VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

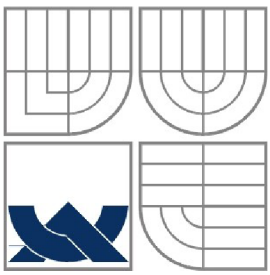
AUTOMATICKÉ TŘÍDĚNÍ FOTOGRAFIÍ DLE OBSAHU

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

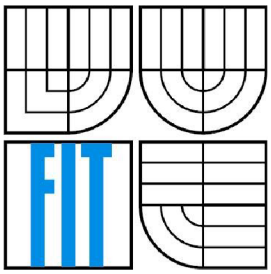
AUTOR PRÁCE
AUTHOR

MARTIN MATUSZEK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ TŘÍDĚNÍ FOTOGRAFIÍ DLE OBSAHU

AUTOMATIC PHOTOGRAPHY CATEGORIZATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN MATUSZEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL

BRNO 2010

Abstrakt

Tato práce se zabývá výběrem metod, návrhem a implementací aplikace schopné automatického třídění fotek dle jejich obsahu do předem daných skupin. Podrobněji se popisují jednotlivé hlavní kroky klasifikace. Vyhledání a popis význačných bodů v obraze metodou SURF, vytvoření vizuálního slovníku metodou k-means, mapování na slova přes strukturu kd-tree. Vytváří se vlastní hodnocení na základě kterého se klasifikuje. Je zde popsáno jak byly jednotlivé kroky implementovány s pomocí knihoven OpenCV a Qt. A taktéž jsou ukázány výsledky pro různá nastavení běhu aplikace a snahy o zlepšení výsledku, kdy aplikace dokáže roztřídit fotky správně, ale úspěšnost je kolísavá.

Abstract

This thesis deal with choosing methods, design and implementation of application, which is able of automatic categorization photos based on its content into predetermined groups. Main steps of categorization are described in greater detail. Finding and description of interesting points in image is implemented using SURF, creation of visual dictionary by k-means, mapping on the words through kd-tree structure. Own evaluation is made for categorization. It is described, how the selected steps were implemented with OpenCV and Qt libraries. And the results of runs of application with different settings are shown. And efforts to improve outcome, when the application can categorize right, but success is variable.

Klíčová slova

automatická kategorizace obrazu, shlukování, význačné body, příznakový vektor, vizuální slovník, SURF, k-means, kd-tree

Keywords

automatic image categorization, clustering, interesting points, feature vector, visual dictionary, SURF, k-means, kd-tree

Citace

Martin Matuszek: Automatické třídění fotografií dle obsahu, bakalářská práce, Brno, FIT VUT v Brně, 2010

Automatické třídění fotografií dle obsahu

Prohlášení

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

……
Martin Matuszek
19.5.2010

Poděkování

Chtěl bych poděkovat svému vedoucímu, Ing. Michalovi Španělovi, za jeho rady, čas a pomoc při řešení bakalářské práce.

Obsah

1 Úvod

Pokud člověk sám často fotí, dostane fotky od kamarádů, nebo jen pracuje s různými obrázky, může se dostat do situace, kdy by potřeboval větší množství obrazových dat roztřídit do různých skupin, aniž by této činnosti věnoval příliš mnoho času. Tato myšlenka zaujala i mne při výběru tohoto tématu. Jelikož ne vždycky je potřeba obrázky rozdělit podle času pořízení, nebo vytvoření, ale klidně i podle samotného obsahu. Za předpokladu, že už máte vytvořených několik skupin, do kterých chcete nové fotky zařadit, i vy sami se při třídění soustředíte na podobné rysy vyskytující se v dané skupině. Jedna skupina fotek se psy, další jenom s květinami a další třeba jenom se západy slunce. Těchto podobností si člověk jednoduše všimne a není pro něj problém určit správnou skupinu. Při větším počtu fotek to ale může být časově náročné a unavující. A právě návrhem a realizací nástroje pro takovouto automatickou kategorizaci jakkoliv velkého objemu fotografií do předem daných tříd se zabývá tato práce.

Cílem je tedy navrhnout aplikaci, která by dokázala správně zařadit nové fotky, o kterých není žádná informace, do odlišných skupin. Ty jsou předem známy a aplikace by měla využít právě fotografie v těchto skupinách k vytvoření určitého postupu rozlišování fotek nových. Je třeba se vypořádat s reprezentací informací o každé fotce, se zjištěním určitých zajímavých oblastí, neboli příznaků společných fotkám dané skupiny. Dále s efektivním řešením přiřazení do skupiny právě na základě zmíněných zajímavých oblastí. A také maximalizováním pravděpodobnosti že vybraná skupina je správná. To vše implementovat zvoleným programovacím jazykem do aplikace, schopné tyto činnosti automaticky vykonat. Práce není zaměřena na porovnávání rozličných metod používaných v různých postupech klasifikace obrazu, ale pouze na využití obvyklých postupů při vlastní realizaci.

Samotné řešení je rozděleno do několika částí. V první fázi je třeba zjistit nějakým způsobem zajímavé oblasti fotky. K tomu je využita metoda pro extrakce příznaků význačných bodů *SURF*. Tak dokážeme každý obrázek popsat daleko menším množstvím dat. Tyto body jsou navíc vždycky pro danou fotku klíčové. Další fází je vytvoření vizuálního slovníku z těchto bodů, pomocí shlukování do předem daného počtu shluků – slov. K tomu je využita metoda *k-means*. Vytváří se jeden slovník společný pro všechny skupiny, tedy ze všech význačných bodů. Následným krokem je na základě vytvořeného slovníku sestavit bodové ohodnocení slov pro každou skupinu. Finálně dochází na řadu třídění nových fotek. Také se v každé detekují význačné body, přiřadí do shluků s využitím metody průchodu *k-dimenzionálních stromů* a spočítá se celkový počet bodů pro každou skupinu. Na základě nejvyššího počtu se vybere nejvhodnější skupina.

Zde jsou představeny následující kapitoly. Jejich obsah a čím se zabývají. 2. kapitola detailně popisuje principy hlavních metod použitých v aplikaci. Kapitola 3 prezentuje „State of Art“, tedy

současný stav v tomto odvětví IT. Poukazuje na postupy, které se v současnosti běžně používají pro řešení problému, kterým se tato práce zabývá. Následující 4. kapitola uvádí konkrétní použitý postup a v jakých krocích bude klasifikace probíhat. 5. kapitola se zaměřuje na implementaci programovacím jazykem C++ a uvádí použité knihovny. Prezentace konkrétních výsledků je představena v kapitole 6. Závěr, shrnutí výsledků a návrh na budoucí postup lze najít v kapitole 7.

2 Teoretická část

Zde se podrobně popisují převzaté metody, použité při návrhu a následné implementaci.

Při rozpoznávání obrazu se využívá různých přístupů, ať už k zjišťování zajímavých oblastí v obrazech, nebo následného zpracování získaných dat. Zde je představeno několik postupů využitých v této práci. Na vyhledání význačných bodů v obraze je využita metoda *SURF*. Následné zpracování popisů těchto bodů se provádí pomocí shlukování, konkrétně metodou *k-means*. A dalším postupem, určeným pro přiřazení příznakového vektoru význačného bodu do odpovídajícího shluku, je procházení vícedimenzionálních prostorů s pomocí *kd-trees*.

2.1 Význačné body

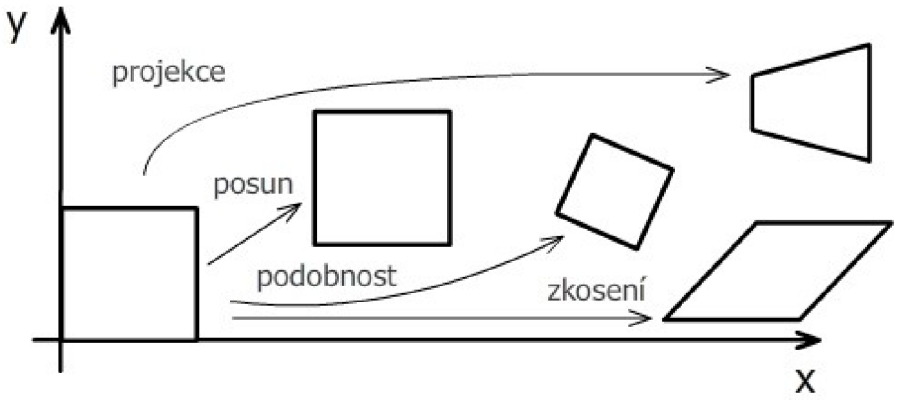
Význačné body a jejich detekce v obraze je jednou z velmi důležitých součástí rozpoznávání podobných či různě upravených obrazů [1]. Význačný bod je druh obrazového příznaku (eng. *feature*). Obecným hlavním cílem obrazových příznaků je z celkového počtu bodů popisujícího obrázek vybrat jenom ty důležité. Tedy prochází se postupně každý bod obrázku a rozhodne se, jestli se jedná o příznak podle okolí daného bodu. Definice jak je najít jsou různé, ale primárně, pokud je v obraze takovýto bod detekován, musí být detekován i v rozdílných obrazech stejné scény.

Obrázek 2.1: Detekce stejných příznaků [1]

Celkově v detekci a extrakci význačných bodů nejde jen o to najít nějak zajímavé body, ale o reprezentaci celého obrazu umožňující popsat v něm obsažené objekty. Význačné body mají obecně následující vlastnosti:

- robustnost – šum, rozmazání, komprese, diskretizace nemají velký dopad na význačný bod
- jsou lokální – není třeba předchozí segmentace obrazu
- rozlišitelnost – individuální význačný bod lze porovnat s jinými v databázi objektů

- počet – mnoho význačných bodů může být generováno i k malým objektům
- přesnost – precizní lokalizace v obraze
- výkonnost – blízké real-time výkonu detekce (závisí na zvoleném typu detekce)
- invariantní k fotometrickým deformacím
- invariantní ke geometrickým transformacím



Obrázek 2.2: Geometrické transformace

Do skupiny obrazových rysů patří spolu s *význačnými body* i *rohy*. Z metod zpracování obrazu pomocí detekce hran se brzy zjistila místa, kde hrany mění ostře svůj směr – detekovaly se rohy. Oproti tomu význačné body nehledají pouze rohy, ale dalo by se říct, že podobné rysy. Místa s intenzivním barevným přechodem, např. bílý bod na černém pozadí, což se nedá považovat za tradiční roh.

2.2 SURF

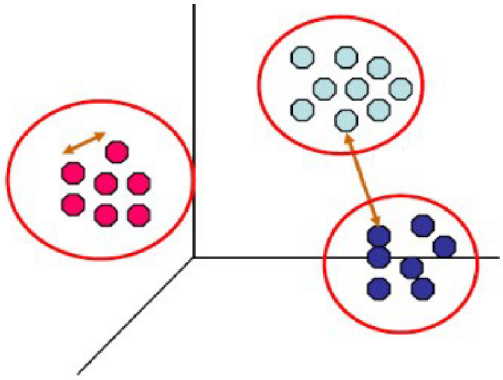
Tento pojem znamená v orig. Speeded-Up Robust Feature. Jedná se o detektor a deskriptor význačných bodů představený roku 2006 [2], který je invariantní vůči rotacím a změně měřítka. Snažil se přiblížit, nebo dokonce překonat tehdejší používané techniky s ohledem na zachování opakovatelnosti (schopnost najít stejné rysy ve více různých obrazech stejné body), robustnosti, rozlišitelnosti a s větší rychlostí. Aby toto mohl dokázat, spoléhá se na integrační obrazce pro obrazovou konvoluci. Využívá silné stránky už představených detektorů a deskriptorů (speciálně Hessiannovy matice pro detekci a Haarovy vlnky pro deskriptor) se zjednodušením těchto metod jen na to nejzákladnější

Je částečně inspirován SIFTem (taktéž metoda na detekci a deskripci význačných bodů [3]). Okolí každého vyhledaného význačného bodu je popsáno příznakovým vektorem, který musí právě splňovat základní kritéria dostatečné robustnosti vůči rušivým vlivům a rozlišitelnosti. Zaměření se na invariantnost pouze vůči změně měřítka a rovinné rotaci, je kvůli tomu, že právě tyto představují

dobrý kompromis mezi komplexností rozeznání a současně robustností. Skosení a perspektivní změny jsou považovány až za druhořadé. Toto „zanedbání“ také dodává SURFu na rychlosti.

2.3 Shlukování

Metody shlukování lze zařadit do počítačového učení – bez učitele [4]. Kdy je třeba rozdělit data do tříd podle zadaného kritéria, ovšem nejsou k dispozici informace od učitele (*non-supervised learning*). A kritérium vlastně stanovuje, že data v jedné třídě jsou si mezi sebou podobnější než data v jiných třídách.



Obrázek 2.3: Různě stanovené kritérium:
Vzdálenost od prvků stejné skupiny?
Nebo mezi nejbližšími prvky různých skupin? [4]

Kritéria mohou být různá:

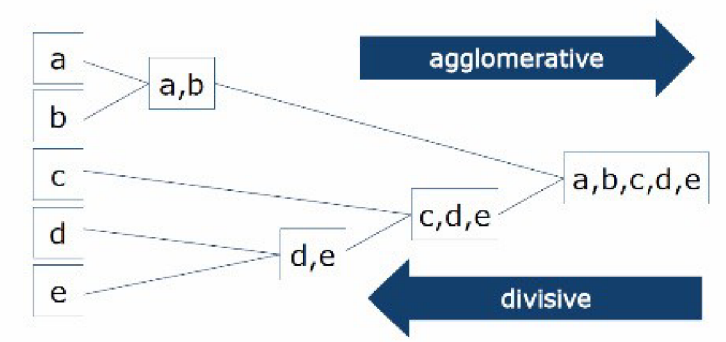
- MIN – podle vzdálenosti k nejbližšímu prvku skupiny
- MAX – podle vzdálenosti k nejvzdálenějšímu prvku skupiny
- CENTROIDS – podle vzdálenosti ke středům shluků
- GROUP AVERAGE – vzdálenost ze všech možných dvojic skupiny
- další...

Jsou zde zmíněny vzdálenosti bodu k jiným, lze použít různé typy: Euclidovskou, Manhattanskou, Cosinovu vzdálenost aj.

Samotné způsoby shlukování se dělí do dvou hlavních větví:

- **Hierarchické shlukování (Hierarchical clustering)** – Vytváří se stromové struktury (tzv. dendrogramy). Není přesně daný počet shluků a to může být výhodou při řešení klasifikace. Mezi nevýhody patří problémy s *outliers (osamocené body)*, ze kterých může vytvořit samostatné shluky a s nestejně velkými shluky.
  - Typ Divisive – Na začátku je jeden velký shluk a postupně dělíme.

- Typ Agglomerative – Jde opačnou cestou. Na začátku tvoří každý vzorek samostatný shluk, vybíráme nejbližší a postupně spojujeme.



Obrázek 2.4: Dendrogram [4]

- **Ne-Hierarchické shlukování (Non-Hierarchical clustering)** – Jedná se o statistické přístupy. Patří sem například metoda *k-means*.

2.4 K-means

Je to postup pro ne-hierarchické shlukování. Kdy chceme určitý počet bodů daných *n*-dimenzionálnímy vektory x_i zařadit do K shluků (*clusterů*), kdy každý shluk je dán *středem* (*centroid*) μ_i taktéž o n dimenzích. K je předem daný počet. Využívá se *Euclidovská* vzdálenost každého bodu do středu shluku. Algoritmus vyhledání středů a přiřazení všech bodů do správných shluků je následující:

1. Náhodně vyber středy shluků μ_i z x_i daných k roztřídění.
2. Klasifikuj vektory x_i vždy do nejbližšího středu $\mu_{i.}$.

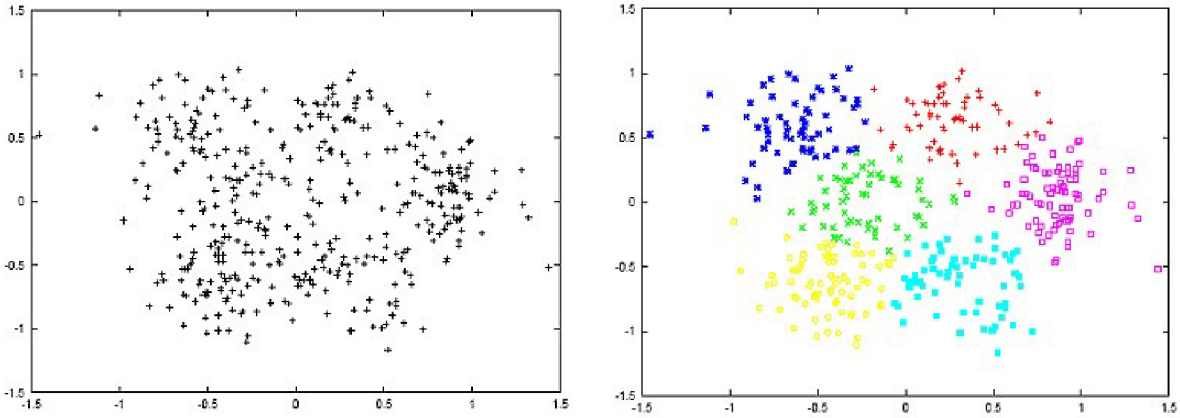
$$y_i = \arg \min_{k \in \{1,..,K\}} \|x_i - \mu_k\|^2 \qquad (2.1)$$

3. Vypočti nové středy shluků.

$$\mu'_k = \frac{1}{N_{y_i=k}} \sum_{i:y_i=k} x_i \qquad (2.2)$$

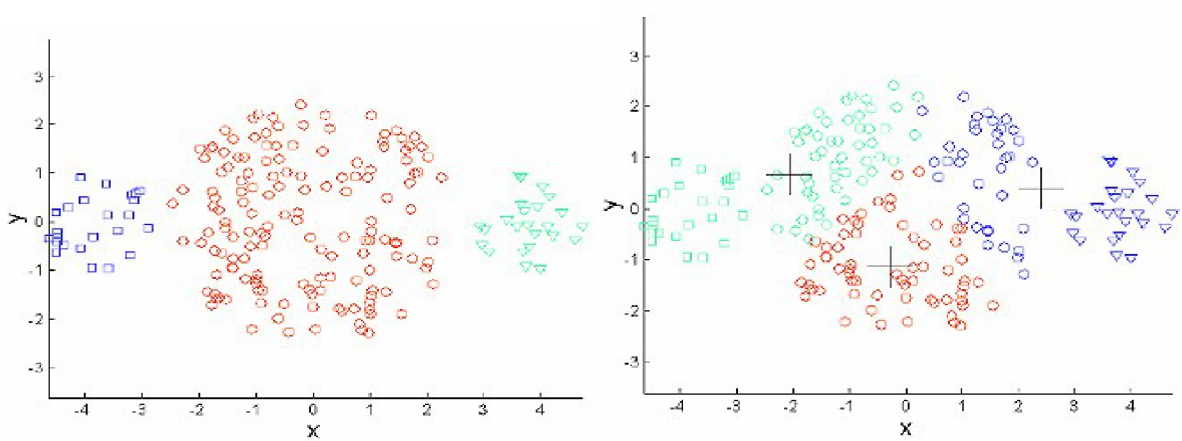
4. Opakuj kroky 2. a 3. dokud se středy shluků mění.

K-means dělí data do kompaktních celků:

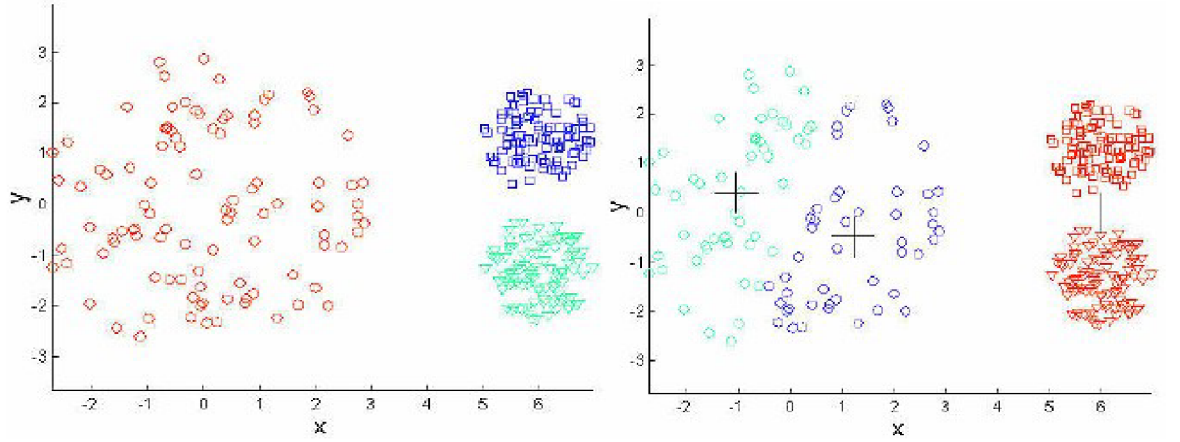


Obrázek 2.5: Ukázka klasifikace bodů do šesti skupin [4]

K-means je už podle algoritmu jednoduchá metoda, která ale dokáže vytvořit dobré výsledky. Ne samozřejmě vždy. Závisí na vstupních datech. Následují ukázky špatné klasifikace, kvůli specifickému rozložení, na které je tato metoda nevhodná. Na ukázkových obrázcích je vždy nejprve vyobrazeno opravdové rozložení skupin a pak rozložení jak jej vytvořil *k-means*.

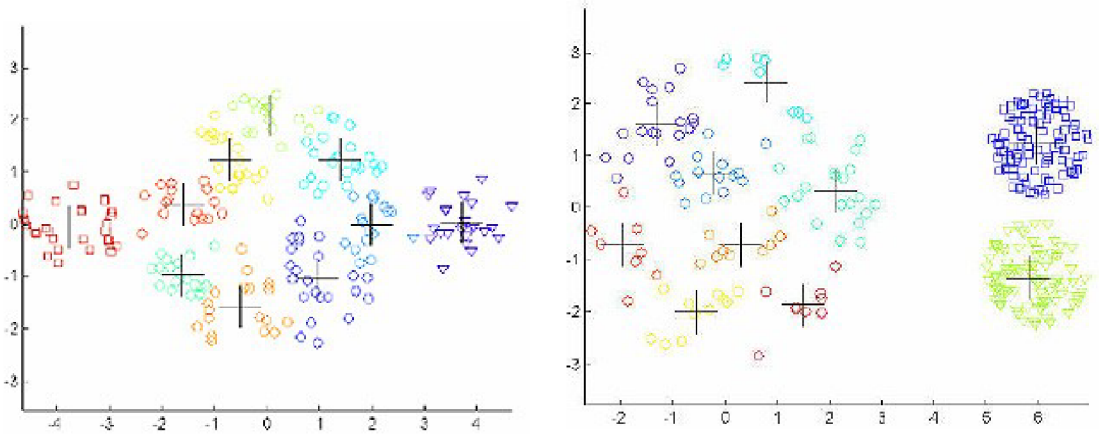


Obrázek 2.6: Rozdílná velikost 3 shluků [4]



Obrázek 2.7: Rozdílná hustota 3 shluků [4]

Řešením by mohlo být použít větší počet shluků. Nedošlo by k promíchání dat z různých tříd. Výsledné shluky by ale bylo třeba správně sloučit.



Obrázek 2.8: Použití většího počtu shluků pro zamezení promíchání dat z různých tříd [4]

Je vidět, že pro vhodný výsledek k-meansu, jehož algoritmus je založený na předem daném neměnném počtu shluků do kterých se bude klasifikovat, je klíčové správně určit počet clusterů [4].

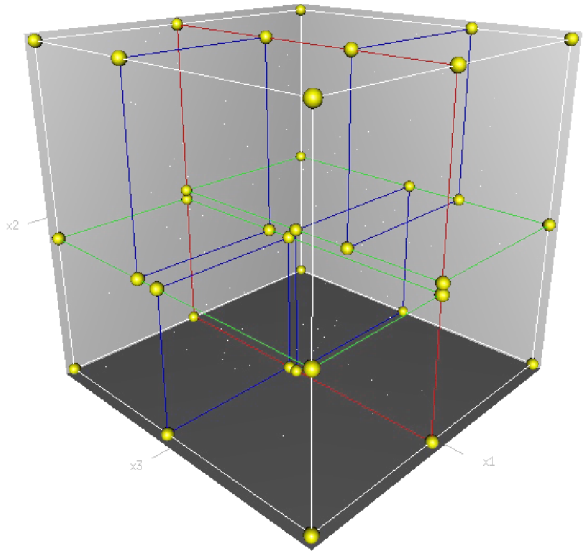
2.5 Kd-tree

Kd-tree je zkrácená anglická verze názvu pro *k-dimenzionální strom* [5]. Jedná se o prostor dělící datovou strukturu určenou pro uspořádaní bodů v k-dimezionálním prostoru. *Kd-tree* je binární strom, jehož každý uzel *k*-dimenzionální bod. Každý ne-listový bod může být uvažován jako implicitně generující dělící nadrovinu, která dělí prostor do dvou částí, známé jako podprostory. Body napravo od roviny reprezentují pravou větev stromu a body nalevo zase levý podstrom. Směr nadroviny je vybrán následujícím způsobem: každý uzel ve stromě je asociován s jednou z *k*-dimenzí s nadrovinou kolmou na osu této dimenze. Takže, například, pokud pro určité dělení je vybrána *x*-ová osa, všechny body s menší *x*-ovou hodnotou, než je hodnota uzlu, budou zařazeny do levého podstormu a všechny větší hodnoty zase do pravého. V tomto případě tedy bude nadrovina rozdělena podle *x*-ové hodnoty bodu a její normála bude *x*-ová osa.

Je mnoho způsobů jak sestavit *kd-tree*, jelikož je mnoho způsobů jak vybrat dělící roviny podle os. Metody ale mají souhrnně tato pravidla:

- S pohybem ve stromě směrem dolů, postupně se cyklicky používají jednotlivé osy jako dělící roviny (např. Kořen *x*-ově zarovnanou osu, jeho syn *y*-ovou, jeho vnuk *z*-ovou a další uzel bude mít zase *x*-ovou atd.).
- Uzly jsou vybírány podle *mediánu* z bodů vkládaných do podstromu s respektováním souřadnic os, které byly použity pro vytvoření dělící roviny.

Tato metoda vede k vytvoření vyváženého *kd-tree*, ve kterém je každý listový uzel podobné vzdálen od kořene. Ačkoliv vyvážené nejsou nutně optimální pro všechny aplikace.



Obrázek 2.9: Zobrazení postupného dělení 3D prostoru podle kd-tree [5]

Také není nutné stanovovat medián, v tom případě ve výsledku akorát není garantováno, že strom bude vyvážený. Jednoduchá heuristika pro vyhledání mediánu a vyhnutí se lineární-časové složitosti při hledání, či složitosti O(n log n) při řazení, je stanovení menšího, pevně daného počtu náhodně vybraných čísel, ze kterých se medián vyhledá. V praxi tento postup vede často k vytvoření hezky vyvážených stromů.

K vyhledávání ve stromě může být využit **Nearest Neighbour search** (Nejbližší Soused):

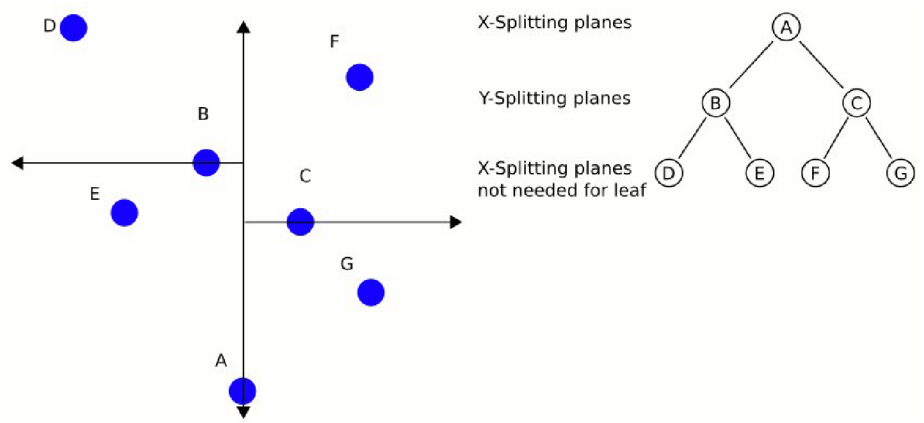
- Tento algoritmus se snaží o vyhledání bodu ve stromě, který je nejblíže danému vstupnímu bodu. Toto vyhledávání může díky vlastnostem stromu rychle eliminovat rozsáhlé oblasti prostoru. Prohledávání Postupuje v těchto krocích:

1. Začíná se u kořene, postupně směrem dolů rekurzivně. Rozhoduje se jít nalevo, nebo napravo podle toho, ke kterému hraničnímu uzlu je blíž.
2. Jakmile se dostane až k listu, uchová si jej jako „současný nejlepší“
3. Algoritmus se vrací z rekurzivního zanoření zpět a provádí tyto kroky:
    1. Pokud je současný uzel blíž ke vstupnímu bodu než současný nejlepší, je tento novým současným nejlepším.
    2. Dále zkontroluje, jestli jsou na druhé straně dělící roviny nějaké body bližší body než současný nejlepší. To je koncepčně provedeno protnutím dělící nadroviny s hyperkoulí okolo hledaného bodu, která má poloměr rovný vzdálenosti k současnému nejlepšímu bodu. Jelikož jsou všechny roviny osově zarovnané, je toto implementováno jako

jednoduché porovnávání, jestli je rozdíl mezi dělícími koordináty hledaného bodu a současného uzlu menší než vzdálenost z hledaného bodu do současného nejlepšího

1. Pokud hyperkoule protne rovinu, je možnost že jsou na druhé straně bližší body, takže algoritmus se musí přesunout na opačnou větev stromu, než je současný uzel a pokusit se najít bližší body, podle stejného rekurzivního procesu, jako celkové hledání.
2. Pokud hyperkoule neprotne dělící rovinu, algoritmus normálně pokračuje stromem směrem nahoru a celá opačná větev aktuálního uzlu je eliminována.

4. Až algoritmus skončí tento proces pro kořenový uzel, prohledávání je ukončeno.



Obrázek 2.10: Ukázka dělení prostoru pro 2 dimenze [5]

Složitost:

- Vytvoření statického kd-tree z n bodů trvá $O(n \log^2 n)$ času, pokud je $O(n \log n)$ řazení použito ke spočítání mediánu v každém levelu. A složitost $O(n \log n)$, pokud je k vyhledání mediánů použit lineární přístup.
- Vkládání nového bodu do vyváženého stromu trvá $O(\log n)$.
- Odstranění bodu z vyváženého stromu trvá $O(\log n)$.
- Vyhledání bodu pomocí *NN* algoritmu má časovou složitost $O(\log n)$ s náhodně rozloženými body. Nejhorší možná varianta $t_{worst} = O(k \cdot N^{1-1/k})$ může nastat, jestli je n v počtu řádu dimenzí.

Pro zvýšení rychlosti hledání se dají použít přibližné algoritmy – Approximate NNs. Stačí drobně upravit stávající algoritmus přidáním horní hranice por počet bodů prověřený ve stromě, nebo přerušením hledání po určité době podle real-time hodin.

3 State of Art

V současnosti je téma rozpoznávání obrazu velmi populární a rozvíjející se odvětví IT. Kamerové systémy či snímače fotek jsou na každém rohu. Jsou využívány v továrnách pro kontrolu kvality výrobků, pro rozpoznávání obličejů při hledání osob, při automatickém dodržování bezpečného odstupu od kolejiště na nádražích, pro hledání podobných obrázků kdekoli ze světa na Googlu, nebo pro poznávání místa na zemi pomocí podobných fotek s technologií Photosynth. Klasifikace fotek a rozpoznávání objektů ve fotkách se musí obecně vypořádat s generalizací tohoto procesu. Jelikož obvykle dochází k celé řadě rušení a deformací objektů ve scéně, které představují stejnou věc. Na ukázku jsou vybrány následující postupy používané pro klasifikaci fotek a rozpoznávání určitých objektů ve fotkách.

Základní přístupy pro rozpoznávání objektů jsou různé[6]. Například *constellation modely* [7], *fragment-based modely* [8] a *pictorial struktury* [9] se snaží lokalizovat rozlišitelné části objektů a určit meze v prostorovém rozložení. Potencionálně jsou tyto přístupy velmi mocné, ale nedokáží řádně zpracovat významné deformace, jako např. prostorové rotace. Také nezahrnují objekty s rozdílným počtem částí, jako budovy a stromy. Jedna z vyvinutých technik na rozpoznávání objektů na těchto přístupech zakládá, ale zjednodušuje je. Kdy rozpoznává části objektů jako jsou třeba listy nebo okna, ale bere v úvahu jen relativní velikost a snaží se o rozpoznání celkového objektu.

Jedná se o techniku pro objektovou kategorizaci pomocí naučeného vizuálního slovníku, na jejíž vývoji se podílel Microsoft s univerzitou v Cambridge [6]. Soustřeďují se na vytvoření kompaktního a diskriminačního modelu objektových třída a následné automatické rozpoznání objektů v obrazech. V obrázku *3.1* lze vidět výstup algoritmu, kde správně pojmenoval ručně vybrané oblasti v obrázcích.

Obrázek 3.1: Ukázka rozpoznávání objektů v obraze [6]

Tato metoda vychází z techniky používané v [10], ale oproti pevně danému počtu slov vizuálního slovníku, nechá se optimální velikost odhadnout automaticky. Trénovací obrázky jsou nejprve nahrubo manuálně segmentovány (pomocí „malovacího prostředí“). Na každý obrázek je následně použita konvoluce s filtry z banky filtrů [11], např. Gaussian, první a druhá derivace

Gaussiana, pro zjištění reakce na filtry (pro každý pixel), které jsou sloučeny ze všech obrázků (nezávisle na třídě) a se-shlukovány pomocí *K-means*. Použitá vzdálenost je *Mahalanobis*. Ve výsledku sada středů shluků a odpovídající kovarianční matice definují univerzální vizuální slovník (*UVD*). Zde má slovník velké množství slov (v řádech tisíců). Na testovací fotky se pak taky použijí filtry a následně se spočítají vzdálenosti do jednotlivých slov. Normalizované histogramy clusterů mohou být spočítány na určitých regionech a vyhodnoceny třídy, alternativně by šlo využít *Gaussův model* tříd s posteriorními pravděpodobnostmi parametrů, který se používal při trénování. V konečných výsledcích si právě *Gaussův model* vedl velmi dobře. A vlastní *appearance-based model*, který je právě zaměřen jen na určité části obrázku, prokázal velmi dobrou schopnost kategorizovat texturově i strukturově bohaté obrázky.

O trochu výše zmiňovaná práce *Visual Categorization with Bags of Keypoints* [10] se zabývá kategorizací obrazů na základě tzv. *„Bags of Keypoint“*. Pro extrakci význačných bodů využívá *SIFT* [3], kterému dává přednost před Gaussovými deriváty. Jelikož se očekává, že by měly být méně náchylné, vůči různým nepříznivým vlivům. Také deskriptory popisující význačné body jsou 128místné (než od 12 až 16 u Gausse). Pro zpracování získaných vektorů se používá *k-means*. Jeho nevýhoda, stanovení předem daného shluků, zde nehraje velkou roli, protože je důležité především správné roztřídění, než určení počtu. Proto shlukování spouští několikrát s různými počty shluků a vybírají s nejmenší chybovostí přiřazení. Tak se vytvoří celkový vizuální slovník a právě sady slov využité v jednotlivých kategoriích jsou ony *bags of keypoints*. A pro konečnou kategorizaci používá *multi-class supervised learning* se stejným počtem tříd jako je definovaný počet kategorií. Kategorizace probíhá ve dvou oddělených kolech. Trénování a pak testování. Označená data jsou předána klasifikátoru k natrénování pro adaptaci statistické rozhodovací procedury. Tato práce používá *Naïve Bayes klasifikátor* [12], kvůli jeho jednoduchosti a rychlosti a taky *Suport Vector Machine* (*SVM*) jelikož je často používaný pro řešení mnohodimenzionálních problémů. A podle konečných výsledků použití *SVM* daleko převyšovalo jednoduchý *Naïve Bayes klasifikátor*.

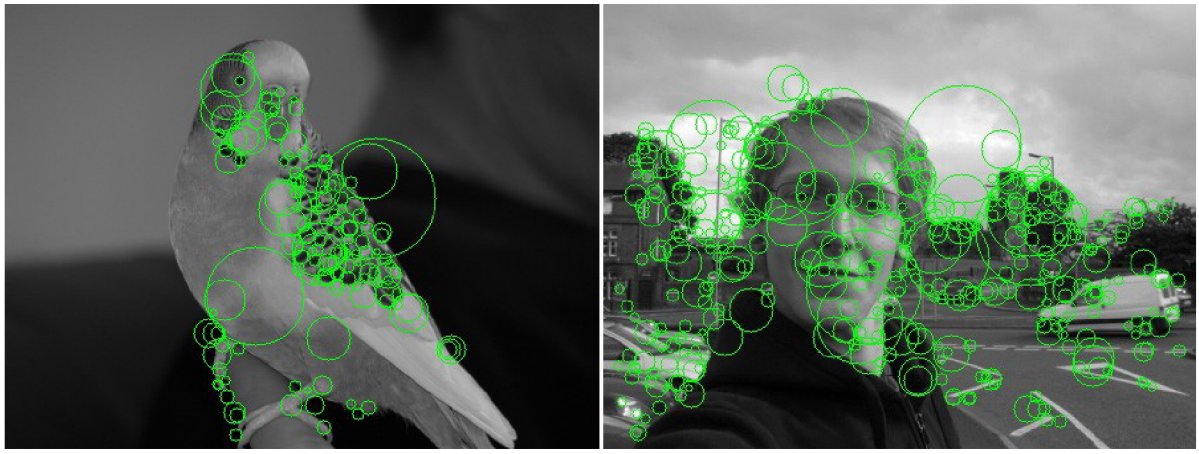
4 Návrh řešení

Základní myšlenkou této práce je vytvořit systém, schopný zařazení vstupních fotografií, do předem daných skupin. Při návrhu postupu pro klasifikaci fotografií je vycházeno z podobných prací na toto téma (kap.3). O fotografiích na vstupu nemá žádné informace a sám musí poznat správnou skupinu, nebo alespoň odhadnout tu nejpravděpodobnější. Klíčem k tomu bude využít informace, získané při definování skupin:

1. Stanoví se skupiny fotek – každá musí obsahovat větší objem reprezentativních dat, která budou základem pro informace o skupinách.
2. Z fotek ve všech skupinách se zjistí význačné body a z jejich deskriptorů se vytvoří jedna velká matice.
3. Tyto deskriptory se rozdělí do tříd pomocí shlukování (bez informace o původu deskriptorů) – jednotlivé shluky jsou slova nově vytvořeného vizuálního slovníku společného všem kategoriím.
4. Následně na základě statistického zpracování počtu slov ve skupinách je vytvořeno hodnocení každé skupiny.
5. Z nových fotografií určených ke kategorizaci se opět detekují a popíší význačné body. Ty jsou namapovány k příslušným slovům. A na základě vytvořeného hodnocení a počtů jednotlivých slov jsou spočítány celkové body každé skupině. Fotka je přiřazena do skupiny s největším počtem bodů.

4.1 Vyhledání a popis význačných bodů.

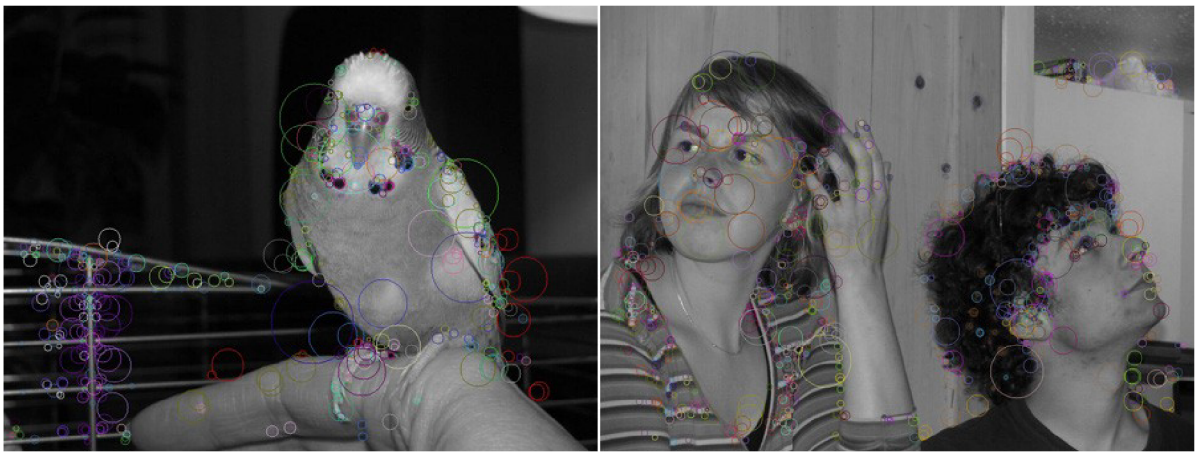
Pro zjištění obrazových příznaků (*feautures*) je použita metoda *SURF* (viz kap. 2.2). Ta nejen vyhledá v obraze význačné body, ale dokáže je i popsat příznakovým vektorem, který popisuje okolí bodu. Délka vektoru je stanovena na 64 na základě [2], kde je ukázáno v testových grafech, že popis tolika dimenzemi už produkuje dobré výsledky a oproti třeba 128 se moc neliší a šetří paměť. Všechny body se totiž spojují dohromady do jedné velké matice f x d, kde f je celkový počet všech *feature vektorů* a d je počet dimenzí vektorů. A pokud metoda SURF najde v každé fotce tisíce význačných bodů, tak už hraje roli, jestli mají vektory jednou tolik parametrů. Nakonec se ještě snižuje rozlišení všech trénovacích fotek na velikost delší strany 480 pixelů, právě z důvodu paměťové a časové náročnosti následného zpracování takovéhoto velkého objemu dat. Po změně jsou počty vyhledaných bodů v řádu stovek.



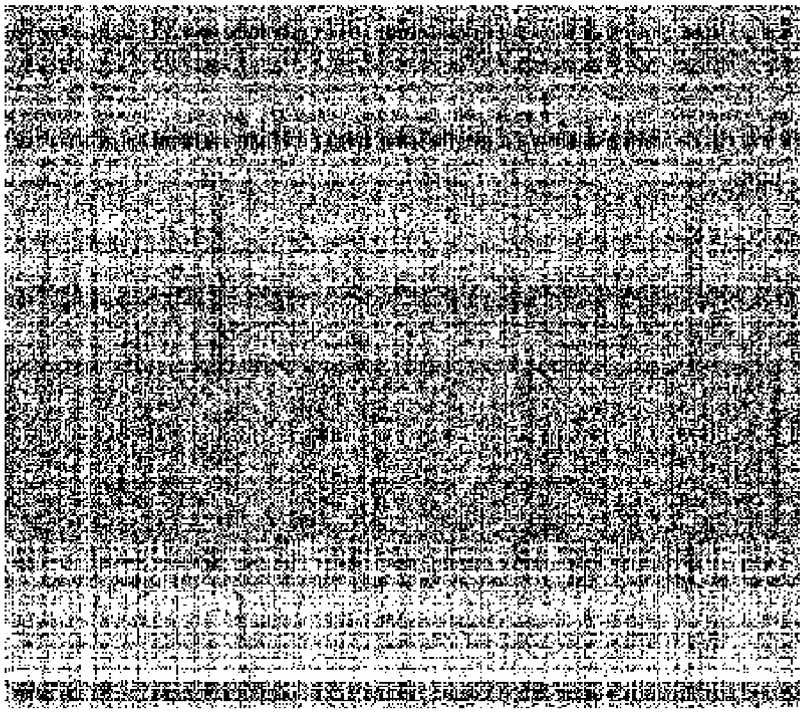
Obrázek 4.1: Zobrazení nalezených význačných bodů ve fotkách 480x360px → 147 a 395 bodů

4.2 Vizuální slovník

Dalším krokem systému je vytvoření jednoho velkého společného vizuálního slovníku z deskriptorů o kterých byla řeč v minulé kapitole. Na vstupu je jedna velká skupina význačných bodů, popsaná vektory. Nejsou žádné informace o tom ke které fotce nebo do které skupiny jakýkoliv bod patří. V tomto okamžiku se použije postup kategorizace shlukováním, podobně jako v [10]. Cílem je rozdělit všechny vektory do *k* skupin, kde *k* bude počet slov výsledného slovníku. Je tedy použita metoda *k-means* (viz. Kap 2.4) s počtem shluků *k*, na základě které je na konci každý bod přiřazen do clusteru, který má nejblíže. Použitá vzdálenost je Eucleidovská. Každý shluk je popsán podobným vektorem jako význačný bod a na konci shlukování se tyto hodnoty uchovají. V této práci použité názvy – *shluk*, *cluster*, nebo *slovo* jsou si významem rovny.



Obrázek 4.2: Význačné body přiřazené do clusterů (zde 100), každý znázorněn jinou barvou.



Obrázek 4.3: Zobrazení použitých slov každou fotkou. Každý řádek je jedna fotka, každý sloupec jedno slovo. Bíle body značí výskyt slova ve fotce.

4.3 Vytvoření váhového hodnocení skupin

Informace o tom, který význačný bod patří ke které fotce, zatím nebyla využita. Je ale k dispozici a teď bude použita ke spočítání váhového ohodnocení jednotlivých slov slovníku pro každou skupinu. Postupně je procházen každý význačný bod z jednotlivé skupiny a podle informace o shluku do kterého je přiřazen se vytváří histogram vyskytujících se slov. Jelikož se počty bodů ve skupinách mohou lišit, pro sjednocení hodnot jsou převedeny na procenta celkového počtu a hodnota každého slova je řádově zvýšena až do řádu desítek. Tak dostaneme různá hodnocení stejného slova pro různé skupiny. To je ale jen polovina hodnocení. Druhá se vypočítá jako procentuální rozložení z počtu bodů všech skupin v jednom slově, čímž se zvýší váha pro skupinu, která slovo využívá více než ostatní. Ve výsledku se obě hodnoty sečtou a tak se dostanou konečná ohodnocení slov a ta se uchovají.

Využito je ještě několika úprav výsledných počtů. Jedna z možností je vybrat určitý počet (třeba 1%) z nejčastěji použitých slov v každé skupině a tyto počty vynulovat. Může se jednat o data,

která jsou sice často zastoupena, ale pro danou skupinu nemají dostatečný rozlišovací význam, nebo se mohou vyskytovat i v jiných skupinách. A další možností s ještě selektivnějším způsobem, právě na potlačení obvyklých slov potlačujících rozlišitelnost. Prověření určitého počtu nejlepších hodnot (3%) a vynulování jenom těch, které se vyskytují i v alespoň jedné z ostatních skupin.

Počty použitých slov skupinou				**Zváhované procenta z celku zvlášť v rámci skupiny**				**Procenta rozložení daného slova v rámci skupin**			
113	77	80	47	23,598	21,061	18,904	17,455	29,126	25,995	23,333	21,544
74	59	83	39	15,453	16,138	19,613	14,484	23,525	24,567	29,858	22,049
108	84	134	55	22,554	22,976	31,665	20,426	23,103	23,536	32,436	20,923
98	40	148	17	20,465	10,941	34,974	6,313	28,153	15,051	48,111	8,685
139	52	69	88	29,027	14,223	16,305	32,682	31,470	15,420	17,677	35,43

Tabulka 4.1: Ukázka hodnocení vzorku jednotlivých slov (v řádcích) každou ze čtyř skupin.

4.4 Přiřazování fotek do skupin

Všechny předchozí kroky byly vlastně jen příprava systému pro samotnou kategorizaci. Výsledné ohodnocení je uchováváno, a proto předchozí procedury nemusí být pokaždé znovu opakovány. Ze středů shluků (slov slovníku) popsaných vektory (viz *4.3*) se vytvoří prostorová datová struktura *kd-tree* (viz. *2.5*). Nové vstupní fotky určené k roztřídění jsou zpracovávány jedna po druhé. Nejdříve se ve fotce najdou význačné body (stejně jako v *4.1*). Ty jsou následně mapovány na uzly *kd-tree*, čímž vlastně zjistíme přiřazení význačných bodů slovům slovníku. A v tomto okamžiku už stačí jen projít nalezená slova a sečíst jejich ohodnocení v rámci každé skupiny zvlášť. Skupina s nejvyšším ohodnocením je vítězná. Jednoduše se dá i z celkového počtu vypočítat, na kolik si je systém přiřazením jistý.

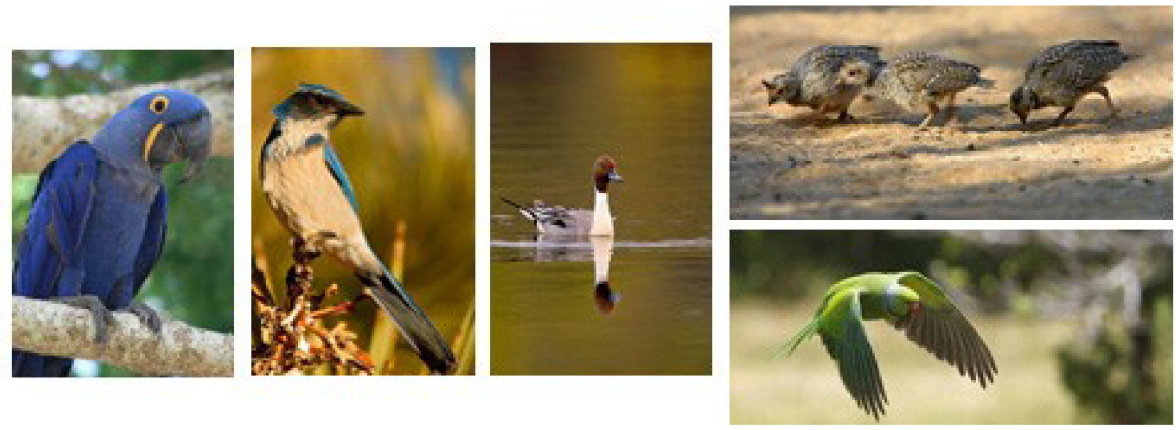
4.5 Volby knihovny počítačového vidění

Jelikož nebylo cílem práce nově implementovat vymyšlené algoritmy z oblasti počítačového vidění, bylo třeba vybrat nějakou knihovnu, která by už měla navržené algoritmy efektivně implementované. Byla vybrána OpenCV (Computer Vision). Pracuje s obrázky jak na základní úrovni, načítání, ukládání, změny formátu, tak obsahuje funkce pro mnoho pokročilých algoritmů, mezi jinými i pro SURF, shlukování s k-means i práci s kd-tree.

4.6 Vybrání trénovacích dat

Systém je testován na 4 různých skupinách, které by měl umět rozpoznávat:

- Ptáci – obvykle detail na celého ptáka, z blízka i z trochu větší dálky focené za dne. Pozadí vetšinou ne moc ostré kvůli tomu, že je zaostřeno na ptáka.



Obrázek 4.4: Ukázka z trénovacích fotografií skupiny Ptáci

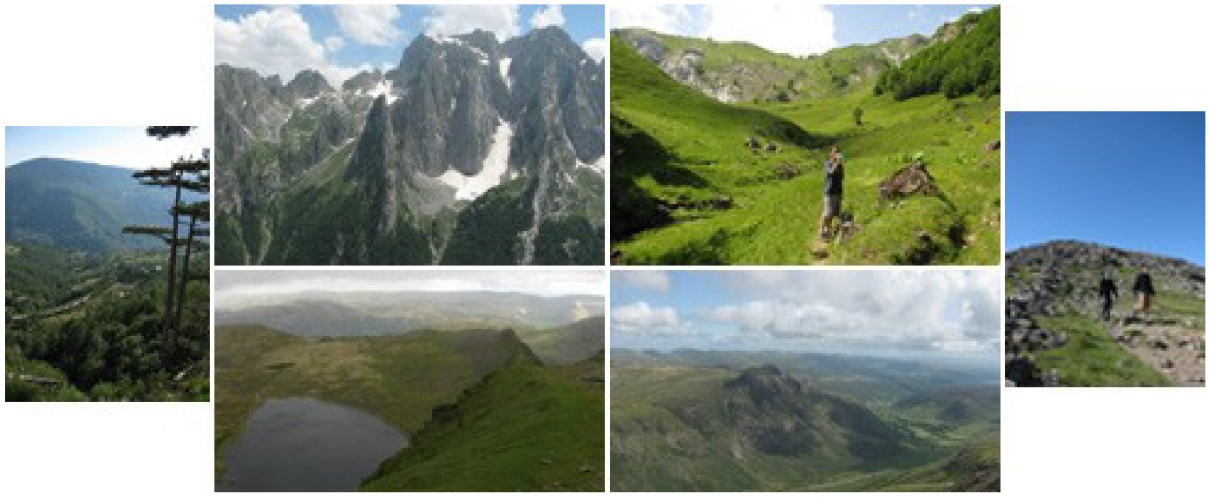
- Lidé – na fotkách jsou jak skupinky lidí, tak „busty“, v různých prostorách (v interiérech i venku). A s různými světelnými podmínkami (denní světlo, umělé, i nedostatečné světlo blesku ve tmě či tmavých prostorách)

Obrázek 4.5: Ukázka z trénovacích fotografií skupiny Lidé

- Auta – všechny fotek aut jsou focené za dne, nebo s velmi dobrým světlem například v autosalonech. Na většině jsou auta zblízka. Několik „fotek“ jsou realistické počítačové obrázky.

Obrázek 4.6: Ukázka z trénovacích fotografií skupiny Auta

- Hory – opět jsou všechny fotky focené za dne. Povrchy hor jsou skalnaté, zelené i se sněhem s nebem všech možných stavů. Pohledy jsou na hory i dolů z hor. Na některých fotkách jsou ne moc velké postavy.



Obrázek 4.7: Ukázka trénovacích fotografií skupiny Hory

Z větší části jsou fotky ptáků pořízeny z webového atlasu ptáků a foto alb [13], stejně tak fotky aut [14]. Všechny fotografie hor a lidí jsou mé vlastní. V každé skupině je okolo 100 fotek a jsou kvůli trénovacím účelům zmenšené.

5 Implementace

Aplikace je implementována podle návrhu řešení popsaného v kap. 4. a proto je rozčleněna přesně podle zmíněného návrhu do na sobě nezávislých částí. Jedná se o konzolovou aplikaci načítající klasicky parametry programu při spuštění. Implementační jazyk je C++, jehož základní funkcionalita je doplněna o funkce a třídy z toolkitu Qt. Využité algoritmy z oblasti počítačového vidění nejsou nově implementovány, ale je využita specializovaná knihovna OpenCV. Jedna z částí využívá matematický nástroj Matlab a proto je napsán i krátký skript používající fce. Matlabu. Celý program je implementován tak, že je možné spustit jen určité části klasifikace a není nutné vždy spouštět vše.

5.1 Toolkit Qt

Program je implementován sice do funkcí a ne do objektových tříd a ovládá se z konzole, budoucí možností ale je vytvoření GUI a především proto byl ke zpracování vybrán Toolkit Qt, který obsahuje knihovny pro vytvoření GUI a ke zlepšení práce s C++. Následují významné typy a třídy, které byly použity. Je využit typ `QFile` a `QTextStream` pro práci se soubory. Dále nástavba nad standardním typem `string` – `QString`, který obsahuje větší množství metod k manipulaci. Využité především parsování, převody na čísla. Pokud jsou v následujících popisech zmíněné seznamy a vektory (pokud o nich nejedná ve spojitost s knihovnou OpenCV), jsou tím výhradně myšleny třídy `QList`, `QStringList` a `QVector`, které zajišťují práci s dynamicky alokovanými poli. Qt také podporuje orientaci v systémových cestách a práci se soubory přes třídu `QDir`. Toho je využito při vytváření cest k fotkám v dané složce. Dalo by se to taky využít při přímém přesouvání fotek na základě přiřazení. Jsou podporovány i metody pro konkrétní formát dat (viz 5.2).

5.2 Knihovna OpenCV

Jak již bylo zmíněno (kap. 4.6), konkrétně knihovna OpenCV2.0, obsahuje řadu pokročilých funkcí z oblasti počítačového vidění. Poskytuje i vlastní datové struktury používané ve funkcích – např. `CvMat`, `CvSeq` pro matici a sekvenci. Proto u funkcí, které pracují s těmito datovými typy bylo třeba se snažit vyhnout přepisování z pole do pole kvůli rozdílným typům. Proto některé fce., třeba `readClusterDescript()` rovnou načítají do specifických formátů. Další nesnází je, že knihovna neposkytuje funkce pro převod mezi vlastními typy. Tak bylo třeba pro výstup `CvMat` z `readClusterDescript()` napsat funkci `cvMatToCvSeq()`, která data převede na typ `CvSeq`, potřebný pro procházení *kd-tree* v `cvFindFeatures()`. Pro načítání obrázků je použita fce. `CvLoadImage()`, která podporuje jenom následující typy obrazových dat: bmp, dib, jpeg, jpg,

jpe, png, pbm, pgm, ppm, sr, ras, tiff, tif. Ošetření vybraných formátů se provádí už při vytváření seznamu fotek pro načtení pomocí Qt filtru ve fci. `MakeFileList()`.

Pro integrace OpenCV knihovny do projektu v Qt se hlavičky normálně načítají přes `#include`. Navíc je potřeba přidat do projektového souboru (*.pro) cestu ke zdrojovým souborům, např:

- `INCLUDEPATH += C:/OpenCV2.0/include/opencv`

  A přidání cest k dynamickým knihovnám (ukázka přidání pouze jedné):
- `LIBS += C:/OpenCV2.0/lib/libcv200.dll.a`

5.3 Uchování dat

Jednotlivé části spolu přímo nekomunikují, ale některé jsou závislé na výstupech těch předcházejících a tato data dále zpracovávají. Proto se vytváří několik souborů uchovávajících specifická data do kterých se zapisuje po zpracování nové informace. Jsou uloženy ve složce `./resources`. Pouze soubor s výsledky se ukládá do aktuální složky s aplikací.

Zde je přehled použitých pevně pojmenovaných souborů s daty používanými během chodu aplikace:

- **groupNamesSums.txt** – název každé skupiny (název složky) a počet fotek v ní, postupně na řádcích, stylem:

  @birds

  #106
- **descriptorsMat.txt** – společná matice deskriptorů význačných bodů všech skupin (`float`). Rozměry F x D, kde F je počet význačných bodů a D je počet dimenzí vektoru. Sloupce jsou oddělené mezerami.
- **featureNums.txt** – uchovávají se zde cesty ke každé fotce a počet nalezených význačných bodů. Jeden údaj na řádku.

  @D:/Foto/2010/birds/040128-028.jpg

  #495
- **centroids.txt** – matice `double` deskriptorů jednotlivých středů clusterů. C x D, kde C je počet clusterů a D počet dimenzí. Sloupce jsou odděleny mezerami. Tyto údaje jsou zapsány i do souboru (# - clustery, | dimenze) na první dva řádky.

  `#450`

  `|64`

  `-9.4229189e-004 -4.8615764e-004  3.6546342e-003 ...`
- **clustLabels.txt** – jen indexy přiřazených shluků příznakovým vektorům. Jeden na řádku.

- **groupValues.txt** – výsledné ohodnocení skupin. Na prvních dvou řádcích je zapsán počet clustrů C (značka #) a počet skupin G (značka |). Samotné ohodnocení je zapsané v matici C x G. Sloupce jsou oddělené mezerami.

```
#450
|4
9.24327 23.3882 21.1567 5.97752
```

- **resultAssign.txt** – soubor s výsledky. Na začátku jsou vypsané názvy jednotlivých skupin. U každé fotky je nejdříve vypsaná absolutní cesta. Následují body jednotlivých skupin a přepočet na procenta. A výsledně přiřazená skupina.

```
-->D:/Skola/Bakalarka/work/fotoDoSkupin/car05.jpg
body: 23146 procenta: 24.33
body: 23657.9 procenta: 24.87
body: 28508.7 procenta: 29.97
body: 19806 procenta: 20.82
nej==> cars
```

5.4 Jednotlivé části

Tato sekce popisuje podrobnější popis implementace jednotlivých částí aplikace, popřípadě jaké implementační problémy byly řešeny. Každá z částí je implementována jednou hlavní funkcí a dalšími pomocnými, které tato funkce volá Jsou uloženy v samostatných souborech. Odpovídající hlavičkový soubor obsahuje definice k použitým funkcím. Následující podkapitoly jsou krom první pojmenované podle názvu souborů, které popisují. Podrobnější popis lze najít v programové dokumentaci.

main.cpp

Tenhle soubor je klasicky kořenový, obsahující hlavní funkce. První je funkce `processArgs()`, která zpracuje a zkontroluje vstupní parametry programu. Ve výsledku sestaví seznam stavů pro konečný automat, který řídí a postupně volá hlavní funkce ve správném pořadí v závislosti na parametrech. Všechny funkce jsou postupně následující. `makeFileList()` vytváří z parametru „cesty ke složkám s fotkami“ seznam cest k obrazovým souborům. Navíc zapisuje do souboru jméno skupiny (jméno složky) a počty fotek v nich. `processPicture()` má na starosti vyhledání význačných bodů a vytvoření jednoho velkého seznamu vektorů deskriptorů ze všech fotek všech skupin. Funkce na zpracování příznakových vektorů a vytvoření vizuálního slovníku je implementována skriptem v Matlabu, proto je další funkcí `processClustLabels()`. Ta vytváří

váhové ohodnocení slov na základě `float` vektorů, představujících vizuální slovník a seznamu přiřazených clusterů příznakovým vektorům. Poslední fce. `classPhoto()` spočítá pravděpodobnost pro každou z fotek určených k zařazení. Použité chybové kódy pro ošetření chyb i hlášení jsou spolu s globálními konstantami v `globals.h`

processPicture.cpp (processPicture.h)

Obsahuje především fci. `processPicture()`. Ta je založena na funkcích pracujícími s obrázky z knihovny OpenCV. K načítání používá `cvLoadImage()`. Tahle funkce dokáže načítat obrázky do speciálního datového formátu `IplImage`. OpenCV má implementovanou vlastní fci. na extrakci příznaků pomocí SURF – `cvExtractSURF()`. Ta na výstupu produkuje dvě sekvence typu `CvSeq`. Jedna s informacemi o každém nalezeném bodu a druhou s vektorem `float` hodnot popisujících bod. V nastavení funkce se dá nastavit počet dimenzí vektorů – 64/128. Je použito pouze 64 kvůli paměťové náročnosti, navíc by tato hodnota měla být dostačující. Dalším parametrem je práh pro Hessianovy matice. Je stabilně nastaven na 600, detekuje se o něco menší počet význačných bodů, pro zaměření na ty důležitější a snížení objemu dat. Zapsání dat z vektorů, přidáním na konec souboru, probíhá po zpracování každé fotky. Co vektor, to jeden řádek. Čísla oddělená mezerami. Tento formát je jednoduše zpracovatelný Matlabem. Do dalšího souboru se zapisují k názvům fotek počty nalezených význačných bodů.

kmeansMatlab.m

Tento jednoduchý skript realizuje načtení matice všech `float` vektorů popisujících význačné body, díky podporovanému formátu, v jakém jsou zapsané, lze použít funkci `importdata()` a do prostředí se načte rovnou celá matice. A jejich seshlukování → vytvoření vizuálního slovníku je provedeno pomocí fce. Matlabu `kmeans()`. Funkce je spouštěna 3x, s náhodně inicializovanými středy shluků. Nastavení počtu iterací lze provést následujícím přízem `statset('MaxIter', iters)` před spuštěním kmeans. Nejlepší výsledky se zapíší do souborů. Do jednoho vektory popisující středy shluků. Je využita fce. `fprintf()` k formátovanému výpisu pro první dva řádky, vypisuje se počet clusterů a dimenze. Samotná matice pomocí standardní fce. `save()` s parametrem `'-append'` pro přidání nakonec. Do druhého souboru index shluku přiřazeného každému původně načtenému vektoru opět fcí. `fprintf()`.

processLabels.cpp (processLabels.h)

Soubor obsahující `processLabels()` a podpůrné fce. Důležitá pro načítání potřebných údajů je `readFeatStats()`. Ta do tří seznamů načte počty počet ve skupinách, počty bodů v jednotlivých fotkách a indexy přiřazených shluků. Na základě těchto údajů je v cyklu procházen seznam shluků

a jsou přesně spočítány histogramy shluků (slov) pro jednotlivé skupiny. Ty jsou uloženy ve vektorech v seznamu `clusterAssign`. Následně je použita důležitá fce `omitBest()`.

Ta má na starosti vynulování několika nejčastějších hodnot, pokud se vyskytují i v jiných skupinách. Nejprve se projde každý vektor (histogram) skupiny a zaznamenávají se do seznamu 3% nejčastějších hodnot – u každé počet, index ve vektoru a `bool` příznak nastavený na `false`. Tyto dvou rozměrné seznamy se pak zapíší do jednoho společného pro všechny skupiny a vznikne tak 3-rozměrný seznam. Nastavení, které hodnoty vynulovat je následující: Od první skupiny po předposlední se prochází postupně každý prvek – aktuální prvek. U toho se kontroluje, jestli má příznak nastavený na 0. Pokud ano, prochází se ty prvky druhé až poslední skupiny, které mají příznak nastaveny na `false` (nebyly ještě poznačeny) a kontroluje se jejich index s aktuálním. Jestli se shoduje, nastaví se u obou prvků příznak na `true`. Nakonec se tyto seznamy projdou a u prvků které mají nastavený příznak na `true`, se pomocí poznačeného indexu vynulují hodnoty v původních seznamech.

V `processLabels()` se uchovávají tři 2-rozměrné seznamy. `clusterAssign` – počty vyskytujících se slov ve skupinách, `procOfGrps` – tyto počty v procentech v rámci skupiny a `procAmongGrps` – procentuální rozložení bodů v rámci slova.

Po úpravě počtů se je převedení hodnot na procenta realizováno ve fci. `CountProc()`. A následné spočítání procentuálního rozložení v `getProcAmongGrps()`. Finální funkcí, která vyhodnotí procenta a vytvoří z nich bodové ohodnocení slov v rámci skupiny je `makeAndPrintEval()`. Převádí procenta v rámci skupin do řádu desítek vynásobením 100 a následně sečte s rozložením v rámci slova a zapíše do souboru.

Součástí této funkce jsou i zakomentované, ale ponechané kusy kódu, které vytvoří ve složce s pracovními soubory obrázek vizuálního slovníku. Zakomentovaný kód je vždy značen `V_D`. Obrázek je primárně černý, bíle body značí použití. V řádcích jsou fotky, ve sloupcích použité clustery. Značení je po jednotlivých pixelech. O vyplnění obsahu se stará funkce `makeDictionaryImg()` procházením histogramu použitých shluků a nastavováním jednotlivých pixelů.

classPhoto.cpp (classPhoto.h)

V tomto souboru je obsažena především fce. `classPhoto()`. Ta je závislá pouze na výstupech z předchozích funkcí uložených v souborech. Pokud ty alespoň jednou proběhly, může už být volána pouze samostatně a načítat pokaždé dřívější data. Také využívá fci. `makeFileList()` pro vytvoření seznamu cest k fotkám, ovšem už bez volitelného parametru, takže se informace o tom kolik fotek se načetlo nikam neukládají. Vyhodnocení probíhá postupně pro každou fotku v jednom

cyklu `for()`. Nejprve je vytvořena struktura `CvFeatureTree` pro *kd-tree* funkcí `cvCreateKDTree` z popisů clusterů načtených fcí. `readClusterDescript()`, která rovnou vrací ukazatel na strukturu `cvMat`. Pro zjištění význačných bodů je implementována vlastní funkce `findFeatDescriptors()`, menší obdoba `processPicture()`, která ale taky využívá `cvExtractSURF()` a vrací ukazatel na sekvenci vektorů `cvSeq`. Jelikož ale tyto vektory půjdou do fce. pro mapování na *kd-tree* `cvFindFeatures()`, která je ale přijímá ve struktuře `CvMat`, je napsána vlastní funkce `cvSeqToCvMat()` pro převod.

Finální část výpočtu přiřazení provádí `chooseGroup()`. Ta na základě načtení bodových ohodnocení jednoduše prochází namapované indexy a používá je jako indexy do pole s ohodnocením a hodnoty sčítá. Tento postup se opakuje pro každou skupinu. Hodnoty se vrací v poli o rozměrech počtu skupin.

A konečné zapsání bodů a spočítaných procent, jakožto i přiřazení výsledné skupiny do souboru přes fci. `writeFinalEval()`. Názvy skupin do kterých přiřazuje se načítají ze souboru fcí. `readGroupNames()`, která z nich vytváří seznam řetězců.

readAndWrite.cpp (readAndWrite.h)

V tomto souboru se nacházejí funkce, které slouží k načítání, nebo zapisování do konkrétních souborů. Jsou využity extérně jednou nebo více funkcemi z jiných souborů.

globals.cpp (globals.h)

Soubor obsahuje globální proměnné využité obecně ve funkcích, nebo hlavní funkcí. Jsou zde chybové konstanty `errCode`, pro odchytávání chybových návratů z funkcí a pole `errMsg[]` odpovídajících chybových hlášení. Také funkce `printError()`, která tiskne chybovou hlášku, podle předaného kódu a sekundárně další řetězec, který je nepovinný.

5.5 Spuštění programu

Program je konzolový. Při spuštění lze pomocí parametrů nastavit, jaké přesně akce budou provedeny. Následuje seznam parametrů:

`-f <pathsFile>`	Spustí první část, která ze souboru `<pathsFile>`, ve kterém jsou cesty ke složkám s fotkami zvolených skupin, načte fotky, zjistí jejich význačné body a zapíše výsledky do souborů.
`-l`	načte ze souboru přiřazení bodů do slovníku a vytvoří bodové ohodnocení, zapíše jej do souboru.

`-c <pathsFile>` Načte soubory s hodnocením a fotky určené k roztřídění ze složky (složek). Cesta k ní je předaná přes `<pathsFile>`. Vytvoří návrh na umístění do odpovídající třídy fotek a zapíše jej do souboru.

`-cl <pathsFile>` Tímto parametrem je možné spustit obě dvě části za sebou.

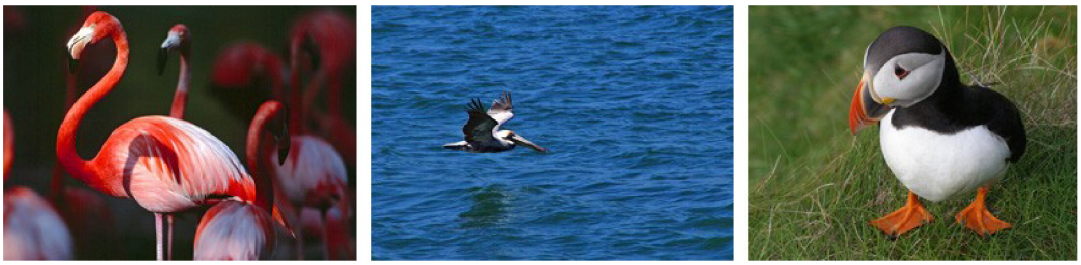
Skript s *kmeans* stačí jednoduše spustit v Matlabu bez dodatečného nastavení. Ale je možné změnit proměnné pro nastavení počtu clusterů a iterací

6 Výsledky

Tato kapitola představuje data na kterých bylo testováno, výsledky průběžného testování navržené aplikace a také finální konečnou schopnost kategorizovat fotky.

6.1 Testovací data

Jsou vybrána zcela náhodně, podobným způsobem jako v kap. 4.6 [13][14]. Testovacích fotek je 120, 30 pro každou kategorii. Žádná se nevyskytuje v trénovacích datech. Následuje ukázka.



Obrázek 6.1: Ukázka z fotografií k roztřídění - Ptáci

Obrázek 6.2: Ukázka z fotografií k roztřídění - Auta

Obrázek 6.3: Ukázka z fotografií k roztřídění - Lidé

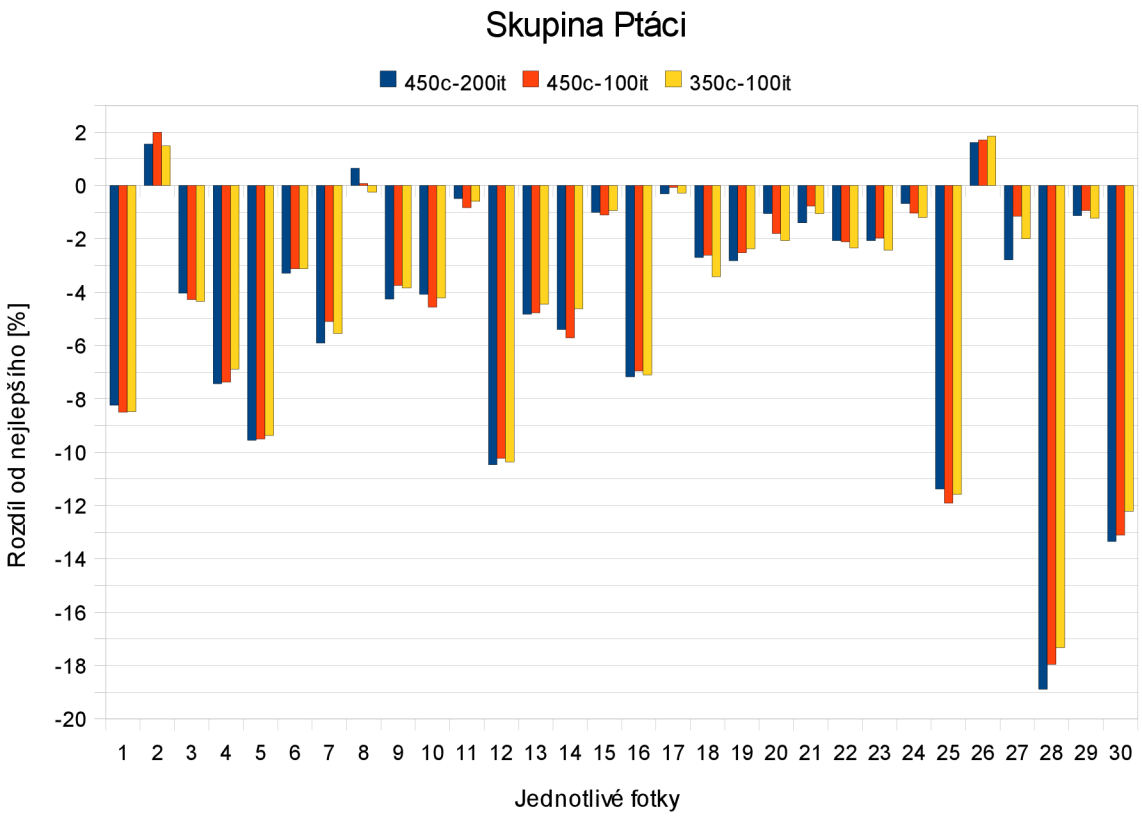
Obrázek 6.4: Ukázka z fotografií k roztřídění - Hory

6.2 Testování nastavení k-means

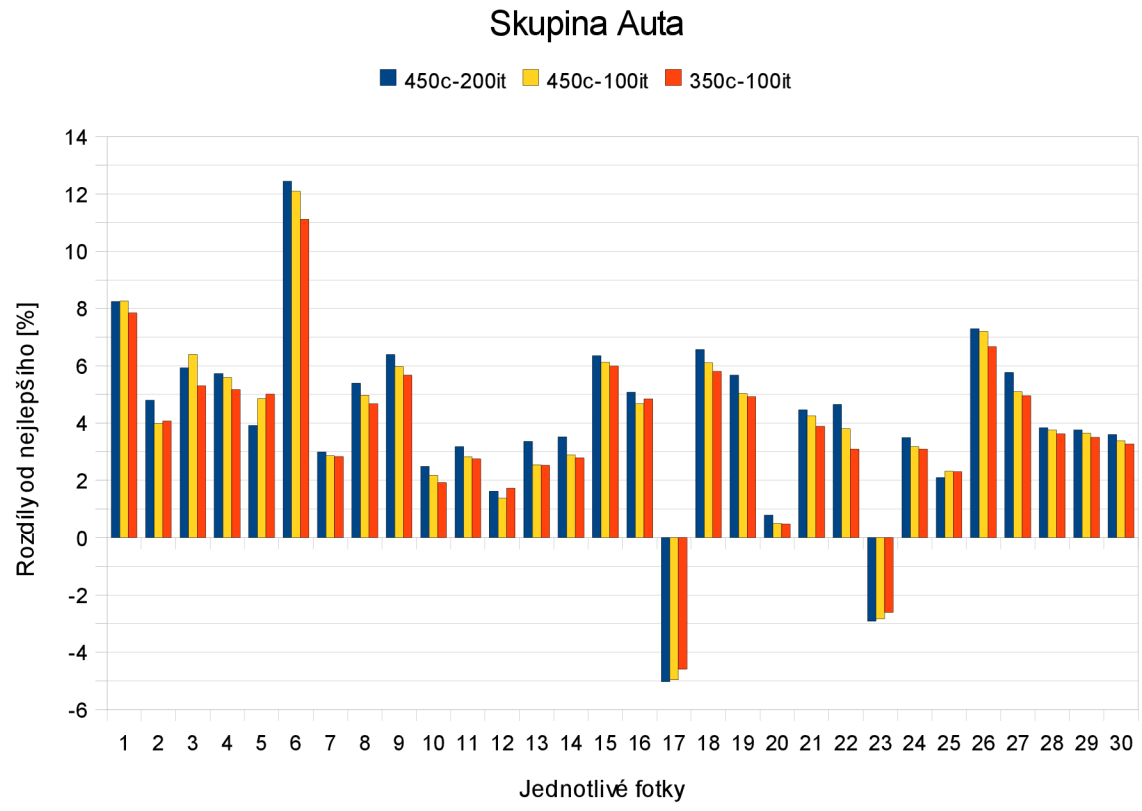
Vytvoření aplikace a návrh byla jedna část řešení. Druhá je správné nastavení běhu jednotlivých částí, aby podávaly co nejlepší výsledky. Důležitou roli hraje velikost vizuálního slovníku. Experimentálně byla stanovena hodnota 450 shluků. Pro porovnání i 350. Bohužel s vyššími počty (od 500) si už Matlab neporadil a způsobovaly chybu nedostatek paměti – `Out of memory`. To je také důvod snížení celkového počtu zaznamenaných význačných bodů přibližně na 153 000 z původního počtu okolo 860 000. Běhy *k-means* mají vždy 3 opakování s různě inicializovanými středy clusterů. Bylo ponecháno defaultní nastavení *k-means* na 100 iterací jednoho běhu. Ale testováno i 200 iterací, jestli se nedosáhne lepších výsledků.

V následujících grafech vidíte výsledky 3 testových běhů pro zjištění optimálního nastavení. Výsledky jsou zobrazeny v procentech jako rozdíl procentuálního ohodnocení správné skupiny s tou s největším skórem z ostatních. Proto kladné hodnoty značí, že odpovídající fotka byla správně přiřazena a jak moc se liší od druhé nejlepší, zatímco záporné ukazují kolik procent ztrácí na nejlepší skupinu, tedy aby byla správně zařazena. Písmena za čísly v legendě mají následující význam:

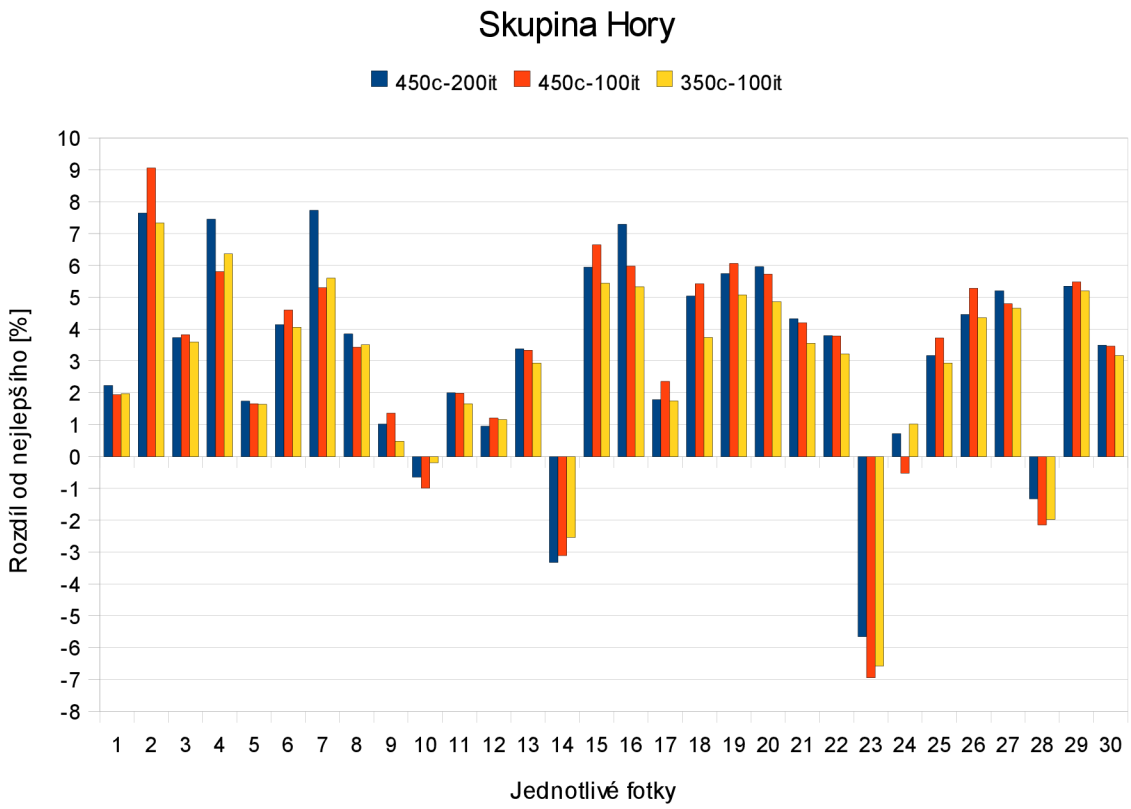
- `c` – počet použitých shluků
- `it` – počet iterací v rámci jednoho běhu.



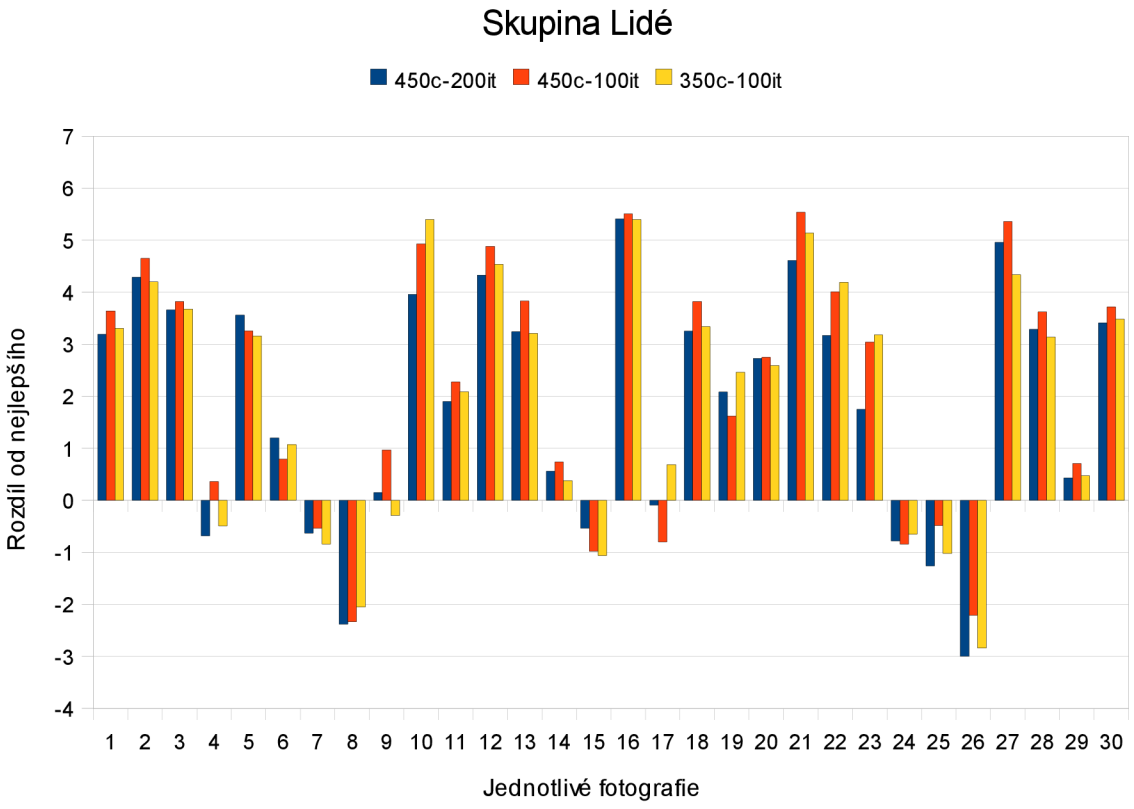
Tabulka 6.1: Rozdíl v odhadu přiřazení do skupiny pro skupinu Ptáci



Tabulka 6.2: Rozdíl v odhadu přiřazení do skupiny pro skupinu Auta



Tabulka 6.3: Rozdíl v odhadu přiřazení do skupiny pro skupinu Hory



Tabulka 6.4: Rozdíl v odhadu přiřazení do skupiny pro skupinu Lidé

Malý rozdíl zobrazený v grafu ještě nemusí znamenat, že skupina byla slabě ohodnocená. Mohla dostat velké množství bodů, ovšem spolu s některou další skupinou. Poukazuje to tedy na fakt, že je fotka špatně rozlišitelná. Další podrobnosti uvádí následující tabulky:

Skupiny	450c-200it	450c-100it	350c-100it
Ptáci	-132,99	-129,97	-130,26
Auta	125,46	118,04	112,59
Hory	97,19	92,71	83,31
Lidé	55,77	65,67	60,22
Celkem	145,43	146,45	125,86

Tabulka 6.5: Celkové součty rozdílů pro nastavení v rámci skupin

Skupiny	450c-200it	450c-100it	350c-100it
Ptáci	3 (10,00%)	3 (10,00%)	2 (6,66%)
Auta	28 (93,33%)	28 (93,33%)	28 (93,33%)
Hory	26 (86,66%)	25 (83,33%)	26 (86,66%)
Lidé	22 (73,33%)	23 (76,66%)	22 (73,33%)
Celkem	79 (65,82%)	79 (65,82%)	78 (65,00%)

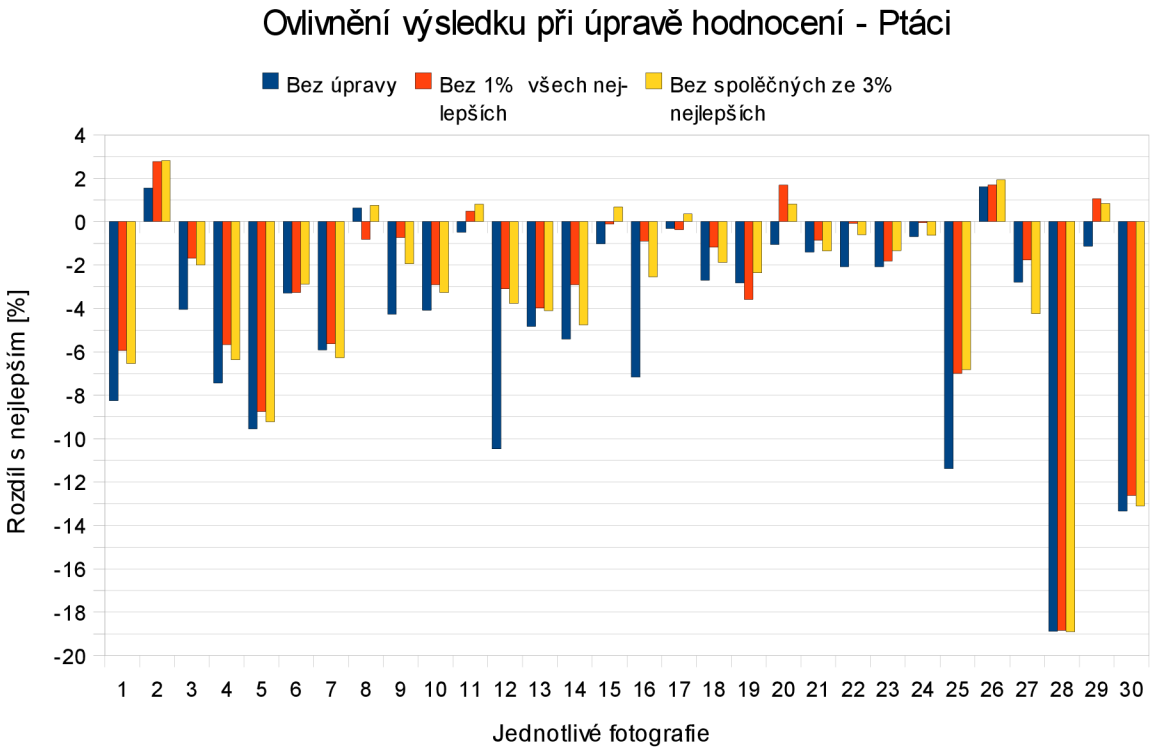
Tabulka 6.6: Počty správně určených fotek (z celkového počtu 30)

Na počty správně přiřazených fotek jsou si shlukování s rozdílným nastavením téměř rovny. O jednu fotku hůře odhadlo přiřazení do skupiny spuštění s 350 clustery. Z tabulky 6.5 lze vyčíst vlastně naprosté zprůměrování přes všechny fotky i přes skupiny. Výsledné hodnoty ukazují, že použití více clusterů dává přeci jen o něco lepší výsledky. Zatímco u zvýšeného počtu iterací v *k-means* se dá konstatovat že nepřinesl žádná zlepšení.

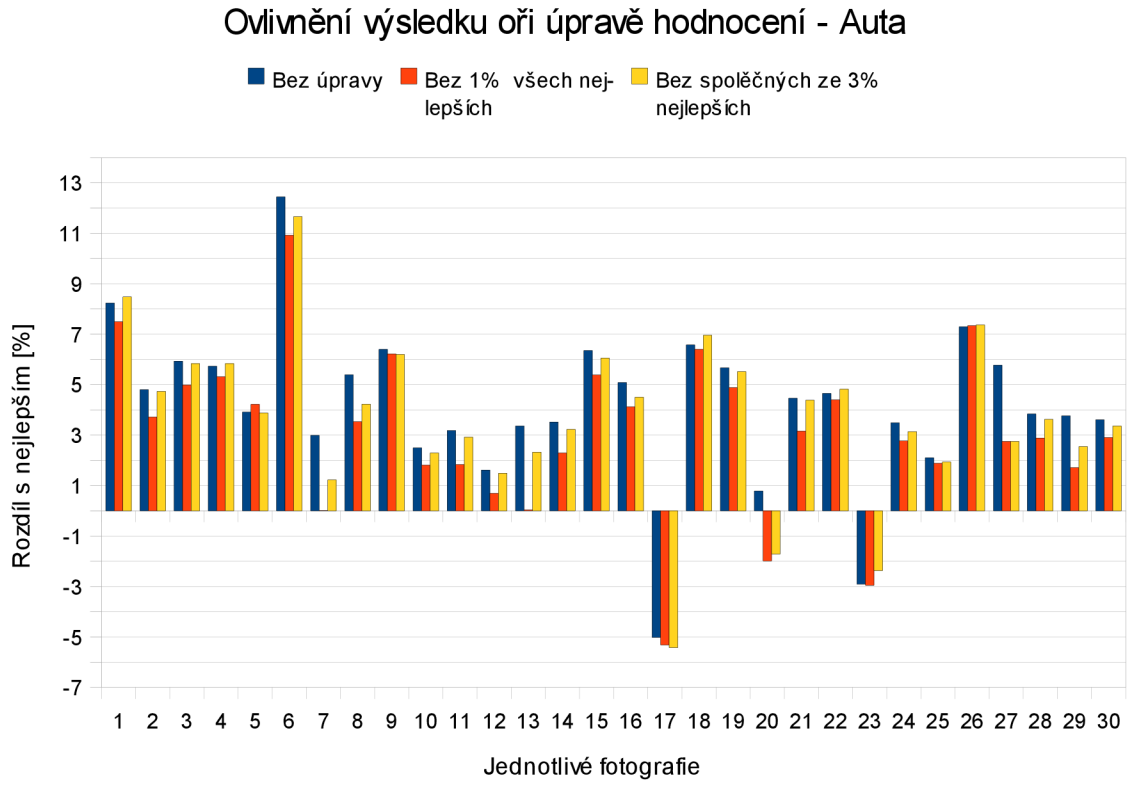
Už v ukázaných grafech, ale i v tabulce 6.6 je patrná kolísavost úspěšnosti přiřazení. Přestože auta ale i hory systém poznává velmi dobře. U lidí už je to horší a u fotografií ptáků opravdu špatné.

6.3 Vylepšování vyhodnocování

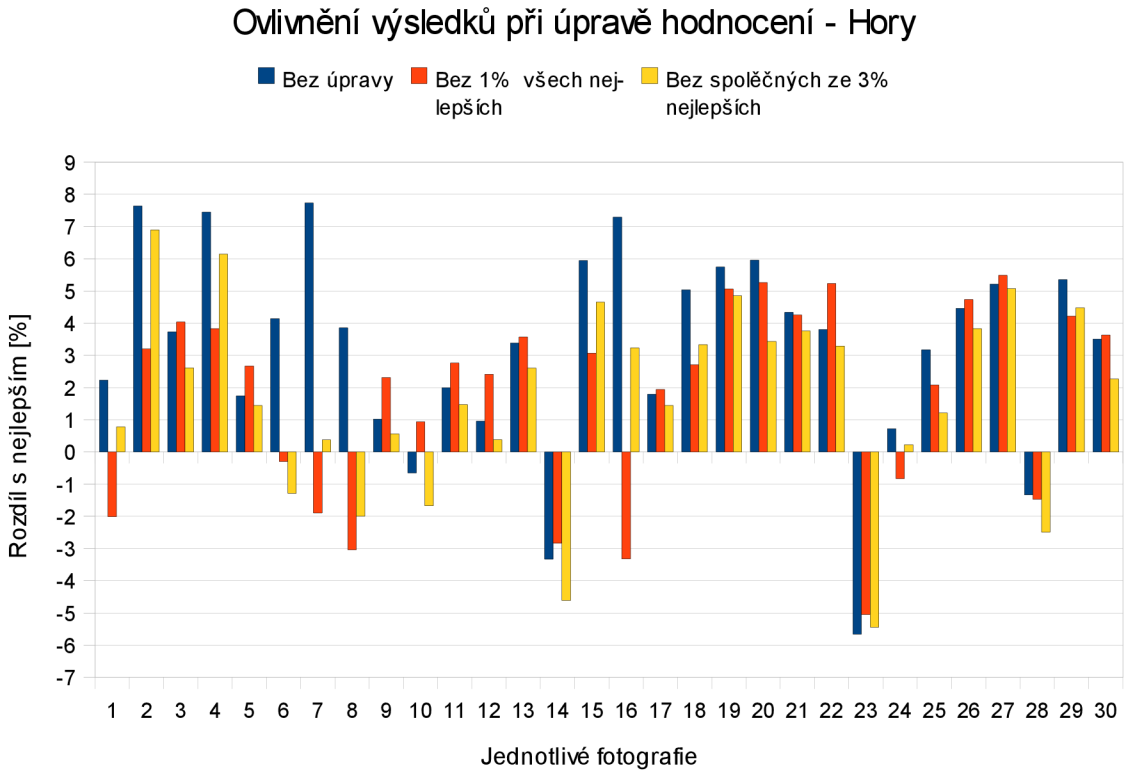
O vylepšení hodnocení se snaží úpravy ve vytváření ohodnocení skupin. Výsledky jsou prezentovány v následujících grafech. Nastavení shlukování je ponecháno na 450 clusterech a 200 iteracích. Položka v grafu „Bez úprav“ *(modrá)* značí nijak neupravované výsledky shlukování. Následující (*oranžová*) položka značí, že 1% z nejčastěji použitých slov bylo vynulováno u poslední *(žluté)* byly brány v úvahu 3% nejlepších výsledků, ze kterých se vynulovaly ty počty slov, které byly i mezi 3% alespoň jedné jiné skupiny. Výsledné ohodnocení bylo vytvořeno až na základě upravených počtů.



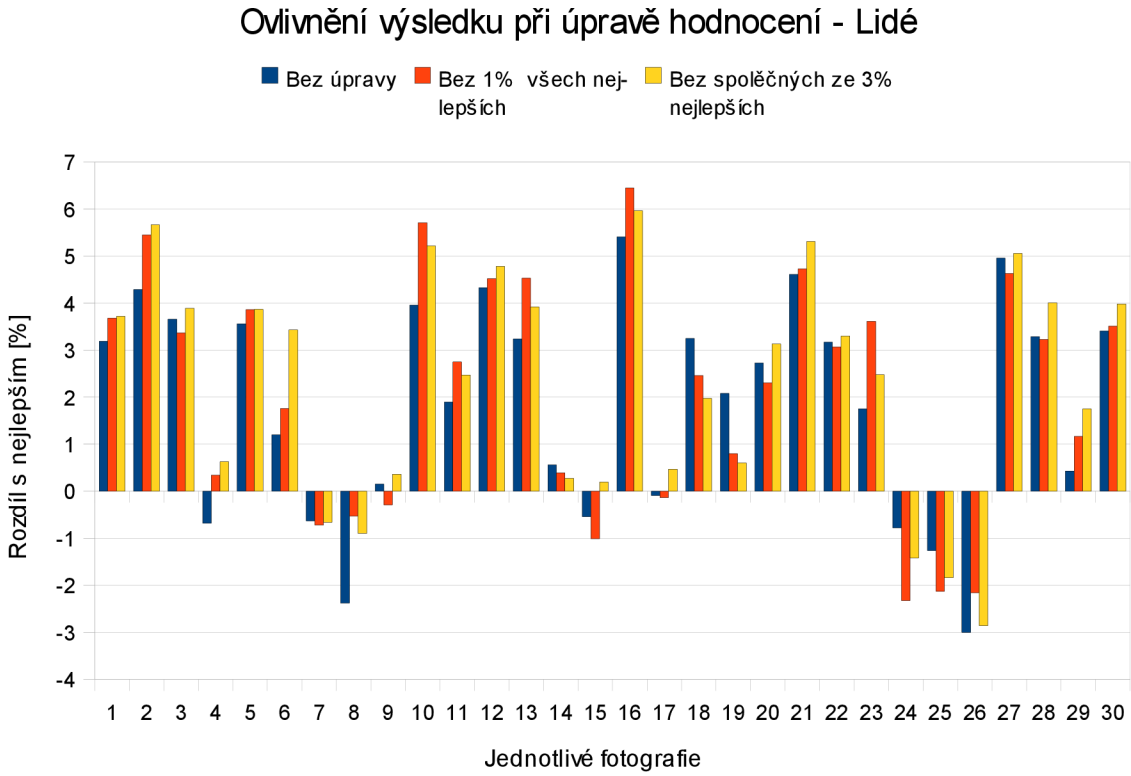
Tabulka 6.7: Různé výsledky hodnocení skupiny Ptáci



Tabulka 6.8: Různá hodnocení skupiny Auta



Tabulka 6.9: Různá hodnocení skupiny Hory



Tabulka 6.10: Různá hodnocení skupiny Lidé

Shrnuté hodnoty z grafů:

Skupiny	Bez úprav	Bez 1% všech nejlepších	Bez společných ze 3% nejlepších
Ptáci	-132,99	-86,72	-95,8
Auta	125,46	93,39	111,82
Hory	97,19	52,64	50,78
Lidé	55,77	63,02	68,78
Celkem	145,43	122,33	135,58

Tabulka 6.11: Celkové součty rozdílů pro různá ohodnocení

Nějaký text

Skupiny	Bez úprav	Bez 1% všech nejlepších	Bez společných ze 3% nejlepších
Ptáci	3 (10,00%)	5 (16,66%)	8 (26,66%)
Auta	28 (93,33%)	27 (90,00%)	27 (90,00%)
Hory	26 (86,66%)	21 (70,00%)	24 (80,00%)
Lidé	22 (73,33%)	22 (73,33%)	25 (83,33%)
Celkem	79 (65,82%)	75 (62,50%)	84 (70,00%)

Tabulka 6.12: Počty správně určených fotek (30 na skupinu)

Pokud se vyhodnotí pouze tabulka s rozdíly 6.11, obě vylepšení dopadla o něco hůře než původní výsledky. To vyplývá především z odebráním nejlépe ohodnocených slov, které zvyšovaly celkové body a snížily se rozdíly mezi skupinami. Z tabulky 6.12 je vidět, že s těmi slovy, které rušily hodnocení a odstranění mohlo zvýšit rozlišitelnost fotky pro skupinu, se odebraly i ty, které ji výrazněji pomáhaly. V případě odebrání 1% nejlepších, přestože se jednalo o malou část slov, stačilo to na snížení úspěšnosti. Situace je ale jiná u selektivního odstranění. Přestože se taktéž snížil průměrný rozdíl při přiřazení, počet dobře klasifikovaných fotek vzrostl. K vynulování slov stačila i pouhá jedna shoda s nejlepšími slovy z jiné skupiny. Tedy největší závislost mezi dvěma skupinami byla sníženy. Podle grafů (tab 6.7 – 6.10) je vidět, že zlepšení nebylo jednoznačné a především v těch úspěšnějších skupinách došlo k mírnému zhoršení.

Výsledná nejlepší konfigurace – 450 slov slovníku z 200 iterací, a selektivním nulováním 3% nejlépe ohodnocených slov z každé skupiny se dosáhlo na 120 testovacích fotkách celkové úspěšnosti 70%. V rámci skupin dopadlo nejlépe rozlišování hor: 90% a nejhůře ptáků: 26,66%.

6.4 Zhodnocení použitých knihoven

Využití zmíněných knihoven (kap 5.1, 5.2) bylo dobrou volbou. Práce s Qt šla dobře od ruky, díky množství implementovaných metod, které usnadňují práci. Vše je také lépe nachystáno pro vytvoření GUI, které nakonec zůstalo nerealizováno.

Knihovna OpenCV je pro práci s grafikou vhodná, jak jsem si ověřil. Hlavní problém, který mě ale potkal, byl ve funkci `cvKMeans2()` určené ke shlukování a kterou jsem měl v plánu použít. Po začlenění do vlastního kódu a nastavení všech potřebných parametrů bohužel nefungovala jak by měla. Z celkového počtu clusterů byly zpracovávané vektory vždy přiřazeny jen k několika středům a většina zůstala prázdná. Předpokládám, že samotná implementace je správná, protože je to velmi rozšířená knihovna. Tento problém se začleněním *k-means*, přestože jsem na něm strávil mnoho času hledáním vlastních chyb, se mi nepodařilo vyřešit. To je důvod, proč nakonec došlo k rozdělení celé aplikace a použití Matlabu. S další potíží jsem se setkal při použití funkcí pro *kd-tree*. K těm totiž chybí dokumentace. Proto jsem až při prohledávání různých fór zjistil, jak se přesně jmenují funkce, které chci použít a přesný formát použití až studiem z hlavičkového souboru. Tyto nepříjemnosti mi poněkud zhoršily dojem z OpenCV, i přesto byla knihovna k mému řešení velmi příhodná.

6.5 Možná rozšíření

Zde je několik návrhů na rozšíření a možné vylepšení programu. Například začlenění shlukování přímo do programu, aby uživatel nebyl závislý na Matlabu a program se stal ucelenějším. Tedy nejspíš dokázat využít funkci `cvKMeans2()`.

Uživatelsky příjemnější grafické rozhraní oproti konzolovému určitě stojí za zmínění. Kdy by si uživatel například mohl podle míry jistoty nechat fotky natvrdo roztřídit a pak v náhledech a se zobrazením spočítaných pravděpodobností ručně roztřídit zbytek.

V současném stavu program uchovává informace vždy jen o jednom vizuálním slovníku a ohodnocení. Dal by se vytvořit nějaký lepší systém uchovávání informací, třeba s využitím XML a podporou několika různých slovníků a ohodnocení, ke kterému by se i lépe přistupovalo právě přes grafické rozhraní.

7 Závěr

V této práci je prezentována myšlenka vytvoření aplikace schopné automaticky kategorizovat fotky na základě předem daných skupin. Je popsán její návrh, použité algoritmy, samotná implementace i dosažené výsledky.

Členění do skupin je založeno na vizuálním slovníku, vytvořeného ze všech skupin. Nejdříve se najdou a popíší význačné body metodou *SURF*. Ty jsou pak seshlukovány pomocí metody *k-means*. Jednotlivé shluky jsou vlastně slovy vytvořeného vizuálního slovníku. Na základě statistické analýzy počtů vyskytujících se slov v jednotlivých skupinách je vytvořeno ohodnocení každého slova v rámci skupiny. Nové fotky, určené ke kategorizaci, jsou také popsány pomocí *SURF*u. Příznakové vektory jsou namapovány na slova přes *kd-tree*. Pomocí už vytvořeného hodnocení se spočítá celkový počet bodů pro každou skupinu a určí se pravděpodobnost přiřazení. Nejvyšší pravděpodobnost značí nejvhodnější skupinu.

Trénovací skupiny byly vytvořeny z přibližně 100 fotek pro každou ze 4 skupin. Testovacích fotek bylo celkem 120, po 30 pro každou ze skupin. Celková průměrná úspěšnost zařazení je 70%. Nejlepší 90% a nejhorší 26,66% v rámci jedné skupiny pro finální konfiguraci programu. Ta byla v rámci testování ustanovena na 450 slov vizuálního slovníku, při 200 iteracích při jeho sestavení a selektivního nulování nejlepších společných hodnot při vytváření bodového ohodnocení skupin. Cíl této práce byl splněn. Pomocí popsaných kroků a navržených vylepšení dokáže aplikace úspěšně přiřadit fotce správnou skupinu téměř ve třech čtvrtinách případů. Tato schopnost ale kolísá u různých skupin. Je tedy možnost budoucího pokračování pro zlepšení výsledků.

Další prací na projektu by mohlo být přepracování vizuální slovníku. Vytváří se jeden velký pro všechny skupiny. Lepším řešením by bylo vytvořit pro každou skupinu vlastní slovník. Pokud totiž ve stávajícím řešení chce uživatel změnit jednu skupiny do které se roztřiďuje, musí vytvořit kompletně nový slovník. To je velmi časově náročné. Takhle by stačilo zpracovat fotky jen nové skupiny. Taky by se tím vyřešil problém paměťové velikosti najednou zpracovávaných příznakových vektorů a mohl by se zvýšit počet slov ve slovníku, což by zase mohlo vést k lepším výsledkům.

Pokud se jedná o změny hlavního návrhu, dal by se poměrně naivní způsob vytváření ohodnocení, na základě kterého se klasifikuje nahradit metodou strojového učení *SVM* (*Support Vector Machines*) tak jako je použit v [10].

Literatura

[1] Darrell, T.: Visual object and activity recognition [pdf]. 6.9.2009 [cit. 11.5.2010]. Dostupný z WWW: <http://www.eecs.berkeley.edu/~trevor/CS294PublicFiles/04Local%20Features%20Lecture/Local%20Features%20Lecture.pdf>.

[2] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-Up Robust Feature In Computer Vision and Image Understanding [pdf]. Elsevier Science Inc., New York, 2008 [cit. 11.5.2010]. Svazek 110, 3. 346-359. ISSN: 1077-3142. Dostupný z WWW: <ftp://ftp.vision.ee.ethz.ch/publications/articles/eth_biwi_00517.pdf>.

[3] Lowe, D., G.: Object Recognition from Local Scale-Invariant Featuters In Seventh International Conference on Computer Vision [pdf]. University of British Columbia, Vancouver, Sept. 2009 [cit. 11.5.2010]. Svazek 2. ISBN 0-7695-0164-8. Dostupný z WWW: <http://www.eecs.berkeley.edu/~trevor/CS294PublicFiles/04Local%20Features%20Lecture/Local%20Features%20Lecture.pdf>.

[4] Španěl, M.: Klasifikace a rozpoznávání: Unsupervised techniky [pdf]. FIT VUT Brno, Brno, 2009.

[5] kd-tree [online]. Poslední aktualizace: 5.5.2010 [cit. 13.5.2010]. Dostupné z WWW: <http://en.wikipedia.org/wiki/Kd-tree>

[6] Winn, J., Criminisi, A., Minka, T.: Object Categorization by Learned Universal Visual Dictionary In ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision [pdf]. IEEE Computer Society, Washington DC, 2005. Svazek 2. 1800 - 1807. ISBN 0-7695-2334-X-02. Dostupné z WWW: <http://research.microsoft.com/pubs/67408/criminisi_iccv2005.pdf> .

[7] Fergus, R., Perona, P., Zisserman, A.: Object class recognitionby unsupervised scale-invariant learning In Computer Vision and Pattern Recognition [pdf]. Madison, 2003. 264-271. Dostupné z WWW: <http://people.csail.mit.edu/fergus/papers/fergus03.pdf>.

[8] Ullman, S., Sali, E., Vidal-Naquet, M.: A Fragment-Based Approach to Object Representation and Classification In Proceedings of the 4th International Workshop on Visual Form. Springer-Verlag, London, 2001. Svazek 2059/2001. 85-100. ISBN 3-540-42120-3. .

[9] Kumar, M., P., Torr, P.,H., S., Zisserman, A.: Extending pictorial structures for object recognition In Proc. of BMVC [online]. 2004. 158-164. Dostupné z WWW: <http://eprints.pascal-network.org/archive/00000252/01/kumar04.pdf>.

[10] Csurka, G., Dance, C., R., Fan, L., Willamowski, J., Bray, C.: Visual categorization with bags of keypoints In Workshop on Statistical Learning in Computer Vision [pdf]. ECCV, 2004. 1-22. Dostupné z WWW: <http://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/csurka-eccv-04.pdf>.

[11] Varma, M., Zisserman, A.: A Statistical Approach to Texture Classification from Single Images In International Journal of Computer Vision [pdf]. 2005. Svazek 62. 61-81. ISSN: 0920-5691. Dostupné z WWW: <http://research.microsoft.com/en-us/um/people/manik/pubs%5Cvarma05.pdf>.

[12] Domingos, P., Pazzani, M.: On the Optimality of the Simple Bayesian Classifier under Zero-One Loss In Machine Learning [pdf]. Kluwer Academic Publishers, Hingham, 1997. Svazek 29. 103-130. ISSN: 0885-6125. Dostupné z WWW: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.40.1930&rep=rep1&type=pdf>.

[13] Přejaté fotografie ptáků [online] [přejato 15.5.2010]. Dostupné z WWW: <http://birdphoto.cz/>, <http://www.wildbirdgallery.com>, <http://www.wallpapers247.com/images/wallpapers>, <http://animals.nationalgeographic.com/animals/photos/bird-wallpapers/>, <http://www.animalspapper.com/category/birds/>, <http://crazy-frankenstein.com>.

[14] Přejaté fotografie aut [online] [přejato 15.5.2010]. Dostupné z WWW: <http://www.carsbase.com/>, <http://lethistorylive.com/>, <http://cars.88000.org/>.

Seznam příloh

Příloha 1. DVD obsahuje:

- spustitelný soubor
- všechny zdrojové kódy a potřebné knihovny, aby byl možný překlad
- programovou dokumentaci
- demonstrační data
- ukázkové výsledky z předchozích běhů programu
- PDF verzi této bakalářské práce
- ODT verzi této bakalářské práce
- README.txt popisující obsah DVD a nápovědu k instalaci a spuštění