jeho komponentov.

## 5.1 Databázový systém PostgreSQL

PostgreSQL, často označovaný ako Postgres, je objektovo relačný databázový systém. Patrí medzi tzv. open-source projekty a je vydávaný, podľa [25], pod licenciou PostgreSQL license, ktorá je podobná licenciám BSD (Berkeley Sotware Distribution) a MIT (Massachusetts Institute of Technology). Celkový vývoj tohto systému trvá už viac ako 15 rokov. Primárne je vyvíjaný a určený pre operačný systém LINUX, resp. pre unixové systémy, avšak v súčasnosti poskytuje podporu aj pre operačný systém Windows.

Základnou vlastnosťou PostgreSQL databáze je splnenie tzv. ACID podmienok. Tie zaručujú atomickosť všetkých operácií nad databázou, stálu konzistenciu dát, izolovanosť operácií a trvanlivosť vykonaných zmien v databáze. Podporuje tiež prácu s cudzími kľúčmi, pohľadmi, triggermi a uloženými procedúrami. Súčasťou je aj natívne rozhranie pre väčšinu programovacích jazykov (C/C++, Java, .NET, Python, atď.).

Podľa [18] poskytuje PostgreSQL funkcie, ktoré je zriedka možné nájsť v iných druhoch relačných databáz. Medzi hlavné prednosti patria, schopnosť rozšírenia systému o nové, užívateľom zadané dátové typy, funkcie či operátory a správa súbežnosti cez MVCC dizajn (Multi-Version Concurrency Control), ktorý zaručuje výborný výkon aj za podmienok silno zbiehavého prístupu. Jedinou nevýhodou tohto systému je menšia rozšírenosť oproti MySQL, ktorý tiež patrí medzi voľne dostupné databázové systémy.

Dáta potrebné pri zhlukovaní boli uložené práve v databázovom systéme PostgreSQL, ako reprezentácia dátovej sady TRECVid.

### 5.1.1 Dátová sada TRECVid

TREC konferencie, sponzorované organizáciou NIST (National Institute of Standards and Technology), majú za úlohu podporiť výskum v oblasti vyhľadávania informácií v multimédiách, a to hlavne vo videu. Poskytujú veľkú testovaciu množinu dát, nad ktorou sú riešené zadané úlohy.

V roku 2008 boli poskytnuté celkovo štyri dátové sady [4]: Sound and Vision, BBC rushes, TRECVid 2008 surveillance video a MUSCLE-VCD-2007. Táto diplomová práca, pri testovaní vlastností rôznych metód zhlukovania vektorov rysov multimediálnych dát, využíva dátovú sadu Sound and Vision z roku 2008. Jedná sa o kolekciu dokumentárnych, spravodajských a výukových

televíznych programov doplnených o archívne videá. Všetky videá sú vo formáte MPEG-1 a poskytla ich inštitúcia Netherlands Institute for Sound and Vision.

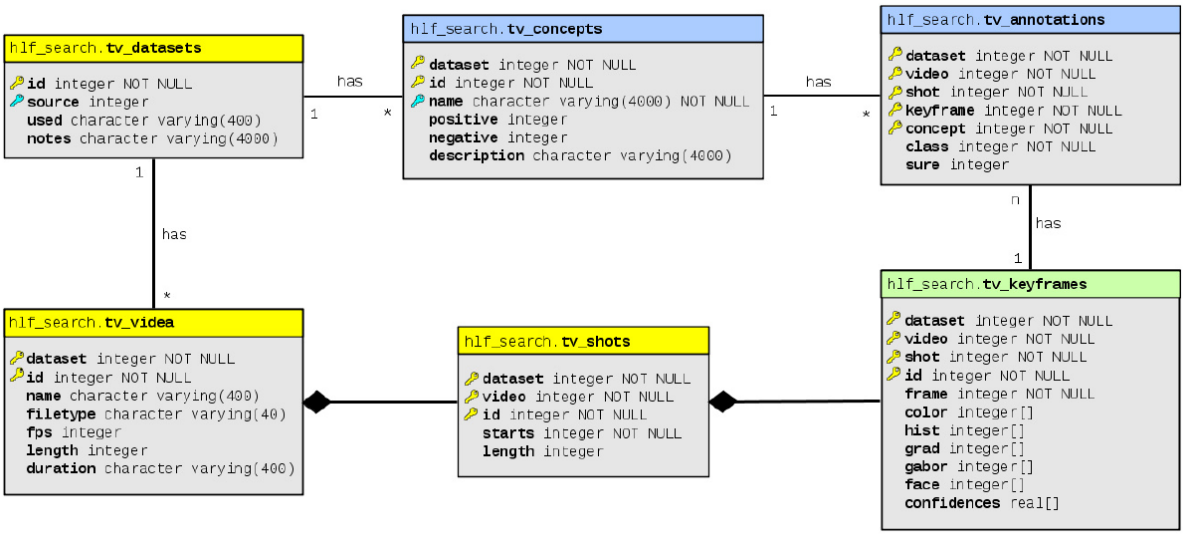
Dátová sada je rozdelená na dve časti. Prvou je podmnožina určená na vývoj aplikácií. Tvorí ju 219 videí o celkovej dĺžke približne 100 hodín. Druhú časť tvoria testovacie dáta pre evaluáciu, ktorých počet je totožný s vývojovou podmnožinou a ich dĺžka je opäť približne 100 hodín. Vzhľadom k veľkému objemu dát, ktoré by bolo potrebné spracovať, sa obrazové dáta jednotlivých videí skladajú z menších častí. Tie sa nazývajú zábery (shots) a tvoria významové celky videa. Zábery sa ďalej skladajú z jednotlivých snímok (keyframes), pre ktoré sú vytvorené nízkoúrovňové rysy (features). Každé z videí je tiež reprezentované zvukovou stopou (speech), ktorá je rozdelená na časové úseky zodpovedajúce jednotlivým záberom.

Pre popis kľúčových snímok sa používajú štyri deskriptory, prevzaté z [9]: deskriptor škálovateľného rozloženia v obrázku, deskriptor založený na histograme vo farebnom priestore HS, deskriptor založený na viacstupňovom gradiente a deskriptor textúr pomocou Gaborových filtrov. Ak je kľúčová snímka anotovaná podľa definovaných konceptov, nazývame ju pozitívnou vzorkou. Pre dátovú sadu Sound and Vision 2008 bolo zverejnených 20 konceptov, ktoré majú byť v obraze nájdené. Príkladom týchto konceptov je napríklad letiace lietadlo, demonštrácia alebo protest, kvetina, ulica či dvaja ľudia. Koncepty sa môžu v snímkach aj prekrývať, avšak táto diplomová práca pracuje len s jediným konceptom pre jednu snímku. Záber tiež nemusí obsahovať žiaden koncept a vtedy je označovaný ako negatívna vzorka.

### 5.1.2 Schéma dátovej sady

Databáza TRECVid je umiestnená na školskom serveri minerva2. Táto databáza pôvodne obsahovala všetky dáta používané pri evaluácii, avšak v školskom roku 2010/2011 musela byť z dôvodu jej straty opätovne spustená z nekompletnej zálohy. Je rozdelená na niekoľko schém, podľa logickej príslušnosti k zadaným úlohám. Vstupné dáta používané pri riešení úloh zameraných na zhlukovanie rysov vysokej úrovne sú uložené v schéme s názvom *hlf_search*. Táto schéme obsahuje celkovo 18 tabuliek. Na obrázku 5.1 je uvedený ER diagram časti tejto schémy. Dôležitými tabuľkami z hľadiska spracovania nízkoúrovňového popisu dát sú nasledovné.

- **tv_dataset:** Obsahuje jednotlivé dátové sady odlíšené jednoznačnými identifikátormi. V tejto diplomovej práci boli použité kolekcie s kľúčmi 821 pre vývoj a 822 pre testovanie.
- **tv_video:** Obsahuje všetkých 219 videí pre vývojovú aj pre testovaciu podmnožinu dát
- **tv_shots:** Tabuľka obsahujúca jednotlivé zábery priradené k videám
- **tv_concepts:** Obsahom tabuľky je definícia všetkých používaných konceptov spolu s ich stručným popisom. Koncepty sú naviazané na jednotlivé datasety.
- **tv_annotation:** Tabuľka anotácií. Uvádza či je v danom zábere obsiahnutý daný koncept. Je previazaná s tabuľkou tv_keyframes.
- **tv_keyframes:** Najdôležitejšia tabuľka z hľadiska tejto diplomovej práce. Obsahuje vyššie popísané štyri deskriptory, spolu s hodnotou o počte veľkých, stredných a malých tvárí na snímke. Je naviazaná na jednotlivé snímky, videá, datasety a anotácie.




**Obrázok 5.1 – ER diagram schémy hlf_search (prevzaté od vedúceho diplomovej práce)**

### 5.1.3 Dodatočné informácie k video dátam

Databázová schéma využívaná pri tvorbe a experimentoch tejto diplomovej práce bola z vyššie uvedených dôvodov značne zmenšená. V konečnom dôsledku sa využíva len tabuľka *tv_keyframes*, ktorá poskytuje dostatočné informácie potrebné pre zhlukovanie nízkoúrovňových rysov. Pri evaluácii však nastal problém s rozlíšením pozitívnych a negatívnych snímok. Z tohto dôvodu bolo potrebné načítať informácie o priradení jednotlivých konceptov k daným záberom manuálne.

TRECVid na svojich stránkach zverejňuje súbory, ktoré dokážu plnohodnotne nahradiť tabuľky *tv_annotation* a *tv_concept*. Tieto súbory je možné nájsť na adrese [4]. Súbory relevantné vzhľadom k získaniu informácií o konceptoch nachádzajúcich sa v danom zábere majú tvar napr. *Emergency_Vehicle.dat* a ich sémantika má tento tvar:

WX MCG-ICT-CAS Emergency_Vehicle BG_3273 TRECVID2007_12/shot12_114_RKF P

Prvé dva reťazce označujú identifikáciu tvorcov anotácie. Ďalší v poradí je názov samotného konceptu, ku ktorému daná anotácia patrí. Posledným zaujímavým reťazcom je *shot12_114_RKF*, ktorý identifikuje v prvej časti video, v druhej záber a v tretej reálnosť danej snímky. V tomto prípade možno teda určiť, že záber 114 z videa 12 je reálny a je anotovaný konceptom *Emergency_Vehicle*. Posledný znak určuje, že daný záber patrí medzi pozitívne vzorky. V opačnom prípade by bolo na mieste posledného znaku písmeno *N*.

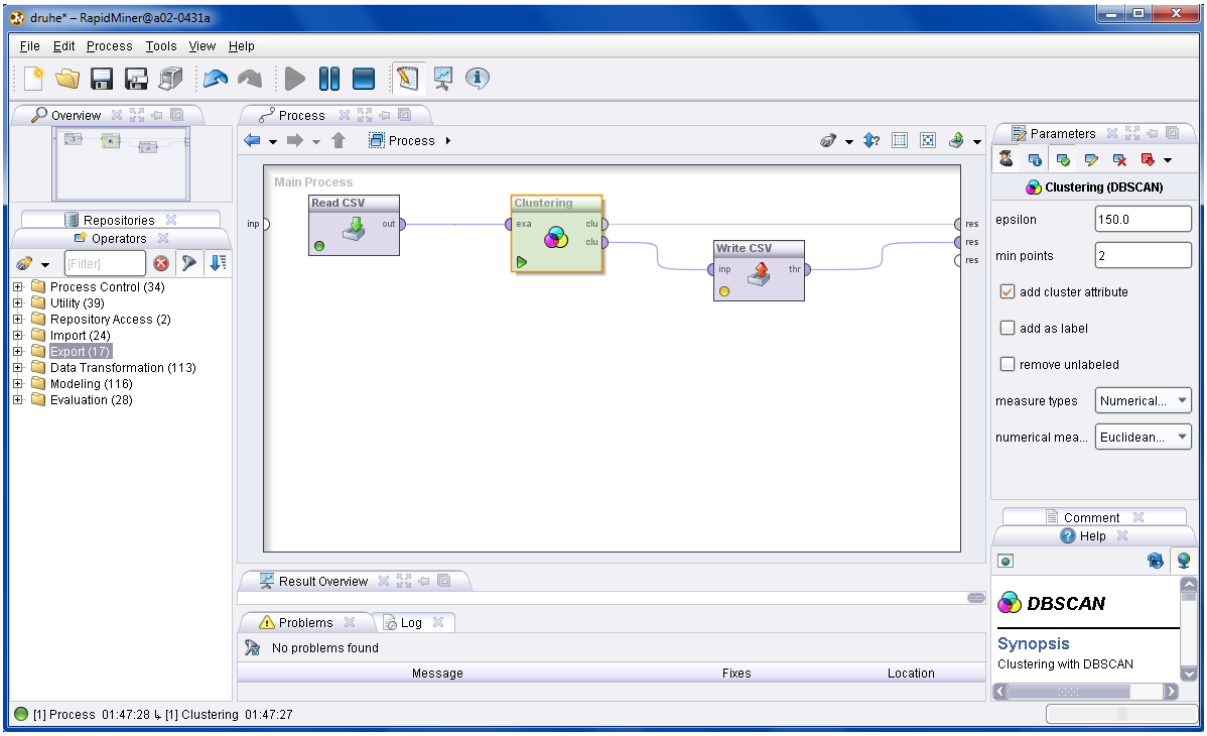
## 5.2 Použité nástroje pre dolovanie dát

Táto kapitola popisuje nástroje, ktoré boli použité pri experimentoch nad dátami popísanými v predchádzajúcej kapitole. Testy mali za úlohu demonštrovať funkčnosť aplikácie vytvorenej v rámci praktickej časti diplomovej práce. Výsledky z nižšie uvedených nástrojov boli porovnávané s výsledkami aplikácie. Pomocou týchto nástrojov boli tiež určené metódy použité pri samotnej implementácii. Týmito nástrojmi sú RapidMiner a Weka.

### 5.2.1 RapidMiner

Nasledujúci odsek čerpá informácie z [19]. RapidMiner je nástroj používaný ako samostatná aplikácia pri analýze a dolovaní dát, alebo ako prostriedok určený pre integráciu do vlastnej aplikácie. V súčasnej dobe ide o najrozšírenejší voľne dostupný program pre získavanie znalostí z dát. Existujú dve verzie RapidMineru, Community a Enterprise edition. Community edition je voľne dostupná ako open-source a je primárne určená pre výzkum, testovanie a súkromné použitie. Aj napriek tomu však ponúka veľké množstvo funkcií pre dolovanie informácií z dát. Je možné ju využiť pri akýchkoľvek dolovacích úlohách, napr. pri získavaní znalostí z multimediálnych databáz. Enterprise edition je platená verzia s bohatším repertoárom možností a funkcií a využíva sa v komerčných oblastiach.

RapidMiner je celý implementovaný v jazyku Java, čím je zabezpečená prenositeľnosť na veľké množstvo platforiem a operačných systémov. Grafické rozhranie umožňuje jednoduché a intuitívne vytváranie a spúšťanie dolovacích úloh. Na obrázku 5.2 je ukážka aplikácie RapidMiner s práve prebiehajúcim zhlukovaním pomocou metódy DBSCAN.




**Obrázok 5.2 – Ukážka aplikácie RapidMiner so spustenou metódou DBSCAN**

RapidMiner definuje každú úlohu ako proces pozostávajúci z operátorov. Tieto pojmy sú podľa [9] definované približne nasledovne.

Proces sa skladá z rôzneho počtu operátorov, ktoré je možné rôzne kombinovať a zamieňať. Táto vlastnosť poskytuje možnosť riešenia rozmanitých dolovacích úloh. Konfigurácia procesu, čo značí použité operátory a ich vzájomné prepojenie, sa počas vytvárania procesu v grafickom prostredí ukladá do súboru XML. Vytvorenie konfiguračného súboru je teda možné aj bez použitia grafického prostredia. Na základe týchto súborov je možné spúšťať dolovacie úlohy na pozadí z príkazového riadku.

Operátorov je v RapidMineri obsiahnutých viac ako 400 a reprezentujú jednotlivé časti dolovacích úloh. Výhodou tohto programu je možnosť vytvárať vlastné operátory, pomocou ktorých sa dá dodefinovať funkcionalita. Postup vytvárania nového operátora je popísaný v [20].

Pre každý operátor je presne definovaný požadovaný vstup a produkovaný výstup. Taktiež je možné operátory nastaviť pomocou množiny vnútorných operátorov a parametrov. Vstupom je väčšinou model alebo samotné dáta. Operátory produkujú okrem výstupov aj rôzne hodnoty, ktoré sa dajú zaznamenať do logovacieho súboru. Jedná sa napríklad o dĺžku trvania procesu. Vo všeobecnosti platí, že na vstup operátora je možné priviesť iba kompatibilný výstup iného operátora. Výnimku tvoria prvý a posledný operátor, ktoré prijímajú, prípadne poskytujú dáta od nadradeného procesu. Z operátorov sa dajú vytvárať tzv. stavebné bloky, ktoré je následne možné opakovane využívať vo viacerých procesoch.

Operátory je možné rozdeliť podľa [20] do niekoľkých skupín. Každá skupina implementuje inú časť dolovacej úlohy.

- **Riadenie procesu:** Operátory riadiace tok procesu. Patria sem rôzne slučky, podmienky či vetvenia.
- **Utility:** Prídavné operátory, slúžiace k spájaniu sub-procesov. Do tejto skupiny patria tiež makro operátory a operátory pre logovanie.
- **Prístup k repozitárom:** Vstupné dáta z databáze alebo zo súboru je možné uložiť do repozitára, ku ktorému je pomocou tejto skupiny operátorov ľahší a rýchlejší prístup.
- **Import,Export :** Tieto skupiny zaisťujú vstup a výstup dát procesu. Ponúkajú širokú škálu podporovaných formátov súborov (napr. *csv*), ale aj priame pripojenie k rôznym typom databáz.
- **Transformácia dát:** Najdôležitejšia skupina operátorov, čo sa počtu a významu týka. Všetky sa využívajú pri transformácií (predspracovanie a postprocesing) dát a metadát.
- **Modelovanie:** Obsahuje operátory implementujúce samotné dolovacie algoritmy, medzi ktoré patria klasifikačné metódy, regresné metódy, zhlukovanie, váhovanie koeficientov, metódy pre asociačné pravidlá, korelačné a podobnostné analýzy. Taktiež sem patria operátory aplikujúce výsledné modely na ďalšie dátové sety.
- **Vyhodnotenie:** Operátory vyhodnocujúce kvalitu dokončených dolovacích úloh pomocou grafov, alebo textovou formou.

### 5.2.2 Weka

Weka je kolekciou algoritmov strojového učenia, určenej pre úlohy získavania znalostí z dát. Tieto algoritmy môžu byť, rovnako ako v prípade RapidMineru, použité priamo nad určitou dátovou množinou, alebo ich je možné zakomponovať do vlastného kódu v programovacom jazyku Java. Weka obsahuje nástroje pre predspracovanie dát, klasifikáciu, regresiu, zhlukovanie, asociačné pravidlá a vizualizáciu výsledkov alebo samotných dát. Podporuje tvorbu nových schém strojového učenia a je vydaná pod licenciou GNU GPL (GNU General Public License). Informácie v tejto časti textu boli čerpané z [24].

Weka je v tejto práci využitá ku grafickému zobrazeniu dát pred aj po ich spracovaní zhlukovacími algoritmami. K tomu slúži funkcia Visualization, ktorá dokáže zobraziť viacero druhov grafov, prípadne grafických výstupov. Využité sú však iba dve z nich. Prvým je možnosť načítania súboru s príponou arff a zobrazenie jednotlivých atribútov v dvojdimenzionálnom grafe. Druhý z grafov tiež načítava arff súbory, avšak ako výstup vytvára tzv. ROC krivky.

## 5.3 TRECVid Search

Cieľom praktickej časti diplomovej práce je rozšírenie existujúceho programu TRECVid Search. Tento program je určený pre vyhľadávanie v dátovej sade TRECVid. Je vyvíjaný na Fakulte informačních technologií VUT v Brne. Software je implementovaný v jazyku Java s využitím databázového systému PostgreSQL.

TRECVid Search má vytvorené funkčné grafické rozhranie, na ktoré bolo potrebné nadviazať. Doterajšia verzia programu umožňovala vyhľadávanie vo videu podľa rôznych metód uvedených nižšie. Posledná z uvedených metód, teda zhlukovanie na základe nízkoúrovňových rysov, je predmetom tejto práce. Nasledujúci zoznam je prevzatý z [23].

- Vyhľadávanie na základe lokálnych vlastností obrazu, využíva automatické hľadanie podobností medzi dvojicou obrázkov pomocou algoritmov SIFT a SURF popísaných v kapitole 4.3.1.
- Vyhľadávanie na základe globálnych vlastností obrázkov, napr. farby alebo textúry.
- Vyhľadávanie na základe kombinácie dvoch predchádzajúcich typov.
- Jednoduché textové vyhľadávanie.
- Textové vyhľadávanie s využitím sémantického slovníka WordNet.
- Vyhľadávanie na základe nájdených obrázkových HLF konceptov, využíva internetové vyhľadávače.
- Zhlukovanie na základe nízkoúrovňových rysov, využívajúce metódu BIRCH.

## 5.4 Analýza a návrh aplikácie

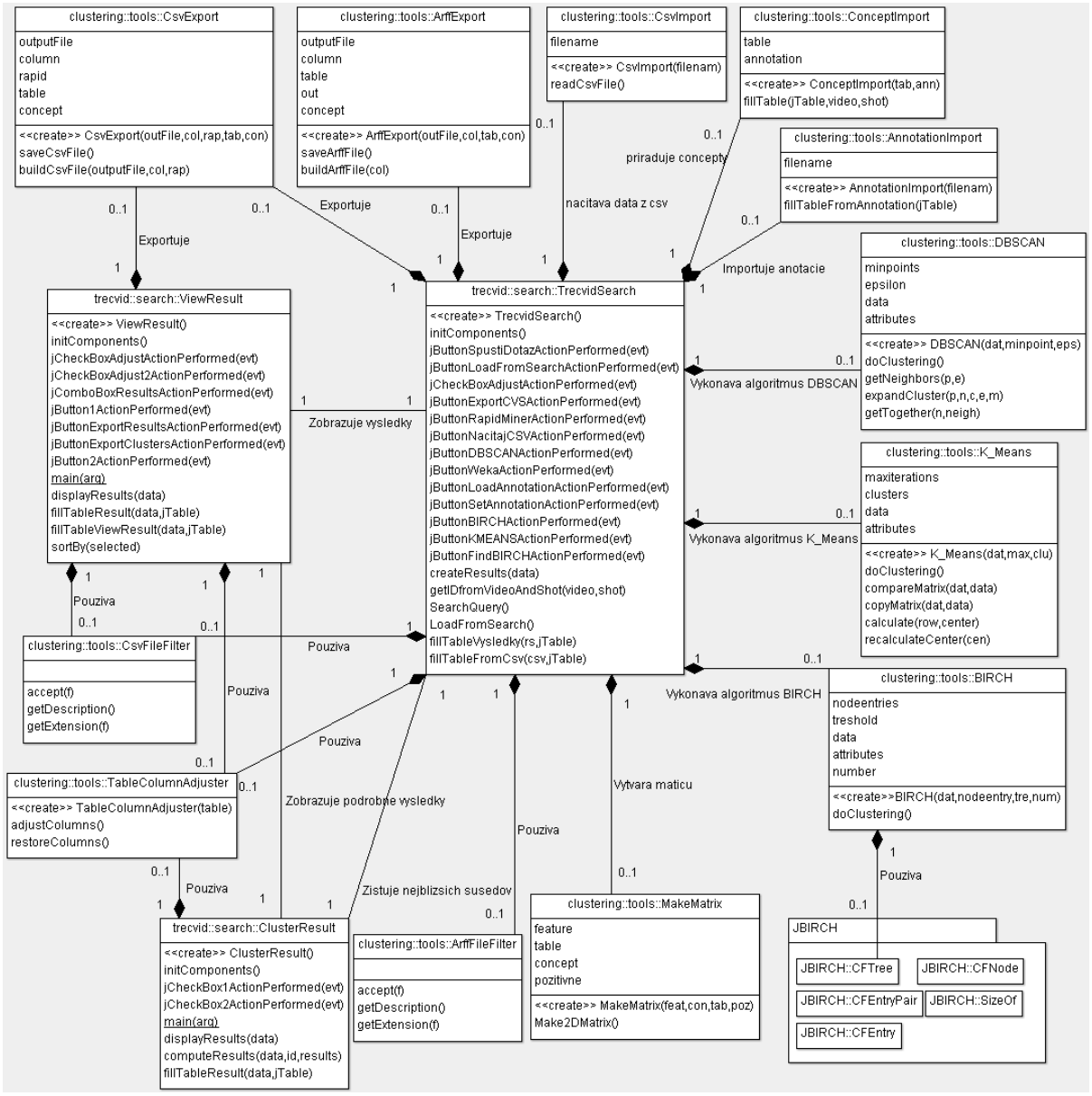
V tejto kapitole bude popísaná analýza a návrh diplomovej práce. Ako už bolo vyššie spomínané, bolo potrebné čiastočne prepracovať existujúcu aplikáciu TRECVid Search a pridať do nej modul, vykonávajúci zhlukovanie multimediálnych dát. Pre samotné zhlukovanie boli po experimentoch, vykonaných v nástroji RapidMiner, vybrané metódy DBSCAN a K-Means. Jedným z dôvodov výberu týchto metód bola možnosť porovnania výsledkov dokončenej praktickej časti a výsledkov poskytovaných RapidMinerom. Poslednou implementovanou metódou sa stala, po dohode s vedúcim diplomovej práce, BIRCH. BIRCH bol zvolený, pretože je niekoľkonásobne rýchlejší ako predchádzajúce dve metódy. Popis jednotlivých metód sa nachádza v kapitole 3.2.

Zhlukovanie slúži samo o sebe ako vyhľadávanie. Z jeho výsledkov je pri ich správnej prezentácii možné vybrať objekty, ktoré sú najbližšie zadanému objektu. Táto charakteristická črta zhlukovania je využitá pri vyhľadávaní podobných obrázkov na základe jednotlivých nízkoúrovňových rysov.

### 5.4.1 Návrh aplikácie

Cieľom návrhu je rozšíriť aplikáciu TRECVid Search. Rozšírenie poskytuje načítanie vstupných dát z PostgreSQL databáze, prípadne z csv súboru, prehľad o vstupných dátach, možnosť načítať k týmto dátam koncepty, export načítaných dát a napokon vykonanie samotného zhlukovania nízkoúrovňových rysov dát. Zhlukovanie je možné vykonať na všetkých dátach, alebo len na pozitívnych vzorkách, teda na dátach, ku ktorým sú priradené koncepty. Po ukončení zhlukovania sa zobrazí nové dialógové okno zachytávajúce jeho výsledky. Tie sú prezentované formou tabuliek zobrazujúcich jednotlivé objekty a k nim priradené zhluky. Je umožnené uložiť výsledky zhlukovania

do csv súboru, pre ich ďalšie použitie. Napokon sa dajú v novom dialógovom okne zobraziť objekty, ktoré sú vybranému objektu najbližšie.




**Obrázok 5.3 – Diagram tried vytvorenej aplikácie**

Čiastočne odtrhnutá od tohto, inak homogénneho celku, ktorý by mohol byť implementovaný aj ako samostatná aplikácia, je možnosť vyhľadania podobných obrázkov na záložke *Search*. Vyhľadávanie prebieha pomocou metódy BIRCH a s využitím rozličných tried a funkcií obsiahnutých v hlavnej časti rozšírenia.

Diagram tried, zobrazený na obrázku 5.3, zobrazuje statickú štruktúru aplikácie prostredníctvom samotných tried a vzťahov medzi nimi. Tento diagram znázorňuje iba komponenty pridané v rámci riešenia praktickej časti diplomovej práce. Triedy sú rozdelené do týchto troch balíčkov *trecvid.search*, *clustering.tools* a pomocného balíčka *JBIRCH*.

Balíček *trecvid.search* je hlavným balíčkom celej aplikácie a bol doplnený o triedy *ViewResult*, ktorá zobrazuje výsledky zhlukovania a *ClusterResult*, ktorá zobrazuje objekty najviac podobné vybranému objektu. Obe tieto triedy sú reprezentované samostatnými grafickými oknami

aplikácie. Trieda *TrecvidSearch* bola taktiež upravená z hľadiska vyhľadávania v záložke *Search*, ale aj v rámci vytvorenia novej záložky v hlavnom programe s názvom *Clustering*.

Balíček *clustering.tools* obsahuje pomocné triedy pre import a export dát, zarovnanie stĺpcov tabuľky a taktiež tri hlavné metódy zhlukovania (BIRCH, DBSCAN a K-Means). *JBIRCH* pozostáva z tried prevzatých z GNU GPL projektu JBIRCH. Tieto triedy reprezentujú tvorbu, úpravu a prehľadávanie CF-Stromu potrebného pri výpočte algoritmu BIRCH.

## 5.5 Implementácia

Pre implementáciu aplikácie bol zvolený objektovo orientovaný programovací jazyk Java, ktorého detailný popis je možné nájsť na [16]. Ako vývojové prostredie bol zvolený NetBeans IDE 7.0, avšak prvé časti boli vyvíjané v NetBeans IDE 6.9.1. V aplikácii sú využité štandardne dostupné knižnice a triedy jazyka Java. Okrem nich boli použité aj dve voľne dostupné rozširujúce knižnice OpenCvs a Colt.

OpenCsv je balíček, obsahujúci triedy pre zjednodušenie práce so súbormi vo formáte csv. Trieda CSVReader umožňuje jednoduché načítanie csv súborov. Naopak trieda CSVWriter obsahuje metódy pre vytváranie súborov tohto formátu. Obe triedy sú schopné vysporiadať sa s rôznymi druhmi oddeľovačov hodnôt a s rôznymi znakmi, ktoré ohraničujú text. Všetky operácie so súbormi formátu csv, teda načítanie vstupných dát, ich ukladanie pre ďalšie použitie a tiež export výsledkov pre porovnanie s nástrojom RapidMiner, sú v aplikácii riešené pomocou tejto knižnice. Bližšie informácie o tomto balíčku sú na [22]. Druhý z rozširujúcich balíčkov je Colt. Predstavuje celú množinu knižníc implementujúcich výkonné, optimalizované vedecké a technické výpočty. Obsahuje tiež implementácie optimalizovaných maticových dátových štruktúr, nad ktorými je možné jednoducho vykonávať maticové operácie, výber dát a zároveň aj štatistické operácie (napr. výpočet priemeru). Podrobnejšie je tento balíček popísaný na [3]. Vo vytvorenej aplikácii sú tieto maticové štruktúry využívané predovšetkým pre ukladanie veľkého objemu dát. Následne je nad týmito dátami vykonaná jedna z metód zhlukovania. Jednou z nevýhod maticových štruktúr balíčka Colt je ich statická veľkosť. Ak by bolo možné matice dynamicky zväčšovať, výpočty by boli ešte výkonnejšie.

Väčšina dát v aplikácii je z maticových štruktúr prevádzaná priamo do tabuliek. Tieto dáta sú však načítané dynamicky a preto vzniká problém s určením rozsahu vstupných hodnôt pre jednotlivé stĺpce. Bolo nutné zaistiť schopnosť prispôsobiť veľkosť tabuľky načítaným dátam pre ich správne zobrazenie. Túto schopnosť ponúka trieda TableColumnAdjuster, ktorej autorom je Rob Camick. Je voľne dostupná na [11] a bola taktiež použitá vo vytvorenej aplikácii.

Aplikácia TRECVid Search ponúka plne funkčné grafické rozhranie. Tvorí ho okno s niekoľkými záložkami združujúcimi funkcionalitu jednotlivých častí do logických celkov. V rámci praktickej časti diplomovej práce je vytvorená nová záložka s názvom *Clustering* a vytvorené dve samostatné okná, slúžiace k prezentácii výsledkov. Do určitej miery je upravená aj funkcionalita záložky *Search*.

Celú doplňujúcu časť je možné rozdeliť do štyroch hlavných celkov. Prvým z nich je načítanie a úprava vstupných dát. Druhá časť sa dá nazvať aj hlavnou, pretože zahrňuje algoritmy zhlukovania nízkoúrovňových dát a teda aj popis implementovaných zhlukovacích metód. Ďalší celok má za úlohu prezentovať výsledky získané z predchádzajúcej časti. Napokon poslednou časťou je vyhľadávanie podobných obrázkov na základe výsledkov zhlukovania. Nasledujúce kapitoly postupne popíšu spôsob implementácie týchto celkov.

### 5.5.1 Načítanie a úprava vstupných dát

Aplikácia TRECVid Search umožňuje v záložke *Database* pripojenie k databáze PostgreSQL, z ktorej čerpá všetky dáta z dátovej sady TRECVid. Doplnená časť obsahuje možnosť vytvorenia a vykonania dotazu nad touto databázou. Nie je žiadnym spôsobom kontrolovaná správnosť dotazu, pretože sa jedná o experimentálnu aplikáciu a predpokladá sa znalosť danej databáze. Vstupné dáta je možné načítať tiež zo súboru csv. Pri výbere tohto spôsobu získania dát sa zobrazí dialógové okno určené k otvoreniu súborov. Pomocou triedy *CsvFileFilter* je prednastavený výber súborov s príponou csv a nie je možné vybrať súbor iného formátu. Následne je tento súbor načítaný pomocou vyššie uvedenej knižnice OpenCsv. Aplikácia poskytuje aj funkciu načítania iba tých dát, ktoré sú k dispozícii v záložke *Search*. Táto možnosť kombinuje výber informácií z ComboBoxov, obsahujúcich čísla videí a záberov, s vytvorením dotazu smerujúceho do databáze.

Všetky tri spôsoby načítania dát využívajú pre ich zobrazenie tabuľku, ktorá je vďaka triede TableColumnAdjuster schopná prispôsobovať sa veľkosti týchto dát. Do tejto tabuľky sa dodatočne doplní identifikačné číslo popisujúce jednotlivé riadky. Načítanie zo záložky *Search* a z databáze využívajú, pre naplnenie tabuľky, *java.sql.ResultSet*, ktorý dokáže poskytovať obsah výsledku dotazu po jednotlivých riadkoch, tak ako sú uložené v databáze. Rovnako CSVReader obsahuje metódu *readNext*, ktorá má podobnú funkcionalitu.

Kvôli dôvodom uvedeným vyššie je potrebné manuálne načítavať súbor, ktorý obsahuje anotácie jednotlivých záberov. Táto procedúra je riešená podobným spôsobom ako načítavanie dát z csv súboru, s tým rozdielom, že v tomto prípade je použitá vstavaná trieda *Scanner* a je naplnená odlišná tabuľka. Samotné anotácie sú prakticky bezvýznamné, ak sa nepriradia ku správnym dátam. K tomuto účelu slúži trieda *ConceptImport*, ktorá doplní do tabuľky, obsahujúcej dáta, správne koncepty v číselnej reprezentácii. Zábery, ktoré nie sú anotované majú priradenú hodnotu nula.

Vstupné dáta obsiahnuté v tabuľke je ďalej možné, bez rozdielu na spôsob načítania, exportovať do csv súboru dvoma spôsobmi. Prvým je ich export ako celku. Takto získaný súbor je potom vhodný pre opätovné načítanie a použitie týmto programom. Druhý spôsob umožňuje export dát vhodných ako vstup pre nástroj RapidMiner. Ukladá sa vždy len jeden z nízkoúrovňových rysov, avšak existuje možnosť uložiť iba pozitívne vzorky. Rovnakým spôsobom je vyriešený aj export pre program Weka. Ten však využíva, namiesto triedy CSVWriter, štandardné knižnice jazyka Java.

Na obrázku 5.4 je ukážka grafického rozhrania aplikácie TRECVid Search, konkrétne záložky Clustering. Je tu vidieť tlačidlá vykonávajúce všetky popísané možnosti načítania a úpravy dát.

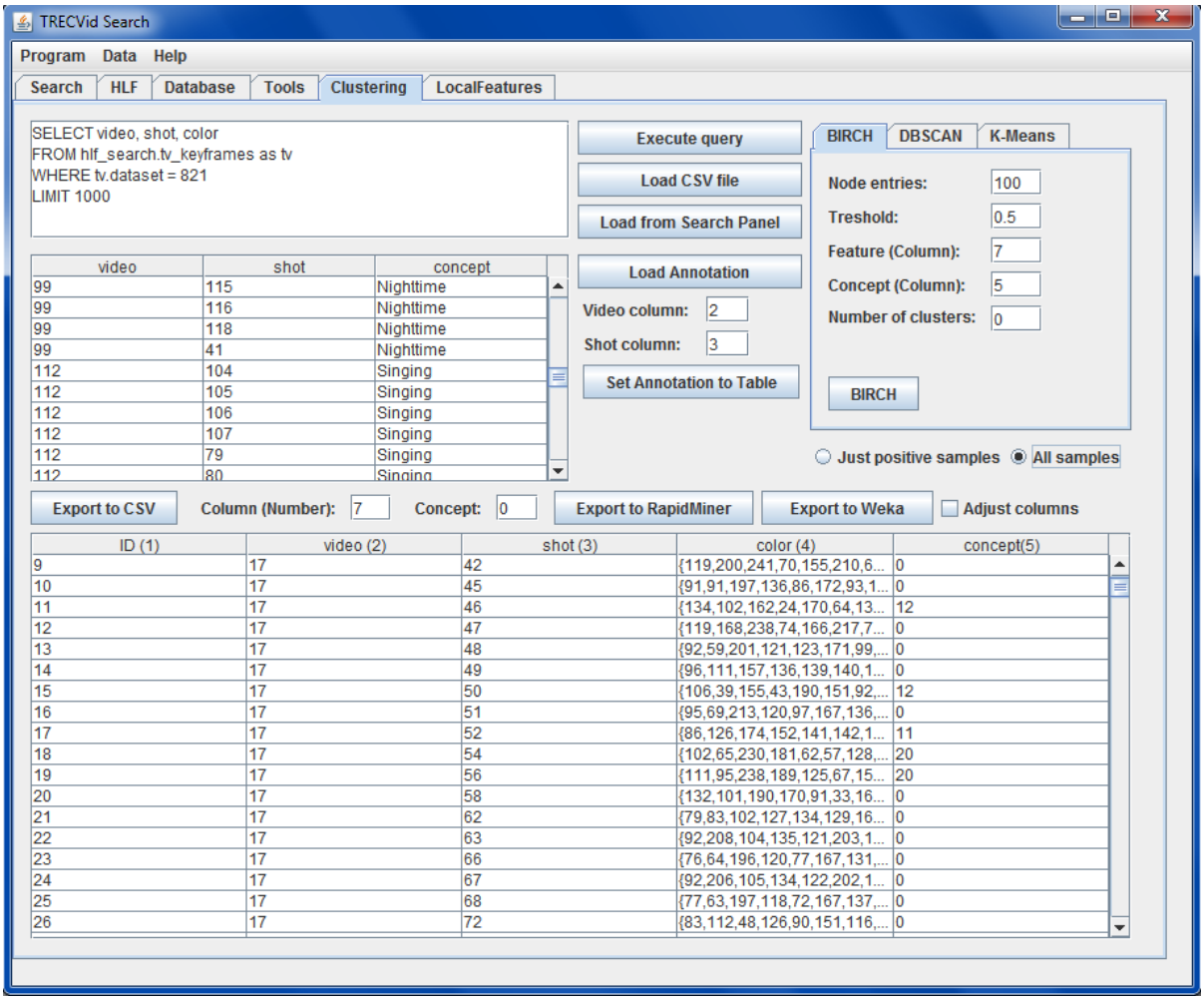
### 5.5.2 Zhlukovanie

V aplikácii sú implementované metódy BIRCH, DBSCAN a K-Means. Ešte pred začiatkom samotného procesu zhlukovania je potrebné vykonať viacero nastavení. Prvým je výber dát. Aplikácia umožňuje zadať, či sa majú použiť všetky vopred načítané dáta, alebo len tie z nich, ktoré majú priradený koncept, teda tzv. pozitívne vzorky. V ďalšom kroku je potrebné určiť, podľa ktorého z rysov má prebehnúť zhlukovanie. Pri zadaní stĺpca tabuľky, obsahujúceho koncepty, je vo výsledných dátach zohľadnený aj tento faktor. V opačnom prípade sa uvažuje o všetkých dátach akoby boli bez priradeného konceptu. Tieto tri nastavenia sú pre všetky tri metódy rovnaké. Ďalšie hodnoty ovplyvňujúce výsledky zhlukovania sú pre každú z metód odlišné a budú popísané nižšie.

Pri spustení procesu zhlukovania sa vo všetkých troch prípadoch načítajú vstupné dáta do optimalizovanej maticovej štruktúry. K tomu slúži trieda *MakeMatrix*, ktorá pomocou metódy *Make2DMatrix* prevedie relevantné informácie z tabuľky, naplnenej vstupnými dátami, do

pripravenej matice. Riadok tejto matice predstavuje jednu vstupnú vzorku dát. V prvom stĺpci sa nachádza koncept priradený danej vzorke. Nasledujú jednotlivé dátové atribúty vybraného nízkoúrovňového rysu spolu s pomocnou hodnotou pre algoritmus DBSCAN a pripraveným stĺpcom pre výsledný zhluk, do ktorého bude daná vzorka patriť. Poslednou hodnotou riadku matice je jednoznačný identifikátor odkazujúci na tabuľku vstupných dát.

.




**Obrázok 5.4 – Okno aplikácie TRECVid Search (záložka Clustering)**

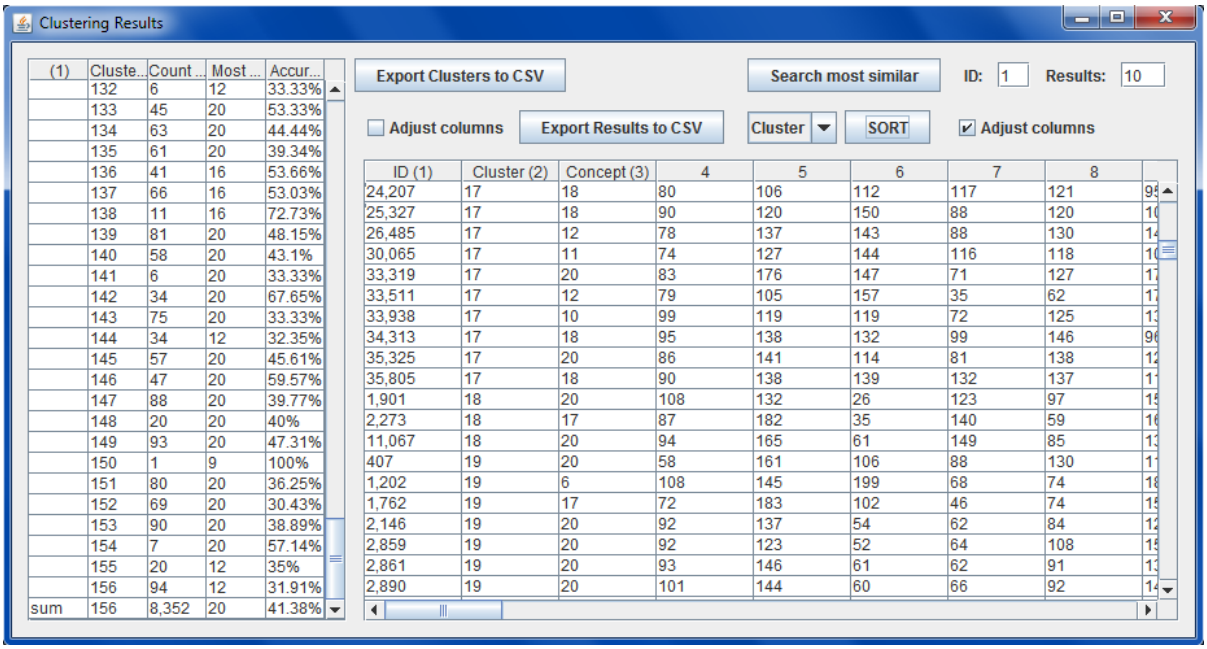
Metóda BIRCH popísaná v kapitole 3.2.3 je implementovaná ako trieda *BIRCH* s jedinou vnútornou metódou *doClustering*. Pre správne vytvorenie inštancie tejto triedy je potrebných viacero parametrov. Prvým je najvyššie možné množstvo potomkov, ktoré môžu uzly, alebo listy v danom CF-Tree dosiahnuť. Druhý parameter udáva prah, pri ktorom je potrebné zväčšiť veľkosť listového uzlu. Kľúčovou z hľadiska výpočtu je však matica obsahujúca dáta. Tá sa pomocou balíčka JBIRCH prevedie na výškovo vyvážený strom, splňujúci zadané parametre. Balíček JBIRCH vytvoril Roberto Perdisci, je distribuovaný pod licenciou GNU GPL a viac informácií je na [12]. Umožňuje zadanie veľkosti pamäte, do ktorej je potrebné dané dáta zmestiť. V tejto práci bola zvolená štandardná veľkosť 2 GB. Pôvodne priraďoval JBIRCH každej vstupnej hodnote nový identifikátor, avšak bola do neho pridaná funkcia generujúca nový identifikátor každému listovému uzlu stromu. Posledným, ale voliteľným parametrom je množstvo výsledných zhlukov. Pri zadaní určitého množstva sa spustí algoritmus K-Means, ktorý je popísaný v ďalšom texte, nad priemernými hodnotami jednotlivých

atribútov všetkých listových uzlov. Ak nie je tento parameter zadaný (nastavená nula), tak je počet zhlukov ako aj ich obsah identický s listovými uzlami vytvoreného stromu.

DBSCAN metóda zhlukovania je popísaná v kapitole 3.2.3. V tejto aplikácii je predstavovaná triedou *DBSCAN*, ktorá má rovnako ako predchádzajúca, ale aj nasledujúca trieda hlavnú vnútornú metódu *doClustering*. Parametrami tejto triedy sú minimálny počet bodov, ktoré môže vytvorený zhluku obsahovať, a hraničná hodnota $\varepsilon$, udávajúca do akej najväčšej vzdialenosti sa môžu v takomto zhlukoch vyskytovať jednotlivé body. Pri samotnom zhlukovaní sa využíva jeden zo stĺpcov matice, ktorý udáva, či sa daný bod už nachádza v niektorom z vytvorených zhlukov. Ako vzdialenostná funkcia bola zvolená euklidovská vzdialenosť, ktorej výpočet je vo vzorci 3.5 tejto diplomovej práce.

Poslednou z implementovaných metód zhlukovania je K-Means, predstavovaná triedou K_Means. Jej parametrami sú maximálny počet iterácií a počet zhlukov, ktoré by mal výsledný model obsahovať. Podrobnejšie je táto metóda rozoberaná v kapitole 3.2.1. Jedným z problémov tejto metódy je určovanie počiatočných bodov, predstavujúcich jednotlivé zhluky. V tejto aplikácii je využitá štandardná trieda jazyka Java *Random*, ktorá generuje čísla v rozsahu danom identifikačnými číslami jednotlivých vzoriek dát. Následne sú za počiatočné body zvolené vzorky prislúchajúce k týmto vzorkám. Tento algoritmus je možné použiť na výsledné listy metódy BIRCH a tak zredukovať ich počet na požadované množstvo.

Výstupom všetkých troch metód sú optimalizované maticové štruktúry zhodné so vstupnými maticami. Rozdielna je len hodnota udávajúca príslušnosť k danému zhluku. Zhluky sú vždy číslované od jednotky, pričom v algoritme DBSCAN sú zašumené, teda príliš vzdialené dáta, označené hodnotou mínus jedna.



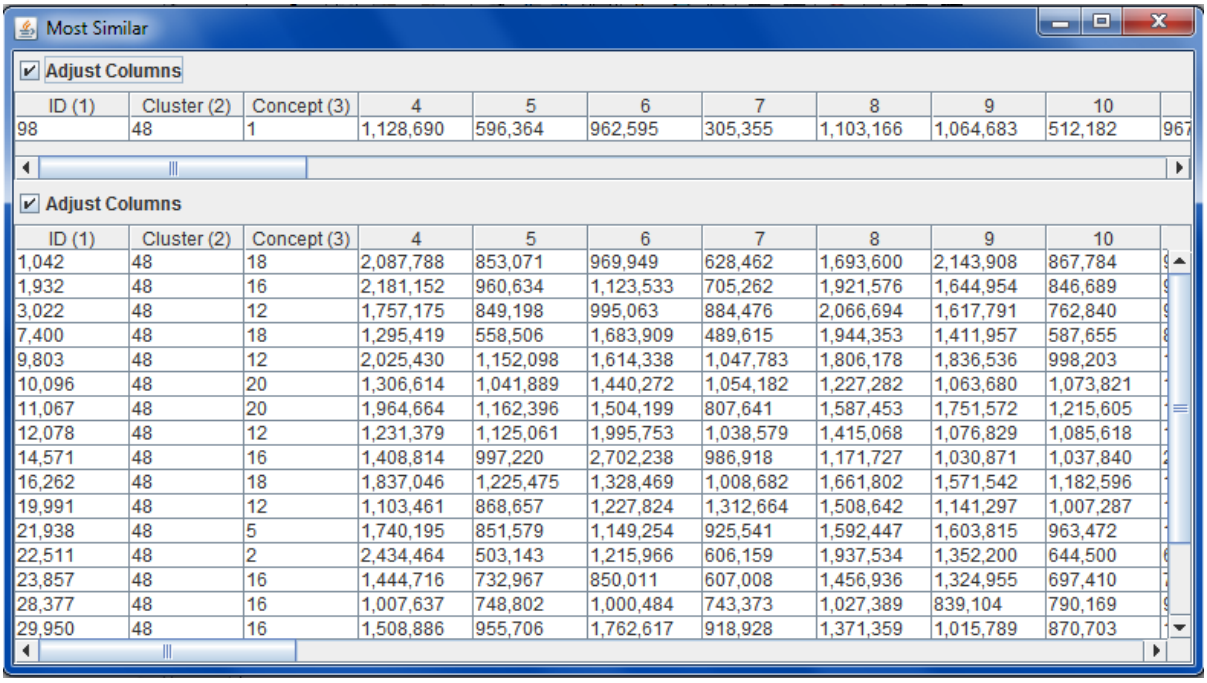

**Obrázok 5.5 – Okno aplikácie s výsledkami zhlukovania pomocou metódy BIRCH**

## 5.5.3 Prezentácia výsledkov

Výsledky zhlukovania sa predávajú metóde *displayResults* triedy *ViewResult*. Táto trieda vytvára nové okno, ktorého ukážka je na obrázku 5.5. Zobrazovanie výsledkov je následne rozdelené do dvoch krokov. V oboch sa využívajú, na prezentáciu výstupných dát zhlukovania, tabuľky. Nasleduje popis týchto krokov.

- **Zhlukový model:** Tento model je vložený do menšej z tabuliek. Obsahuje štyri hodnoty pre každý vytvorený zhluk. Prvou z hodnôt je identifikačné číslo zhluku. Druhou potom počet dátových vzoriek náležiacich tomuto zhluku. Táto hodnota, ako aj všetky ostatné, je získaná z výslednej matice zhlukovania, ktorá sa predáva tejto triede. V treťom zo stĺpcov je hodnota udávajúca číslo konceptu, ktorý je priradený najväčšiemu počtu vzoriek v danom zhluku. Napokon štvrtá hodnota je percentuálne zastúpenie počtu týchto najviac zastúpených konceptov v pomere k celkovému počtu vzoriek. V prípade, že daný zhluk neobsahuje žiadnu vzorku, ktorej by bol priradený koncept, teda žiadnu pozitívnu vzorku, je táto hodnota nastavená na nulu. V poslednom riadku tabuľky sa nachádzajú sumárne hodnoty všetkých štyroch stĺpcov. Sú to celkový počet zhlukov, celkový počet vzoriek, najčastejšie zastúpený koncept a priemerná percentuálna úspešnosť.
- **Dátový set:** Je ďalším zo spôsobov ako prezentovať výsledky zhlukovania. Predstavuje vstupné dáta obohatené o hodnotu, udávajúcu náležitosť danej vzorky ku zhluku s jednoznačným identifikačným číslom. Tento set je vložený do väčšej tabuľky a obsahuje, okrem jednotlivých atribútov vybraného nízkoúrovňového rysu vstupných vzoriek, aj tri doplňujúce hodnoty. Prvou z nich je identifikačné číslo zhodné z číslom uvedeným pri načítaní vstupných dát. Druhá hodnota udáva, ku ktorému zo zhlukov daná vzorka patrí a napokon tretia hodnota vyjadruje príslušnosť vzorky ku konceptu. Výsledná tabuľka sa dá radiť pomocou metódy *sortBy* podľa zhlukov, ako je ukázané na obrázku 5.5, alebo podľa identifikačného čísla.

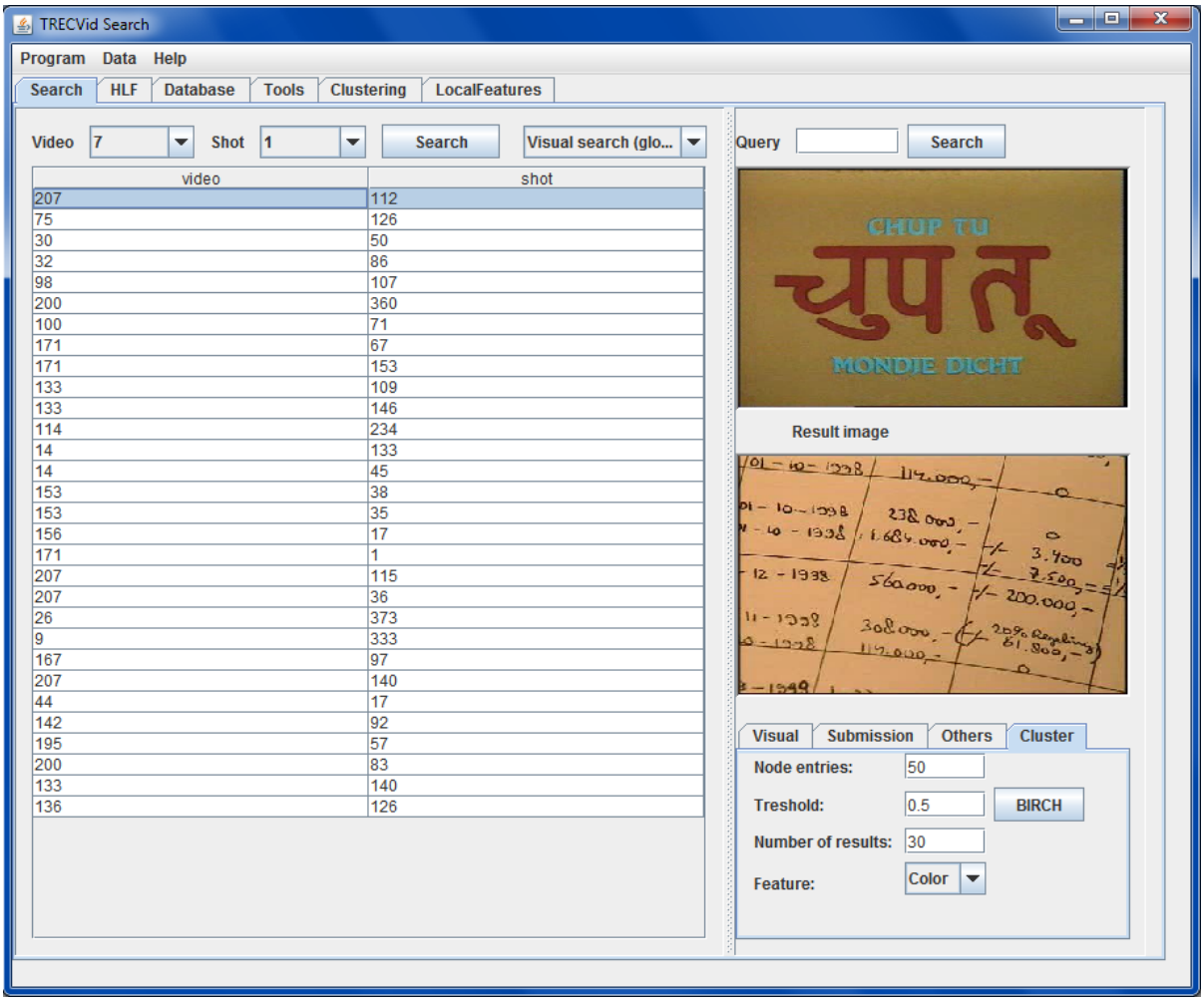
Obe tabuľky je možné roztiahnuť pomocou triedy *TableColumnAdjust* na požadovanú veľkosť. Rovnako existuje možnosť výsledky oboch tabuliek exportovať do súboru vo formáte csv. K tomuto účelu je opäť použitá trieda *CsvExport*, využívajúca *CSVWriter*. Takto uložené súbory sú vhodné pre opätovné použitie týmto programom, prípadne nástrojmi popísanými v kapitole 5.2.




**Obrázok 5.6 – Okno aplikácie s výsledkami vyhľadávania najpodobnejších vzoriek**

### 5.5.4 Vyhľadávanie

Vyhľadávanie je určené k zobrazeniu vzoriek, ktoré sú, vzhľadom ku náležitosti ku zhluku, najbližšie vybranému bodu. Tento bod je možné určiť v okne, obsahujúcom výsledky zhlukovania zadaním jeho identifikačného čísla. Ďalej je potrebné udať požadovaný počet najbližších bodov. Samotné vyhľadávanie je následne v réžii triedy *ClusterResult*. Tá pomocou vnútornej metódy *computeResults* vyberie z výslednej matice body, ktoré sú v rovnakom zhluku ako vstupný bod a zároveň spĺňajú nasledovnú podmienku. Za najbližšie body sú považované tie, ktorých výsledok, vzniknutý pri výpočte euklidovskej vzdialenosti od daného bodu, má najnižšiu hodnotu. Následne sa tieto body vypíšu pomocou metódy *displayResults* do tabuľky, v okne vytvorenom jej triedou. Do menšej tabuľky sa vložia informácie o vyhľadávanej vzorke. Ukážka takto vytvoreného okna je na obrázku 5.6. Vzorky sú zoradené podľa vzdialenosti, od najbližšej po najvzdialenejšiu. Výsledky sú ukladané do maticovej štruktúry triedy *DoubleMatrix2D*, z dôvodu popísanom v nasledujúcej kapitole.




**Obrázok 5.7 – Okno aplikácie s výsledkami vyhľadávania podobných obrázkov**

### 5.5.5 Vyhľadávanie podobných obrázkov

Pri vyhľadávaní podobných obrázkov bolo potrebné upraviť samotnú triedu *TrecvidSearch*. Po dôkladnom zvážení bolo rozhodnuté využiť existujúcu tabuľku *jTableSearch* pre zobrazovanie výsledkov vyhľadávania. Využívajú sa viaceré funkcie pripravené v predchádzajúcich etapách. V prvom rade je vytvorený dotaz do databáze a následne naplnená tabuľka v záložke *Clustering*,

všetkými vzorkami vyhovujúcimi datasetu s číslom 821. Nasleduje vytvorenie vstupnej matice, ktorá sa neskôr predá zhlukovaciemu algoritmu. Ešte predtým je však potrebné nastaviť požadované parametre metódy BIRCH. Táto metóda bola zvolená vďaka svojej rýchlosti a efektívnosti. Za týmto účelom bola v dolnej časti záložky *Search* vytvorená nová záložka s názvom *Cluster*. Po spustení vyhľadávania sa vykoná nad vývojovým datasetom zhlukovanie pomocou metódy BIRCH. V ďalšom kroku sa zavolá metóda *computeResults* triedy *ClusterResult*. Tá potrebuje ku svojej činnosti identifikačné číslo vyhľadávanej vzorky. Hodnota tohto čísla sa zisťuje zo vstupnej matice, načítanej z databáze, vyhľadaním zadaného videa a snímky. Výstupom metódy *computeResults* je, ako už bolo spomínané vyššie, matica výsledných vzoriek. Z tejto matice sa opäť prevodom cez vstupné dáta zistí video a snímka náležiaca k identifikačným číslam. Tabuľka zobrazujúca výsledky vyhľadávania sa naplní týmito snímkami, ktoré sú napokon dostupné k nahliadnutiu. Ukážka výstupu vyhľadávania podobných obrázkov je na obrázku 5.7.

### 5.5.6 Spustenie aplikácie

Hlavnou triedou aplikácie TRECVid Search je *TrecvidSearch* uložená v balíčku *trecvid.search*. Pomocou tejto triedy sa spúšťa a zobrazuje celá aplikácia. Ostatné okná vytvorené v rámci tejto diplomovej práce sú spúšťané práve z tejto triedy.

Používané vývojové prostredie dokáže kompilovať aplikáciu do archívu s príponou jar. Tento archív je spustiteľný na ktoromkoľvek operačnom systéme vybavenom tzv. Java Virtual Machine. Podmienkou spustenia programu je však dostupnosť všetkých dodatočných knižníc. Tieto knižnice sú uložené v zložke lib, ktorá musí byť dodávaná spolu so spustiteľným jar archívom.

# 6 Experimenty

Cieľom tejto kapitoly je popísať experimenty vykonané nad dátovou sadou Sound and Vision 2008 zo súťaže TRECVid, ktorá je detailnejšie popísaná v kapitole 5.1.2. Pri experimentoch sa používali štyri nízkoúrovňové rysy. Boli to, škálovateľné rozloženie farby v obrázku, deskriptor založený na histograme, deskriptor založený na viacstupňovom gradiente a deskriptor textúr pomocou Gaborových filtrov. Všetky tieto rysy budú v ďalšom texte uvádzané ako color, hist, grad a gabor. Ich veľkosť je však rozdielna a tiež sa líšia rozmedzím a rozložením hodnôt svojich atribútov. Color je rozčlenený na 49, hist na 144, grad na 40 a napokon gabor na 31 atribútov. Nízkoúrovňové rysy hist a grad navyše obsahujú nulové hodnoty v niektorých z atribútov a preto je vo všeobecnosti ich výpočet náročnejší.

Pre experimenty nad dátovou sadou sa používali dve množiny vstupných dát. Prvou množinou je celá vývojová množina dátovej sady Sound and Vision 2008. Táto množina obsahuje 36262 dátových vzoriek. Druhá množina obsahuje len tie vzorky, ktoré boli anotované aspoň do jedného konceptu. Pre jednoduchosť je pri viacerých konceptoch, náležiacich tej istej vzorke, braný v úvahu len jeden z nich. Takéto vzorky sa tiež nazývajú pozitívne a je im priradená hodnota v rozmedzí 1 až 20. Naproti tomu vzorky z prvej množiny, ktoré sú bez anotácie (tzv. negatívne vzorky), majú priradenú hodnotu 0. Veľkosť druhej množiny je 8352 dátových vzoriek.

Pre každú metódu bolo vytvorených a následne vykonaných niekoľko druhov experimentov. Tieto experimenty budú v nasledujúcich kapitolách zhrnuté. Na záver bude ukázaná funkčnosť vyhľadávania podobných obrázkov na základe jednotlivých nízkoúrovňových rysov.

## 6.1 Metóda DBSCAN

Pre metódu DBSCAN boli navrhnuté tri druhy testov. Prvý z nich má za úlohu názorne ukázať aký dôležitý je pre túto metódu parameter epsilon. Druhým testom je porovnanie výstupných dát z nástroja RapidMiner a výstupu z vytvorenej aplikácie, a tretím testom je zisťovanie časovej závislosti algoritmu.

### 6.1.1 Určenie parametru epsilon

Parameter $\varepsilon$ je prakticky jedinou potrebnou hodnotou pre správny výpočet algoritmu DBSCAN. Ak sa zvolí príliš nízke číslo, metóda označí veľké množstvo vstupných dát za šum. Na druhej strane, ak je táto hodnota vysoká, väčšina bodov bude priradená jedinému zhluku. Druhý z parametrov metódy DBSCAN sa vo veľkej miere považuje za zbytočný. Hlavným kritériom pre úspešné zhlukovanie na základe hustoty, je totiž vytvorenie čo najväčšieho množstva zhlukov. Toto kritérium zohrávalo úlohu aj pri tomto teste. Preto sú v tabuľke 6.1, reprezentujúcej výsledky tohto experimentu, zvlášť vyznačené výsledky, ktoré ho spĺňajú.

Pri samotnom vykonávaní tohto experimentu bolo možné postupovať čiastočne systematicky. Pre každý zo štyroch nízkoúrovňových rysov bolo určené náhodné číslo a prebehol algoritmus nad pozitívnymi vzorkami. Pri neuspokojivých výsledkoch sa zvolili nové čísla a bolo opätovne vykonané zhlukovanie. Týmto spôsobom sa vylúčili najnepravdepodobnejšie hodnoty parametru epsilon. Napokon bolo potrebné zmenšovať interval hodnôt, pokiaľ nebol dosiahnutý požadovaný výsledok. Vo všetkých prípadoch bolo určené, že minimálny počet bodov vo výsledných zhlukoch musí byť 2. V tabuľke 6.1 je zachytená záverečná časť určovania hodnoty parametra epsilon pre každý

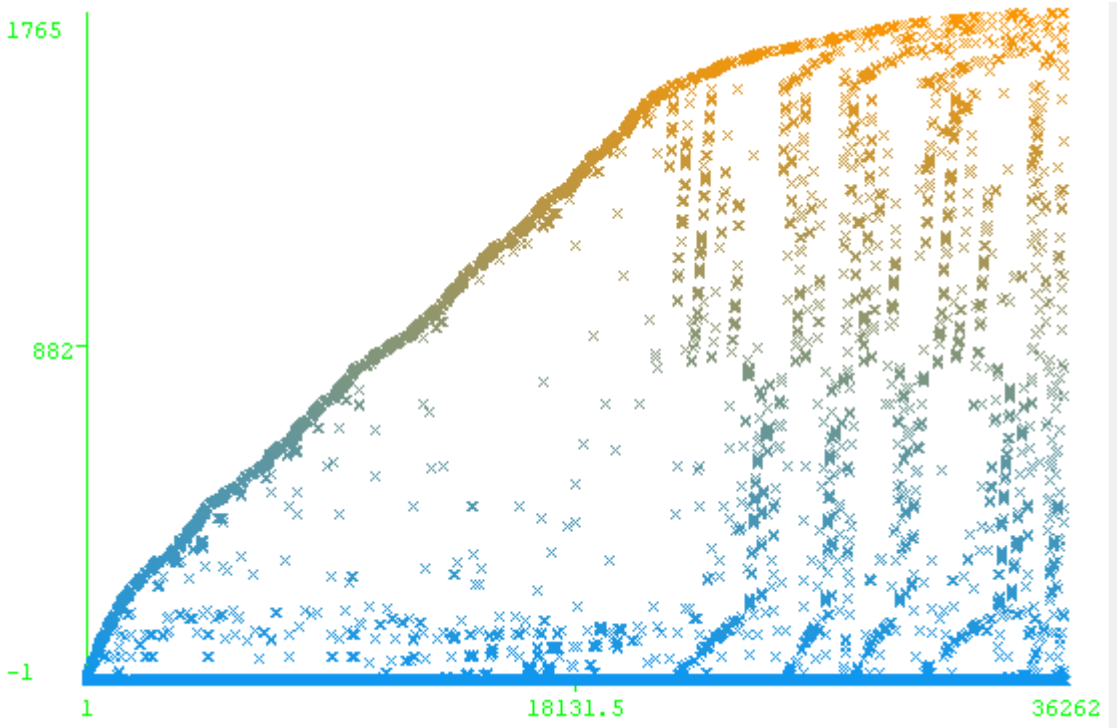
z nízkoúrovňových rysov. V prvej časti sú hodnoty epsilon a v druhej počet zhlukov vytvorených pri zhlukovaní metódou DBSCAN.

| COLOR | | HIST | | GRAD | | GABOR | |
|---|---|---|---|---|---|---|---|
| **epsilon** | **zhluky** | **epsilon** | **zhluky** | **Epsilon** | **zhluky** | **epsilon** | **zhluky** |
| 59,0 | 378 | 1 050 000 | 504 | 2 100 000 | 372 | 30,00 | 624 |
| 60,0 | 382 | 1 100 000 | 507 | **2 200 000** | **397** | 30,50 | 638 |
| 60,5 | 383 | **1 150 000** | **511** | 2 300 000 | 395 | 30,75 | 639 |
| **61,0** | **386** | 1 175 000 | 507 | 2 350 000 | 376 | **31,00** | **642** |
| 62,0 | 377 | 1 200 000 | 497 | 2 500 000 | 394 | 32,00 | 618 |

**Tabuľka 6.1 – Závislosť počtu zhlukov od hodnoty epsilon**

### 6.1.2 Porovnanie výsledkov

Nástroj RapidMiner poskytuje dva druhy výsledkov zhlukovej analýzy, dátový set a zhlukový model. Podobné výsledky poskytuje aj vytvorená aplikácia, a preto sa naskytla možnosť porovnať výstupy oboch programov. DBSCAN je jedna z mnohých metód poskytovaných RapidMinerom, avšak ako jediná je vhodná pre porovnávanie, pretože pri opätovnom spustení sú výsledky vždy rovnaké. Experimenty prebiehali pre rôzne hodnoty epsilon a s rôznymi vstupnými dátami. Pri všetkých bolo preukázané, že oba programy poskytujú zhodný počet a obsah zhlukov.




**Obrázok 6.1 – Graf závislosti identifikačného čísla na číslach zhlukov**

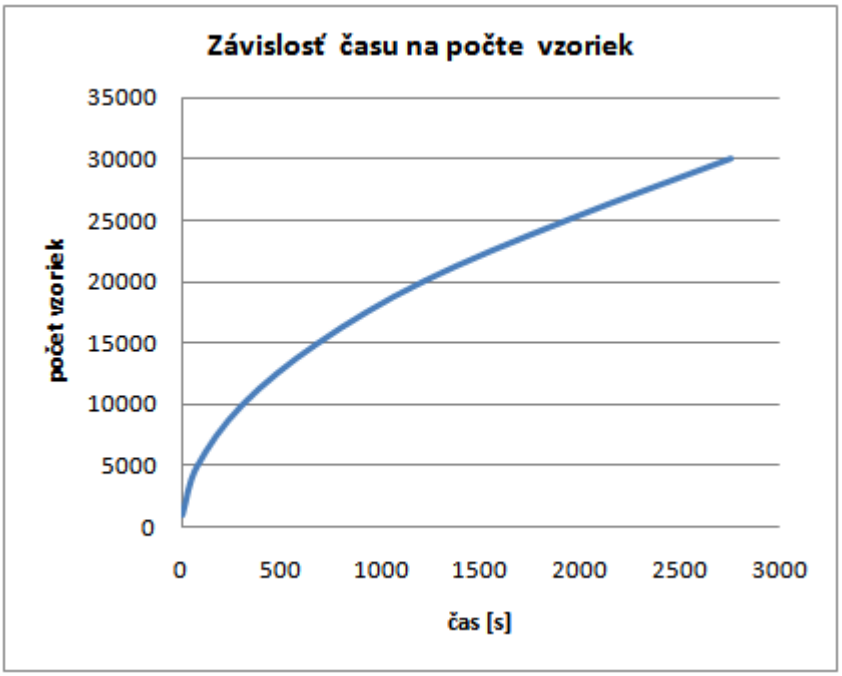
Názorná ukážka výstupov oboch programov by presiahla rozsah tejto práce. Zaujímavým úkazom je však fakt, že vzorky s podobným, myslené približne rovnako veľkým, identifikačným číslom, majú tendenciu vytvárať zhluky. Tento úkaz je možné demonštrovať pomocou nástroju Weka, ktorý umožňuje zobraziť dáta z formátu arff v podobe rozličných grafov. Na obrázku 6.1 je

ukážka výstupu tohto programu a zároveň dôkaz platnosti tvrdenia o danom úkaze. Experiment produkujúci dáta obsiahnuté v tomto obrázku, bol vykonaný na celej dátovej množine, pomocou nízkoúrovňového rysu color a s hodnotou epsilon 50.

### 6.1.3 Časová závislosť DBSCAN

Meranie časovej závislosti je značne náročná úloha, pretože na každom počítači a operačnom systéme sa čas vykonávania, nielen metódy DBSCAN, líši. Preto je potrebné robiť testy týkajúce sa času v rovnakom prostredí. Využíva sa pri tom nástroj RapidMiner, ktorý vďaka svojej schopnosti zaznamenávať logovacie súbory po každej práci s dátami, dokáže relatívne ľahko poskytnúť aj údaje o dobe trvania výpočtu.

Tento experiment zisťuje závislosť času na počte vzoriek predaných zhlukovacej metóde vo forme vstupných dát. Prekvapivo sa ukázalo, že dĺžka trvania zhlukovej analýzy nie je v prípade metódy DBSCAN závislá ani od počtu atribútov, ani od zvoleného epsilon. Jediná závislosť času, teda naozaj existuje len na počte vstupných dát. Na obrázku 6.2 je graf závislosti času na počte vzoriek, pre nízkoúrovňový rys color.




**Obrázok 6.2 – Graf závislosti času na počte vzoriek pre algoritmus DBSCAN**

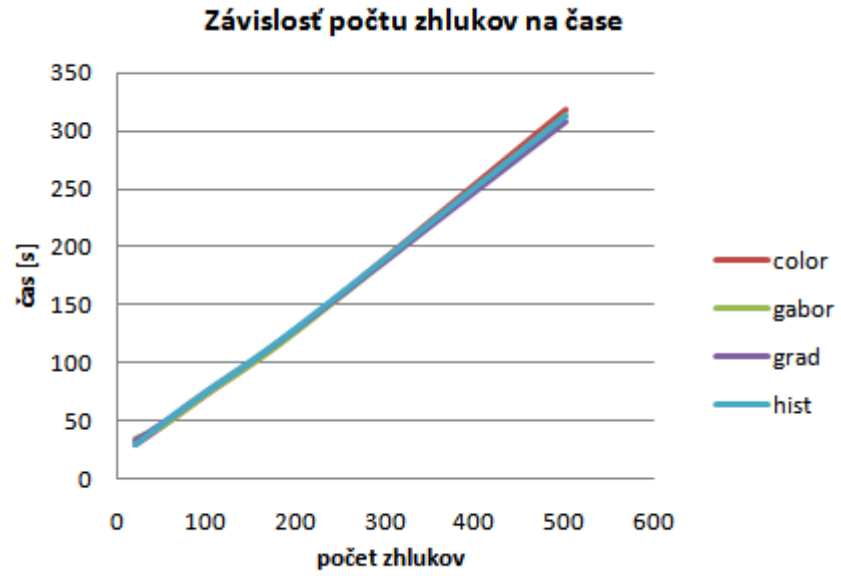
## 6.2 Metóda K-Means

Metóda K-Means bola testovaná dvoma rôznymi spôsobmi. Prvý z testov je podobný predchádzajúcemu testu metódy DBSCAN a ide o časovú závislosť na rôznych parametroch. Druhý test vyjadruje presnosť algoritmu, určenú podľa počtu vzoriek v zhluku s priradeným rovnakým konceptom.

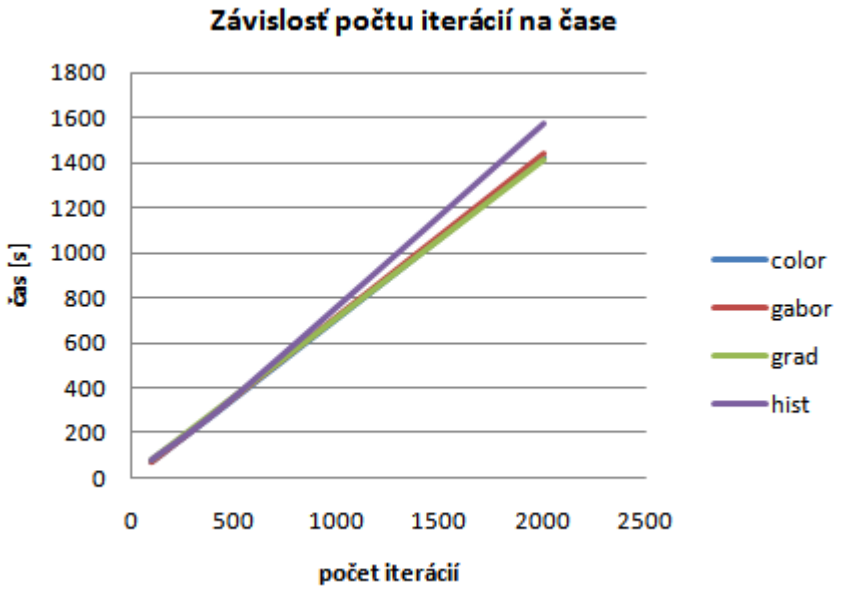
### 6.2.1 Časová závislosť K-Means

Tak isto ako v prechádzajúcom prípade bol k vykonaniu tohto experimentu použitý nástroj RapidMiner. Pri algoritme K-Means je však vhodnejšie testovať dĺžku vykonávania procesu

v závislosti na zadaných vstupných parametrov. Ako sa totiž ukázalo, tie majú značný vplyv nielen na čas potrebný k vykonaniu zhlukovania, ale aj na výpočtový výkon potrebný k jeho úspešnému dokončeniu. Testované boli všetky štyri nízkoúrovňové rysy, pre prípad, že by mala ich veľkosť určitý vplyv na konečnú podobu experimentu. Časová závislosť pri oboch parametroch algoritmu K-Means má približne lineárny priebeh. Výsledky experimentu sú na obrázkoch 6.3 a 6.4.




**Obrázok 6.3 – Graf závislosti počtu zhlukov na čase pre algoritmus K-Means**




**Obrázok 6.4 – Graf závislosti počtu iterácií na čase pre algoritmus K-Means**

### 6.2.2 Presnosť K-Means

Presnosť výpočtu algoritmu je daná vzorcom 6.1, kde $v_i$ je počet najčastejšie vyskytovaných konceptov v zhluku $i$ a $c$ je celkový počet vzoriek vo vstupných dátach.

$$presnosť = \frac{\sum_{i=1}^{n} v_i}{c} \tag{6.1}$$

Táto hodnota je vo vytvorenej aplikácii využívaná pri zobrazovaní výsledkov. Je vhodná pre vyhodnotenie celkovej úspešnosti zhlukovej analýzy. V prípade algoritmu K-Means bolo uskutočnených viacero pokusov zameriavajúcich sa na presnosť, avšak nie všetky priniesli úspech. Využívali sa všetky štyri nízkoúrovňové rysy a rovnako aj oboje dátové množiny. Presnosť zhlukovania nad všetkými, teda aj pozitívnymi aj negatívnymi, dátami sa ukázala byť nedostačujúca k tomu, aby bolo možné prehlásiť tento algoritmus za presný. Vo väčšine prípadov bola menšia ako 10%. Množina obsahujúca iba pozitívne vzorky však vykazovala prekvapivo vysokú presnosť presahujúcu 40%. Testovali sa oba parametre algoritmu, vždy po desiatich pokusoch na dvojicu parametrov, rys a dátovú množinu. Prezentované sú priemerné hodnoty výsledkov.

Parameter, určujúci počet iterácií, mal na výsledky zhlukovania iba minimálny dosah. Počiatočné testy pracovali s hodnotami 10, 50, 100, 200 a 500 pre počet iterácií a 50 pre počet zhlukov. Následne, po predchádzajúcom zistení, boli hodnoty ponechané na stálej hodnote 10. Priemerné presnosti pre jednotlivé nízkoúrovňové rysy, sú uložené v tabuľke 6.2, v ktorej sú množiny pozitívnych vzoriek označené veľkým P na začiatku názvu. Druhý parameter, udávajúci očakávaný počet zhlukov, už vykazoval pomerne väčší stupeň závislosti na celkovej presnosti. Výsledky týchto zistení sú uložené v tabuľke 6.3, ktorá je však, z dôvodu rozsiahlosti obmedzená na rysy grad a color. Práve zhlukovanie pomocou týchto dvoch rysov bolo najviac, prípadne najmenej presné.

| Feature | color | Pcolor | hist | Phist | grad | Pgrad | gabor | Pgabor |
|---|---|---|---|---|---|---|---|---|
| **priemer [%]** | 8,87 | 38,64 | 9,11 | 39,79 | 9,41 | 40,92 | 9,04 | 39,25 |

**Tabuľka 6.2 – Priemerné hodnoty presnosti pri premenlivom počte iterácií**

| počet zhlukov | color [%] | color_pos [%] | grad [%] | grad_pos [%] |
|---|---|---|---|---|
| **10** | 8,74 | 37,98 | 8,88 | 38,18 |
| **100** | 9,20 | 39,56 | 9,75 | 42,21 |
| **200** | 9,56 | 42,25 | 10,01 | 44,11 |
| **500** | 10,24 | 45,55 | 10,68 | 47,32 |
| **1000** | 11,24 | 49,96 | 11,63 | 51,41 |
| **2000** | 12,58 | 57,45 | 12,84 | 58,69 |
| **5000** | 15,04 | - | 15,59 | - |
| **10000** | 17,53 | - | 17,97 | - |

**Tabuľka 6.3 – Priemerné hodnoty presnosti pri premenlivom počte zhlukov**

## 6.3 Metóda BIRCH

Implementovaná metóda BIRCH má dva módy použitia, čomu zodpovedajú aj testy. Prvým z nich je samostatné uloženie vstupných dát do štruktúry CF-Tree, ktorej listové uzly reprezentujú zhluky a ich obsahom sú jednotlivé vzorky priradené týmto zhlukom. Druhou možnosťou použitia je zadanie počtu výsledných zhlukov. V tomto prípade sa použije na listové uzly metóda K-Means, ktorá zredukuje ich počet na požadovanú hodnotu.

### 6.3.1 Presnosť BIRCH

Vzorec na výpočet presnosti 6.1 sa využíva pri výstupoch všetkých metód a BIRCH nie je výnimkou. Rovnako ako pri K-Means boli predmetom testovania oba požadované parametre. Experimenty zistili, že treshold má minimálny dopad na celkovú presnosť. Maximálna odchýlka sú dve stotiny na každú pripočítanú jednotku k tejto hodnote. Naproti tomu má treshold vplyv na celkovú veľkosť stromu. Pri nižších hodnotách sa vytvorí viac listov ako pri hodnotách vyšších. Avšak aj tento rozdiel sa pohybuje rádovo v jednotkách. Navyše je potrebné si uvedomiť, že samotná štruktúra CF-Tree upravuje hodnotu tresholdu dynamicky, ak sa daný strom nevojde do pamäte. Druhý parameter, udávajúci maximálny počet synovských uzlov, prípadne hodnôt v liste, je dôležitý z hľadiska celkovej veľkosti stromu. Čím menšia je táto hodnota, tým je vytvorených viacero zhlukov. Experimentmi sa zistilo, že platí priama úmernosť medzi počtom zhlukov a výslednou presnosťou algoritmu. Dosiahnuté výsledky sú zaznamenané v tabuľke 6.4, ktorá opätovne z dôvodu rozsahu obsahuje len nízkoúrovňové rysy gabor a hist.

| *feature* | GABOR | | GABOR_POSITIVE | | HIST | | HIST_POSITIVE | |
|---|---|---|---|---|---|---|---|---|
| | % | počet | % | počet | % | počet | % | Počet |
| **5** | 18,76 | 11 640 | 58,63 | 2 628 | 19,08 | 12 978 | 62,31 | 2 958 |
| **10** | 15,77 | 5 928 | 50,29 | 1 330 | 16,32 | 6 799 | 53,22 | 1 531 |
| **20** | 13,20 | 2 913 | 44,90 | 659 | 13,74 | 3 367 | 47,39 | 753 |
| **50** | 10,97 | 1 140 | 41,39 | 260 | 11,53 | 1 275 | 42,78 | 293 |
| **100** | 10,05 | 568 | 40,23 | 128 | 10,32 | 615 | 40,88 | 137 |
| **200** | 9,54 | 291 | 38,92 | 64 | 9,73 | 308 | 39,70 | 70 |
| **500** | 9,10 | 108 | 38,67 | 25 | 9,28 | 116 | 39,26 | 31 |
| **1000** | 9,04 | 58 | 38,41 | 15 | 9,08 | 59 | 38,19 | 14 |

**Tabuľka 6.4 – Hodnoty presností pri premenlivom počte parametra node_entry**

### 6.3.2 Porovnanie presnosti BIRCH a K-Means

BIRCH algoritmus je schopný, ako už bolo naznačené, zredukovať počet vytvorených zhlukov na požadovaný počet, pomocou metódy K-Means. Vďaka tejto vlastnosti je možné tieto dva algoritmy porovnať. Pre porovnanie je dostupná jediná metrika, a tou je presnosť. Nad oboma metódami boli prevedené experimenty s rôznym počtom požadovaných výstupov. Využívali sa všetky štyri nízkoúrovňové rysy. Parametre BIRCH boli nastavené na treshold rovný 1,0 a node_entry sa rovnalo 50. K-Means používal 100 iterácií. Výsledky tohto experimentu sú v tabuľke 6.5, pričom vždy v prvom riadku sa nachádza presnosť K-Means a v druhom BIRCH. Zaujímavým faktom je menšia presnosť algoritmu BIRCH. Z predchádzajúcich experimentov a ich výsledkov sa zdalo, že táto

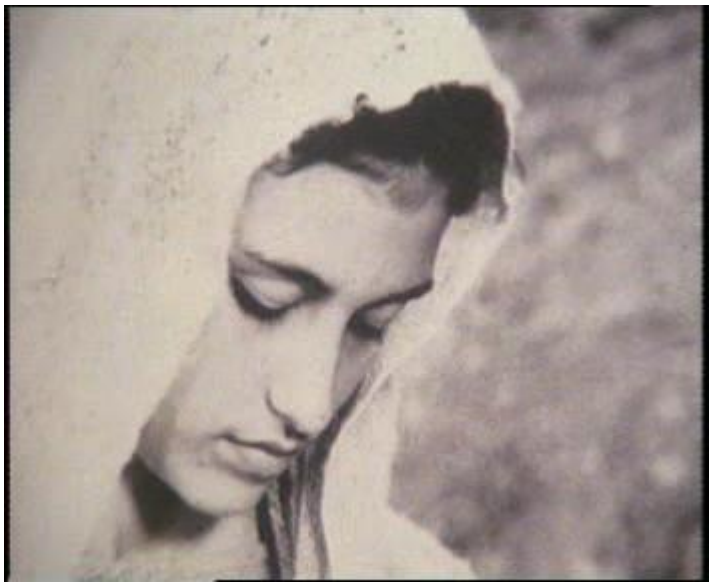
metóda dosiahne najlepšie výsledky. Avšak je potrebné spomenúť jej rýchlosť. Nie je možné vytvoriť hmatateľný dôkaz jej rýchlosti, pretože do aplikácie neboli vložené žiadne funkcie pracujúce s časom a ani nástroje RapidMiner a Weka neimplementujú túto metódu.

| **feature** | | **COLOR** | | **HIST** | | **GRAD** | | **GABOR** | |
|---|---|---|---|---|---|---|---|---|---|
| **hodnoty [%]** | | *positive* | *all* | *positive* | *all* | *positive* | *all* | *positive* | *All* |
| **50** | *K-Means* | 38,43 | 8,85 | 39,89 | 9,15 | 41,09 | 9,36 | 39,02 | 9,01 |
| | *BIRCH* | 38,49 | 8,79 | 39,89 | 9,11 | 40,03 | 9,32 | 39,18 | 9,01 |
| **100** | *K-Means* | 40,19 | 9,22 | 41,03 | 9,35 | 42,35 | 9,64 | 39,51 | 9,22 |
| | *BIRCH* | 38,72 | 9,01 | 40,59 | 9,17 | 41,26 | 9,38 | 39,62 | 9,16 |
| **500** | *K-Means* | 45,38 | 10,34 | 45,91 | 10,46 | 47,45 | 10,75 | 45,06 | 10,21 |
| | *BIRCH* | 41,71 | 9,69 | 42,56 | 10,12 | 43,68 | 10,34 | 41,21 | 9,83 |

**Tabuľka 6.5 – Porovnanie presnosti algoritmov K-Means a BIRCH**

## 6.4 Vyhľadávanie podobných obrázkov

Jednou z funkcií vytvoreného programu je vyhľadávanie podobných obrázkov. Táto funkcionalita vyberie najbližšie vzorky, v zmysle euklidovskej vzdialenosti pomocou metódy BIRCH. V praxi to znamená, že aplikácia vyhľadá, na základe zadaného nízkoúrovňového rysu, najpodobnejšie obrázky. Implementovaná je len funkčnosť na vývojovú sadu dát Sound and Vision 2008, a nie je preto možné vyhľadávať v testovacej sade. Výsledok takéhoto vyhľadávania sa nedá nijako overiť, pretože je zaručené, že vybratá vzorka je skutočne najbližšie k vstupnému bodu. Preto pri zobrazení vzorky je na užívateľovi, aby subjektívne posúdil správnosť výberu. Obrázok 6.5 bol zadaný pri jednom z experimentov, ako vstup vyhľadávania. Ďalšie obrázky 6.6 a 6.7 predstavujú najbližšie snímky z hľadiska všetkých štyroch rysov.



**Obrázok 6.5 – Obrázok určený pre vyhľadávanie**


**Obrázok 6.6 – Výsledné obrázky vyhľadávania rysu color vľavo a rysu hist vpravo**

**Obrázok 6.7 – Výsledné obrázky vyhľadávania rysu grad vľavo a rysu gabor vpravo**

# 7 Záver

Témou tejto práce je získavanie znalostí z multimediálnych databáz. Teoretický základ, z ktorého sa vychádza bol vytvorený v rámci semestrálneho projektu. V úvodnej časti je tu načrtnutá povaha úloh zameraných na problematiku získavania znalostí z databáz vo všeobecnosti. Nosnou témou práce bolo zhlukovanie, inak nazývané aj zhluková analýza. Jednotlivé metódy a postupy, ktoré využíva, sú podrobne popísané v kapitole 3. V práci sa nachádza aj úvod do oblasti dolovania multimediálnych dát, ktorý je zameraný hlavne na nízkoúrovňové rysy, používané pre popis obrázkov a video dát. Z týchto rysov sú popísané v kapitole 4.3 len tie najzákladnejšie, ako farba, textúra, tvar a umiestnenie. V praktickej časti sú však využívané rysy založené na škálovateľnom rozložení farby v obrázku, deskriptory vytvorené na základe histogramu, deskriptory používajúce viacstupňový gradient a napokon deskriptory popisujúce textúry pomocou Gaborových filtrov.

Implementačná časť práce sa zaoberá rozšírením aplikácie Trecvid Search, ktorá je vyvíjaná na Fakultě informačních technológií VUT v Brně. Toto rozšírenie prináša do aplikácie nové prvky, majúce za úlohu vykonávať zhlukovú analýzu. Týmito prvkami sú algoritmy BIRCH, DBSCAN a k-means, ktoré boli zvolené ako štandardný zástupcovia jednotlivých kategórií zhlukovacích metód. Tieto algoritmy sú implementované ako samostatné triedy, prijímajúce vstupné dáta a parametre, vďaka čomu je možné ich budúce využitie v nadväzujúcich rozšíreniach, prípadne programoch. Ďalšou pridanou funkciou je možnosť vyhľadávania podobných obrázkov na základe nízkoúrovňových rysov a zobrazenie výsledkov priamo v aplikácii. Túto možnosť je však nutné detailnejšie prepracovať, pretože následkom zlyhania databázového systému, nebola k dispozícii kompletná databáza. Z tohto dôvodu bola do aplikácie pridaná možnosť načítania a priradenia anotačných dát jednotlivým vzorkám dátovej sady.

Záverečná časť je venovaná experimentom nad dátovou sadou Sound and Vision 2008 súťaže TRECVid. Z dôvodu časovej náročnosti úloh boli všetky testy robené len nad vývojovou podmnožinou tejto dátovej sady. Avšak aj tá bola dostačujúca, aby sa dalo prehlásiť, že jednoznačne najlepším implementovaným algoritmom pre zhlukovú analýzu multidimenzionálnych dát je BIRCH. Nedosahuje kvalitu algoritmu k-means, ak je ten dobre nastavený, ale vynahrádza to neuveriteľnou rýchlosťou výpočtu. Jediným obmedzujúcim faktorom je veľkosť priradenej operačnej pamäte. Pomôckou pri jeho implementácii bol aj balíček JBIRCH, vydávaný pod licenciou GNU GPL. Tento balíček dokáže vytvárať hĺbkovo vyvážené CF-stromy. Niektoré metódy tried tohto balíčka však museli byť upravené pre optimalizovanejší výkon vo vytvorenej aplikácii. Pri experimentovaní bol tiež využitý nástroj RapidMiner, ktorý poskytoval potrebné doplňujúce funkcie, ako napr. zisťovanie času z logovacích súborov, či vytváranie výsledkov slúžiacich ako referenčné riešenia daných zhlukovacích metód.

Možným rozšírením tejto práce by mohla byť implementácia ďalších metód zhlukovej analýzy, prípadne doladenie vyhľadávacej funkcie tak, aby spolupracovala priamo s celým programom. To by prinieslo užívateľsky prívetivejšie prostredie a jednoduchšie nastavenie programu. Optimalizácia implementovaných algoritmov, pre rýchlejšie štruktúry, ako súčasne využívané matice by bola rovnako vítaným pokračovaním tejto diplomovej práce.

# Literatúra


[1] ANKERST, M.; BREUNIG, M. M.; KRIEGEL, H.-P. a kol. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA, 1999 [cit. 2011-01-06], s. 49-60. Dostupný z WWW: < http://www2.cs.uh.edu/~ceick/7363/Papers/dbscan.pdf >.

[2] BERKA, P. *Dobývání znalostí z databází*. Praha: Academia, 2003. 366 s. ISBN 80-200-1062-9.

[3] COLT [online]. c1999 [cit. 2011-05-20]. Dostupný z WWW: <http://acs.lbl.gov/software/colt/>.

[4] *Guidelines for the TRECVID 2008 Evaluation* [online]. 2007 , 5.1.2009 [cit. 2011-05-18]. Dostupný z WWW: <http://www-nlpir.nist.gov/projects/tv2008/tv2008.html>.

[5] HAN, J.; KAMBER, M. *Data Mining: Concepts and Techniques*. second Edition. San Francisco (California): Morgan Kaufmann Publishers, 2006. 770 s. ISBN 1-55860-901-6.

[6] HINNEBURG, A.; KEIM, D. A. *An Eficient Approach to Clustering in Large Multimedia Databases with Noise. In Knowledge Discovery and Data Mining*, 1998 [cit. 2011-01-06], s. 58-65. Dostupný z WWW: < http://www.aaai.org/Papers/KDD/1998/KDD98-009.pdf >.

[7] HUNG, L. *Birch: An efficient data clustering method for very large databases*. Elektronický učebný text. Atlanta, Georgia Institute of Technology, [cit. 2011-01-08]. Dostupný z WWW: < www.cc.gatech.edu/~lingliu/courses/cs4440/notes/**Lai**.**ppt** >.

[8] CHMELAŘ, P.; STRYKA, L. *Multimediální databáze*. Elektronický učebný text. Brno, FIT VUT v Brně, 2008.

[9] CHMELAŘ, P. *Vyhledávání TRECVid 2008*, elektronický učební text [online]. 2008. [cit. 2011-09-01]. Dostupný z WWW: < http://www.fit.vutbr.cz/research/pubs/TR/2008/sem_uifs/s081020slidy1.pdf > .

[10] CHMELAŘ, P. *Získávání znalostí z multimediálních databází*. Elektronický učebný text. Brno, FIT VUT v Brně, 2008.

[11] JAVA TIPS WEBLOG*: Table Column Adjuster* [online]. 2008 [cit. 2011-05-20]. Dostupný z WWW: <http://tips4java.wordpress.com/2008/11/10/table-column-adjuster/>.

[12] JBIRCH – *Roberto Perdisci* [online]. 2011 [cit. 2011-05-21]. Dostupný z WWW: < http://roberto.perdisci.com/projects/jbirch>.

[13] JIRMÁSEK, T. *Získávání znalostí z multimediálních databází*. Diplomová práca. Brno, FIT VUT v Brně, 2009.

[14] KOLÁŘ, D. *Pokročilé databázové systémy*, Prednášky [online]. 2008-2009 [cit. 2011-01-07]. Dostupný z WWW: < http://www.fit.vutbr.cz/study/courses/PDB/private/ >.

[15] *MPEG-7 Overview (version 10)* [online]. 2004 [cit. 20011-10-01]. Dostupný z WWW: < http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm >.

[16] ORACLE AND JAVA*: Technologies* [online]. c2010 [cit. 2011-05-20]. Dostupné z WWW: <http://www.oracle.com/us/technologies/java/index.html>.

[17] PETRUSHIN, V. A.; KHAN, L. *Multimedia Data Mining and Knowledge Discovery*. Springer US, 2007. 521 s. Computer Science. ISBN 978-1-84628-799-2.

[18] POSTGRESQL*: The world's most advanced open source database* [online]. c1996 . [cit. 2011-5-18] Dostupný z WWW: <http://www.postgresql.org/>.

[19] RAPID – I - RAPIDMINER*: Community edition* [online]. 2000 . [cit. 2011-5-18] Dostupný z WWW: < http://rapid-i.com/>.

[20] RAPIDMINER 5.0 *: Manual*. 2010. Dostupný z WWW: <http://switch.dl.sourceforge.net/project/rapidminer/1.%20RapidMiner/5.1/>.

[21] SIKORA, T. *The MPEG-7 Visual Standard for Content Description – An Overview*. Interactive Media – Human Factors, Berlin, Germany, 2001 [cit. 2011-01-10], s. 7. Dostupný z WWW: < www.ee.columbia.edu/~sfchang/course/vis/REF/sikora01.pdf >.

[22] SMITH, G.: *Opencsv* [online]. 2009 [cit. 2011-05-21]. Dostupný z WWW: <http://opencsv.sourceforge.net/>.

[23] ŠABATKA, P.: *Vyhledávání informací*. diplomová práce, FIT VUT v Brně, 2010.

[24] WEKA 3 – *Data Mining with Open Source Machine Learning Software in Java* [online]. 2011 [cit. 2011-05-21]. Dostupný z WWW: <http://www.cs.waikato.ac.nz/ml/weka/>.

[25] WIKIPEDIA: PostgreSQL | Wikipedia, The Free Encyclopedia. [online], 2011, [cit. 2011-5-18]. Dostupný z WWW < http://cs.wikipedia.org/wiki/PostgreSQL>

[26] XU, R.; WUNSCH, D. *Clustering*. Hoboken : John Wiley & Sons, Inc., 2009. 358 s. ISBN 978-0-470-27680-8.

[27] ZENDULKA, J. a kol. *Získávání znalostí z databází*. Elektronický učebný text. Brno, FIT VUT v Brně, 2006.

[28] ZHANG, T.; RAMAKRISHNAN, R.; LIVNY, M. *BIRCH: An Efficient Data Clustering Method For Very Large Databases*. In ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD‘96), Montreal, Canada, 1996 [cit. 2011-01-07], s. 103-114. Dostupný z WWW: < http://www.lans.ece.utexas.edu/course/ee380l/1999fall/papers/list2/p103-zhang.pdf >.

# Zoznam príloh

Príloha 1.DVD

- zdrojové súbory aplikácie
- textová práca vo formáte pdf a docx
- vstupné dáta získané z aplikácie pomocou dotazov
- súbor obsahujúci všetky anotácie
- spustiteľný program so všetkými potrebnými knižnicami
- inštalačné súbory RapidMineru a nástroja Weka