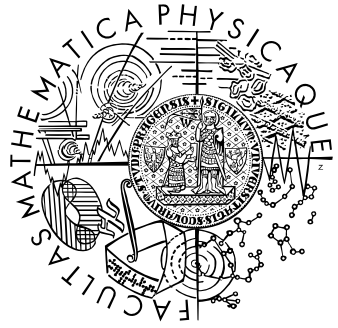
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Ivana Lukšová

Klasifikace bitmapových obrázků

Katedra teoretické informatiky a matematické logiky

Vedúci bakalárskej práce: RNDr. Pavel Surynek, Ph.D.

Študijný program: Obecná informatika

2010

Na tomto mieste by som chcela poďakovať vedúcemu tejto práce, RNDr. Pavlovi Surynkovi, Ph.D., za ponúknutie zaujímavej témy, konzultácie a pomoc pri vytváraní práce. Na druhom mieste ďakujem Vladimírovi Lukšovi a Lenke Ocetovej za poskytnutie fotografií použitých na testovanie.

Prehlasujem, že som svoju bakalársku prácu napísala samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a s jej zverejňovaním.

V Prahe dňa 1. 8. 2010 Ivana Lukšová

Obsah

Názov práce: Klasifikace bitmapových obrázků

Autor: Ivana Lukšová

Katedra: Katedra teoretické informatiky a matematické logiky

Vedúci bakalárskej práce: RNDr. Pavel Surynek, Ph.D.

E-mail vedúceho: Pavel.Surynek@ktiml.mff.cuni.cz

Abstrakt:

V tejto práci navrhujeme metódu na automatickú klasifikáciu bitmapových obrázkov na základe obsahu do tried ako budovy, krajinky, fotografie apod. Metóda využíva koncepty strojového učenia, konkrétne mechanizmus rozhodovacieho stromu. Z obrázkov sú extrahované charakteristiky ako kontrast, rozloženie farieb, výskyt priamych línií. Na základe týchto charakteristík je vybudovaný rozhodovací strom použitím algoritmu ID3. Súčasťou práce je vytvorenie programu, ktorý umožňuje operácie s databázou obrázkov vzhľadom k výsledkom klasifikácie – vyhľadávanie, triedenie do adresárov atď. Metóda má dosahuje úspešnosť 75-85% v rámci tried.

Kľúčové slová: klasifikácia obrázkov, rozpoznávanie obrazu, rozhodovací strom, spracovanie obrazu

Title: Classification of bitmap pictures

Author: Ivana Lukšová

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: RNDr. Pavel Surynek, Ph.D.

Supervisor's e-mail address: Pavel.Surynek@ktiml.mff.cuni.cz

Abstract:

In this work we propose a method of automatic content-based bitmap pictures classification into classes like buildings, landscapes, photographs etc. The method uses concepts of the machine learning, concretely the mechanism of a decision tree. We extract characteristics like the contrast, the distribution of colors, the presence of straight lines. On the basis of these characteristics the decision tree is built using the algorithm ID3. Creation of the program, which allows operations with the database of images based on the classification results – retrieval, sorting into directories etc. is also a part of the work. The method shows rate 75-85% correct within the classes.

Key words: picture classification, image recognition, decision tree, image analysis

1 Úvod

Bitmapový obrázok je označenie jednak celej kategórie digitálneho obrazu, jednak skupiny formátov súborov, kde je grafická informácia uložená v podobe dvojrozmernej matice farebných bodov - pixelov do bitovej mapy. Výraz bitová mapa je stále používaný z historických dôvodov, presné označenie by bolo pixelová mapa. Spolu s vektorovým obrázkom predstavujú dva základné spôsoby, ako je v počítači možné uložiť obrazovú informáciu. Bitmapový obrázok je pre jeho jednoduché zobrazovanie, nenáročnú editovateľnosť a priamu získateľnosť zo zariadení ako digitálny fotoaparát či skener, najbežnejší formát používaný v súčastnosti: je využívaný na ukladanie digitálnych fotografií, naskenovaných dokumentov, rôznych renderovaných obrazov, počítačových ilustrácií a pod.

Pri bitmapových obrázkoch sa využíva viacero reprezentácii, najčastejšie je farba konkrétneho pixelu popísaná pomocou trojice hodnôt v celočíselnom rozsahu, zvlášť pre červenú, zelenú a modrú farbu – toto označenie vychádza z farebného modelu RGB obrazoviek monitorov, kde každý odtieň je možné vytvoriť kombináciou týchto troch základných farieb.

V praxi sa často vyskytuje prípad, keď užívateľ pracuje s veľkou sadou bitmapových obrázkov, ktoré potrebuje klasifikovať - triediť do rôznych kategórií v závislosti na tom, čo jednotlivé obrázky predstavujú, či vyhľadávať obrázky podľa ich obsahu. Ako príklad si môžeme predstaviť veľkú internetovú databázu obrázkov, alebo archivovanie fotografií na pevnom disku počítača. Indexovanie tejto databázy na základe obsahu do kategórií je nevyhnutné pri vyhľadávaní v reálnom čase. Táto úloha sa nedá riešiť priamo, bitmapový obrázok takúto informáciu neposkytuje. Môžeme z neho priamo získať údaj o farbe každého pixelu, či spočítať jednoduché charakteristiky ako počet odtieňov alebo histogram, ale prítomnosť budov, prírody, ľudí na obrázku nezistíme. Jedinou možnosťou je ručne klasifikovať celú databázu, čo je veľmi časovo náročné a nepohodlné.

Jedným z riešení tohto problému môže byť metóda na automatickú klasifikáciu bitmapových obrázkov predstavená v tejto práci. Metóda je založená na princípe, že obrázky, ktoré patria do tej istej triedy, sú v určitom zmysle podobné - majú špecifické určité charakteristiky, zároveň obrázky v rozdielnych triedach majú niektoré charakteristiky dostatočne odlišné. Na základe analýzy obrazovej informácie a extrakcií podstatných charakteristík, následne s použitím inteligentných postupov strojového učenia - konkrétne algoritmom rozhodovacieho stromu – vytvoríme také mechanizmy, ktoré túto klasifikáciu umožnia. Tieto mechanizmy následne použijeme vo vytvorenej aplikácií, ktorá bude tvoriť účinný nástroj pre správu bitmapových obrázkov. Aplikácia zároveň overí účinnosť použitia rozhodovacieho stromu ako algoritmu umelej inteligencie v rámci dôležitej úlohy rozpoznávania obrazu.

Práca je členená do siedmich kapitol. Po úvode, kde prestavíme problém, ktorý rieši táto práca a princíp, na ktorom spočíva jeho riešenie, nasleduje druhá kapitola, v ktorej definujemie ciele úlohy, ktorú sa snažíme splniť. Ďalej nasleduje popis a zdôvodnenie použitia techník našej metódy. V štvrtej kapitole opíšeme implementačné detaily týchto techník, v piatej nájdeme popis testovacieho prostredia a prehľad výsledkov našej metódy. Na záver zhodnotíme dosiahnuté výsledky a navrhneme možnosti budúceho rozvoja.

2 Definícia úlohy

Názov automatická klasifikácia bitmapových obrázkov naznačuje, že cieľom nášho programu bude spracovanie sady obrázkov a automatické zaradenie každého obrázku do istých tried na základe toho, čo daný obrázok reprezentuje. Presnejšie sa bude jednať o priradenie istej množiny príznakov ku každému obrazu, kde každý príznak bude jednoznačne určovať, či daný obraz patrí alebo nepatrí do konkrétnej triedy. Triedy obrázkov budú predom určené, bude sa jednať o bežné „kategórie“ obrazov – napríklad trieda fotografie, trieda kreslené obrázky, trieda príroda apod. Triedy nebudú disjunktné, to znamená, že každý obrázok môže, ale nemusí patriť do viacerých tried.

Na základe týchto príznakov bude následne uskutočnená ďalšia manipulácia s obrázkami či celými sadami obrázkov, v ktorých budeme môcť vyhľadávať podľa príznakov, triediť do adresárov a pod. Keďže predpokladáme, že klasifikácia nebude mať stopercentnú úspešnosť, príznaky bude možné editovať aj ručne.

Získanie mechanizmu, ktorý bude umožňovať túto klasifikáciu, bude spočívať z viacerých krokov.

2.1 Určenie klasifikačných tried

Pre dosiahnutie čo najpoužiteľnejších výsledkov v rozpoznávaní obrazu je nutné vybrať vhodné triedy, do ktorých budeme obrázky klasifikovať. Pri vytváraní klasifikačného mechanizmu predpokladáme, že obrázky v rámci jednej triedy budú mať určité obrazové charakteristiky dostatočne podobné, zatiaľ čo obrázky nepatriace do danej triedy by sa mali dať na základe rovnakých či iných charakteristík rozlíšiť.

Pri výbere tried je rovnako kladený dôraz na ich praktickú použiteľnosť z pohľadu užívateľa, preto použitie klasifikačného mechanizmu by malo byť možné pre čo najširšiu paletu bitmapových obrázkov.

Z vyššie uvedeného vyplýva, že každá trieda by nemala byť priveľmi všeobecná, aby bolo možné nájsť vhodné charakteristiky a zároveň množina tried by mala pokrývať čo najväčšiu oblasť z bitmapových obrázkov, a to najmä kreslených či renderovaných obrázkov a digitálnych fotografi.

V rozsahu bakalárskej práce sme pre klasifikáciu vybrali päť nasledujúcich zaujímavých tried: Konkrétne sa jedná o tieto triedy: *fotografie* – pre obrázky získané digitálnym fotoaparátom alebo naskenovaním papierových fotografií,

kreslené obrázky, ktoré budú zahŕňať najrôznejšie počítačové ilustrácie, *krajinky* pre obrázky znázorňujúce panorámu krajiny z väčšej vzdialenosti, ďalej *budovy* pre obrázky predstavujúce exteriér budov a nakoniec *makro objekty,* tie budú predstavovať fotografie rôznych objektov z veľmi malej vzdialenosti. Naša metóda sa však dostatočne obecná, aby sa dala použiť aj pre ďalšie kategórie obrázkov.

2.2 Určenie a extrakcia vhodných charakteristík

V rámci každej triedy je potrebné nájsť obrazové charakteristiky, ktoré sú špecifické pre obrázky v danej triede. Ako príklad si uveďme triedu *budovy* – môžeme predpokladať, že na obrázkoch s budovami sa bude nachádzať zvýšené množstvo vertikálnych a horizontálnych línií. Takýchto výstižných charakteristík je vhodné nájsť ideálne viacero, aby sme dosiahli čo najlepšie výsledky klasifikácie.

Následne je potrebné nájsť a vymyslieť algoritmy a techniky, pomocou ktorých tieto charakteristiky dokážeme vyextrahovať z obrazu a reprezentovať vo forme vhodnej pre klasifikačný mechanizmus, to znamená že implicitne vyjadríme hodnotu konkrétnej charakteristiky, najčastejšie v číselnej podobe – desatinná alebo celočíselná hodnota. Z hodnôt týchto charakteristík pre každý obrázok následne zostavíme vektor, kde každá jeho zložka bude mať presný význam – napríklad bude vyjadrovať počet farieb a pod. Pre obrázky s n charakteristikami tento vektor bude nad R^n.

2.3 Výber klasifikačného mechanizmu

V našej aplikácií sa pracuje s veľkými sadami dát – bitmapovými obrázkami, pričom z každého obrázku je možné získať viacero charakteristík. Potrebujeme preto nájsť vhodný klasifikačný mechanizmus, ktorý dokáže spracovať toto veľké množstvo dát a na základe hodnôt týchto charakteristík vytvoriť systém pravidiel, podľa ktorých sa bude prebiehať klasifikácia. Tieto požiadavky spĺňajú mechanizmy strojového učenia:

Strojové učenie je vedecká disciplína zaoberajúca sa mnohými algoritmami a technikami – všetky však majú spoločné to, že správanie týchto algoritmov sa vyvíja od určitých dát, ktoré sú získané „bežným“ spôsobom. Presnú definíciu nájdeme v knihe Machine Learning [7] :

> *„O počítačovom programe tvrdíme, že sa učí zo skúsenosti E s ohľadom na určitú triedu úloh T a meraním výkonnosti P, ak jeho výkonnosť v úlohách v T, meraná pomocou P, sa zvyšuje so skúsenosťou E.“*

Medzi algoritmy vytvárajúce triediace mechanizmy z určitých zdrojových dát patria neurónové siete a rozhodovacie stromy, v našej aplikácií sme použili jednu z implementácií posledného spomenutého algoritmu – zdôvodnenie je vysvetlené v nasledujúcej kapitole.

2.4 Proces tvorby klasifikačného mechanizmu

Po nájdení charakteristík každej triedy obrázkov a výbere vhodného klasifikačného mechanizmu je potrebné tento mechanizmus vhodným algoritmom vytvoriť, pričom proces tvorby bude prebiehať zvlášť pre každú triedu. Tento proces bude pozostávať najprv z výberu trénovacej množiny, kde každý prvok bude tvorený obrázkom a jeho priradením do triedy, ktoré môže byť buď pozitívne alebo negatívne. Na týchto obrázkoch sa bude následne prevádzať analýza a extrakcia charakteristík a nakoniec prebehne samotný algoritmus učenia, ktorý na základe hodnôt týchto charakteristík vytvorí klasifikačný mechanizmus .

Keďže výber trénovacích dát má vplyv na proces tvorby, budeme testovať viacero rozličných množín a vyberieme tú, pre ktorá náš klasifikačný mechanizmus vracia najpresnejšie výsledky. Úspešnosť klasifikačného mechanizmu budeme overovať na testovacej množine, ktorá je odlišná od trenóvacej. Obe množiny musia byť dostatočne veľké a rozmanité, aby sme čo najvernejšie overili účinnosť nášho klasifikačného algoritmu v praxi.

2.5 Užívateľské prostredie a klasifikácia

Súčasťou úlohy je vytvorenie prehľadného grafického užívateľského prostredia, ktoré bude využívať náš klasifikačný mechanizmus - bude umožňovať klasifikáciu sady obrázkov pri zadaní rôznych parametrov, prezeranie a editáciu výsledkov. Užívateľské prostredie bude vytvorené s ohľadom na čo najväčšiu možnú funkčnosť, takže aplikácia bude umožňovať nielen klasifikáciu, ale aj manipuláciu s už klasifikovanými sadami obrázkov – vyhľadávanie podľa príznakov priradených klasifikáciou a triedenie obrázkov do zložiek. Takisto bude umožňovať prezeranie obrázkov, zobrazenie základných charakteristík ako histogram a pod.

Výsledky klasifikácie – príznaky patriace ku každému obrázku, ktoré budú označovať príslušnosť či neprísluśnosť do istej triedy. Pretože klasifikácia, ktorá zahŕňa proces extrakcie charakteristík z obrazu je časovo náročná, budú vypočítané výsledky ukladané na disk spolu s obrázkami. Ukladanie výsledkov bude takisto zvyšovať použiteľnosť programu a umožňovať pohodlné testovanie metódy. Klasifikačné výsledky budú obsahovať výsledky klasifikácie pre rôzne triedy a pre rôzne parametre analýzy.

3 Techniky

Úspech našej metódy je závislý na dôkladnom výbere techník a algoritmov jednak pri spracovaní obrazovej informácie a extrahovaní príslušných charakteristík, jednak pri samotnej klasifikácií. V našej aplikácií boli použité viaceré techniky z oblasti spracovania obrazu a umelej inteligencie, podrobne ich popíšeme a zároveň zdôvodníme ich použitie v nasledujúcej kapitole.

3.1 Rozhodovací strom

Rozhodovací strom je kľúčovým mechanizmom celej aplikácie. Práve tento mechanizmus je zodpovedný za správne priradenie obrázku do triedy. Základný princíp rozhodovacieho stromu je popísaný v knihe Machine Learning [7]: Princíp spočíva vo vytvorení pravidiel, podľa ktorých sa objekty budú klasifikovať do tried, na základe určitej trénovacej sady objektov, u ktorých je zaradenie do tried známe.

3.1.1 Úloha rozhodovacieho stromu

Ako už bolo popísane vyššie, úloha rozhodovacieho stromu je na základe množiny klasifikačných pravidiel priradiť triedu k ľubovoľnému objektu. Presne formulované, je rozhodovací strom reprezentuje

> *„funkciu, ktorej vstupom je vektor hodnôt a výstupom je rozhodnutie – jediná výstupná hodnota“* [11].

Základným pojmom je množina *objektov*. Platí, že každý objekt v tejto množine má pevný počet charakteristík – atribútov. Hodnota atribútu pre daný objekt je prvok z oboru hodnôt tohto atribútu, takže každý objekt je charakterizovaný vektorom – usporiadanou n-ticou hodnôt atribútov. Zároveň platí, že každý objekt patrí do práve jednej klasifikačnej triedy, to znamená, že klasifikačné triedy sú disjunktné.

Keďže v našej aplikácii chceme mať viacero tried, ktoré nemusia byť disjunktné, toto obmedzenie sme vyriešili nasledujúcim spôsobom: Vytvoríme viacero rozhodovacích stromov, jeden pre každú triedu, do ktorej chceme zatrieďovať obrázky. Triedy v rámci každého rozhodovacieho stromu budú potom len dve – v jednej sa budú nachádzať objekty, ktoré patria do našej triedy (pozitívna trieda), v druhej naopak tie, ktoré nepatria (negatívna trieda).

Ďalším dôležitým pojmom je *trénovacia množina* objektov, to znamená množina objektov, ktorých priradenie do tried je známe. Z tejto trénovacej množiny je na základe induktívnych metód učiaceho algoritmu vytvorený rozhodovací strom.

3.1.2 Štruktúra rozhodovacieho stromu

Štruktúra rozhodovacieho stromu je jednoduchá, ale zato veľmi výkonná. Ako to vyplýva z jeho názvu, štruktúra má stromovú formu, najčastejšie v podobe binárneho stromu.

Obecne platí, že rozhodovací strom je orientovaný strom s ohodnotenými uzlami a hranami, v ktorom každý uzol predstavuje test – výber medzi dvomi alebo viacerými možnosťami a každý list predstavuje rozhodnutie – priradenie objektu do istej triedy.

Pri klasifikácii objektu rozhodovací strom prechádzame od koreňa až po list, vždy testujeme atribút určený daným uzlom, vyberieme správneho potomka na základe hodnoty atribútu a následne algoritmus rekurzívne opakujeme, až kým neskončíme v liste. Zapísané v pseudokóde:

```
function classify(item, root) returns class_type
      if isLeaf(root) then
            return classType(root);
      end if
      attribute = testAttribute(root);
      children_index = perform_test(item, attribute);
      children = childrenAtIndex(root, children_index);
return classify(item, children)
```

Metóda `perform_test(item, attribute)` vráti index hrany na základe hodnoty atribútu objektu a metóda `childrenAtIndex(root, children_index)` vráti potomka aktuálneho uzlu s daným indexom.

3.1.3 Vlastnosti rozhodovacieho stromu

Pri vytváraní rozhodovacieho stromu sa môže vyskytnúť problém, či hodnoty atribútov trénovacej množiny poskytujú dostatočne veľa informácii na vybudovanie rozhodovacieho stromu. Hovoríme, že atribúty sú *adekvátne* vzhľadom k úlohe postavenia rozhodovacieho stromu, ak sa v trénovacej množine nenachádzajú žiadne dva objekty, ktoré by mali pri všetkých atribútoch rovnaké hodnoty, ale patrili do rôznej triedy [10]. Keď táto skutočnosť neplatí, v trénovacej množine môžeme nájsť objekty, ktoré nie je možné na základe informácie poskytnutej v atribútoch rozlíšiť. Riešením tejto situácie je použitie algoritmu učenia rozhodovacieho stromu, ktorý ošetruje túto situáciu, ako napríklad algoritmus ID3 opísaný nižšie.

Ak sú atribúty adekvátne, rozhodovací strom je vždy možné postaviť. Rozhodovacích stromov, ktoré správne klasifikujú danú trénovaciu množinu, je zvyčajne veľký počet. Našou snahou je však nájsť korektný rozhodovací strom, ktorý správne klasifikuje nielen objekty patriace do trénovacej množiny, ale aj akékoľvek iné, neznáme objekty. Optimálny rozhodovací strom so stopercentnou úspešnosťou

je veľmi ťažko možné vytvoriť – problém zostrojenia binárneho rozhodovacieho stromu je NP-úplný, ako je dokázané v [5]. Podmienkou aspoň čiastočnej úspešnosti je zachytenie v klasifikačných pravidlách dôležité súvislosti medzi hodnotami atribútov a priradením do triedy. Túto úspešnosť môžeme ovplyvniť vhodným výberom učiaceho algoritmu.

klient	príjem	konto	pohlavie	nezamestnaný	**úver**
k1	vysoký	vysoké	žena	nie	áno
k2	vysoký	vysoké	muž	nie	áno
k3	nízký	nízké	muž	nie	nie
k4	nízký	vysoké	žena	áno	áno
k5	nízký	vysoké	muž	áno	áno
k6	nízký	nízké	žena	áno	nie
k7	vysoký	nízké	muž	nie	áno
k8	vysoký	nízké	žena	áno	áno
k9	nízký	střední	muž	áno	nie
k10	vysoký	střední	žena	nie	áno
k11	nízký	střední	žena	áno	nie
k12	nízký	střední	muž	nie	áno

Tabuľka 1. Príklad dát pre tvorbu stromu, prvé štyri údaje sú atribúty, piaty je zaradenie do triedy.

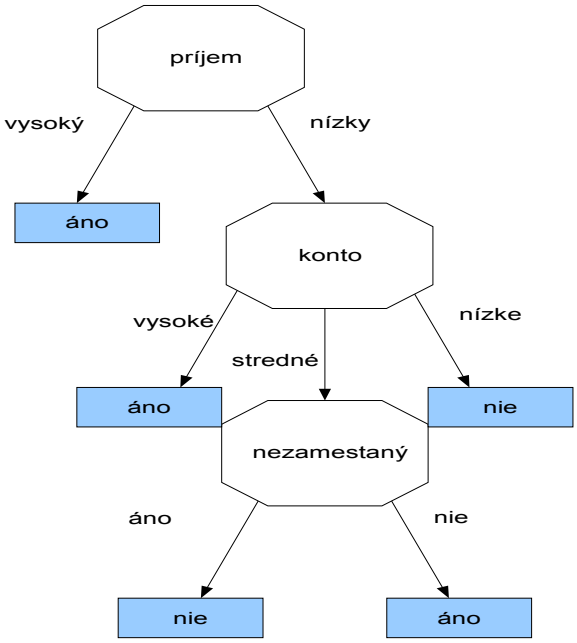


Diagram 1. Príklad rozhodovacieho stromu, ktorý správne klasifikuje trénovaciu množinu.

3.1.4 Algoritmus učenia rozhodovacieho stromu

Existuje viacero možností, ako je možné vybudovať z trénovacej sady rozhodovací strom, ktorý správne klasifikuje objekty z tejto sady. Najtriviálnejší prístup je vygenerovať všetky stromy, ktoré správne klasifikujú trénovaciu množinu a vybrať ten najjednoduchší z nich. Na prvý pohľad sú jasné nevýhody tohto prístupu – korektných stromov je už pri priemernej veľkej trénovacej množine obrovský počet a nemáme istotu, že takto vytvorený rozhodovací strom dáva najpresnejšie výsledky pri klasifikácií neznámych objektov. Tento prístup je vhodný len pre malé trénovacie množiny a objekty s málo atribútmi.

Existuje viacero algoritmov, ktoré v praxi nájdu dobrý rozhodovací strom, nie nutne optimálny, medzi ne patria algoritmy CLS, C4.5, ID3 a iné.

V našej aplikácií bol na postavenie rozhodovacieho stromu použitý mierne upravený algoritmus **ID3** (**Iterative Dichotomiser 3**), ktorý bol publikovaný Rossom Quinlanom [10].

Algoritmus ID3 bol navrhnutý práve pre budovanie rozhodovacích stromov z veľkej trénovacej množiny, obsahujúcej objekty s mnohými atribútmi. ID3 algoritmus stavia rozhodovací strom zhora nadol, použitím metódy rozdeľuj a panuj. ID3 algoritmus preferuje stromy s menšou hĺbkou pred stromami s veľkou hĺbkou a veľkým počtom testov na základe princípu Occamovej britvy, ktorú môžeme interpretovať ako:

„Pokiaľ pre nejaký jav existuje viacero vysvetlení, je lepšie uprednostňovať to najmenej komplikované.“

V nasledujúcom texte bude uvedený algoritmus na postavenie rozhodovacieho stromu, ktorý klasifikuje do dvoch tried, nie je ťažké rozšíriť tento algoritmus na viacero tried.

Označme C neprázdnu množinu objektov. Ak C obsahuje objekty jednej triedy, najjednoduchší korektní rozhodovací strom bude tvorený len jediným listom, ktorý priradí danú triedu objektu. Inak algoritmus najprv nájde vhodný atribút, potom rozdelí trénovaciu množinu na podmnožiny na základe hodnôt daného atribútu a tento postup rekurzívne opakuje na týchto podmnožinách, až kým všetky objekty v podmnožine patria len do jednej triedy. Tieto podmnožiny budú nakoniec tvoriť listy rozhodovacieho stromu, to znamená, že budú predstavovať rozhodnutie, ktoré priradí danú triedu k objektu.

Nie je ťažké ukázať, že v každom kroku je možné nájsť vhodný test. Keďže atribúty sú adekvátne, vždy je možné vybrať atribút, v ktorom sa dva objekty líšia hodnotou. Toto v najhoršom prípade vedie k tvorbe listov obsahujúcich len jednoprvkové podmnožiny, stále však spĺňajúce pravidlo jedna trieda v každom liste.

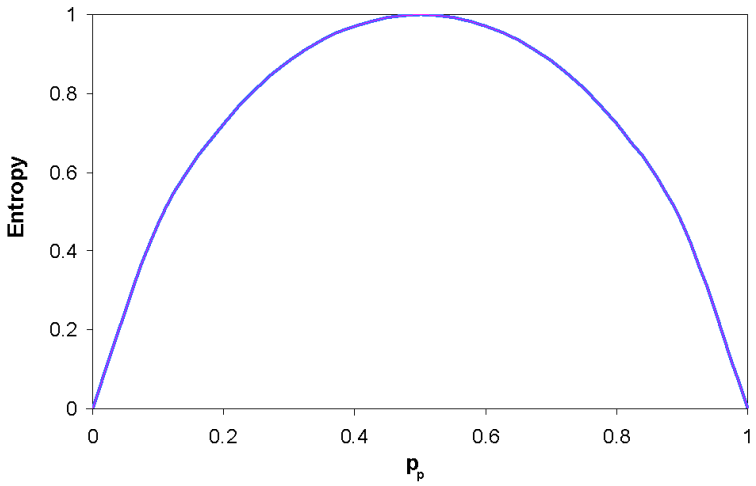
Kľúčový problém v každom kroku spočíva vo výbere správneho testovacieho atribútu. ID3 algoritmus používa metódu založenú na získavaní informácie, ktorá vychádza z dvoch predpokladov [10]:

„Nech C obsahuje p objektov triedy P a n objektov triedy N.

1. *Každý korektný rozhodovací strom postavený na množine C zatriedi objekty v rovnakom pomere ako je ich zastúpenie v C. Ľubovoľný objekt bude patriť s pravdepodobnosťou p/(p+n) do triedy P a s pravdepodobnosťou n/(p+n) do triedy N.*
2. *Pri zatrieďovaní objektu rozhodovací strom vráti triedu. Na rozhodovací strom sa teda môžeme pozerať ako na zdroj správ „P“ a „N“, s očakávanou informáciou potrebnou na generovanie tejto správy:*

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}\ \text{“}$$

Funkciu I(p,n) nazývame entropia, udáva homogenitu množiny informácií. Čím vyššia je entropia, tým väčšie je „znečistenie“ množiny. Z uvedeného vzorca je zrejmé, že entropia množiny, v ktorej sa nachádzajú objekty rovnakej triedy je 0.



Obrázok 2. Graf entropie v závislosti na pravdepodobnosti pozitívnej triedy.

Ak atribút A má hodnoty $\{a_1, a_2, a_3, a_4,\}$, množina C bude rozdelená na 4 podmnožiny C_1, C_2, C_3, C_4, kde každá podmnožina C_i obsahuje objekty z C, ktoré majú hodnotu atribútu A a_i. Označme p_i objekty triedy P ležiace v C_i, podobne n_i objekty triedy N. Očakávaná informácia vyžadovaná z podstromu C_i je $I(p_i,n_i)$.

Očakávaná informácia vyžadovaná zo stromu s koreňom A je vážený priemer

$$E(A)=\sum_{i=1}^{v}\frac{p_i+n_i}{p+n}I(p_i,n_i)$$

kde váha i-tej vetvy je pomer počtu objektov C_i k počtu objektov C.

Mieru získanej informácie (information gain, IG) určíme výpočtom, o koľko sa znížila miera entropie po rozdelení množiny C na základe atribútu A:

$$IG(A)=I(p,n)-E(A)$$

Algoritmus počíta mieru získanej informácie pre všetky atribúty a do koreňa stromu vyberie ten, ktorý ten, ktorý má najväčší prínos – najväčšiu hodnotu IG. Algoritmus ošetruje aj prípad, ak atribúty sú neadekvátne, alebo pri tvorbe uzlu sa nevyskytuje v trénovacej množine nejaká hodnota atribútu.

Samotný algoritmus zapísaný v pseudokóde prebieha nasledovne:

```
function ID3 (train_set, attr_set, clazz_set) returns node
let T new node;
if entropy(train_set)=0 then
       c=classType(firstItem(train_set));
       class_type(T)=c;
       return T;
end if
```

```
//ošetrenie prípadu, ak atribúty sú neadekvátne
if atrr_set is empty then
       c=findMostProbableClassType(train_set);
       class_type(T)=c;
       return T;
end if
A = findAttributeWithTheHighestIG(attr_set, train_set);
testAttribute(T) = A;
for each atrr_value a of A;
       Da = findItemsWithAttributeValue(train_set, A, a)
       //ošetrenie prípadu, ak sa chýba nejaký hodnota atribútu
       if Da is empty then
              let Ta new node;
              c=findMostProbableClassType(train_set);
              class_type(Ta )=c;
       else
              Ta = ID3(Da, atrr_set \{A}, class_set)
       end if
       addBranchToTreeWithValue(T,  Ta, a)
end for
return root
```

Podľa [10] zložitosť algoritmu ID3 môžeme zistiť nasledovnou úvahou: V každom uzle rozhodovacieho stromu, ktorý nie je listom, musíme nájsť vhodný atribút A s najväčším informačným ziskom. Pre výpočet miery získanej informácie musíme pre každú hodnotu atribútu nájsť objekty s touto hodnotou a overiť, či patria do pozitívnej, alebo negatívnej triedy. Výpočetná zložitosť pri každom teste je následne O(|C|*|A|), kde |C| je veľkosť trénovacej množiny a |A| je počet zostávajúcich atribútov. Celková zložitosť algoritmu ID3 je potom O(|C|*|A|*n), kde n je počet testov stromu.

3.1.5 Zdôvodnenie použitia

Rozhodovací strom sme požili v našej metóde najmä z dôvodu, že svojou povahou predstavuje vhodné riešenie problému klasifikácie obrázkov – nielenže je tento mechanizmus priamo určený na klasifikáciu, ale podstatné sú jeho indukčné metódy – dokáže odvodiť relevantné pravidlá medzi hodnotami atribútov a príslušnosťou k triede.

Ďalším z dôvodov jeho použitia je jednoduchosť implementácie a dobrá čitateľnosť – na rozdiel od neurónových sietí klasifikačné pravidlá rozhodovacieho

stromu sú zrozumiteľné a ľahko reprezentovateľné. Klasifikačný proces je takisto veľmi rýchly – jedná sa o jednoduchý priechod stromom.

Pri procese stavania rozhodovacieho stromu bol použitý mierne upravený algoritmus ID3, ktorý je vhodný práve pre množiny dát, aké sa vyskytujú v našej aplikácií – veľké počty objektov s viacerými atribútmi. ID3 algoritmus vo výraznej miere redukuje veľkosť stromu, dokáže rozlíšiť podstatné atribúty od nepodstatných a stromy postavené týmto algoritmom majú dostatočne dobrú úspešnosť. Zložitosť tohto algoritmu je vhodná na to, aby mohol byť využitý aj pre rozsiahle úlohy.

3.2 Spracovanie a analýza obrazovej informácie

Spracovaním obrazu nazývame akýkoľvek proces, ktorého vstupom je obrazová informácia a výstupom je buď spracovaný obraz, alebo k nemu vzťahujúca sa sada informácií či atribútov. Pod analýzou obrazovej informácie rozumieme samotné extrahovanie dôležitých charakteristík.

Obidve techniky majú nezastupiteľné miesto v klasifikačnom procese – číselné vyjadrenia extrahovaných charakteristík sú použité ako atribúty klasifikovaných obrázkov. Pri tejto extrakcii boli použité mnohé techniky, bližšie si ich popíšeme v nasledujúcich odstavcoch.

3.2.1 Predspracovanie obrazu

Pre relevantné výsledky analýzy obrazovej informácie je potrebné ešte pred akokoľvek analýzou obrazu zjednotiť určité vlastnosti bitmapového obrázku – konkrétne máme na mysli jeho rozlíšenie, tj. rozmery pixelovej matice.

Zmenšenie veľkosti obrázku prebieha fyzickým obmedzením počtu pixelov. Metódy zmenšovania rozlíšenia sa líšia svojou kvalitou, obecne platí, že čím je presnejší je výsledok, tým je algoritmus časovo náročnejší. My sme použili bilineárnu interpoláciu, ktorá pre každý výstupný bod nájde súradnice vstupného bodu, ktoré môžu byť neceločíselné. Výsledná farba pixelu sa vypočíta z malého počtu najbližších susedných pixelov tohto bodu. V našom prípade je to okolie 2x2 pixelov, hodnota výstupného pixelu sa počíta z aritmetického priemeru týchto štyroch hodnôt.

Pre zjednotenie veľkosti vyberáme jednu konkrétnu hodnotu rozmeru, tj. najdlhšia strana bude mať po zmenšení danú presnú dĺžku. Tato hodnota musí byť dostatočne primeraná – to znamená dosť veľká, aby sa zachovali všetky dôležité obrazové informácie, ale pritom musíme brať ohľad na to, že spracovanie obrázkov s veľkým rozlíšením je omnoho viac náročné na pamäťové aj výpočtové prostriedky. Závislosť medzi potrebnými pamäťovými i časovými prostriedkami a veľkosťou

pixelovej matice môže byť pri istých algoritmoch až kvadratická či kubická, spracovanie zmenšeného obrázku následne trvá rádovo kratšiu dobu, výsledky analýzy pritom zostanú neovplyvnené.

Zmenšenie rozlíšenia zároveň eliminuje výskyt šumu, ktorý môže byť prekážkou pri analýze obrazu. Pri istých technikách analýzy je zjednotenie rozmerov dokonca nevyhnutné – keď extrahované charakteristiky priamo závisia na rozmeroch obrázku – ako napríklad stredná dĺžka úsečiek a pod.

3.2.2 Analýza farebnej informácie

Farba jednotlivých pixelov je základná informácia, ktorú môžeme bezprostredne získať z bitmapového obrázku. Priama práca s touto informáciou je veľmi ťažko možná, môžeme však z tejto informácie extrahovať viacero charakteristík.

3.2.2.1 Kódovanie farebnej informácie

Farebná informácia v rámci obrazu je uložená v jednotlivých pixeloch – ako sme už spomenuli skôr, najčastejšia používaná reprezentácia je pomocou troch hodnôt, jedná pre každú základnú farbu z farebného spektra. Na obrazovke monitora vidíme farbu ako zloženie troch zložiek - červenej (red – R), zelenej (green – G) a modrej (blue – B). Hodnota každej farebnej zložky dosahuje hodnoty z určitého intervalu – môže to byť buď spojitý interval <0, 1>, alebo množina hodnôt {0, 1, … , 255}, či {0, 1, … , 4096} a iné (toto číslo je určené tým, do koľkých bitov kódujeme hodnotu každej zložky. Hodnota 0 znamená, že sa daná zložka vo výslednej farbe nevyskytuje, maximálna hodnota zase znamená, že zložka je vo farbe zastúpená vo svojej maximálnej intenzite.

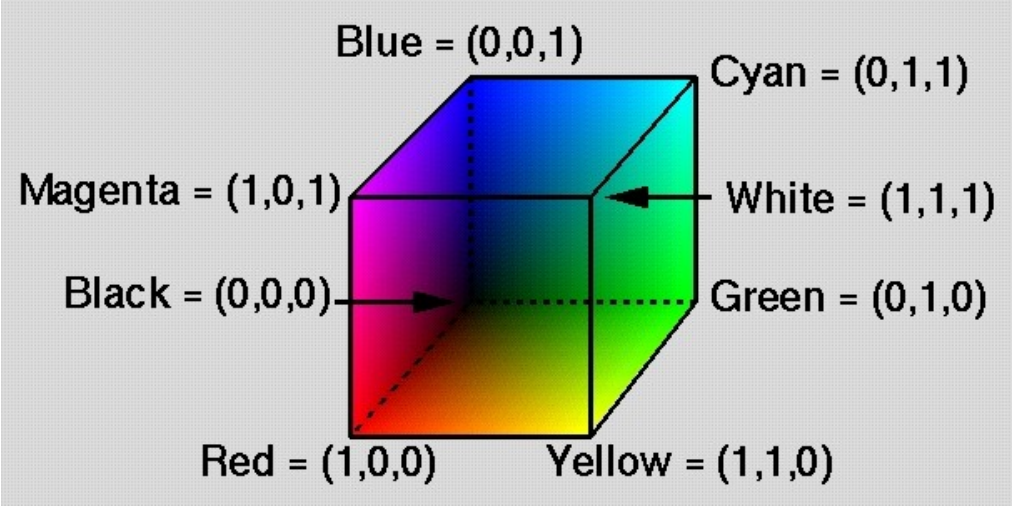
Reprezentácia farby tromi zložkami R, G, B nie je jediná z možností. Používajú sa tiež farebné priestory CMY či HSV - kde základné farebné zložky sú tvorené inými farbami či inými parametrami. Pre náš prípad je však reprezentácia v RGB najvhodnejšia – pretože väčšina bitmapových obrázkov je ukladaná práve v tejto podobe, nie sú nutné žiadne konverzie, farebnú hodnotu je možné zistiť priamo z každého pixelu.

3.2.2.2 Kocka RGB

Pri analýze farebnej informácie je neúnosné pracovať zvlášť s každým odtieňom (už pri reprezentácií 256 hodnôt na jednu zložku farby máme celkový počet farieb $256^3 = 16\ 777\ 216$) . Efektívnejšie obmedziť farebný priestor a pracovať so zástupcami farieb – každému odtieňu priradíme čo najpodobnejší odtieň z pevne určenej palety. Obmedzením farebného priestoru síce dochádza ku strate informácií, ale pre naše potreby sú jeho dôsledky zanedbateľné.

Pri výbere a priraďovaní farebných zástupcov je použitá nasledovná technika. Farebnú škálu priestoru RGB si môžeme predstaviť v priestore ako jednotkovú kocku, umiestnenú v strede súradnicového systému na osách označených r, g, b – r pre červenú, g pre zelenú a b pre modrú farbu. Každému farebnému vektoru zodpovedá jeden bod v kocke.

Túto farebnú kocku rozrežeme v každom súradnicovom smere v pravidelných intervaloch rovnakej dĺžky pevným počtom rezov n, následne sa kocka rozpadne na $(n+1)^3$ menších kociek. Každému odtieňu patriacemu do určitej menšej kocky priradíme ako zástupcu stred tejto kocky. Takto získame paletu o veľkosti $(n+1)^3$, takže namiesto veľkého počtu odtieňov pracujeme len s malým počtom zástupcov.



Obrázok 3. Kocka RGB s vyfarbenými zadnými stenami pre interval <0, 1>.

Pseudokód vytvorenia takéto rozdelenia pre jeden obrázok, kde použitá kocka má na jednej hrane 4 menšie kocky môže vyzerať nasledovne:

```
let c new cube; //vytvoríme kocku s hranou 4
edgeDiv = maxValue/4;
for y = 0 to ImageHeight do
       for x = 0 to ImageWidth do
              red = redValue(x,y);
              green = greenValue(x,y);
              blue = blueValue(x,y);
              i,j,k = 0;
              if red >  edgeDiv*2 then
                     if red >  edgeDiv*3 then i=4
                            else i=3
                     end if
              else
                     if red >  edgeDiv then i=2
                            else i=1
```

```
                    end if
                end if
                … // analogicky pre ostatné farby
                cube(i, j, k)=cube(i, j, k)+1;
            end for
        end for
```

Pre nájdenie správnej pozície v kocke sme použili jednoduchú podobu binárneho delenia, premenná `edgeDiv` reprezentuje, koľko odtieňov leží na hrane menšej kocky, premenná `maxValue` udáva maximálnu hodnotu farebnej zložky.

3.2.2.3 Štatistické rozdelenie farieb

Pri klasifikácií obrázkov sa snažíme nájsť určité farebné charakteristiky obrázkov – snažíme sa zistiť, aké farebné odtiene a v akom počte sa vyskytujú v daných triedach.

Ak skúmame rozloženie odtieňov v rámci jednej triedy, zistíme, farebné spektrum triedy je väčšinou obmedzené - niektoré farby prevažujú a zároveň väčšina odtieňov sa s veľkou pravdepodobnosťou nachádza v pomerne úzkej časti RGB kocky. Pre presné číselné vyjadrenie tejto charakteristiky je nutné štatisticky zanalyzovať veľký počet obrázkov triedy a vyjadriť pravdepodobnosť výskytu každého odtieňu. Platí, že pre odtieň h pravdepodobnosť p_h výskytu v danej triede je rovná

$$p_h = \frac{\sum_{o \in O} f}{n}$$

kde O je množina všetkých obrázkov, f je počet pixelov odtieňu h v obrázku a n celkový počet pixelov všetkých analyzovaných obrázkov.

Práve tu sa osvedčí paleta zástupcov farieb – namiesto toho, aby sme zisťovali pravdepodobnosť každej z milióna hodnôt, pracujeme len s rádovo desiatkami až stovkami.

3.2.2.4 Iné farebné charakteristiky

Významnú informáciu o obraze nesie celkový *počet farieb* použitých v obraze. Táto hodnota vyjadruje absolútny počet všetkých využitých farieb v obrázku v rámci daného farebného spektra. Tento počet je prirodzene obmedzený rozmermi obrázku, pri zmene rozlíšenia nadol sa farebná informácia stráca úmerne pomeru počtu pixelov zmenšeného a pôvodného obrázku.

Bez ohľadu na počet pixelov obrázku môžeme predpokladať, že isté triedy obrázkov, ako kreslené obrázky bez výskytu šumu majú túto hodnotu pomerne nízku, zatiaľ čo renderované obrázky môžu obsahovať vysoký počet odtieňov. Fotografie

majú túto hodnotu pomerne vysokú, ale zase obmedzenú vzhľadom k úzkemu farebnému spektru, ktoré sa bežne vyskytuje v okolitom svete.

Ďalej môžeme skúmať *rozloženie špecifickej farebnej informácie na obraze*, ako príklad si môžeme uviesť fotografie s oblohou, kde sa v hornej časti obrázku nachádza prevažne modrá farba. Toto rozloženie môžeme vyjadriť v percentuálnej podobe, ktorá bude reprezentovať pomer počtu pixelov hľadanej farby v danej časti obrázku oproti celkovému počtu pixelov v oblasti.

3.2.3 Analýza histogramu

Dôležitou charakteristikou obrazu je histogram, ktorý reprezentuje rozloženie farieb obsiahnutých v obraze - je to jednorozmerný vektor, ktorého zložky vyjadrujú absolútne počty farebných hodnôt. Hodnota histogramu H teda pre index *i* hovorí, koľko pixelov má intenzitu *i* [12]. V histograme sa tmavé odtiene nachádzajú blízko nulovej hodnoty, s rastúcou hodnotou rastie aj intenzita. Pretože histogram vyjadruje absolútne počty hodnôt, plynie z toho nasledujúci vzťah [12]:

$$\sum_{i=0}^{max} H(i) = x * y \qquad (*)$$

kde x,y sú rozmery obrázku a max je dĺžka histogramu..

Základným pojmom pri analýze histogramu je *jas*, ktorý technicky odpovedá intenzite svetla. Čím vyššia je intenzita, tým sa javí zdroj jasnejší [12].

Jas predstavuje podstatnú informáciu, pretože je oddelený od farebnej informácie v obraze, kde obrovský počet farebných odtieňov a veľká farebná variabilita v rámci jednej triedy nám znemožňuje extrahovať dôležité charakteristiky. Jasovú hodnotu si môžeme predstaviť ako odtieň šedej – tá ma všetky zložky RGB rovnako veľké. V RGB modeli tak ku každej farbe pripadá práve jeden odtieň šedej, ktorý má rovnaký jas ako daná farba. Pre výpočet jasu nemôžeme použiť jednoduchý priemer zo všetkých farebných zložiek, keďže ľudské oko je na rôzne zložky rôzne citlivé. Hodnota jasu bola určená empiricky [12]:

$$I = 0.299\,R + 0.587\,G + 0.114\,B$$

Tento vzťah je ma rovnakú podobu nezávisle na intervale, v ktorom vyjadrujeme hodnoty farebných zložiek. Ak jasová vychádza neceločíselná, je v prípade potreby zaokrúhlená bežným spôsobom.

Z vyššie uvedeného dôvodu budeme v našej metóde histogram počítať z jasových hodnôt použitím jedného jednorozmerného vektoru. Ak by sme potrebovali počítať histogram zvlášť pre každú zložku RGB, použili by sme histogram zložený z troch jednorozmerných vektorov.

Obrázok 4. Príklad obrázku a grafy jeho histogramov pre jednotlivé farebné zložky, úplne hore červená, v strede zelená, dole modrá.

Histogram je štatistická veličina, pre konkrétnu hodnotu *i* vyjadruje, aká je pravdepodobnosť výskytu pixelu s jasom *i* v obraze. Môžeme z neho získať charakteristiky ak stredná jasová hodnota, či rozptyl – tieto majú dôležitý význam pri výpočte kontrastu, ako bude uvedené neskôr.

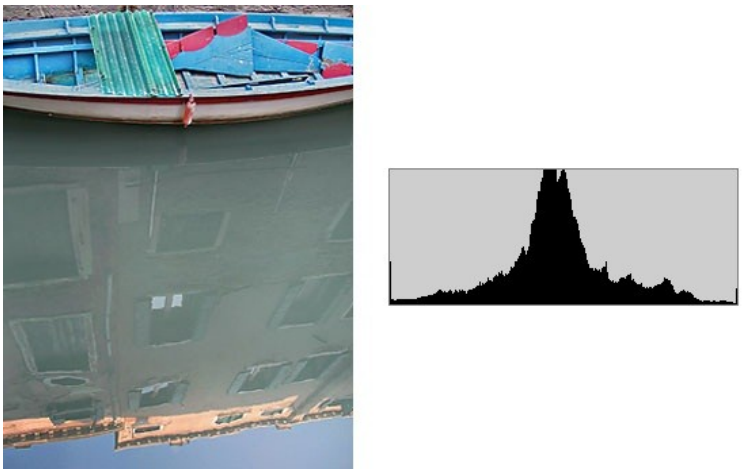
3.2.3.1 Základné histogramové charakteristiky

Histogram nesie informáciu o jasových pomeroch v obraze, nijako však nevypovedá o rozložení tejto informácie na ploche obrázku. Napriek tomu môžeme z histogramu vyčítať dôležité charakteristiky.

Podľa umiestnenia dominujúcich jasových hodnôt môžeme rozlíšiť tri základné druhy obrázkov:

- *tmavé obrázky (low-key)* majú jasové hodnoty s najväčším výskytom blízko nulovej hodnoty
- *svetlé obrázky (high-key)* majú naopak najväčšie jasové hodnoty v hornej časti histogramu
- *strednotónové (mid-key)* obrázky majú väčšinu farieb zastúpenú v okolí stredu histogramu

Obrázok 5. Tmavý obrázok a jeho histogram



Obrázok 6. Strednotónový obrázok a jeho histogram

Obrázok 7. Svetlý obrázok a jeho histogram

Charakterizovať obrázky na základe typu histogramu môžeme nasledujúcim spôsobom: Rozdelíme definičný obor histogramu na tri rovnako veľké pásma, jedno pre tmavé tmavé odtiene označené ako *tiene*, druhé pre *stredné tóny*, poslednú pre svetlé odtiene – *svetlá*. Dôležitú charakteristiku bude následné predstavovať percentuálne zastúpenie týchto pásiem v obrázku.

3.2.3.2 Priebeh histogramu

Dôležitou vlastnosťou každého histogramu je jeho priebeh. Ak sa na histogram pozeráme ako na funkciu početnosti pixelov v závislosti od jasu, môžeme analyzovať jej priebeh a pomocou derivácie nájsť jej extrémy – minimá a maximá – známym postupom: V každom kroku počítame hodnotu diferenciálu d*f(a)(h)* pre veľkosť prírastku *h=1*, ak diferenciál zmení znamienko, našli sme extrém.

Pretože táto metóda nájde všetky lokálne extrémy v histograme, i tie, ktoré nie sú pre jeho priebeh veľmi podstatné, preto je potrebné použiť pri hľadaní

extrémov orezávanie, ktoré vráti len extrémy, ktoré prekračujú určitú prahovú hodnotu. Táto prahová hodnota môže postihovať vertikálny i horizontálny smer.

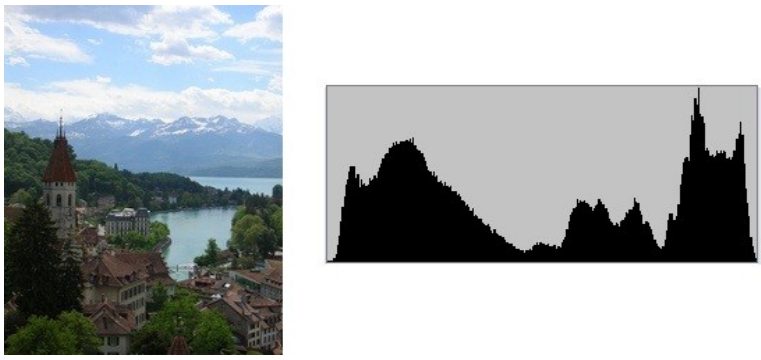
Z priebehu histogramu je pre nás zaujímavý najmä počet a ostrosť jeho kladných extrémov. Väčšina histogramov fotografií je charakteristická tým, ich má len malý počet a veľmi ostré výrazné skoky sú zriedkavé, zatiaľ čo pre obrázky s rozsiahlejšími rovnako farebnými plochami nájdeme v histograme takýchto skokov viacero a väčšinou majú veľmi ostrý priebeh. Ostrosť a veľkosť maxima môžeme reprezentovať pomocou *rozptylu v okolí extrému*. Vysoké a úzke vrcholy majú nízky rozptyl, zatiaľ čo široké a nízke majú rozptyl vysoký. Technicky výpočet môžeme previesť tak, že rozdelíme histogram na viacero pásiem, každé bude ohraničené dvomi minimami, ktoré obklopujú jedno maximum. S každým takto určením pásmom pracujeme ako so samotným rozdelením pravdepodobnosti a počítame pre neho rozptyl. Pritom berieme do úvahy aj veľkosť maxima, vypočítaný rozptyl násobíme pomerom hodnoty maxima a najväčšej hodnoty v histograme. Následným spriemerovaním týchto hodnôt získame vážený priemer rozptylov v okolí maxím, ktorý bude tvoriť ďalšiu dôležitú charakteristiku.

3.2.3.3 Kontrast

Kontrast je ďalšia dôležitá charakteristika, ktorú môžeme vyčítať z histogramu. Veľkosť kontrast je daná mierou rozdielu medzi strednými hodnotami na jednej strane a svetlami a tieňmi na druhej strane [12].

Histogramy, ktorých hodnoty ležia v širokom pásme a obsahujú podstatné množstvo svetiel a tieňov, sú označované ako histogramy s veľkým kontrastom, zatiaľ čo úzke histogramy, ležiace prevažne v strednej časti odrážajú malé množstvo kontrastu.

Kontrast má podstatný vizuálny dopad na obrázok zdôraznením textúry a hrán, ľudské oko subjektívne považuje obrázok s vyšším kontrastom za „krajší“. Pri porovnávaní dvoch histogramov vieme viac-menej určiť, ktorý z nich je kontrastnejší – ten ktorý je „širší“ a má väčšie hodnoty v oblasti svetiel a tieňov, ako v strednom pásme.



Obrázok 8. Fotografia krajinky je typický príkladom obrázku s vysokým kontrastom.

Problém je, ako presne matematicky vyjadriť kontrast ako funkciu histogramu. Existuje viacero definícií kontrastu, niektoré zahŕňajú farbu, iné nie. Podľa článku [9] sa tieto definície líšia od účelu:

- kontrast pre periodicky sa opakujúce vzory je meraný pomocou Michelsonového vzťahu:

$$C = \frac{L_{max} - L_{min}}{L_{max} + L_{min}}$$

Kde L_{min} a L_{max} prestavujú najmenšie resp. najväčšie jasové hodnoty.

- pre meranie kontrastu obrázkov s malými objekty ležiacimi na jednotnom pozadí, kde priemerný jas je približne rovný jasu pozadia sa používa Weberov vzťah:

$$C = \frac{\Delta L}{L}$$

Kde ΔL je prírastok alebo úbytok jasu objektu oproti jasu rovnomerného pozadia L.

- root-mean-square (rms) kontrast sa používa tam, kde potrebujeme porovnávať kontrast dvoch rozdielnych obrázkov – práve preto sme ho využili aj v našej aplikácií. Rms kontrast nezávisí na priestorovom rozložení kontrastu v obrázku, definuje kontrast ako štandardnú odchýlku intenzity pixelov:

$$C = \sqrt{\frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} (I_{ij} - \bar{I})^2}$$

kde N a M sú rozmery obrázku a Ī je stredná hodnota jasu pixelov:

$$\bar{I} = \frac{1}{MN} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} I_{ij}$$

Zo vzťahu (*) dostaneme vzťah

$$\bar{I} = \frac{1}{MN} \sum_{i=0}^{max} H[i] * i$$

pre strednú hodnotu a vzťah

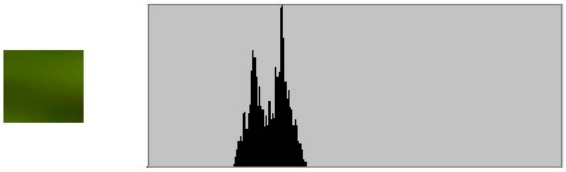
$$C = \sqrt{\frac{1}{MN} \sum_{i=0}^{max} H[i] * i^2}$$

pre kontrast (max je dĺžka histogramu), čo sú štatistické vzorce pre strednú hodnotu a štandardnú odchýlku. Keďže *rms* hodnota nezávisí na priestorom rozložení kontrastu, nemusí vždy dávať odpovedajúce výsledky – vnímanie kontrastu človekom, ale pre použitie v našej metóde je dostatočná.

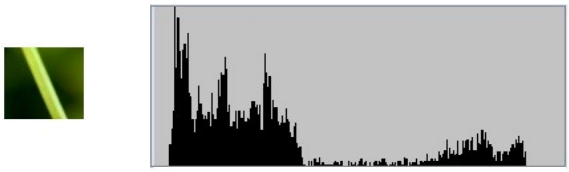
3.2.3.4 Lokálny kontrast

Doposiaľ sme sa zaoberali meraním kontrastu obrázku ako celku. Užitočné pre nás môžu byť však i lokálne hodnoty kontrastu – hodnoty počítané z menších plôch obrázku. Pomocou merania hodnoty lokálneho kontrastu dokážeme odlíšiť na obrázku plochy s uniformnou výplňou, ktoré majú nízky kontrast, od plôch kde sa nachádzajú hrany či výrazná textúra, ktoré majú naopak kontrast vysoký.

Táto charakteristika je podstatná najmä pri obrázkoch, kde sa v popredí nachádza objekt a za ním je rozostrené pozadie. Medzi takéto obrázky patria napríklad fotografie pri snímaní objektov z veľmi malej vzdialenosti – nazývané aj makro fotografie.



Obrázok 9. Výrez pozadia a jeho histogram.



Obrázok 10. Výrez hranového prechodu a jeho histogram

Touto technikou dokážeme rozoznať nielen prítomnosť pozadia na obrázku, ale môžeme ho lokalizovať a spočítať, koľko percent plochy zaberá. Keďže presná hodnota lokálneho kontrastu už nie je natoľko podstatná, môžeme na jeho meranie použiť jednoduchšiu techniku – napríklad šírku histogramového pásma, ktorého hodnoty prekračujú určitú prahovú hodnotu a pod.

3.2.4 Analýza hranovej informácie

Hrany sú dôležité miesta v obraze, kde sa náhle mení hodnota jasu. Miesta na obrázku, ktoré odpovedajú významným hranám, nesú viac informácie ako iné miesta, odpovedajúce napr. pozadiu alebo výplni. Určité triedy obrázkov majú charakteristickú hranovú informáciu – napríklad môžeme predpokladať, že na obrázkoch s budovami sa bude nachádzať viacero dlhých rovných hrán, ktoré bude orientované horizontálnym či vertikálnym smerom. Z tohto dôvodu je pre nás podstatné extrahovať hranovú informáciu a vyjadriť ju v implicitnej podobe – napríklad počiatočnými a koncovými bodmi úsečiek. Pri prechode od jasovej informácie jednotlivých pixelov k implicitnému vyjadreniu hrán boli v našej metóde použité nasledujúce techniky.

3.2.4.1 Detekcia hrán

Detekcia hrán je netriviálna operácia, ktorej úlohou je odflitrovať nepodstatné časti obrázku od dôležitých - vstupom detekcie hrán je pôvodný obrázok, výstupom je prefiltrovaný obraz prevažne v odtieňoch šedej, v ktorom sú jasne črtajú hrany oproti pozadiu. Netrivialita tejto operácie spočíva v tom, že nedá sa jednoznačne určiť, aká veľká musí byť zmena jasu, aby sme mohli o konkrétnom pixele tvrdiť, že tvorí hranu.

V ideálnom prípade je výsledkom detekcie sada prepojených kriviek, ktoré tvoria hrany objektov. V bežných obrázkoch nie je vždy možné získať ideálne prepojené krivky, najčastejšie získame množinu fragmentovaných nesúvislých kriviek, ktoré bývajú narušené šumom a falošnými hranami.

V priebehu histórie bolo predstavených viacero metód tejto detekcie v snahe nájsť čo najoptimálnejší výsledok.

Väčšina týchto detektorov sa zakladá na matematickom popise hrany pomocou *gradientu*. Ako už bolo uvedené vyššie, neformálna definícia hrany popisuje hranu ako miesto v obraze s náhlou zmenou jasu. Túto zmenu môžeme vyjadriť práve v podobe gradientu. Gradient popisuje smer najväčšieho rastu skalárneho poľa, je tvorený veľkosťou a smerom. Smer je určený diferenciálnym operátorom *nabla*

$$\nabla f(x,y)=\left(\frac{\partial f(x,y)}{\partial x},\frac{\partial f(x,y)}{\partial y}\right) \quad [12]$$

kde funkcia f je zobrazenie z R^2 do jasu a predpokladáme, že toto zobrazenie je spojité a diferencovateľné. Veľkosť gradientu je potom veľkosť vektoru

$$|\nabla f(x,y)|=\sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2+\left(\frac{\partial f(x,y)}{\partial y}\right)^2} \quad [12].$$

V skutočnosti však pracujeme s jasovou funkciou I, ktorá je diskrétna, takže veľkosť a smer gradientu v obrázkoch je diskrétna aproximácia „skutočnej“ hodnoty.

Detekcia hrán odfiltruje hranové body, tj. body s veľkým gradientom od nehranových, čím sa dostávame k definícií hrany: hrana je tvorená bodmi na obrázku, ktoré vráti tento algoritmus.

Na odhad gradientu sa používajú viaceré postupy využívajúce analýzu okolia bodu, ktoré sú najčastejšie založené na *konvolúcií*.

Konvolúcia funkcií I(x) a h(x) je podľa [12] definovaná ako

$$I(x)\circ h(x)=\int_{-\infty}^{+\infty} I(x-\alpha)h(\alpha)\,d\alpha \ .$$

Funkciu *h(x)* označujeme *konvolučné jadro*. Konvolučné jadro si môžeme predstaviť ako okno, ktoré prechádza celý obraz a na výpočet novej hodnoty bodu *x* sa sa využíva jeho malé okolie.

Tento vzťah je možné ľahko zobecniť do dvoch rozmerov. Pri práci s bitmapovými obrázkami sa používa diskrétna dvojrozmerná podoba predchádzajúceho vzorca:

$$I'_{i,j} = I_{i,j} \circ h_{i,j} = \sum_{x=-k}^{k} \sum_{y=-k}^{k} I_{i-x,j-y} h_{i,j}$$ [12],

kde *i* a *j* predstavujú súradnice aktuálneho pixelu.

Konvolučné jadro diskrétnej konvolúcie potom bude tvoriť tabuľku o rozmeroch *(2k+1)*(2k+1)*. Rozmery konvolučného jadra majú nepárnu veľkosť, pretože v tom prípade môžeme aktuálne spracovávaný pixel položiť presne do stredu tabuľky a vypočítať súčet podľa vzťahu uvedeného vyššie.

Na detekciu hrán sa používa viacero konvolučných operátorov, ktoré boli zostrojené tak, aby zachytili veľké zmeny gradientu v obraze. Podľa [12] ak

> *„zoberieme ľubovoľnú funkciu a spočítame jej prvú deriváciu, zmeny gradientu pôvodnej funkcie sa prejavia ako lokálne extrémy derivácie. Ak prevedieme druhú deriváciu, extrémy prvej derivácie odpovedajú bodom, kde funkcia prechádza nulovou hodnotou.“*

Na základe toho môžeme operátory rozdeliť do dvoch skupín:

- **gradientové operátory** aproximujúce veľkosť prvej derivácie v obraze hľadaním jej maxima a minima.
- **laplaciánové operátory**, ktoré aproximujú veľkosť druhej derivácie hľadaním jej priechodov nulou.

Každý z týchto operátorov má svoje výhody a nevýhody. Do prvej skupiny patrí napr. *Sobelov operátor* a *Robertsov operátor,* ktoré môžu byť za istých okolností nepresné. Tieto operátory produkujú nerovnomerne široké hrany.

Obrázok 11. Originálny obrázok pred detekciou hrán

Obrázok 12. Výsledok detekcie hrán Sobelovým operátorom

Do druhej patrí *Laplaceov operátor* a *Cannyho operátor*.

Laplaceov operátor vracia stabilné výsledky pri všetkých smeroch hrán, pretože berie do úvahy len veľkosť hrany. Z tohto dôvodu bol použitý v našej metóde. Jeho nevýhoda je väčšia citlivosť na šum, pre naše účely to však postačuje.

Obrázok 13. Výsledok detekcie Laplaceovým operátorom

Laplaceov operátor Δ má viacero tvarov, najčastejšie sa používa konvolučné jadro

$$h=\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$ pre výpočet zo štvorokolia bodu, alebo

$$h=\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$ pre výpočet z osmiokolia, ktorý sme využili aj my v našej metóde.

Tento operátor aproximuje druhú mocninu hodnoty gradientu [12]. Gradient sa odhaduje ako súčet diferencií vo dvoch na seba kolmých smeroch

$$\Delta f(x,y)=\left(\frac{\partial^2 f(x,y)}{\partial^2 x},\frac{\partial^2 f(x,y)}{\partial^2 y}\right).$$

Presné matematické odvodenie jeho konvolučného jadra môžeme nájsť v [6].

Algorimus detekcie hrán v pseudokóde prebieha nasledovne:

```
for y = 0 to ImageHeight do
       for x = 0 to ImageWidth do
              sum = 0
              for i= -k to k do
                     for j = -k to k do
                            sum = sum + h(j,i) * f(x - j, y - i)
                     end for
              end for
              g(x,y) = sum
       end for
end for
```

Laplaceov operátor poskytuje dobré výsledky aj pri jeho vyššej citlivosti na šum.

3.2.4.2 Houghova transformácia

Výsledkom detekcie hrán je obraz, na ktorom sa nachádzajú krivky predstavujúce hrany. Našou snahou je rozložiť tieto krivky na rovné segmenty a vyjadriť ich v implicitnej podobe, čo znamená získať ich koncové súradnice.

Musíme teda vyriešiť problém, ako so súradníc jednotlivých hranových bodov nájsť rovné segmenty. Najtriviálnejším riešením by bolo skúsiť všetky kombinácie bodov, ktoré by mohli tvoriť úsečky. Toto riešenie nie je veľmi efektívne vzhľadom k veľkému množstvu bodov, ktoré môžeme očakávať – jeho časová náročnosť $O(n^{5/2})$, kde n je počet všetkých bodov obrázku. (K uvedenej zložitosti sme prišli nasledovnou úvahou: pre každú dvojicu bodov, ktorých je n^2, musíme otestovať všetky body ležiace na úsečke medzi týmito dvomi bodmi. Vzdialenosť dvoch bodov na obrázku môžeme odhadnúť ako $n^{1/2}$.)

Tento problém optimálnejšie rieši technika nazvaná *Houghova transformácia,* ktorá umožňuje nájdenie parametrických objektov v obraze. Princíp tejto transformácie spočíva v tom, že problém nájdenia kolineárnych bodov prevádza na problém nájdenia zbiehajúcich sa priamok. Táto technika transformuje každý hranových bod na priamku v parametrickom priestore. Presný popis tejto techniky predstavíme v nasledujúcej podkapitole.

3.2.4.2.1 Princíp

Nech (x,y) sú súradnice nejakého hranového bodu. Cez tento bod prechádza nekonečný počet priamok, všetky však môžeme zapísať v smernicovom tvare

$$y = mx + b \ ,$$

kde *m* je *smernica priamky* a *b* je *úsek* vyťatý priamkou na ose y.

Každú z priamok, ktorá prechádza bodom *(x,y)*, môžeme jednoznačne charakterizovať bodom *(m,b)* v smernicovo-úsekovom priestore, pre konkrétne m je hodnota b daná vzťahom

$$b = y - mx \ .$$

Vidíme, že sada hodnôt *(m,b)* korešpondujúcich k priamkam prechádzajúcich cez bod *(x,y)* tvorí priamku v smernicovo-úsekovom priestore.

Ku každému bodu v obrazovom priestore *(x,y)* teda pripadá jedna priamka v priestore *(m,b)* a naopak, ku každému bodu v priestore *(m,b)* prislúcha jedna priamka v priestore *(x,y)*.

Ak v obrazovom priestore leží určitý počet bodov na jednej priamke, ich transformácia v priestore *(m,b)* budú tvorená rovnakým počtom priamok, ktoré sa pretínajú v jednom bode. Súradnice tohto bodu budú tvoriť parametre hľadanej priamky.

Houghova transformácia prevádza body na priamky a na základe počtu zbiehajúcich sa priamok v smernicovo-úsekovom priestore hľadá kandidátov v obrazovom priestore ležiacich na jednej čiare.

3.2.4.2.2 Normálová reprezentácia priamky

Problém smernicovo-úsekovej reprezentácie je, že v prípade priamky rovnobežnej s osou y sú *m* i *b* nekonečné. Riešenie prestavené v [1] navrhuje normálovú reprezentáciu, ktorá definuje priamku vzťahom

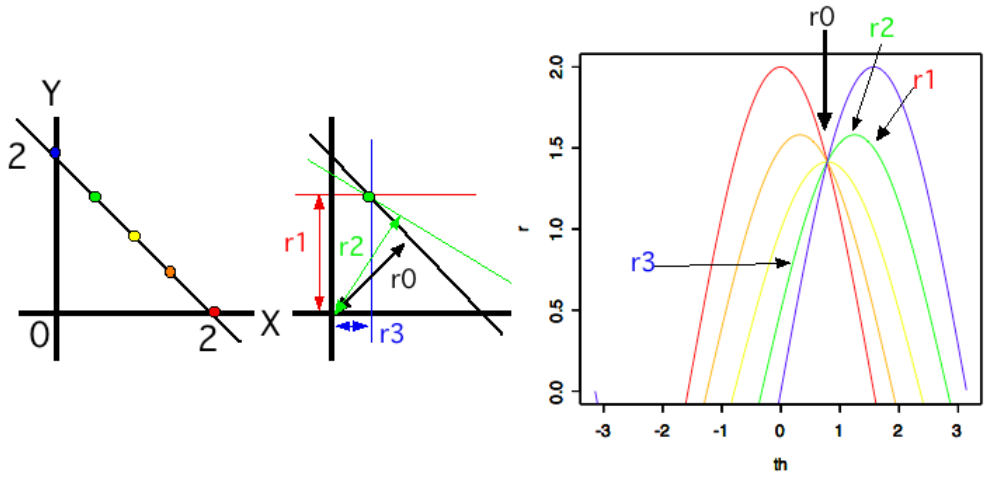
$$x\cos\theta + y\sin\theta = \rho \ .$$

Namiesto neobmedzených parametrov m a b používa parametre ρ a θ, kde ρ predstavuje vzdialenosť medzi priamkou a počiatkom súradnicovej sústavy, θ je uhol medzi vektorom z počiatku súradnicovej sústavy a najbližším bodom priamky. Uhol θ je ohraničený intervalom $<0,\pi)$ a ρ má maximálnu veľkosť rovnú diagonále obrázku.

Následne pre bod (x,y) patriaci do obrazového priestoru, všetky priamky prechádzajúce týmto bodom spĺňajú rovnicu

$$\rho = x\cos(\theta) + y\sin(\theta) \ ,$$

ktorá korešponduje k unikátnej sínusoidnej krivke v priestore (ρ, θ). Jediný rozdiel oproti predchádzajúcej reprezentácií spočíva v hľadaní priesečníkov sinusoidných kriviek namiesto priamok.



Obrázok 14. Znázornenie princípu Houghovej transformácie a detekcie čiar. Na ľavom obrázku sa nachádza päť bodov ležiacich na jednej priamke v obrázkovom priestore. V strede sú priamky prislúchajúce k týmto bodom v priestore *(m,b)*. Napravo priestor *(ρ ,θ)* a sinusoidne krivky pretínajúce sa v jednom bode.

3.2.4.2.3 Algoritmus Houghovej transformácie

Houghova transformácia využíva na zisťovanie počtu priesečníkov v priestore *(ρ,θ)* dvojrozmernú maticu nazvanú *akumulátor*. Pre každý bod *(x,y)* obrazového priestoru vypočíta všetky hodnoty *(ρ,θ)* korešpondujúce k priamkám, ktoré prechádzajú týmto bodom. Tieto hodnoty vykreslia sinusoidnu krivku, pre každý bod tejto krivky zvýšime hodnotu v akumulátore na pozícií určenej súradnicami nášho bodu.

Vyhľadaním lokálnych maxím v akumulátore nájdeme priamky, na ktorých leží v obrazovom priestore väčšie množstvo bodov.

Algoritmus zapísaný v pseudokóde:

```
function HoughTransform(edgePoints, acc) returns param_lines
        for each point (x,y) in edgePoints do
                for theta = 0 to thetaMax do
                        rho = calculateRho(x,y,theta);
                        acc(theta, rho)=acc(theta, rho)+1;
                end for
        end for
        // nájdeme lokálne maximá v akumulátore
        maxima = findLocalMaximaInAcc();
        // zo súradníc každého maxima zistíme parametre priamok
        lines = getLinesFromMaxima(maxima);
return lines
```

kde *acc* prestavuje akumulátor použitý pri výpočte.

Algoritmus nájde parametre priamok, o ktorých vieme, že na nich leží veľký počet bodov. Je teda vysoko pravdepodobné, že týchto priamkách ležia dlhé hranové segmenty. Následne jednoduchým spätným preskúmaním každej nájdenej priamky sme schopní získať ich koncové a počiatočné súradnice.

3.2.4.3 Hranové charakteristiky

Houghova transformácia nám umožnila získať implicitné vyjadrenie hranových segmentov v obraze, z ktorých dokážeme určiť celé spektrum zaujímavých charakteristík.

Medzi charakteristiky, ktoré môžeme priamo získať zo zoznamu hranových segmentov, patrí *počet priamok*, ktoré sa podarilo nájsť Houghovou transformáciou a *počet segmentov*, keďže predpokladáme, že na jednej priamke môže ležať viac nesúvislých úsečiek.

Medzi základné charakteristiky každej úsečky patrí *dĺžka*. Dĺžku úsečky s koncovými súradnicami (x_1,y_1) a (x_2,y_2) vypočítame pod známeho vzorca:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Z dĺžok segmentov môžeme zostaviť histogram a pracovať s ním ako so štatistickou veličinou, medzi užitočné charakteristiky bude určite *stredná dĺžka segmentu* či *rozptyl dĺžok*.

Podobne ako sme histogram jasu obrázku rozdelili na tiene, stred a svetlá, môžeme na základe histogramu dĺžok úsečiek rozdeliť tieto na *krátke, stredné dlhé a dlhé*. Percentuálne zastúpenie každej skupiny bude tvoriť významný znak pre isté triedy obrázkov.

Niektoré obrázky sú charakteristické tým, že je len malá pravdepodobnosť, že sa na nich vyskytuje hranová úsečka s podstatne veľkou dĺžkou. Medzi takéto obrázky patria napríklad fotografie prírody, kde nachádzame prevažne krivky a krátke hrany, takže *dĺžka najdlhšej úsečky* bude ďalšia podstatná charakteristika.

Ďalšou základnou charakteristikou každej úsečky je *uhol*, ktorý zviera s osou x, ak prenesieme jej počiatočný bod do stredu súradnicovej sústavy. Veľkosť tohto uhlu je :

$$\alpha = \arccos\left(\frac{|x_2 - x_1|}{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}\right)$$

Veľkosť uhlu α leži v intervale $<0,\pi)$, pre naše potreby tento spojitý interval nahradíme množinou diskrétnych hodnôt, napr. 0 až 179π /180. Na základe diskretizácie hodnôt uhlov úsečiek môžeme taktiež vytvoriť histogram, kde hodnota *h[i]* pre uhol veľkosti *i* bude udávať, koľko úsečiek má danú veľkosť uhlu, prípadne lepšou charakteristikou je celkový súčet dĺžok úsečiek s daným uhlom. Na základe

tohto histogramu môžeme zistiť štatistické charakteristiky ako stredná hodnota uhlov úsečiek či ich rozptyl.

Niektoré uhly sú pre vnímanie človeka špecifickejšie ako ostatné - jedná sa o uhly 0 a $\pi/2$, podľa ktorých definujeme horizontálne a vertikálne úsečky. Zvýšený výskyt vertikálnych a horizontálnych hrán môžeme nájsť napríklad na fotografiách ľudských výtvorov či budov, zatiaľ čo na fotografiách prírody je rozloženie uhlov hranových úsečiek prevažne rovnomerné. Zaujímať nás bude teda pomer celkového súčtu dĺžok vertikálnych či horizontálnych úsečiek oproti priemeru.

Ďalším špecifikom obrázkov ľudských výtvorov je výskyt pravých uhlov v hranách objektov. Dôležitou charakteristikou teda bude *počet pravých uhlov* v obrázku.

Pre zjednodušenie ich detekcie sa môžeme sústrediť len na horizontálne a vertikálne hrany. Navrhli sme jednoduchú techniku na ich nájdenie:

Do bitovej mapy o rozmeroch obrázku najprv vykreslíme všetky vertikálne úsečky, prípadne len ich koncové body. Následne pri vykresľovaní každej horizontálnej úsečky budeme kontrolovať malé okolie koncových bodov, či sa v blízkosti nenachádza koncový bod vertikálnej úsečky.

Priebeh algoritmus zapísaný v pseudokóde:

```
for vertical in V do
      draw(vertical);
end for
for horizontal in H do
      // skontrolujeme okolie bodu
      if checkNeighbourhood(horizontal.start) then
            rightAngles=rightAngles+1;
      end if
      if checkNeighbourhood(horizontal.end) then
            rightAngles=rightAngles+1;
      end if
end for
```

V prípade, že pravé uhly zisťujeme len z koncových bodov segmentov je zložitosť algoritmu je O(n), kde n je celkový počet horizontálnych i vertikálnych segmentov.

4 Implementácia

Našu metódu na klasifikáciu obrázkov sme implementovali v jazyku Java vo vývojom prostredí Eclipse s dôrazom na efektívnosť vykonávaných techník. Jazyk Java bol zvolený z dôvodu využiteľnosti v ňom vytvorenej aplikácie na každej platforme, dobrej podpory princípov objektovo orientovaného programovania a takisto z dôvodu, že jeho súčasťou sú knižnice, ktoré umožňujú pohodlnú prácu s obrazovou informáciou a vytvorenie užívateľsky príjemného grafického rozhrania.

Implementáciu môžeme rozdeliť na tri hlavné, prepojené časti: do prvej patria techniky na extrakciu charakteristík z obrázku, ďalšiu časť predstavuje mechanizmus umožňujúci klasifikáciu, tj. implementácia rozhodovacieho stromu a poslednú časť tvorí grafické užívateľské prostredie. Dôležitú úlohu v našom programe hrajú takisto testy, ktoré boli použité na kontrolu a overenie mnohých použitých techník. Aplikácia sa spolu so zdrojovými kódmi a obrázkovými dátami použitými pri učení i testovaní nachádza na priloženom kompaktnom disku.

V Kapitole 3 sme popísali techniky a použité algoritmy v našej aplikácií, takisto sme uviedli ich pseudokódy. V nasledujúcej kapitole opíšeme detaily ich implementácie, predstavíme netriviálne problémy, s ktorými sa pri tom stretávame a uvedieme ich riešenie. Na záver kapitoly popíšeme grafické užívateľské rozhranie, ktoré používa našu klasifikačnou metódu, jeho úlohy a súčasti.

4.1 Rozhodovací strom

Pri návrhu implementácie rozhodovacieho stromu – tj. algoritmu učenia rozhodovacieho stromu sme vychádzali z implementácie ID3 algoritmu učenia rozhodovacieho stromu jaDTi [3], ktorá bola zjednodušená a prispôsobená naším potrebám.

Pôvodný ID3 algoritmus predpokladá, že každý atribút má ako obor hodnôt konečnú diskrétnu množinu, v našom prípade však pripúšťame hodnoty atribútov obrázkov tvorené reálnymi číslami.

Z tohto dôvodu musíme pracovať s dvomi druhmi atribútov, jednak *hodnotové atribúty*, ktoré majú diskrétny obor hodnôt, následne *prahové atribúty* pre atribúty s oborom hodnôt R.

Keďže každý atribút v pôvodnom ID3 algoritme mal obor hodnôt o veľkosti 2, musíme v našom prípade výpočet entropie aj informačného zisku hodnotového

atribútu prispôsobiť ľubovoľnej konečnej veľkosti jeho oboru hodnôt. Toto dosiahneme jednoduchým zobecnením vzorca pre entropiu

$$I(\{n_i\}) = \sum_i -\frac{n_i}{n} \log_2 \frac{n_i}{n} ,$$

kde n_i je počet objektov i-tej triedy a n je celkový počet objektov.

Pri prahovom atribúte sa úloha hľadania atribútu s najväčšou mierou získanej informácie rozširuje aj na nájdenie konkrétnej hodnoty patriacej do R nazvanej *prah,* na základe ktorej rozdelíme sadu obrázkov na dve podsady - v prvej budú tie obrázky, ktoré majú hodnotu atribútu menšiu alebo rovnú ako prahová hodnota, v druhej naopak väčšiu.

Pri hľadaní prahovej hodnoty atribútu zachovávame pravidlo o najväčšom informačnom prínose. Vyberieme takú kombináciu atribútu a prahu, aby so všetkých možných kombinácií mal takto vytvorený test najväčší IG.

Existuje viacero spôsobov, ako nájsť najoptimálnejšiu prahovú hodnotu pre atribút. Jedným z týchto spôsobov by bol odhad maxima funkcie IG pomocou binárneho delenia. My sme si však zvolili metódu s vyššou presnosťou na úkor zložitosti, čo však neprestavuje veľkú zátaž pre náš program – predpokladáme, že proces učenia stromu nebude prebiehať veľmi často a pri veľkosti trénovacej množiny rádovo stovky použitie metódy s horšou zložitosťou nemá podstatný dopad. Naša metóda hodnoty atribútu zoradí podľa veľkosti a následne počíta IG pre prah rovný aritmetického priemeru dvoch po sebe idúcich hodnôt, pričom vráti ako výsledok prah s najväčším IG. Zložitosť tejto metódy je $O(|T|*(|N| \log |N|+|N|))$ v rámci každého uzlu, kde |N| je veľkosť trénovacej sady a |T| je počet prahových atribútov.

```
function findBestThreshold(itemSet, attrValues)
	//zoradíme hodnoty atribútov podľa veľkosti
	sort(attrValues);
	bestIG=0;
	bestThreshold=0;
	for i = 2 to length(attrValues) do
		threshold = (attrValues(i-1) + attrValues(i))/2;
		IG = calculateIGForThreshold(threshold, itemSet);
		if (IG > bestIG) then
			bestThreshold=threshold;
			bestIG=IG;
		end if
	end for
return bestThreshold
```

Funkcia `calculateIGForThreshold` počíta informačný zisk z množiny objektov rozdelenej na dve disjunktné podmnožiny na základe danej hodnoty prahu.

Použitie čisto *prahových atribútov* mení podobu rozhodovacieho stromu, z ktorého sa stáva binárny strom (s použitím atribútov s diskrétnym oborom hodnôt mal každý uzol počet potomkov rovný veľkosti svojho definičného oboru). Rovnako pri použití týchto atribútov už neplatí obmedzenie o maximálnej hĺbke stromu – ktorá bola predtým rovná počtu atribútov – prahový atribút môže byť v ceste od listu ku koreňu stromu použitý na testovanie viackrát, samozrejme zakaždým s inou hodnotou prahu.

Aby sme aj v tomto prípade čo najviac zachovali rozumnú hĺbku stromu a obmedzili zbytočnú „rozkošatenosť“, zrušili sme podmienku vytvárania listu stromu v prípade, že entropia trénovacej množiny v liste je 0. Namiesto toho vytvárame list už v prípade, že entropia trénovacej množiny je menšia ako určitá prahová hodnota. Takto vytvorený list bude vracať triedu s najväčším výskytom v množine objektov – triedu, ktorá s najväčšou pravdepodobnosťou prislúcha tomuto listu.

Aj keď všetky atribúty – obrazové charakteristiky sú v našej implementácií prahové, zachovali sme možnosť využiť pôvodné atribúty s diskrétnym oborom hodnôt napríklad pre neskoršie využitie. Dosiahli sme to použitím abstraktného typu atribútu s deklarovanými základnými operáciami, kde vnútorná implementácia týchto operácií zavisí na tom, či sa jedná o prahový alebo hodnotový atribút.

```
if( right_angles < "8.5") {
       if( longSegmentsRatio < "0.00115589884906061 57") {   class: 0
       } else {
              if( vertical_meanRatio < "1.8721979850023986") {
                     if( peaksCount < "3.5") { class: 0
                     } else {
                            if( variance < "38.523057295335875") { class: 0
                            } else { class:     1}
                     }
              } else {
                     if( variance < "129.64905732206398") { class: 0
                     } else {
                            if( variance < "1069.4722464866436") {
                                   if( maxLineSegment < "69.0") {
                                          if( horizontal_meanRatio <
                                                 "1.7258087334353347") {
                                                 class: 0
                                          } else { class:     1}
                                   } else { class:     1 }
                            } else { class:     0}
```

```
                }
            }
        }
} else {
        if( maxLineSegment < "43.620962268742986") { class:  0
        } else {
            if( longSegmentsRatio < "0.0013279407754701053") {
                if( vertical_meanRatio < "0.9601313122529069") {
                    class: 1
                } else {
                    if( horizontal_meanRatio <
                        "1.3917233268850446") { class:   0
                    } else { class:     1}
                }
            } else { class:     1}
        }
}
```

Príklad 1. Ukážka vytvoreného rozhodovacieho stromu pre triedu budovy.

4.2 Štatistické rozdelenie farieb

V Kapitole 3 sme predstavili techniku zisťovania štatistického rozdelenia farieb a zmenšenia farebného priestoru vytvorením zástupcov farieb na základe rozrezania kocky RGB. V nasledujúcej podkapitole uvedieme metódu, ktorú sme navrhli na číselné vyjadrenie zhody rozdelenia farieb obrázku oproti rozdeleniu farieb konkrétnej triedy:

```
function calculateConcordance(imageColorSet, distributionColorSet,
    threshold) returns concordance_value
    concordance = 0;
    for each color in imageColorSet do
        if getDistribution(color, distributionColorSet) >
            thresold
            concordance=concordance+getRelativeArea(color);
        end if
    end for
return concordance
```

Vstupná premenná `imageColorSet` reprezentuje zoznam farieb spolu s ich relatívnym výskytom na obrázku – vyjadruje, koľko percent zaberá daná farba na obrázku. Parameter `DistributionColorSet` prestavuje štatistické rozdelenie farieb pre konkrétnu triedu, takisto vyjadrené v percentách. Parameter `threshold` musí byť vhodne určený pre každú triedu, podľa jeho hodnoty sú oddelené zriedkavé farby od tých, ktoré sa môžu bežne vyskytovať na obrázku danej triedy. Výsledná premenná

`concordance` bude vyjadrovať, koľko percent plochy obrázku je pokrytých farbou, ktorá sa bežne vyskytuje v danej triede.

4.3 Histogram

Pri implementácií extrakcií charakteristík z histogramu bolo potrebné nájsť algoritmus, ktorý z histogramu dokáže získať súradnice dôležitých extrémov, ktoré majú podstatný dopad na priebeh histogramu. Na odhadnutie takýchto extrémov sme navrhli nasledujúcu metódu:

Maximum sa dá označiť ako *podstatné*, ak spĺňa nasledujúce podmienky:

- ak je jeho vzdialenosť od nasledujúceho susedného maxima (ak existuje) väčšia ako určitá minimálna vzdialenosť nazvaná *maximumDistance*
- ak sa jeho hodnota líši od hodnôt oboch obklopujúcich miním (ak existujú) aspoň o hodnotu nazvanú *maximumThreshold*

Z uvedených podmienok je zrejmé, že v istých prípadoch metóda nemusí nájsť všetky významné extrémy. V bežnej praxi sú však tieto prípady málo pravdepodobné - hodnoty *maximumDistance* a *maximumThreshold* boli vhodne určené experimentálnym spôsobom tak, aby boli výsledky algoritmu v naprostej väčšine prípadov v zhode s pozorovaním.

Priebeh algoritmu môžeme zhrnúť nasledovne: Nájdeme všetky lokálne extrémy, následne prevedieme selekciu pre každé maximum podľa daných podmienok:

- ak maximum nespĺňa prvú podmienku a v blízkej vzdialenosti sa nachádza iné maximum, je odstránené menšie z nich.
- ak maximum nespĺňa druhú podmienku, je odstránené.

Zoznam maxím je priebežne upravovaný, aby sa prípadná chyba minimalizovala. Algoritmus zapísaný v pseudokóde:

```
function findMaxima(hist) returns maximum_list
        maxima = findLocalMaxima(hist);
        for i = length(maxima) down to 0 do
                if not checkMaximumDistance(maxima(i)) then
                        removeLowerMaximum(maxima(i));
                else if not checkMaximumThreshold(maxima(i)) then
                        removeMaximum(maxima(i));
                end if
        end for
return maxima
```

Funkcia `checkMaximumDistance` vracia true alebo false podľa toho, či maximum spĺňa prvú podmienku. Funkcia `checkMaximumThreshold` podobne overuje, či maximum spĺňa druhú podmienku.

4.4 Detekcia hranových úsečiek

Dôležitým parametrom pri detekcii hranových úsečiek v priebehu algoritmu Houghovej transformácie je počet diskrétnych hodnôt uhlu θ. Táto hodnota vypovedá o tom, pre koľko hodnôt pri vykresľovaní každej sinusoidnej krivky budeme počítať druhý parameter ρ, podľa vzorca uvedeného v prechádzajúcej kapitole:

$$\rho = x\cos(\theta) + y\sin(\theta)$$

Tento parameter, nazývaný *thetaSteps*, tvorí jeden z rozmerov použitého akumulátora – na jeho hodnote podstatne závisí rýchlosť Houghovej transformácie, takže jeho hodnota musí byť vybraná veľmi uvážlivo. Čím vyššia hodnota, tým je detekcia rovných čiar presnejšia, ale priebeh detekcie trvá citeľne dlhšiu dobu.

Naša implementácia Houghovej transformácie je založená na implementácií [2] s niekoľkými podstatnými rozdielmi. Kým bežný algoritmus nájde parametre priamok, na ktorých ležia rovné hranové segmenty, v našom prípade potrebujeme nájsť počiatočné a koncové body týchto segmentov. Jednou z možností, ako nájsť jednotlivé hranové body ležiace na jednej priamke, je využiť pomocnú trojrozmernú dátovú štruktúru, *bodový akumulátor*, do ktorého pre každý uhol θ a vzdialenosť od počiatku súradníc ρ ukladáme zoznam hranových bodov, ktoré k ním prislúchajú, viď pseudokód nižšie.

Následne prechodom zoznamu bodov jednej priamky zoradeného podľa súradníc nájdeme počiatočné a koncové body segmentov ležiacich na priamke s parametrami θ a ρ.

4.5 Grafické užívateľské rozhranie

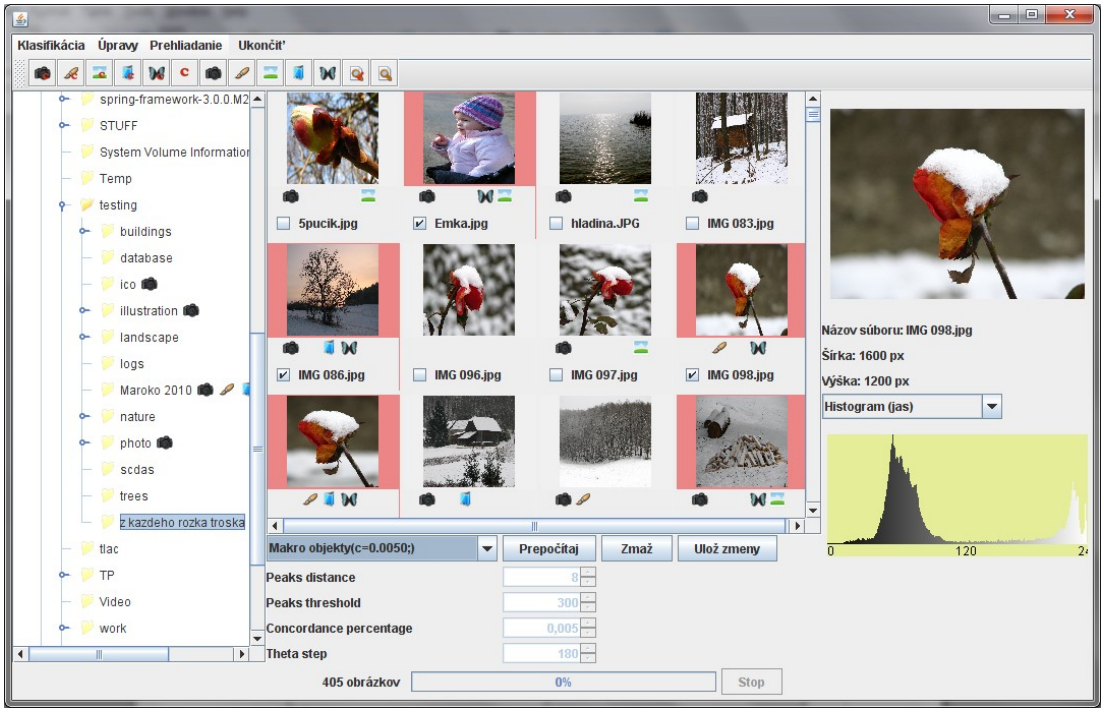
Súčasťou programu je takisto vytvorenie grafického užívateľského rozhrania, ktoré bude využívať klasifikačné mechanizmy vytvorené učiacim algoritmom na trénovacej sade.

Rozhranie umožňuje pohodlne zadávať požiadavky na klasifikáciu použitím konkrétneho klasifikačného mechanizmu, zobrazovať výsledky tejto klasifikácie spolu s náhľadom obrázkov pre ľahké overenie a editovať výsledky klasifikácie v prípade nesprávneho výsledku.

Ďalšou úlohou rozhrania je vykonávanie určitých akcií na základe výsledkov klasifikácie ako vyhľadávanie obrázkov konkrétnej triedy, presúvanie, kopírovanie

obrázkov či už s negatívnym, alebo pozitívnym výsledkom. Rozhranie takisto umožňuje zobraziť základné charakteristiky obrázku ako histogram farebných zložiek a pod.

V procese vytvárania rozhodovacieho stromu boli pri extrakcií charakteristík z trénovacej množiny použité určité pevné parametre, ktorých hodnoty boli určené experimentálnym spôsobom. V samotnom procese klasifikácie sme pomocou grafického rozhrania povolili pozmeniť tieto parametre, čím sa dosiahla schopnosť mierne upraviť výsledky klasifikácie a korigovať rozdiely medzi trénovacou množinou obrázkov a skutočnými dátami.



Obrázok 15. Grafické užívateľské prostredie

4.6 Ďalšie súčasti aplikácie

4.6.1 Vstup a výstup

Každý vytvorený rozhodovací strom je uložený na pevný disk, aby bolo možné jeho opätovné načítanie a použitie bez nutnosti opakovaného behu učiaceho algoritmu. Preto je dôležitou súčasťou aplikácie takisto modul zabezpečujúci načítavanie dát z pevného disku a ich ukladanie. Tento modul zabezpečuje:

- načítavanie obrázkov, vytvorených rozhodovacích stromov a výsledkov štatistického rozdelenia farieb pre jednotlivé triedy z pevného disku
- ukladanie, editáciu a mazanie výsledkov klasifikácie

- presúvanie a kopírovanie obrázkov

Pre implementáciu tejto vstupu a výstupu sme využili štandardné knižnice jazyka Java.

4.6.2 Testy

Dôležitú úlohu pri implementácií nášho mechanizmu hralo testovacie prostredie. V testovacom prostredí sa overovala správnosť fungovania všetkých použitých techník – jednak pri analýze obrazovej informácie, jednak pri procese učenia rozhodovacieho stromu.

Z tohto prostredia boli takisto zadávané požiadavky na vytvorenie rozhodovacích stromov nad konkrétnymi trénovacími obrázkami a na vytvorenie štatistického rozdelenia farieb pre konkrétnu triedu.

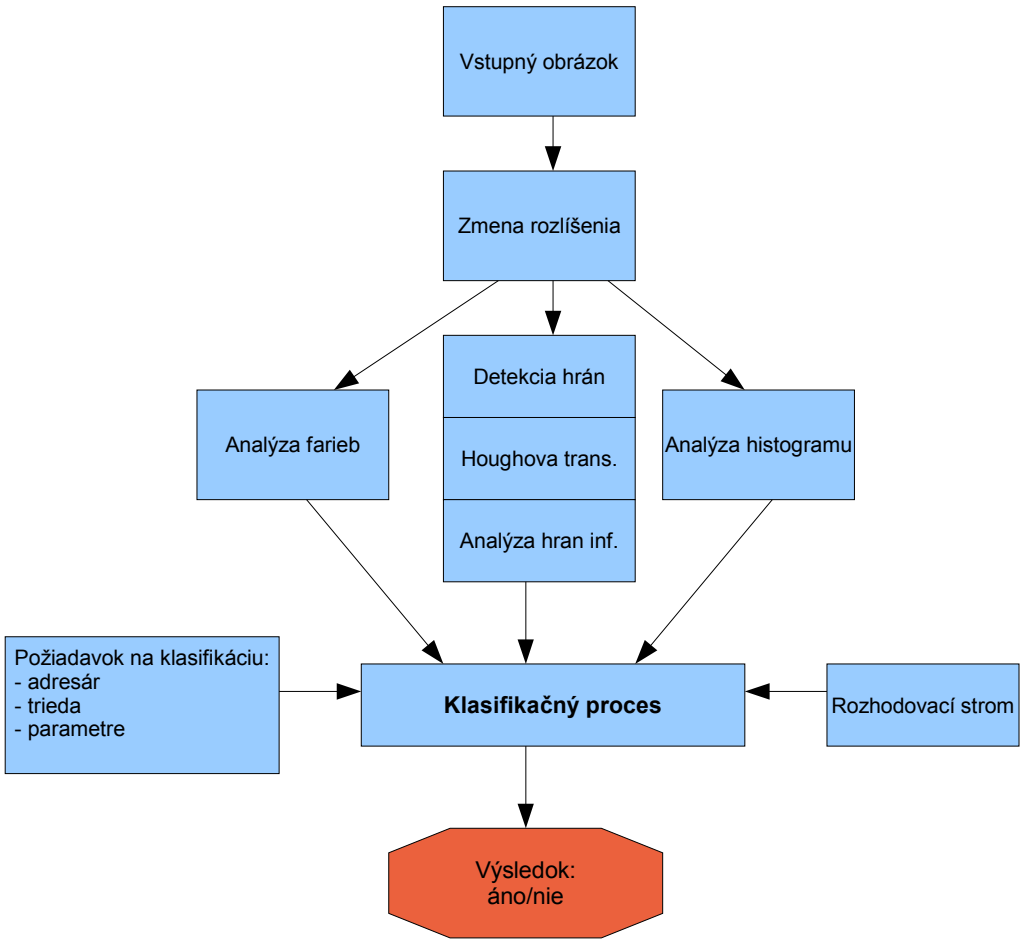


Diagram 2. Znázornenie vstupov a výstupov klasifikačného procesu.

5 Experimentálne výsledky a zhodnotenie

Súčasťou predvedenia našej metódy na klasifikáciu obrázkov je takisto otestovanie jej úspešnosti a zhodnotenie dosiahnutých výsledkov. Úspešnosť nášho algoritmu je daná percentuálnou hodnotou

$$s = \frac{p}{n},$$

kde p je počet správne klasifikovaných obrázkov a n je celkový počet obrázkov, na ktorých prebiehalo testovanie.

Úspešnosť nášho algoritmu môže byť pri niektorých triedach relatívna, pretože priradenie konkrétneho obrázku do triedy je do istej miery subjektívne – napríklad ťažko určiť, čo sa dá ešte považovať za krajinku a čo už nie. Pre čo najobjektívnejšie zhodnotenie výsledkov každú triedu charakterizujeme určitými „typickými“ znakmi. O obrázku môžeme s istotou povedať, že nepatrí do danej triedy, ak sa na ňom žiaden znak nevyskytuje, alebo ak je to z povahy obrázku na „prvý pohľad“ zrejmé. Naopak na typickom reprezentantovi triedy nájdeme všetky tieto znaky.

Neprávne priradenie do triedy nastane, ak klasifikačný mechanizmus priradí obrázok do triedy, o ktorom vieme s istotou, že tam nepatrí alebo ak klasifikačný mechanizmus nezaradí správne typického reprezentanta triedy.

V nasledujúcej kapitole uvedieme popis testovacieho prostredia a následne uvedieme výsledky experimentov pri klasifikácií obrázkov v rámci každej klasifikovanej triedy. Na záver zhodnotíme účinnosť našej metódy.

5.1 Testovacie prostredie

Dáta pre trénovací algoritmus i samotné testovanie úspešnosti klasifikácie boli vybrané z kolekcie obrázkov získaných z Internetu z rôznych zdrojov. Kolekcia celkom obsahovala viac ako 350 najrozmanitejších obrázkov, tieto boli zložené z fotografií, ilustrácií, renderovaných obrázkov a iných.

Pri učení každej triedy boli z tejto sady boli vybraný typický reprezentanti, ktorý utvorili pozitívnu množinu o veľkosti minimálne 40 obrázkov. Následne sme zo sady vybrali obrázky, o ktorých sa dalo s určitosťou povedať, že do danej triedy nepatria, tie utvorili negatívnu množinu o približne rovnakej veľkosti. Minimalizovali sme proces hľadania správnej trénovacej množiny tým, že sme v oboch prípadoch vybrali dostatočne veľkú množinu, ktorá obsahovala čo najširšiu

škálu obrázkov – fotografií z rôznych miest, snímaných pri rôznych podmienkach. Pridanie ďalších obrázkov už nemalo podstatný vplyv na tvorbu rozhodovacieho stromu.

Pri testovaní úspešnosti sme vybrali z kolekcie sadu obrázkov, ktoré boli odlišné od testovacej sady, to znamená prienik trénovacej a testovacej množiny bol Ø. Táto sada bola vybraná náhodne, tak aby sa v nej vyskytoval dostatočný počet reprezentantov triedy. Následne na tejto sade prebehol algoritmus klasifikácie, pri ktorom bola zaznamenaná úspešnosť na základe uvedenom vyššie.

Pri procese učenia i testovania sme použili obrázky z rozličných zdrojov, vytvorené v najrôznejších podmienkach. Z tohto dôvodu sme sa rozhodli overiť účinnosť algoritmu aj v praxi, to znamená na kolekciách obrázkov, aké sa môžu vyskytovať na pevnom disku bežného užívateľa. Získali sme dve sady fotografi od dvoch rôznych zdrojov, pričom každá sada bola zachytená rovnakým digitálnym fotoaparátom.

Prvá sada (A) obsahovala celkom 297 fotografií, ktoré boli získané počas krátkej časovej periódy v podobných podmienkach, druhá (B) celkom 405 fotografií, ktoré boli získané v dlhšej časovej perióde z najrôznejších miest. V obidvoch sadách sa nachádzali fotografie, ktoré mali vysokú obrazovú hodnotu – to znamená ostré s nízkym obsahom šumu, prevažne s dobrou expozíciou, aby sme overili, aké najlepšie výsledky môže naša metóda v reálnych podmienkach dosahovať. Predpokladáme, že pri obrázkoch (fotografiách) s nízkou obsahovou hodnotou táto úspešnosť bude primerane menšia.

Pri testovaní nášho klasifikačného mechanizmu sme využili grafické užívateľské rozhranie, ktoré bolo vyvinuté v rámci tejto metódy. V tomto rozhraní sme zvolili triedu, pre ktorú mala prebiehať klasifikácia a adresár na disku, v ktorom sa nachádzali testovacie obrázky. Aj keď je možné v grafickom rozhraní pozmeniť parametre klasifikácie, testovanie prebiehalo s pôvodnými hodnotami, ktoré boli použité pri učení rozhodovacieho stromu. Užívateľské rozhranie umožňuje zobraziť náhľad obrázku spolu s výsledkom klasifikácie, takže pred testom klasifikácie triedy nie je nutné predspracovať testovaciu sadu a rozdeliť na skupiny podľa priradenia do triedy. Výsledky klasifikácie boli uložené na disk.

5.2 Experimentálne výsledky

Vytvorili sme klasifikačné mechanizmy – rozhodovacie stromy pre nasledovných päť tried: fotografie, kreslené obrázky, budovy, krajinky a makro objekty. Pre každú triedu sme vybrali iné obrazové charakteristiky, ktoré sú viac či menej špecifické, takže úspešnosť klasifikácie je medzi viacerými triedami rozdielna.

Výsledky získané týmito metódami predstavíme v nasledujúcich podkapitolách zvlásť pre každú klasifikovanú triedu.

5.2.1 Trieda fotografie

Ako je zjavné z názvu, do tejto triedy je zaradená prevažná väčšina snímkov, či už získaná priamo digitálnym fotoaparátom, alebo naskenovaním klasickej fotografie.

Pri tejto triede väčšinou nie je problém určiť, či konkrétny obrázok reprezentuje fotografiu alebo nie. Typická fotografia je obrazom okolitého sveta, môže sa na nej nachádzať či už interiér alebo exteriér. Za fotografie sme nepovažovali snímky tlačeného materiálu, kreslených obrázkov, ilustrácií a pod., ktoré sú „technicky“ takisto fotografie.

5.2.1.1 Charakteristiky

Pri klasifikovaní fotografií sa ukázali ako podstatné nasledovné charakteristiky:

- štatistické rozdelenie farieb – väčšina farieb okolitého sveta zachytená na fotografiách sa nachádza v pomerne úzkom spektre farebnej škály. Zaznamenávali sme výskyt odtieňov, ktoré nie sú na fotografiách veľmi bežné, podstatný bol pre nás počet pixelov, ktoré tieto zaberajú.
- počet farebných odtieňov – táto hodnota je pri fotografiách veľmi variabilná, ale pri väčšine snímok nie je príliš nízka alebo vysoká. Extrémne hodnoty naznačujú, že sa nejedná o fotografiu.
- počet maxím histogramu – histogramy fotografií majú obvykle „spojitý“ priebeh bez výrazných skokov, nachádza sa na nich malý počet výrazných extrémov.
- rozptyl v okolí extrémov histogramu – pre fotografie je charakteristický vysoký rozptyl v okolí maxima, veľmi ostré extrémy sú zriedkavé.

Obrázok 16. Príklad štatistického rozdelenia farieb v rámci triedy fotografie, zoradené od najväčšieho výskytu po najmenší. Vidíme, že veľká časť farebnej škály ostala takmer nevyužitá.

5.2.1.2 Výsledky

Trénovacia množina bola zostavená zo 101 fotografií a 54 iných obrázkov, spolu 155 obrázkov. Z tejto sady bol vytvorený rozhodovací strom o hĺbke 7 úrovní, tvorený 15 testami, ktorý za najpodstatnejšie charakteristiky určil počet maxím a zhodu so štatistickým rozdelením farieb.

Testovacia množina sa bola zložená z 17 fotografií a 25 iných obrázkov, spolu 47 obrázkov.

Tento rozhodovací strom dosiahol v testovacej množine presnosť 74,45%, v sade A 81,82 a v sade B 74,07, celkovo 77,17 %.

	Počet obrázkov	Počet reprezentantov	Počet správne klasifikovaných	**Úspešnosť**
Trén. sada	155	101	154	99,35%
Test. sada	47	17	35	74,45%
Sada A	297	297	243	81,82%
Sada B	405	405	300	74,07%
Celkom	749	719	578	77,17%

Tabuľka 2. Výsledky klasifikácie pre triedu fotografie

Počet testov rozhodovacieho stromu	15
Hĺbka rozhodovacieho stromu	7
Pomer počet trén. obr./počet testov	10,33
Atribút v koreni stromu	Počet maxím

Tabuľka 3. Vlastnosti vytvoreného rozhodovacieho stromu

5.2.2 Trieda kreslené obrázky

Do tejto triedy sú zaradené všetky obrázky, ktoré boli vytvorené pomocou počítača a predstavujú najrôznejšie ilustrácie, či už abstraktné, alebo štylizované objekty bežného sveta.

Ich typickým znakom sú väčšie plochy na prvý pohľad rovnakého farebného odtieňa a jasne rozlíšiteľne prechody medzi týmito plochami. Do tejto triedy nepatria renderované obrázky, rôzne diagramy a iné obrázky, u ktorých nie je zrejmé, akým spôsobom boli vytvorené.

5.2.2.1 Charakteristiky

Pri klasifikovaní kreslených obrázkov sa ukázali ako podstatné nasledovné charakteristiky:

- počet maxím histogramu – na histogramoch kreslených obrázkov sa ich obvykle vyskytuje veľký počet, pretože k jednej rovnako farebnej ploche pripadá jedno maximum

- rozptyl v okolí extrémov histogramu – z povahy týchto obrázkov vyplýva, že priemerný rozptyl v okolí extrémov bude veľmi malý.
- počet farieb – táto hodnota tvorí doplňujúcu charakteristiku pri klasifikovaní kreslených obrázkov. Aj keď sa pri kreslených obrázkoch môže zdať, že sa na nich vyskytuje len malý počet farieb, tieto bývajú často znečistené šumom a výsledná hodnota je omnoho väčšia.

Obrázok 17. Príklad typického kresleného obrázku a jeho histogram.

5.2.2.2 Výsledky

Trénovacia množina bola zostavená zo 45 kreslených obrázkov a 59 iných obrázkov, z ktorých prevažovali fotografie, ďalej renderované obrázky, celkovo bolo použitých 104 obrázkov. Z tejto sady bol vytvorený rozhodovací strom o hĺbke 9 úrovní, tvorený 18 testami, ktorý za najpodstatnejšiu charakteristiku určil počet maxím.

Testovacia množina sa bola zložená z 18 kreslených obrázkov a 28 iných obrázkov, spolu 46 obrázkov.

Tento rozhodovací strom dosiahol v testovacej množine presnosť 78,26%, v sade A 84,51% a v sade B 81,73%, celkovo 82,62 %.

	Počet obrázkov	Počet reprezentantov	Počet správne klasifikovaných	**Úspešnosť**
Trén. sada	104	45	104	100,00%
Test. sada	46	18	36	78,26%
Sada A	297	0	251	84,51%
Sada B	405	0	331	81,73%
Celkom	748	18	618	82,62%

Tabuľka 4. Výsledky klasifikácie pre triedu kreslené obrázky

Počet testov rozhodovacieho stromu	19
HÍbka rozhodovacieho stromu	9
Pomer počet trén. obr./počet testov	5,47
Atribút v koreni stromu	Počet maxím

Tabuľka 5. Vlastnosti vytvoreného rozhodovacieho stromu

5.2.3 Trieda budovy

Do tejto triedy boli zaradené fotografie, ilustrácie a iné obrázky reprezentujúce budovy.

Typickým znakom je budova alebo budovy zachytené z exteriéru, ktorá zaberá dominantnú časť obrázku.

Pri obrázkoch, na ktorých budovy zaberajú len malú časť plochy, môže byť problém určiť, či obrázok ešte patrí do triedy budovy alebo nie. V tomto prípade sme sa spoľahli na subjektívny odhad.

5.2.3.1 Charakteristiky

Pri klasifikovaní budov sa ukázali ako podstatné nasledovné charakteristiky:

- počet pravých uhlov
- dĺžka najdlhšieho segmentu
- pomer dlhých úsečiek v obrázku
- pomer dĺžky vertikálnych úsečiek oproti priemernej dĺžke úsečiek vzhľadom k uhlu
- pomer dĺžky horizontálnych úsečiek oproti priemernej dĺžke úsečiek vzhľadom k uhlu

Dôvod výberu týchto charakteristík je zrejmý. Ďalej sme ako doplnkové charakteristiky vybrali:

- počet maxím histogramu
- priemerný rozptyl v okolí maxima

5.2.3.2 Výsledky

Trénovacia množina bola zostavená zo 40 obrázkov budov a 64 iných obrázkov, spolu 104 obrázkov. Z tejto sady bol vytvorený rozhodovací strom o hĺbke 7 úrovní, tvorený 13 testami, ktorý za najpodstatnejšie charakteristiky určil počet pravých uhlov, pomer dlhých úsečiek a dĺžku najdlhšieho segmentu.

Testovacia množina sa bola zložená z 20 obrázkov budov a 24 iných obrázkov, spolu 44 obrázkov.

Tento rozhodovací strom dosiahol v testovacej množine presnosť 81,81%, v sade A 78,11% a v sade B 86,42%, celkovo 82,84% %.

	Počet obrázkov	Počet reprezentantov	Počet správne klasifikovaných	**Úspešnosť**
Trén. sada	104	40	104	100,00%
Test. sada	44	20	36	81,81%
Sada A	297	15	232	78,11%
Sada B	405	18	350	86,42%
Celkom	746	53	618	82,84%

Tabuľka 6. Výsledky klasifikácie pre triedu budovy

Počet testov rozhodovacieho stromu	13
Hĺbka rozhodovacieho stromu	7
Pomer počet trén. obr./počet testov	8
Atribút v koreni stromu	Počet pravých uhlov

Tabuľka 7. Vlastnosti vytvoreného rozhodovacieho stromu

5.2.4 Trieda krajinky

Do tejto triedy boli zaradené prevažne fotografie a iné obrázky, zachytávajúce panorámu krajiny z veľkej diaľky.

Typickým znakom tejto triedy je výrazná línia horizontu, podobne obloha nachádzajúca sa v hornej časti obrázku. Do tejto triedy sú zahrnuté obrázky znázorňujúce z prevažujúcej časti prírodnú krajinu, nie sú sem zaradené panorámy miest a pod.

5.2.4.1 Charakteristiky

Pri klasifikovaní krajiniek sme extrahovali tieto charakteristiky:

- pomer modrých odtieňov v hornej tretine obrázku
- pomer svetiel v hornej časti obrázku
- kontrast obrázku – hodnota kontrastu je pri krajinkách zvyčajne vysoká.

- pomer tieňov – na obrázkoch zachytávajúcich krajiny je malý výskyt tieňov.
- počet maxím histogramu – doplňujúca charakteristika, vyplýva zo špecifického tvaru histogramu tejto triedy.
- rozptyl v okolí extrémov histogramu – doplňujúca charakteristika.

5.2.4.2 Výsledky

Trénovacia množina bola zostavená zo 46 fotografií krajiniek a 44 iných obrázkov, spolu 90 obrázkov. Z tejto sady bol vytvorený rozhodovací strom o hĺbke 7 úrovní, tvorený 12 testami, ktorý za najpodstatnejšie charakteristiky určil pomer modrých odtieňov v hornej časti obrázku, počet maxím a pomer tieňov.

Testovacia množina sa bola zložená z 21 obrázkov krajiniek a 23 iných obrázkov, spolu 44 obrázkov.

Rozhodovací strom dosiahol v testovacej množine presnosť 75,00%, v sade A 84,51% a v sade B 81,73%, celkovo 82,44% %.

	Počet obrázkov	Počet reprezentantov	Počet správne klasifikovaných	**Úspešnosť**
Trén. sada	90	46	89	98,89%
Test. sada	44	21	33	75,00%
Sada A	297	81	251	84,51%
Sada B	405	56	331	81,73%
Celkom	746	158	615	82,44%

Tabuľka 8. Výsledky klasifikácie pre triedu krajinky

Počet testov rozhodovacieho stromu	12
Hĺbka rozhodovacieho stromu	7
Pomer počet trén. obr./počet testov	7,5
Atribút v koreni stromu	Pomer modrej v hor. tret. obrázku

Tabuľka 9. Vlastnosti vytvoreného rozhodovacieho stromu

5.2.5 Trieda makro objekty

Do tejto triedy boli zaradené fotografie, ktoré zachytávajú malé, najčastejšie prírodné objekty z veľmi blízkej vzdialenosti, obvykle menej ako pol metra.

Typickým znakom týchto fotografií je objekt vyskytujúci sa v strednej časti fotografie a za ním je rozostrené pozadie. Objekt je výrazne zväčšení oproti normálnej veľkosti.

Pri určovaní, či daný obrázok je alebo už nie je makro fotografia, sme sa riadili najmä znakom rozostreného pozadia. Objekty odfotografované z malej vzdialenosti, ale bez nezaostreného pozadia sme za makro fotografie nepovažovali.

5.2.5.1 Charakteristiky

Pri klasifikovaní makro objektov sa ukázali ako podstatné nasledovné charakteristiky:

- pomer neostrej plochy, ktorý zaberá v obrázku
- pomer neostrej plochy v okraji obrázku
- pomer svetiel, stredných tónov a tieňov – makro fotografie zvyčajne obsahujú málo tieňov a svetiel.
- kontrast obrázku

5.2.5.2 Výsledky

Trénovacia množina bola zostavená zo 49 fotografií makro objektov a 72 iných obrázkov, spolu 118 obrázkov. Z tejto sady bol vytvorený rozhodovací strom o hĺbke 14 úrovní, tvorený 22 testami, ktorý za najpodstatnejšie charakteristiky určil rozostrenosť okraja a celkovú ostrosť.

Testovacia množina sa bola zložená z 13 makro fotografií a 30 iných obrázkov, spolu 33 obrázkov.

Rozhodovací strom dosiahol v testovacej množine presnosť 74,42%, v sade A 87,20% a v sade B 72,84%, celkovo 78,65 %.

	Počet obrázkov	Počet reprezentantov	Počet správne klasifikovaných	**Úspešnosť**
Trén. sada	118	49	118	100,00%
Test. sada	43	13	32	74,42%
Sada A	297	1	259	87,20%
Sada B	405	57	295	72,84%
Celkom	745	71	586	78,65%

Tabuľka 10. Výsledky klasifikácie pre triedu makro objekty

Počet testov rozhodovacieho stromu	22
HÍbka rozhodovacieho stromu	13
Pomer počet trén. obr./počet testov	5,36
Atribút v koreni stromu	Rozostrenosť okraja

Tabuľka 11. Vlastnosti vytvoreného rozhodovacieho stromu

5.3 Zhodnotenie

Výsledky našej klasifikačnej metódy môžeme zhodnotiť ako dobré. Pre jednotlivé triedy sa pohybujú v rozsahu 75-85 % percent, čo môžeme považovať za prijateľné výsledky na použitie v praxi. Klasifikačná metóda sa ukázala byť stabilná, žiadna trieda neukazuje výraznejšie výkyvy. Pre isté triedy testovacia množina obsahovala príliš málo reprezentantov v pomere k celkovému počtu obrázkov, v tom prípade sa však ukázala schopnosť našej metódy rozoznať, ktoré obrázky nepatria do danej triedy.

Rozhodovací strom sa ukázal ako vhodný klasifikačný mechanizmus i pre takúto komplexnú úlohu. Klasifikačný mechanizmus obstojí i v porovnaní s inými klasifikačnými mechanizmami, ktoré sú založené na iných princípov a využívajú rôzne techniky strojového učenia, ako napríklad klasifikačný mechanizmus obrázkov využívajúcu neurónovú sieť, predstavený v práci [11], ktorý má podobnú úspešnosť.

Experimentálne výsledky ukázali, že neprávna klasifikácia pri istých triedach je spôsobená z veľkej časti podobnou povahou obrázkov z pohľadu charakteristík. Napríklad ak skúmame „ostrosť“ pozadia pri makro objektoch, pri fotografiách s oblohou, ktorá má takisto malú ostrosť, dochádza často k nesprávnym výsledkom. Podobné problémy by sa dali obmedziť špecifickejším výberom charakteristík medzi jednotlivými triedami.

Keďže učiace schopnosti rozhodovacieho stromu ešte nie sú vyčerpané, zároveň je možné nájsť ďalšie špecifickejšie charakteristiky jednotlivých tried, takže zlepšovanie úspešnosti našej klasifikačnej metódy je vysoko pravdepodobné. Rovnako je možné rozšíriť našu metódu o ďalšie klasifikované triedy.

6 Záver

Navrhli sme metódu automatickej klasifikácie bitmapových obrázkov na základe obsahu obrázku. Extrahovali sme užitočné charakteristiky pre každú z tried obrázkov – konkrétne charakteristiky získané analýzou farebnej informácie, jasového histogramu a hranovej informácie. Na základe týchto charakteristík sme vytvorili klasifikačný mechanizmus – rozhodovací strom – pre päť nasledovných tried: fotografie, kreslené obrázky, budovy, krajinky a makro objekty. V rámci práce sme vytvorili taktiež grafické užívateľské rozhranie, ktoré využíva vytvorené klasifikačné mechanizmy, umožňuje pohodlné zadávanie požiadavkov na klasifikáciu, zobrazovanie výsledkov a ďalšie činnosti.

V experimentálnych výsledkoch sme dosiahli úspešnosť pohybujúcu sa medzi 75 - 85% v rámci jednotlivých tried. Výsledky tejto klasifikácie môžem považovať za dostatočne dobré, aby sa náš klasifikačný mechanizmus mohol uplatniť aj v praxi.

Naša práca splnila svoj cieľ – poskytnúť nástroj umožňujúci klasifikáciu bežných sád obrázkov či indexáciu databáz obrázkov na základe obsahu pre rýchle vyhľadávanie. Zároveň naša metóda prestavuje nový pristúp k úlohe rozpoznávania obrazu.

Metóda má vysoký potenciál rozvoja, či už rozšírením klasifikačných mechanizmov na ďalšie triedy, alebo extrahovaním lepších obrazových charakteristík, ktoré môžu zvýšiť presnosť klasifikácie.

Zoznam použitej literatúry

[1] Duda, R. O., Hart, P. E.: Use of the Hough Transformation to Detect Lines and Curves in Picutres, Communications of the ACM, 1972.

[2] Finding Straight Lines with the Hough Tranform, http://vase.essex.ac.uk/software/HoughTransform/, University of Essex, [január 2010].

[3] François, J. M.: jaDTi - Decision Trees: a Java implementation, http://www.run.montefiore.ulg.ac.be/~francois/software/jaDTi/, University of Liège, [január 2010].

[4] How to use trees, http://download-llnw.oracle.com/javase/tutorial/uiswing/components/tree.html, Oracle, [jún 2010].

[5] Hyafil, L., Rivest, R. L. : Contructing optimal binary decision tree is NP-complete. Information Processing Letters, 1976.

[6] Marr, D., Hildreth, E.C.: Theory of edge detection, Proc. Roy. Soc. London., 1980.

[7] Mitchell, T. M.: Machine Learning, McGraw Hill, 1997.

[8] Park, S. B., Lee J. W., Kim, S. K.: Content-based image classification using a neural network, Pattern Recognition Letters, 2004.

[9] Peli, E.: Contrast in Complex Images, Journal of the Optical Society of America, 1990.

[10] Quinlan, J. R.: Induction of Decision Trees. Mach. Learning 1, 1986.

[11] Russel, S., Norvig, P.: Artificial Intelligence – A modern approach, Prentice Hall, 2003.

[12] Žára, J., Beneš, B., Sochor, J., Felkel, P. : Moderní počítačová grafika, 2004.

Prílohy