VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

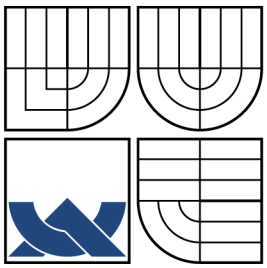
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

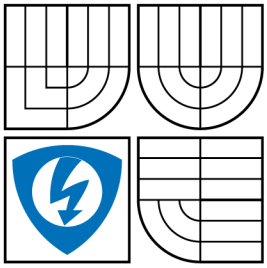
# AUTOMATICKÁ ANOTACE OBRAZU

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. JIŘÍ HEGMON
AUTHOR



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

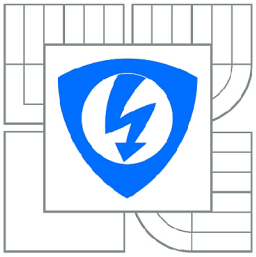
# AUTOMATICKÁ ANOTACE OBRAZU
AUTOMATIC IMAGE ANNOTATION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. JIŘÍ HEGMON
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM BURGET, Ph.D.
SUPERVISOR

BRNO 2013



**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**

**Fakulta elektrotechniky a komunikačních technologií**

**Ústav telekomunikací**

# Diplomová práce

magisterský navazující studijní obor
**Telekomunikační a informační technika**

***Student:*** Bc. Jiří Hegmon ***ID:*** 148268
***Ročník:*** 2 ***Akademický rok:*** 2012/2013

**NÁZEV TÉMATU:**

## Automatická anotace obrazu

**POKYNY PRO VYPRACOVÁNÍ:**

Nastudujte algoritmy, které jsou schopné porovnat obrázky a zjistit jejich míru podobnosti. S pomocí jazyka JAVA implementujte minimálně dva různé algoritmy a zhodnoťte dosažené výsledky. Navrhněte a vytvořte metodu, která se s pomocí podobnosti obrazu pokusí odhadnout obsah v obrazových datech. Zhodnoťte přesnost navrženého řešení.

**DOPORUČENÁ LITERATURA:**

[1] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
[2] Milan Sonka, Vaclav Hlavac, and Roger Boyle. Image Processing: Analysis and Machine Vision. CL-Engineering, 2 edition, September 1998.

***Termín zadání:*** 11.2.2013 ***Termín odevzdání:*** 29.5.2013

***Vedoucí práce:*** Ing. Radim Burget, Ph.D.
***Konzultanti diplomové práce:***

**prof. Ing. Kamil Vrba, CSc.**
*Předseda oborové rady*

**UPOZORNĚNÍ:**

## ABSTRAKT

Rozeznávání a porovnávání obrazu je jedním z hlavních problémů a okruhů oboru počítačového vidění. Tato práce k těmto dvěma problémům připojuje třetí, rozpoznání semantiky, významu obrazu, tzv. anotaci nebo label. Práce využívá znalosti metod rozpoznávání podobnosti obrazů k vytvoření nástroje, který je schopen na základě trénovací množiny obrazů a anotací vytvořit skupinu nejpravděpodobnějších anotací pro danou testovací množinu obrazů. Tato práce představuje několik druhů testovacích množin vhodných pro rozpoznávání anotačních informací u obrazů. Následně je vybrána nejvhodnější množina s potřebnou velikostí trénovací množiny a dostatkem informací v anotacích. Na základě této trénovací množiny je navrhnut algoritmus pro snadné načtení testovací množiny bez velkých nároků na výkon počítače. Vyhodnocení anotačních informací testovací množiny je prováděno na základě různých podobnostních algoritmů. Na počátku této práce byly použity jednoduché, ale nepříliš efektivní metody MSE a porovnání barevných histogramů, postupně bylo ale nutno přejít k použití náročnějších metod (jako je například Tamura, Gabor, CEDD nebo různé druhy hostistogramů). Výsledky tohoto porovnání jsou nakonec brány pro vyhodnocení pravděpodobnosti výskytu dané anotace pro daný obrázek určené testovací množiny. Na závěr práce je provedeno vyhodnocení přesnosti určení anotace na základě informací z použitých trénovacích množin.

## KLÍČOVÁ SLOVA

Obraz, Rozpoznání, Podobnost, Anotace

## ABSTRACT

Recognition and comparison of image is one of the main problems and area of the field of computer vision. This thesis adds to these two issues the third, the recognition image semantics, so called annotations or labels. This work uses the knowledge of methods of recognizing the similarity of images to create a tool that is able based on training dataset of images and annotations, create a group most likely annotation for the test set of images. This work presents several types of test datasets suitable for the detection of annotation information for images. Subsequently, best set with the necessary training dataset size and enough information about annotations is selected. Based on this training dataset algorithm is designed for easy loading test set without large demands on computer performance. Evaluation of annotation information is done based on different similarity algorithms. At the beginning of this work was to use a simple, but not very effective method of MSE and comparison of color histograms, but gradually it was necessary to move to using more advanced methods (such as Tamura, Gabor, CEDD nebo různé druhy hostistogramů). The results of this comparison are then taken to evaluate the likelihood of the annotation for the image specified test set. The last part is an evaluation of the accuracy of annotation based on information from the test set.

## KEYWORDS

Image, Recognition, Similarity, Anotation, Label

HEGMON, Jiří *Automatická anotace obrazu*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 54 s. Vedoucí práce byl Ing. Radim Burget, Ph.D.

# PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Automatická anotace obrazu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ..... .....

(podpis autora)

# PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.


Brno ..... .....

(podpis autora)

# OBSAH

**Úvod** **11**

**1 Popis současných podobnostních metod** **13**
1.1 Aktuální potřeby a implementace měření podobnosti . . . . . 13
1.2 Nejnovější metody měření podobnosti . . . . . 14

**2 Matematické metody a algoritmy využité k měření podobnosti** **16**
2.1 Pixelové indexy . . . . . 16
2.2 Binární indexy . . . . . 17
2.3 Metody pro porovnání podobnosti knihovny LIRE . . . . . 19
2.3.1 Podobnostní metrika AutoColorCorrelogram . . . . . 19
2.3.2 Popisovač barev a hranových směrů - CEDD (Color and Edge Direcivity Descriptor) . . . . . 20
2.3.3 Histogram rozmazaných barev a textur - FCTH (Fuzzy Color and Textur Histogram) . . . . . 20
2.3.4 Gaborova podobnostní metrika - Gabor . . . . . 21
2.3.5 Tamurova podobnostní metrika - Tamura . . . . . 21
2.3.6 Sjednocený kompozitní popisovač - JCD (Joint Composite Descriptor) . . . . . 21
2.3.7 Ostatní podobnostní metriky - JpegCoefficientHistogram, GeneralColorLayout, HSVColorHistogram, SimpleColorHistogram 21
2.4 Informační zisk - Information Gain . . . . . 21

**3 Teorie realizovaných algoritmů pro označení obrázků anotací na základě podobnosti obrazu** **23**
3.1 Vytvoření trénovací množiny . . . . . 24
3.2 Určení histogramu obrazu . . . . . 25
3.3 Určení vlastností obrazu za pomocí knihovny LIRE . . . . . 27
3.4 Vytvoření náhodného stromu . . . . . 28
3.4.1 Náhodný strom bez vazby na informaci v obraze . . . . . 28
3.4.2 Binární strom vytvořený na základě informačního zisku . . . . 28
3.5 Porovnání podobnosti a vyhodnocení pravdivých anotací . . . . . 30

**4 Realizace algoritmů pro označování obrazků anotací** **32**
4.1 Popis konfigurace aplikace . . . . . 32
4.2 Trénovací a testovací DB . . . . . 33
4.3 Trénovací množina jako hashovací tabulka . . . . . 35
4.4 Trénovací množina jako náhodný strom . . . . . 38

4.5 Testovací množina obrázků . . . . . 39
4.6 Vyhodnocení podobnosti, podobnostní test, zápis výsledků . . . . . 40
4.7 Popis struktury a použití aplikace . . . . . 42
4.7.1 Souborová struktura . . . . . 42
4.7.2 Použití aplikace . . . . . 42
4.8 Pomocné výpisy aplikace . . . . . 44

**5 Dosažené výsledky** **45**
5.1 Časová a výpočetní náročnost aplikace . . . . . 45
5.2 Přesnost odhadu anotací . . . . . 46
5.3 Porovnání s podobnými pracemi . . . . . 47

**6 Závěr** **49**

**Literatura** **51**

**Seznam příloh** **54**

# SEZNAM OBRÁZKŮ


1 Přiřazení anotací k obrázku . . . . . 12
2.1 Grafické znázornění histogramů barev . . . . . 17
2.2 Vývojový graf tvorby CEDD . . . . . 20
3.1 Diagram průběhu přiřazení anotace . . . . . 24
3.2 Vyvážený binární strom . . . . . 26
3.3 Extrémně nevyvážený binární strom . . . . . 26
4.1 Příklad konfiguračního souboru config.xml . . . . . 33
4.2 Graf hustoty počtu anotací nepročištěné databáze ESP Game - omezen na počet 300 anotací . . . . . 36
4.3 Graf hustoty počtu anotací pročištěné databáze ESP Game . . . . . 36
4.4 Graf hustoty počtu anotací nepročištěné databáze SUN 2012 - omezen na počet 300 anotací . . . . . 37
4.5 Graf hustoty počtu anotací pročištěné databáze SUN 2012 . . . . . 37
4.6 Grafické zobrazení náhodného stromu [15] . . . . . 41
5.1 Příklad správného odhadu anotace - Obrázek obsahoval anotace: sky, building. Odhadnuto: wall, window, floor, ceiling, door, sky, ceiling lamp, building, curtain, plant . . . . . 48
5.2 Příklad nesprávného odhadu anotace - Obrázek obsahoval anotace: ceiling, curtains, wall, table occluded, sofa, floor, plants, table. Odhadnuto:sky, building, window, door, trees, tree, person, person crop, grass, ground . . . . . 48

# SEZNAM TABULEK

2.1 Přehled binárních podobnostních indexů s koeficienty A, B, C . . . . 18
2.2 Přehled binárních podobnostních indexů s koeficienty A, B, C, D . . . 19
5.1 Přehled přesnosti rozpoznání anotace na základě prahu . . . . . 46
5.2 Přehled přesnosti rozpoznání anotace porovnávané práce [4] . . . . . 47

# ÚVOD

Počítače nevidí jako lidé. Lidé vnímají objekty okolo sebe hlavně jejich významem, barvou, kontrastem barev, jasem a v návaznosti na vzpomínky a zážitky, které jsou s těmito objekty spjaty. Počítače tuto schopnost nemají, vnímají obrazy jako sérii binárních informací, ve kterých nevidí vzpomínky, události, oblíbenou barvu nebo oblíbeného člověka. Účelem této práce je vytvořit aplikaci, která počítač přiblíží alespoň z části schopnostem člověka. Rozpoznání významu obrazu je poměrně komplikovaná záležitost, vždy musí být založena na trénování a pořízení zkušenosti z předešlých obrazů. Stejně to dělají ale i lidé(přiřazení ilustračně provedeno na obrázku 1).

Rozpoznání významu obrazu se zakládá na dvou problémech. Prvním okruhem je vlastní zařazení již získaných informací z trénovacích dat tak, aby je mohl dále počítač efektivně využívat. Druhý problém spočívá v rozhodnutí, jak podobné obrázky porovnat a zvolit, že jsou obrazově a potažmo významově stejné.

Hlavní přínos této práce spočívá ve vytvoření aplikace pro automatickou anotaci obrázků za pomoci měření podobnosti obrázků hlavně na základě trénovací množiny ESP Game[4] a SUN (Scene UNderstanding) verze 2012 [16] [17]. Trénovací databáze ESP Game a SUN 2012 dohromady obsahují cca 80 tisíc obrázků a přibližně 400 tisíců anotačních informací. Zvolené části těchto databází byly použity i pro zpětné otestování přesnosti anotace. V rámci této práce bylo uděláno automatické načtení trénovací množiny do struktury Random Forest (náhodných rozhodovacích stromů) na základě hodnoty informačního zisku anotačních sad obrazů a vytvoření vhodné struktury pro uložení trénovací i testovací množiny do paměti i na disk počítače. V neposlední řadě byly otestováno a realizováno několik algoritmů pro rozpoznání podobnosti obrazů a následně vyhodnocena jejich úspěšnost. Prvním algoritmem byla metoda porovnání střední kvadratické chyby (MSE) [1], druhou metodou bylo porovnání barevných histogramů [2], následně byla zvolena kombinace podobnostních metod knihovny LIRE[18].

Výše popsaná práce byla realizována ve vývojovém open source prostředí NetBeans v. 7.01. K realizaci samotné aplikace byly v práci použity open-source knihovny ImageJ (ij.jar), Apache Log4j v. 1.2 (log4j.jar), Kryo v. 2.20 (kryo-debug-2.20-all.jar) a již zmíněná knihovna Lire v. 0.9.3 (lire.jar). Celá aplikace byla vyvinuta pod licencí FreeBSD Licence. Dokumentace programu byla generována aplikací Doxygen v. 1.8.2.

Zbytek práce je organizován následovně: V kapitole 1 jsou obecně popsány aktuální používané algoritmy. Kapitola 2 se zaobírá matematickou částí měření podobnosti. Kapitola 3 ozřejmuje teorii vybraných porovnávacích podobnostních algoritmů. V kapitole 4 je popsána samotná programová realizace, problémy ve vývoji


Obr. 1: Přiřazení anotací k obrázku

aplikace, slabé i silné místa celého řešení, vysvětluje druhy použitých trénovacích databází a formátů dat v nich těchto databázích obsažených. Samotnou podkapitolou je vysvětlení nastavení vzniklé aplikace. Kapitola 5 zhodnocuje výsledky práce a navrhuje možná další rozšíření ve vyvinuté aplikaci.

# 1 POPIS SOUČASNÝCH PODOBNOSTNÍCH METOD

Současné metody pro měření podobnosti vycházejí z několika přístupů. Jednotlivé přístupy se liší nejenom náročností, použitím, ale také přesností a efektivitou. Při určování podobnosti hraje velkou roli, jaké obrazy chce vlastně uživatel porovnat, kolik jich je, jestli se mohou lišit pohybem v obraze a určitě také vlastnostmi samotného obrazu, jako je například velikost obrazu nebo jeho barevné schéma.

Prvním přístupem je čistě matematické vyjádření podobnosti, které vychází většinou přímo z barevného modelu obrazu. Tyto metody mohou být jednoduše implementovatelné, ale nejsou si v podstatě schopny poradit s různými změnami obrazu, jako jsou posuny, obrazové nerovnosti, rozmazání, nebo změny velikostí a rozměrů testovaných obrazů. Dále absolutně nepracují se samotným významem obrazu, ale pouze s barevnými daty v obraze. Mezi tyto metody patří například vyjádření střední kvadratické chyby (MSE - mean square error) trénovacího a porovnávaného obrázku. Dalším hlediskem těchto metod je poměrně vysoká algoritmická složitost těchto metod.

Za druhé je nutno zmínit algoritmy využívající binární zápis obrázků, jako je například vyjádření Jaccardova indexu. Tyto algoritmy využívají převodu obrazu do bitové podoby a porovnávají obrazy na základě bitové podobnosti nebo na základě ověřeného vzorce porovnají výskyty jednotlivých bitů.

Dalším možným přístupem může být využití informačních indexů, které využívají metod pravděpodobnosti a entropie, jako je například index informačního zisku (InformationGain Index), popřípadě GiNi index. Tyto hodnoty nevyhodnocují samotný obsah obrázku, ale zabývají se změnovostí a informační hodnotou v obraze. Mohou být ale také navázány přímo na anotační informaci obrázků.

## 1.1 Aktuální potřeby a implementace měření podobnosti

Přiřazování anotace obrazů vychází vždy z pevné trénovací množiny, na které si daná aplikace nebo stroj natrénuje, jaké vlastnosti jsou vhodné k danému obrazovému materiálu.

První technickou potřebou může být velké množství nutných operací na zpracování trénovací množiny. Ke správnému vytvoření trénovací množiny je potřeba použít stovky až tisíce obrázků, při použití malého množství informací je většinou porovnání dost nepřesné a zavádějící. Tím pádem je záhodno používat algoritmy,

které neprovádí příliš mnoho matematických operací nad jednotlivými obrázky trénovací množiny (tzn. je nutno použít algoritmy s nízkou algoritmickou složitostí).

Dále mohou být velkou obtíží pro realizaci podobného algoritmu nároky na paměťový prostor počítače. Každý obrázek má řádově stovky kilobytů až megabyty dat, které mohou v rámci trénovací množiny tvořit gigabyty až desítky gigabytů v paměti počítače. Pro použití na osobním počítači se tedy samotné načítání všech obrázků najednou nedá dost dobře realizovat (berme v úvahu stav HW na jaře roku 2013, kdy osobní PC obsahuje okolo 4GB operační paměti [1]). Je tedy nutno v rámci optimalizace využít předpočítaných reprezentací obrazů, jak je například barevný histogram zvolených úrovní barev, nebo matematických transformací. Tímto se samozřejmě značně zvyšuje náročnost vytvoření testovací množiny, ale za cenu obrovského zrychlení v procesu porovnání.

Poslední potřeba, která byla zjištěna při vytváření této práce, je velký důraz na datovou strukturu trénovací množiny. Při prohledávání a přiřazování není možné, aby každý testovací obrázek byl porovnáván vždy se všemi trénovacími obrazy, popřípadě s jejich reprezentacemi. Proto je nutné, aby průchod nad trénovací množinou měl co nejmenší složitost (například logaritmickou). Dále je v této práci poskytnut příklad průchodu při použití čistě sekvenční datové struktury pro reprezentaci trénovací množiny (spojový seznam) a následně porovnání s tou samou realizací přes stromovou strukturu. Při takovéto implementaci samozřejmě dochází k jisté redukci počtu porovnání, ale při vhodně vytvořené trénovací množině by to nemělo mít vliv na výsledek přiřazení anotací.

## 1.2 Nejnovější metody měření podobnosti

Stávající stav anotačních algoritmů využívá prostředky mělé inteligence jako jsou neurální sítě [3], algoritmy pro hledání k-nejbližšího souseda (nearest neighbours), systémy podpůrných vektorů nebo kombinace histogramů. Většina algoritmů využívající neurální sítě je založena na poměrně časově náročné trénovací fázi a následném využití trénovací množiny pro rozpoznání. Naproti tomu algoritmy nejbližších sousedů nevyžadují žádný trénovací čas, ale rozhodnutí podobnosti je založeno na přesné podobě zapamatované historie[5].

V rámci mladších metod a teorií v oblasti podobnosti a zpracování obrazu je nutno se zmínit o algoritmech, které využívají podpůrné vektory, popřípadě strojích podpůrných vektorů (Support Vector Machines - SVN). Tato teorie není z nejmladších - byla uvedena již v devadesátých letech 20. století[7] ,ale je stále rozvíjena

[1] odhad vycházel z autorova subjektivního pozorování a hodnocení prodávaných sestav na internetových obchodech k dubnu 2013

a využívána. Algoritmy využívající SVM jsou v dnešní době na špici, co se týče přesnosti. Novější metody jsou převážně specializované na konkrétní případy. Algoritmus SVN je z teoretického hlediska poměrně jednoduchý, skýtá v sobě poměrně vysokou výkonnost[8] a možnost začlenení do aplikací nejenom v oboru rozpoznání podobnosti, ale i v jiných oborech zpracování obrazu.

Dalším z postupů, které se využívají v klasifikaci obrazu v dnešní době, je metoda využívající jádro protínajících se histogramů (Histogram Intersection Kernel). Tato metoda není příliš složitá, využívá slučování barevných histogramů jednotlivých testovaných obrazů k vyhodnocení podobnosti. Tato metoda je velmi často zmiňována ve spojitosti s SVM[9].

Mezi další metody patří například Color Layout Descriptor (CLD) - velmi rychlá metoda pro zpracování obrazu. Tato metoda byla navržena pro efektivní reprezentaci rozložení barev. Metoda je založená na mřížkové reprezentaci barev a diskrétní kosinově transformaci[13]. V neposlední řadě je nutno zmínit i metodu Fuzzy Color Histogram (FCH). Tento postup využívá podobných struktur jako normální barevný histogram, avšak normální barevný histogram vždy zařadí jeden pixel do jedné histogramové složky, ale FCH rozprostírá každý pixel do více než jedné histogramové složky[14].

# 2 MATEMATICKÉ METODY A ALGORITMY VYUŽITÉ K MĚŘENÍ PODOBNOSTI

K měření podobnosti obrazu se často využívají čistě matematické metody nebo matematické metody, které využívají různé transformace (například DCT-III, DFT atd.). Podobnost obrazu je možné porovnávat i v rámci jiných okruhů, než je jenom okem viditelná podobnost mezi obrazy.

## 2.1 Pixelové indexy

V této části bude uvedeno několik metod, které se používají pro základní měření podobnosti obrazu a využívají rozklad obrazu na pixely [1]. Základní metodou pro porovnání podobnosti je metoda MSE (mean square error).

Metoda MSE je založena na porovnání dvou numerických hodnot z obrazu, nejčastěji se pro vyhodnocení používají hodnoty pixelů v barevném modelu RGB. Pro tuto metodu se dá samozřejmě využít i jiný barevný model, například CMYK. RGB je ovšem nejčastější. Základní vzorec pro výpočet MSE je tento:

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [x(m,n) - x'(m,n)]^2 \tag{2.1}$$

Kde M značí počet porovnávaných pixelů ve výšce obrázku a N značí počet porovnávaných pixelů ve šířce obrázku, x(m,n) označuje hodnotu pixelu na pozici m a n. Nevýhodou této metody je to, že pro model RGB se musí metoda MSE provádět v podstatě 3krát - jedenkrát pro každou barvu daného modelu - a dále používání poměrně velkého množství operací v rámci samotné metody. Právě pro tyto nevýhody byla uvedená metoda odzkoušena v aplikaci, ale nakonec nepoužita.

Další podobnou metodou je metoda MAE (mean absolute error). Tato metoda je zjednodušením výše uvedené MSE o kvadratickou funkci. Základní vzorec je tento:

$$MAE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x(m,n) - x'(m,n)| \tag{2.2}$$

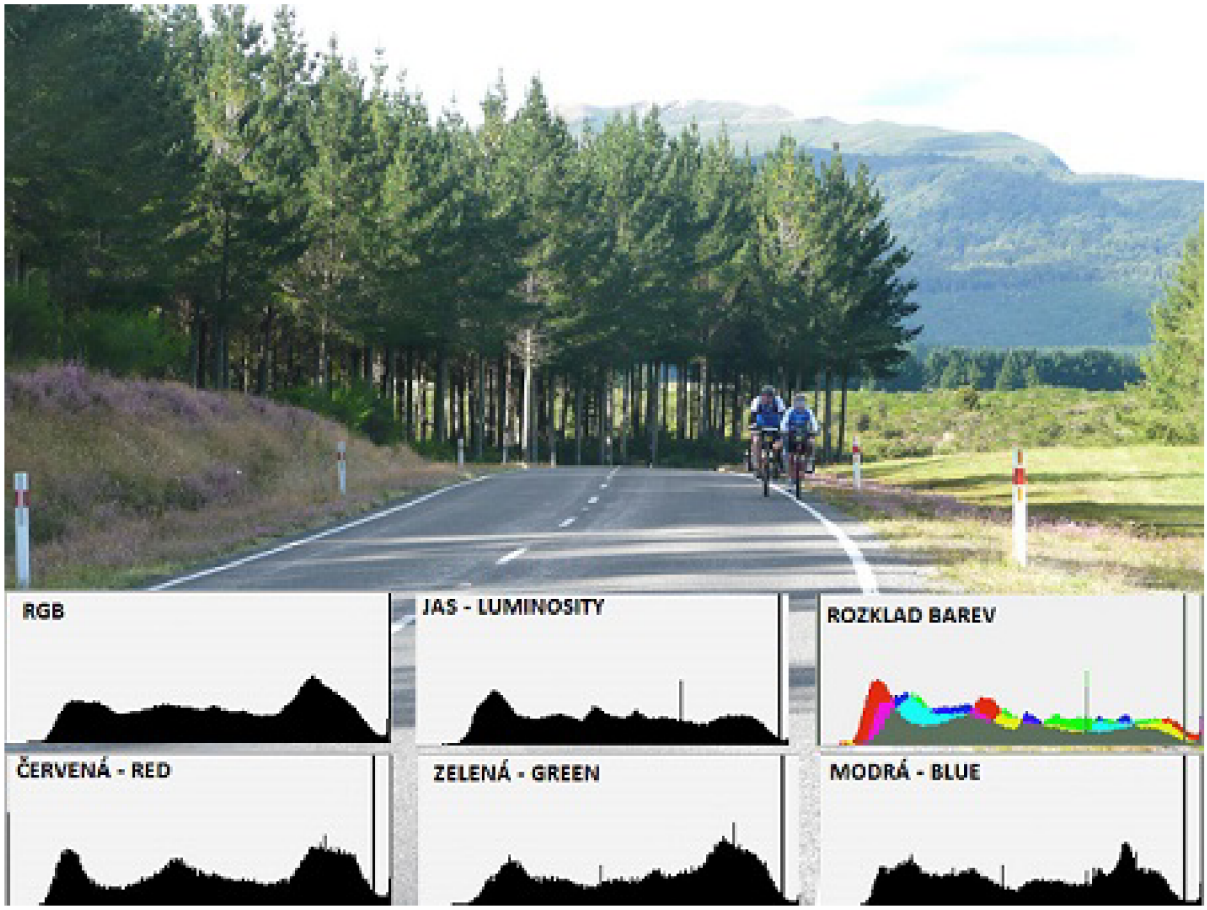
Kde opět M značí počet porovnávaných pixelů ve výšce obrázku a N značí počet porovnávaných pixelů v šířce obrázku, x(m,n) označuje hodnotu pixelu na pozici m a n. Tyto dvě metody jsou si velmi podobné, ale sdílejí také stejné nevýhody. Použití těchto metod pro rozpoznávání velkého množství obrazů je tedy značně náročné na výkon CPU a vzhledem k nutnosti mít kompletní informace o každém pixelu obrázku, zde vzniká i potřeba držet každý obrázek v paměti počítače.

[1] pixel - PIcture ELement - obrazový bod

Další způsob je metoda barevného histogramu. Barevný histogram je ve své podstatě seznam kombinací intervalů hodnot barev v modelu RGB. Takto vydefinované barevné histogramy jsou výhodné v tom, že jsou schopny pokrýt rozmazané části obrazů popřípadě drobné pohyby. Samotný histogram vychází z jednoduchého matematického vzorce:

$$h_{RGB}(a, b, c) = N.P(R = a, G = b, B = c) \tag{2.3}$$

Kde $h_{RGB}$ značí výsledný histogram, $N$ označuje celkový počet pixelů v rámci obrazu, funkce $P(R = a, G = b, B = c)$ reprezentuje pravděpodobnost výskytu barvy $R$ v intervalu $a$, barvy $G$ v intervalu $b$ a barvy $B$ v intervalu $c$.




Obr. 2.1: Grafické znázornění histogramů barev

## 2.2 Binární indexy

V rámci určování podobnosti jsou obecně známé i jednoduché indexy, kterými je možno daný obraz označit. Tyto indexy se většinou počítají na základě binárního

záznamu daného obrázku (tzn. celý obraz je převeden do bitů a ty se vzájemně porovnají) [6].

Pro dva vybrané obrazy se definují koeficienty A, B, C a D. Koeficient A je definován jako počet stejných bitů s hodnotou 1, B je počet bitů obrázku č. 1 s hodnotou 1, C je počet bitů s hodnotou 1 v obrázku č. 2. Posledním indexem je index D, který je definován jako počet shodných bitů s hodnotou 0. Tyto výpočty počítají samozřejmě s tím, že oba vstupní obrázky budou bitově stejně dlouhé. Pokud nejsou, musí se provést úprava velikost a rozměrů jednoho z nich. Hodnoty těchto binárních indexů nabývají povětšinou hodnot mezi 0 až 1, kdy čím více se hodnota indexu $I$ blíží 1, tím více jsou si obrázky podobné: Na základě koeficientů A, B a C jsou definovány tyto podobnostní indexy:

Tab. 2.1: Přehled binárních podobnostních indexů s koeficienty A, B, C

| Název indexu, datum definice | Vzorec | Rozsah |
|---|---|---|
| Dice index , 1945 | $I = \frac{2.a}{2.a+b+c}$ | 0 až 1 |
| Jaccardův index, 1912 | $I = \frac{a}{a+b+c}$ | 0 až 1 |
| Kulczynksi index č.1, 1928 | $I = \frac{a}{b+c}$ | 0 až inf |
| Kulczynksi index č.2, 1928 | $I = \frac{(\frac{a}{2}).(2.a+b+c)}{(a+b).(a+c)}$ | 0 až 1 |
| Simson index, 1960 | $I = \frac{a}{min(a+b,a+c)}$ | 0 až 1 |
| Ochiai index, 1957 | $I = \frac{a}{((a+b).(a+c))^{\frac{1}{2}}}$ | 0 až 1 |
| McConnaughey index, 1982 | $I = \frac{a^2-b.c}{(a+b).(a+c)}$ | -1 až 1 |
| Braun-Blanquet index, 1932 | $I = \frac{a}{max(a+b,a+c)}$ | 0 až 1 |
| Sokal & Sneath index č.2, 1963 | $I = \frac{a}{a+2.b+2.c}$ | 0 až 1 |

Při doplnění koeficientu D byly definovány tyto binární indexy:

Tab. 2.2: Přehled binárních podobnostních indexů s koeficienty A, B, C, D

| Název indexu, datum definice | Vzorec | Rozsah |
|---|---|---|
| Russell & Rao index , 1940 | $I = \frac{a}{a+b+c+d}$ | 0 až 1 |
| Simple Matching index, 1958 | $I = \frac{a+d}{a+b+c+d}$ | 0 až 1 |
| Yule index, 1900 | $I = \frac{(a.d)-(b.c)}{(a.d)+(b.c)}$ | 0 až 1 |
| Rogers & Tanimoto index, 1960 | $I = \frac{a+d}{a+d+2.(b+c)}$ | 0 až 1 |
| Sokal & Sneath index č.1, 1963 | $I = \frac{2.(a+d)}{2.(a+d)+b+c}$ | 0 až 1 |

Mezi tyto další indexy patří například "komplexní vlnkový podobnostní index"(Complex wavelet structural similarity index - dále jen CWSSIM) [6]. Tento index je založen na výpočtu strukturálního podobnostního indexu (Structural Similarity Index - dále jen SSIM), je jeho rozšířením. Výhodou CWSSIM oproti SSIM je to, že není tak náchylný na změny v obraze, tzn. rotace, pohyby a jiné změny obrazu. Tento index je založen na vlnkové a Fourierově transformaci.

## 2.3 Metody pro porovnání podobnosti knihovny LIRE

Výsledky této práce se velmi opírají o metody pro porovnání podobnosti implementované v open source knihovně Lire. Tato knihovna obsahuje v dnešní době okolo 20 různých metriky, pomocí kterých je možno vyjádřit obraz. Následným porovnáváním těchto metrik je pak možno zjistit podobnost dvou obrazů. Každá z těchto metod využívá trochu jiného přístupu. Vzhledem k poměrně velkému množství těchto metod, je v práci použito pouze deset metod. Jsou to AutoColorCorrelogram, CEDD, FCTH, Gabor, GeneralColorLayout, HSVColorHistogram, JCD, JpegCoefficientHistogram, SimpleColorHistogram, Tamura. Ostatní metody nebyly použity hlavně kvůli nekompatibilitě s knihovnou Kryo, která je v aplikaci zaimplementována.

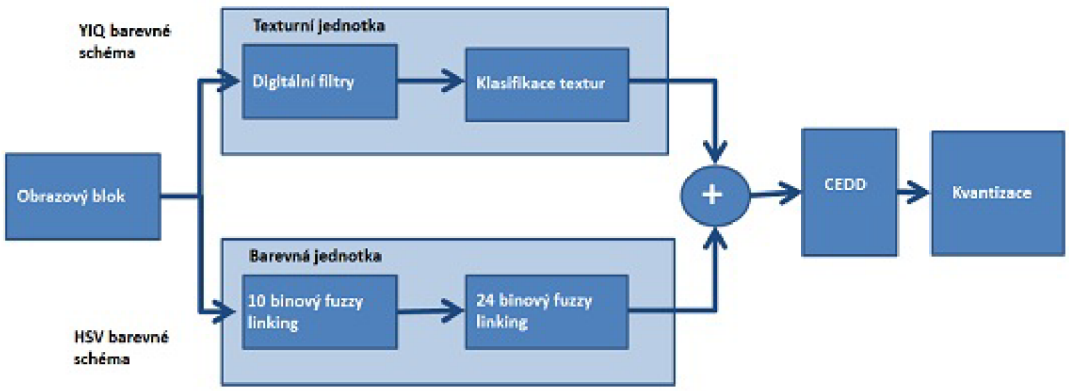
### 2.3.1 Podobnostní metrika AutoColorCorrelogram

Barevný korelogram (dále jen korelogram) je matematickým vyjádřením, jak se mění prostorová korelace barevných párů se vzdáleností. Naproti tomu barevný histogram

(dále jen histogram) zachycuje pouze rozprostření barev v obraze[19]. AutoColor-Correlogram vyjadřuje pro každý pixel $p$ barvy $b$ pravděpodobnost, že pixel $p1$ ve vzdálenosti $k$ má barvu $b$.

### 2.3.2 Popisovač barev a hranových směrů - CEDD (Color and Edge Direcivity Descriptor)

CEDD je blokový postup, který využívá rozkladu obrazu na textury a na barvy. V případě textur využívá digitálního filtrování, následné klasifikace textur obsažených v obraze, v případě barev využívá stejně jako FCTH tzv. fuzzy linking histogramy. V prvním kroku se extrahují data pouze do deseti úrovňového histogramu, v druhém kroku se provádí 24 úrovňová extrakce. Postup při tvorbě CEDD je zobrazen na obrázku níže:




Obr. 2.2: Vývojový graf tvorby CEDD

### 2.3.3 Histogram rozmazaných barev a textur - FCTH (Fuzzy Color and Textur Histogram)

Metoda FCTH vychází z kombinace tří fuzzy (angl. rozmazaný) kroků. V prvním kroku se z obrazu vyextrahuje tzv. Fuzzy-Linking Histogram (histogram s deseti úrovněmi), v druhém kroku se provede extrakce z tohoto deseti úrovňového histogramu do dvaceti čtyř úrovňového histogramu s postupným doplňováním informací o barvách. Ve třetím kroku je provedena Haarova vlnková transformace, na konci třetího kroku je provedena extrakce z dvaceti čtyř úrovňového do sto devadesáti dvou úrovňového histogramu[20].

### 2.3.4 Gaborova podobnostní metrika - Gabor

Metoda Gabor je založena na extrakci informací z obrazu na základě použití Gaborovi transformace (speciální případ Fourierovy transformace). Tato metoda těží z obrázku hlavně informace o textuře a hranách objektů v rámci obrázku.

### 2.3.5 Tamurova podobnostní metrika - Tamura

Tato metoda je implementací vytěžení a porovnání tří Tamurových vlastností. Mezi tyto vlastnosti patří: hrubost obrazu, kontrast, směrovost, pravidelnost, drsnost obrazu a podobnost linií obrazu [22].

### 2.3.6 Sjednocený kompozitní popisovač - JCD (Joint Composite Descriptor)

Extrakce JCD je založena na kombinaci CEDD a FCTH. Tento index je založen na 7 texturních oblastech, kdy každý z nich má 24 podoblastí, které odpovídají rozložení barev na obraze od bílé (podoblast 0) až do tmavé magenta (podoblast 23). Dále je obraz rozložen na 6 texturní oblastí. Hodnota JCD vzniká z kombinace těchto oblastních rozložení[23].

### 2.3.7 Ostatní podobnostní metriky - JpegCoefficientHistogram, GeneralColorLayout, HSVColorHistogram, SimpleColorHistogram

Tyto metody a indexy odpovídají různě složitým a různě úrovňovým barevným histogramům, které jsou vypočítány nad daným obrazem. SimpleColorHistogram a HSVColorHistogram jsou jednoduché histogramy, avšak první z nich je postaven na barevném modelu RGB, druhý využívá barevný model HSV (odstín, sytost, hodnota - angl. Hue, Saturatuion, Value).

## 2.4 Informační zisk - Information Gain

Jak již bylo psáno výše, přidělování anotací není otázkou pouze samotného obrazu, ale z velké části i informací popisující významové vlastnosti obrázku, které má algoritmus dostupné v rámci trénovací množiny. Na základě výskytu každé anotace u obrazu testovací množiny je tak možno spočítat četnost $Ca$ dané anotace, tím pádem pravděpodobnost výskytu anotace $Pa$ v rámci celé množiny anotací. Od pravděpodobnosti anotace je možné se poměrně jednoduše dostat k celkové entropii dané

anotační množiny $H$ a jako poslední krok je možno vyjádřit informační zisk daného obrázku $IGi$.

Pravděpodobnost anotace je dána vzorcem:

$$Pa = \frac{Ca}{C} \tag{2.4}$$

Kdy $Ca$ je počet výskytů konkrétní anotace, $C$ je celkový počet anotací. Po vypočtení pravděpodobnosti je možno poměrně jednoduše zjistit entropii celé anotační množiny. Entropie označuje míru neuspořádanosti systému anotací. Je vyjádřena tímto vzorcem:

$$HS = -\sum_{t=1}^{T}(p_t * log_2(p_t)) \tag{2.5}$$

Kde $HS$ je entropie celého systému, $p_t$ označuje pravděpodobnost výskytu jednotlivé anotace. Informační zisk je hodnota odvozená od entropie, informační zisk je rozdíl entropie celého systému a jednotlivé anotace. Toto je dáno vzorcem:

$$IGt = Ht - HS \tag{2.6}$$

Kdy $IGt$ vyjadřuje informační zisk anotace a $Ht$ je entropie pro uvažovanou anotaci t. Informační zisk je v podstatě redukce trénovací množiny anotací způsobená volbou správné anotace $t$. Tento výpočet je zde hlavně uveden, protože má velký význam v dalším postupu při tvorbě náhodných stromů trénovací množiny obrázků a anotací.

# 3 TEORIE REALIZOVANÝCH ALGORITMŮ PRO OZNAČENÍ OBRÁZKŮ ANOTACÍ NA ZÁKLADĚ PODOBNOSTI OBRAZU

Pro označení obrázků byla použita teorie náhodných stromů, vypočtení pravděpodobností, entropie a informačního zisku a různých podobnostních algoritmů. Postup označení obrázku novou anotací není jednokroková operace, měl by obsahovat následující kroky:

Prvním krokem je načtení trénovací množiny, skládá se z tří podkroků - vypočtení předchystaných hodnot pro reprezentaci obrazu v rámci trénovací množiny, uložení do vhodné struktury pro rychlý následný průchod daty a uložení zpracované trénovací množiny na diskový prostor. Ukládání na disk samozřejmě není nutností, ale vzhledem k paměťové a časové náročnosti přípravy trénovací množiny se zdá jako logický krok.

Dalším krokem je načtení a zpracování testovací množiny. Tento krok obsahuje menší úkoly - vypočtení předchystaných hodnot pro reprezentaci obrazu testované množiny stejně jako u obrazů trénovací množiny, uložení do spojové struktury. Posledním krokem opět může být uložení zpracované testovací množiny na diskový prostor, ale vzhledem k tomu, že testované obrázky se mohou měnit, tak nenabývá takové důležitosti jako u trénovací množiny.

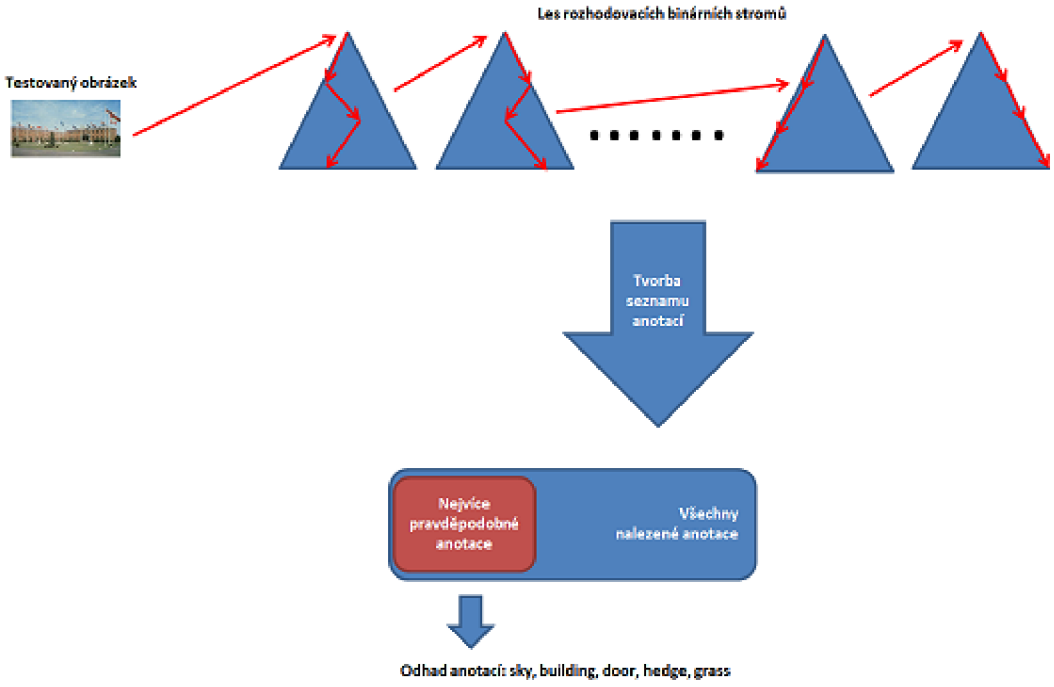
Posledním, avšak nejdůležitějším úkolem je samotné porovnání jednotlivých obrazů testovací množiny s trénovací množinou a vytvoření seznamu anotací jednotlivých testovacích obrazů. Z takto vytvořeného seznamu nejpravděpodobnějších anotací je nutno vzít pouze ty nejsilnější a považovat je za výsledek. Postup přiřazení je možnos vidět na obrázku 3.1.

Popisovaný postup se může velmi lišit na základě vybraných struktur a algoritmů. Prvním zásadním bodem je výběr datové struktury pro vytvoření trénovací množiny, dalším vytvoření předpočítaných hodnot na úrovni jednoho obrazu trénovací množiny, třetím důležitým aspektem je výběr algoritmu pro porovnání, posledním proměnným parametrem je způsob výběru nejpravděpodobnějších anotací z množiny všech nalezených anotací.

V této kapitole budou popsány z teoretického hlediska postupy v této práci použité, co se týče samotné praktické implementace, ta bude popsána v následující kapitole. Jedná se tyto o postupy:

- Vytvoření trénovací množiny

- Určení vlastnostní obrazu - zprvu byl zvolen histogram, ale ve výsledné aplikaci byl tento postup nahrazen knihovnou LIRE
- Tvorba vyváženého náhodného stromu bez vazby na informaci v obraze - první postup
- Tvorba náhodného stromu na základě informačního zisku - druhý postup
- Porovnání podobnosti a vyhodnocení




Obr. 3.1: Diagram průběhu přiřazení anotace

## 3.1 Vytvoření trénovací množiny

Prvním krokem ve zpracovávání trenovací množiny je vytvoření tzv. trénovacího datasetu. Trénovací dataset je struktura, která obsahuje odkazy na konkrétní obrazový materiál, obsahuje předpočítané parametry (indexy, koeficienty, histogramy) pro daný obraz a v případě malých datasetů může obsahovat i samotný obrazový materiál (o této variantě se v práci neuvažuje). Daný trénovací dataset musí být uložen ve vhodné struktuře pro rychlý průchod, protože s trénovací množinou je následně porovnáván každý obrázek z testované množiny.

Prvním postupem, který byl v této práci zvolen, bylo uložení trénovaného obrázku do hashovací tabulky, která měla za klíč anotaci daného obrázku. Obrázek byl následně uložen v této tabulce tolikrát, kolik měl anotací. Avšak po prvních testech

bylo zřejmé, že použití sekvenční struktury bylo zcela nevhodné, ve výsledku vznikala tabulka s desetitisíci klíči. Vytvoření samotné hashovací tabulky nad testovací množinou bylo otázkou sekund, ale porovnávání s testovaným obrazem bylo otázkou hodin, ne-li dní. Hlavním důvodem tohoto nezdaru byla nutnost projít při vyhodnocení podobnosti všechny obrazy trénovací množiny. Proto byla pro uložení trénovací množiny zvolena binární stromová struktura náhodného stromu (Random Tree)[4].

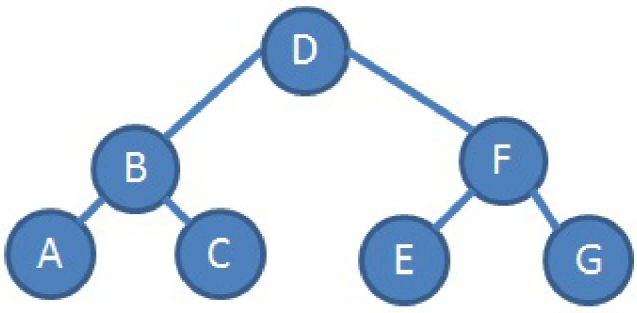
Prvním použitým stromem byl vyvažovaný binární strom, který byl tvořen bez závislosti na vstupních datech, tzn., uzly se do stromu řadili zcela bez závislosti na obsahu a zcela v podstatě náhodně. Přidávaný prvek se vždy založil na další prázdný list z levé strany. Ukázalo po několika pokusech, že tento přístup je zcela nepoužitelný, protože anotace byly při průchodu stromem přiřazovány náhodně a bez nějaké vazby na informaci v obrázku.

Druhý a již poměrně úspěšnější pokus spočíval ve vytvoření binárního stromu rozdělovaného na základě informačního zisku (Information Gain) nad množinou anotací jednotlivých obrázků. Jasným nedostatkem této struktury je to, že při nevhodném vkládání uzlů může dojít k jednostrannému prodlužování stromu a tento strom se v extrémním případě může stát spojovou nebinární strukturou (jako je například spojový seznam). Nevyváženosti se dá předcházet vyvažováním stromu, ale zatím nebylo použito.

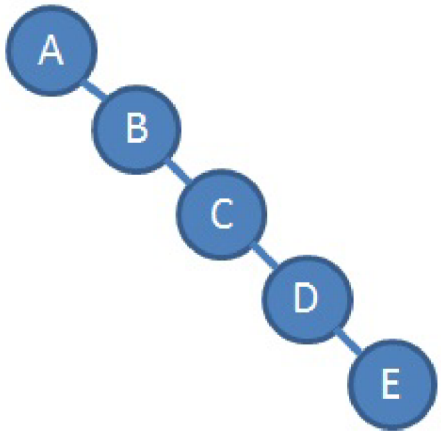
Pro samotnou reprezentaci trénovací množiny se těchto stromů vytváří N. Paramentr N by měl ovlivňovat počet uzlů v daném stromu a následně také počet porovnání, které je nutno provést nad testovaným obrazem. Rozsah N je možnost volit od 1 (počet porovnání je logaritmus o základu 2 z počtu trénovacích obrázků) do počtu trénovacích obrázků (porovnání se vykonává s každým trénovacím obrazem). Na počátku byl zvolen parametr N na hodnotu 10 na základě subjektivního rozhodnutí.

## 3.2 Určení histogramu obrazu

Uvedená metoda nebyla první, která byla zvolena. Ukázalo se, že samotné použití jednoho histogramu také není vhodné. První metodou bylo porovnání MSE (mean square error) testovaného obrazu s každým obrázkem testovací množiny. Vzhledem k tomu, že MSE využívá výpočtu nad každým pixelem, znamenal by tento výpočet několik stovek matematických operací nad jednotlivým obrázkem. Dále muselo být bráno v úvahu, že trénovací množina obsahovala od desetitisíce trénovacích prvků (obrazů), tudíž délka výpočtu byla neuspokojivě dlouhá. Při prvních pokusech nebyl jeden obrázek rozpoznán ani po několika hodinách. Další výraznou nevýhodou této metody byla potřeba načítat a hlavně ukládat obrazové data do paměti počítače.




Obr. 3.2: Vyvážený binární strom




Obr. 3.3: Extrémně nevyvážený binární strom

Po pokusech s MSE byl zvolen přístup porovnání barevných histogramů.

Jako další z postupů použitý v této práci byla zvolena metoda porovnání barevných histogramů. Barevným histogramem je nazvána struktura, která reprezentuje model RGB v intervalech. Tzn., hodnoty jednotlivých barev RGB od 0 - 255 jsou rozděleny do intervalů a následně je do tabulky zapsána četnost kombinací jednotlivých intervalů. Barevný histogram tímto značně rozmlžuje samotný obraz, ale vytvoření histogramu je poměrně časově a paměťově nenáročné. Výhodou je to, že pohyb na obrázku nevyvolá dramatickou změnu hodnot v histogramu. Při vytváření trénovacího datasetu je nad každým trénovacím obrazem vypočten barevný histogram a ten je uložen společně s informací o adrese obrazu. Histogram byl vypočten na základě algoritmu 1.

**Algoritmus 1:** Vytvoření barevného histogramu

**Vstup** : matice RGB M*N *image* a počet barevných intervalů histogram *count*

**Výstup**: histogram barevného rozložení *histogramTable*

```
1  foreach image[m, n] do
2  |  histogramLevel = [hodnotaRimage[m,n]/count, hodnotaimage[m,n]/count, hodnotaimage[m,n]/count];
3  |  if histogramTable.obsahuje (histogramLevel ) then
4  |  |  histogramTable.přičtiJedna(histogramLevel)
5  |  end
6  |  else
7  |  |  histogramTable.vlož(histogramLevel)
8  |  end
9  end
10 return histogramTable;
```

## 3.3 Určení vlastností obrazu za pomocí knihovny LIRE

Tato metoda se ukázala jako správná a finální. Vzhledem k tomu, že samotný barevný histogram byl příliš chudou informací pro určení podobnosti. Knihovna LIRE poskytuje nástroje pro extrakci dalších známých druhů histogramů a barevných nebo obrazových indexů. Vzhledem k tomu, že algoritmy a specifika výpočtu byly popsány v předešlé kapitole nejsou zde uvedeny.

## 3.4 Vytvoření náhodného stromu

V rámci aplikace pro označení obrázku anotace bylo nutno efektivně a úsporně uložit data z trénovací množiny. Na to byla použita struktura náhodného stromu (Random Tree).

### 3.4.1 Náhodný strom bez vazby na informaci v obraze

Prvním pokusem při tvorbě stromu byl zcela vyvážený strom, kterému ale při tvorbě stromu chybí jakákoliv informace o obsahu daného obrazu. Tento strom je binární rozhodovací strom s maximálním rozdílem 1 hloubky pravého a levého podstromu. Tyto stromy jsou řazeny do seznamu náhodných stromů, tzv. Random Forrest. Random Forrest je vytvořen hlavně z důvodu zvýšení pravděpodobnosti výskytu vhodných anotací a zvýšení počtu porovnání podobnosti. Náhodný les (Random Forrest) je tvořen postupem popsaným v algoritmu 2.

**Algoritmus 2:** Vytvoření Random Forrest

**Vstup** : množina obrázků *image* a počet nádodných stromů *countOfTree*
**Výstup**: seznam náhodných stromů (*RandomForrest*)

```
1 foreach image do
2 |   i = i++%(countOfTree);
3 |   vložDoStromu(i,image);
4 end
5 return (RandomForrest);
```

Do každého náhodného stromu jsou vkládány uzly na základě hloubky podstromů. Při vytvoření náhodného stromu se uzel snaží zařadit vždy z pravé strany listů každého ne plně obsazeného uzlu. Tato metoda se nakonec neukázala jako nejvhodnější. Popis je možno nalézt v algoritmu 3.

Tato varianta se z počátku zdála jako vhodná pro svou vyváženost a poměrně přímočarou formu tvorby stromu, ale nakonec z ní bylo upuštěno, protože obsahovala příliš velký prvek náhody.

### 3.4.2 Binární strom vytvořený na základě informačního zisku

Vzhledem k tomu, že vyvážený strom obsahoval příliš velkou složku náhody, bylo nutno zavést pravidlo do tvorby binárního stromu. Pravidlem se stalo vytěžení informačního zisku z anotací příslušného obrázku a následné rozhodování o zařazení do stromu. Informační zisk je počítán na základě pravděpodobnosti výskytu anotací v rámci celé množiny anotací. Údaje o pravděpodobnosti je tedy nutné počítat

**Algoritmus 3:** vložDoStromu()

**Vstup** : index stromu ($i$) a obraz ($image$)

```
1  foreach image do
2      if Strom[i].jePrazdny then
3          kořen = image;
4      end
5      else
6          VypočítejHloubkuPravéhoPodstromu();
7          VypočítejHloubkuLevéhoPodstromu();
8          if pravyPodstrom.hloubka = levyPodstrom.hloubka then
9              VložDoStromu(PravyPodstrom);
10         end
11         else
12             VložDoStromu(LevyPodstrom);
13         end
14     end
15 end
```

před samotnou konstrukcí binárního stromu, informační zisk je následně počítán při samotném zařazení do stromu. Tvorba tohoto stromu je popsána v algoritmu 4:

**Algoritmus 4:** Vytvoření Random Tree na základě Information Gain

**Vstup** : množina obrázků *image* a počet náhodných stromů *countOfTree*

**Výstup**: seznam náhodných stromů ($RandomForrest$)

```
1  foreach image do
2      i = i++%(countOfTree); IG = vypočítejInfoGain(image);
       vložDoStromuNaZákladěIG(i,image,IG);
3  end
4  return (RandomForrest);
```

Do každého binárního stromu jsou vkládány uzly na základě různých hodnot informačního zisku. Při vytvoření tohoto binárního stromu se uzel snaží zařadit vždy jako levý list, pokud má vkládaný uzel větší informační zisk než uzel, který již ve stromu je uložen, tak je nový uzel vložen jako levý list již vloženého uzlu. Pokud má vkládaný uzel nižší informační zisk než již uložený uzel, je vložen jako pravý list. Pokud je pravý nebo levý list již obsazen, je vkládaný uzel posunut o úroveň níže. Následně se rozhodování opakuje. Po ukončení konstrukce stromu by měly být uzly s největším informačním ziskem řazeny z levé strany a uzly s nejnižším informačním

ziskem jsou řazeny z pravé strany. Konstrukce tohoto stromu je pospána v níže uvedeném algoritmu 5

**Algoritmus 5:** vložDoStromuNaZákladěIG()

**Vstup** : index stromu ($i$), obraz ($image$) a informační zisk obrázku $IG$

```
1  foreach image do
2      if Strom[i].jePrazdny then
3          kořen = image
4      end
5      else
6          if uzel.IG >= IG then
7              vložDoStromuNaZákladěIG(LevyPodstrom, IG);
8          end
9          else
10             vložDoStromuNaZákladěIG(PravyPodstrom, IG);
11         end
12     end
13 end
```

## 3.5 Porovnání podobnosti a vyhodnocení pravdivých anotací

Stejně jako i ostatní kusy této práce i vyhodnocení mělo několik úrovní. Prvním stádiem bylo vyhodnocení na základě barevných histogramů.

Barevný histogram testovaného obrazu je následně porovnán s histogramem obrazů trénovací množiny na cestě stromem. Pokud se daný histogramový interval trénovacího obrazu v rámci odchylky 10 % (procentuální hodnota je parametrizovatelná z důvodu odchylek v pohybu v obrázcích a zamlžení a z důvodu testování nejoptimálnějšího nastavení) shoduje s histogramovými hodnotami testovaného obrazu, tak se bere daný interval jako shodný. Obrazy s více než 50% podobnými histogramovými intervaly jsou brány jako stejné a určují cestu průchodu náhodným stromem. V případě podobného obrazu se volí cesta směrem k levému podstromu, v případě nepodobnosti se volí cesta směrem k pravému podstromu. Procentuální odchylky jsou brány jako volitelné a závisí na uživateli, s jakou přesností chce danou podobnost vyhodnocovat. Na základě průchodu stromem je určen seznam anotací, které mohou být použity. V počátku se přidělovaly testovanému obrazu i anotace obrázků, které nebyly vyhodnoceny jako podobné, podobnost určovala pouze cestu

stromem. I po nastavení poměrně velkých odchylek byly výsledky poměrně neúspěšné, pouze barevný histogram byl nedostatečná informace o obrazu.

Druhým a finálním krokem bylo vyhodnocení na základě podobnosti kombinací vlastností, které je schopna extrahovat knihovna LIRE. Každá z vlastností je vyextrahována a uložena v binárním stromě, následně při porovnání jsou stejné vlastnosti testovaného obrazu porovnávány se stromem. Pro vyhodnocení podobnosti je využito vyhodnocení vzdálenosti. Pokud vzdálenost spadá do volitelného limitu dané vlastnosti je vyhodnocena vlastnost jako podobná. Následně pokud je vyhodnoceno, že je počet podobných vlastností více než obecný práh, tak je vyhodnocen obrázek jako podobný. Od takovéhoto obrázku jsou přiděleny anotace k testovanému obrázku, na rozdíl od původní myšlenky, kdy se přidělovaly všechny anotace.

Na základě porovnání histogramů nebo vlastností je vytvořen seznam anotací. Seznam musí být seřazen dle četnosti výskytu u daného obrázku. Dle četnosti výskytu je vybráno N nejpravděpodobnějších anotací a ty jsou zapsány k testovanému obrazu.

# 4 REALIZACE ALGORITMŮ PRO OZNAČOVÁNÍ OBRAZKŮ ANOTACÍ

Aplikace pro označování obrázků anotacemi je rozdělena na zásadní části: realizace trénovací množiny, realizace testovací množiny a samotný porovnávací algoritmus. Nejproblémovějším krokem se stala tvorba trénovací množiny, vzhledem k tomu, že použitá trénovací DB má poměrně velké množství trénovacích prvků. V rámci realizace dané aplikace bylo nutno se vypořádat hlavně s problémem velké spotřeby operační paměti počítače. Již první pokusy s aplikací ukázaly i omezení prostředí NetBeans - spuštění aplikací v prostředí NetBeans je schopno v implicitním modu pracovat v řádu několika desítek MB paměti, proto musel být spouštěn build s parametry `-Xms8M -Xmx7G`, které způsobují zvětšení standardní "haldy"(HEAP) ve vývojovém prostředí na potřebné parametry. V rámci spuštění kompilované aplikace v rámci prostředí Windows bylo zjištěno, že i samotný systém má obdobné omezení. Bylo tedy nutno využít prostředky 64bitového operačního systému a spustit aplikaci v 64 bitové verzi pomocí příkazu

`"c:\Program Files\Java\jre7\bin\java.exe" -d64`, [1]

aby došlo k plnému využití všech paměťových bloků[2]. Realizace a vývoj aplikace byla provedena na počítači s procesorem Intel(R) Core(TM) i7-2677M 1.80GHz s 4.0 GB RAM paměti s použitím ukládání na SSD disky.

## 4.1 Popis konfigurace aplikace

Vytvořená aplikace je poměrně rozsáhle konfigurovatelná, pro spuštění aplikace je možno na základě parametru `-c` . Konfigurace aplikace je uložena v konfigračních souborech s koncovkou `XML` v adresáři `config`. Konfigurační soubor má následující části:

- ProgressSteps - nastavení postupu aplikace, jak daleko v rozpoznání se aplikace dostane, jaké jsou vyžadovány výsledky a jaké jsou zvoleny postupy zpracování trénovacího a testovacího datasetu.
- Datasets - nastavení druhů trénovacího a testovacího datasetu.
- SimilarityTest - obecné nastavení, jakým způsobem se vyhodnocuje podobnost a jakým způsobem se následně ze seznamu anotací vybírají vhodné anotace
- TestLevels - nastavení prahových úrovní jednotlivých vlastností obrazů a celkového podobnostního prahu

[1] Varianta pro Windows 7 64bit

[2] Vzhledem ke standardním nabízeným PC sestavám na internetových obchodech v dubnu 2013 toto nepovažuje autor za omezení

- additionalSettings - dodatečné nastavení, které není pro aplikaci povinné, ale je možno ji pomocí těchto hodnot ovlivnit

```
<root>
    <ProgressSteps>
        <GlobalProgress>ALL</GlobalProgress>
        <!-- enumLabelingProgress
                    PROP,       // po nastavení pravděpodobnos
                    TRAINING,   // po vytvoření tréningových m
                    TEST,       // po vytvoření testovacích mn
                    ALL,        // všechno  -->
        <TrainingDatasetStep>E</TrainingDatasetStep>
        <!-- enumSteps
                    A, // inicializace se stromem včetně obráz
                    B, // inicializace se stromem bez obrázků
```

Obr. 4.1: Příklad konfiguračního souboru config.xml

## 4.2 Trénovací a testovací DB

Trénovací a testovací datasety jsou v rámci aplikace uloženy v adresářích `.\images\TEST_DB` (je možné v rámci konfuigurace aplikace nastavit i jiné vstupní adresáře) a `.\images\TRAINING_DB`. Prvním z požadavků pro úspěšné dokončení dané aplikace byl výběr vhodné trénovací databáze. Pro daný účel se vyskytuje poměrně značné množství databází, ale ne všechny splňují základní požadavky. Těmito požadavky jsou hlavně rozměr databáze a náročnost zpracování. V rámci starší i aktuální literatury se podařilo objevit tyto databáze:

- Vistex-60 database - databáze obsahuje pouze 60 barevných obrázků v rozlišení 128 na 128 pixelů. Tato databáze je rozdělena na kategorie po 2 až 12 obrazech podle sematického zaměření. Zdrojem této databáze je Massachusetts Institute of Technology [10].
- Vistex-167 database - v této databázi je obsaženo 167 obrazů v plném rozlišení 512 na 51 pixelů. Každý obraz je rozčleněn na 16 nepřekrývajících se 128 na 128 pixelů velkých podobrazů. Každý obraz tvoří jednu sématickou skupiny [11].
- Correl-1000 - database - tato databáze obsahuje 1000 barevných obrázků v rozlišení 384 na 256 pixelů. V rámci sématiky obrazu je v této databázi pokryto poměrně velké množství objektů, od přírodních scén, po výtvory člověka a úmělé objekty. Databáze je dělena na 10 kategorií po 100 fotografiích. Tvůrcem této databáze je COREL Corporation [10].

- Corel5K database - tato databáze není poskytována ve formátu JPEG obrázků a anotací, obsahuje všechny data v binární podobě. Je rozdělena na 3 části - 4000 obrazků trénovací části, 500 validačních obrazů a 500 testovacích obrazů. Pro záměr této práce byla tato databáze shledána jako zcela nevhodná [12].
- ESPGAME database - databáze obsahuje 67 tisíc obrazů a 350 tisíc anotačních záznamů. V průměru je přiděleno 5 anotací na jeden obraz, ale pohybují se od 1 do 19 anotací na obraz. Celkově čítá 22 tisíc unikátních anotací. Samotné obrazy jsou ve formátu JPEG a anotace jsou uloženy v podobě TXT souborů [4].
- SUN2012 - Scene UNderstanding verze 2012 - databáze obsahuje 15 tisíc obrazů a 5 tisíc anotačních záznamů ve 130 tisících výskytech. Výskyty anotací jsou velmi odlišné, protože anotace jsou přidělovány na základě uživatelského vstupu na stránkách projektu. Anotace jsou uloženy v poměrně rozsáhlých souborech ve formátu XML. Tyto anotační soubory obsahují nejenom seznam anotačních informací, ale také přímo souřadnice, na kterých se daný objekt v obrazu nachází. Tyto souřadnice se však v rámci této práce nevyužívají. Dalším rozšířením této databáze oproti ostatním je rozložení anotačního i obrazového materiálu do složek a podsložek, které shlukují informace se stejným nebo podobným významem.

Pro vytvoření označovací aplikace bylo nutno vzít co největší množinu anotací a obrazů, avšak i sebevětší množství informací musí být zadáno smysluplně. Z prvního z těchto důvodů byla vybrána databáze ESPGAME, avšak ve výsledku se volba přiklonila k práci nad databází SUN2012. Jasným důvodem bylo, že informace v databázi ESPGAME jsou velmi roztříštěné a občas i nesmyslné. Databáze ESPGame obsahuje například barvy ("yellow", "green"atd.), ale pro získání těchto hodnot existují i jednodušší než uvedené postupy. Dalším argumentem oproti tomu SUN2012 obsahuje povětšinou smysluplné označení objektů na obrázku, ne jejich vlastností. Databáze ESPGame je velmi jednoduše použitelná, protože zpracování JPEG obrazu a textu je standardním úkonem aplikací. Obsahuje širokou škálu zachycených objektů a činností. Navíc narozdíl od ostatních databází nejsou obrázky v ESPGAME v jednom rozlišení, nachází se zde obrázky od 66 na 21 pixelů až po 224 na 169 pixelů. Databáze SUN2012 je také rozměrově různorodá, ale všechny obrázky mají velmi podobnou datovou velikost okolo 645kB. Zpracování materiálu databáze SUN2012 je tedy složitější pouze v tom, že anotační data jsou uloženy v XML dokumentech a ne v plochých souborech jako v ESP Game.

Na základě pokusů bylo zjištěno, že jak databáze ESP Game, tak i SUN 2012 je poměrně zanešena anotacemi s malým počtem výskytů. Takové anotace bylo nutno ignorovat, protože nemají v rámci celkové množiny v podstatě žádnou váhu.

Navíc poukazují pouze na nějaký extrémní objekt na obraze. Práh váhy anotace je nastavitelný v rámci aplikace pomocí nastavení `minimalAnotationCount` v sekci `additionalSettings`, implicitně je nastaven na 10 výskytů. Z uvedených grafů je vidět rozložení anotací v jednotlivých databázích před a po očištění o nevýznamné anotace.

V databázi ESP GAME nepodstatné anotace tvořily necelých třináct tisíc z dva a dvaceti tisíc záznamů. Obecně jde z grafů poměrně dobře vidět, že se tímto opatřením zlikviduje velké množství extrémů a jsou ponechány pouze platné anotace.

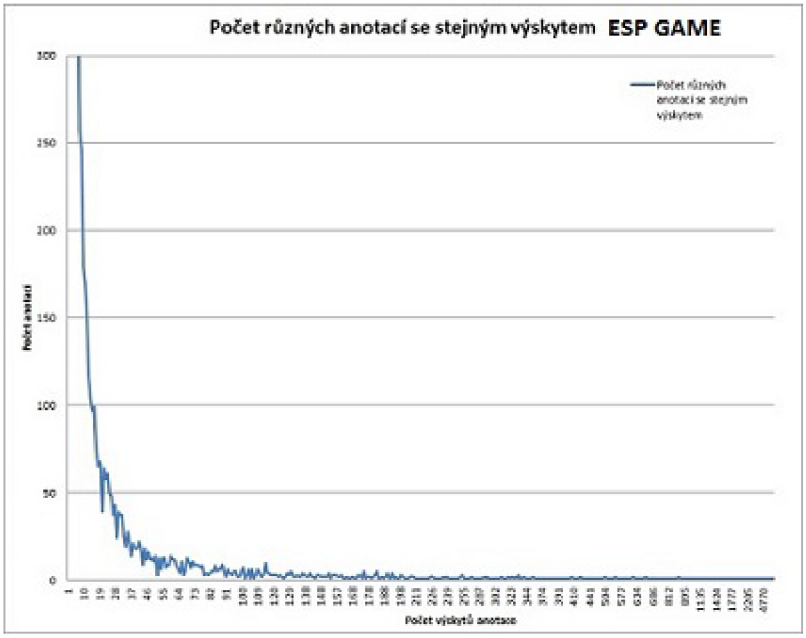
V databázi SUN 2012 tvořily anotace pod 10 výskytů pouze 3000 výskytů, tudíž netvořily takovou výraznou část celé anotační množiny. Na grafech jde vidět, že změna není tak výrazná jako u ESP GAME.

## 4.3 Trénovací množina jako hashovací tabulka

Realizace dané apliakce byla rozdělana na několik částí - první částí bylo vytvoření tříd pro načtení trénovacího datasetu, následně vytvoření tříd pro načtení jednotlivých testovacích datasetů a následně vytváření jednotlivých podobnostních algoritmů pro vyhodnocení podobnosti. Pro trénovací dataset byl vytvořen iface `ifaceTrainingDataset`, který by měl sloužit jako rozhraní nejen pro ESPGAME, SUN2012 ale i další trénovací datasety, které by bylo potřeba do budoucna zabudovat. První zásadním problémem v realizaci bylo, jak vlastně umístit trénovací dataset do paměti počítače jako trénovací dataset.

První pokusy spočívaly ve vytvoření hashovací tabulky anotací s přímou vazbou pouze na adresu obrázku. Toto řešení bylo poměrně jednoduché a velmi rychlé. Avšak nevýhody se projevily při prvním pokusu o vyhodnocení podobnosti s prvním testovaným obrazem. Rozpoznání se stalo neúměrně zdlouhavým, jelikož na základě adresy musel být vždy natažen obraz z disku opět do paměti, přepočítán a následně zpět uložen na disk.

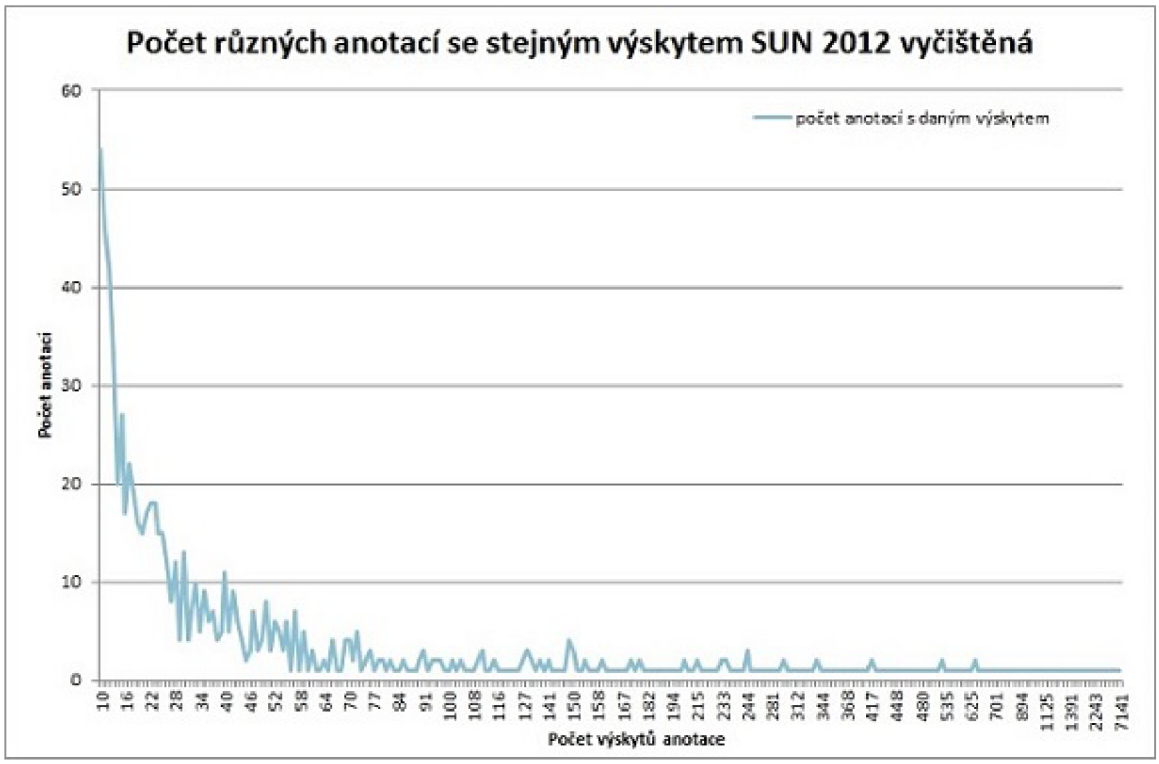
Diagnostika vývojového prostředí NetBeans ukázala, že největší spotřeba času při pokusu o porovnání je při opětovnémi načítání obrázku do paměti ve formátu objektu `ImagePlus`. Proto byl změněn přístup na okamžité a permanentní načtení obrazů do struktury `ESPGameTrainingDataset`, popřípadě `SUN2012TrainingDataset`. Nyní se drobně prodloužilo načítání trénovací množiny, ale přibylo značné vytížení paměťového prostoru. Přesněji na testovacím PC docházelo ke swapování paměti na HDD, protože JVM je schopno virtualizovat podstatně více paměti, než je fyzicky ve stroji zabudováno. Tvorba trénovací množiny vycházela nyní na necelé tři hodiny. Jelikož se nepočítá s přílišnou změnou ve struktuře trénovací množiny, a také není možné vždy celou trénovací množinu tvořit znova, byla zvolena možnost uložení




Obr. 4.2: Graf hustoty počtu anotací nepročištěné databáze ESP Game - omezen na počet 300 anotací


Obr. 4.3: Graf hustoty počtu anotací pročištěné databáze ESP Game


Obr. 4.4: Graf hustoty počtu anotací nepročištěné databáze SUN 2012 - omezen na počet 300 anotací




Obr. 4.5: Graf hustoty počtu anotací pročištěné databáze SUN 2012

na disk, přesněji serializace objektů. Použití objektů typu `ImagePlus` z knihovny ImageJ avšak zabránilo správné serializaci objektů. Tyto objekty nemají implemetováno rozhraní Serializable. Proto byla zvolená knihovna `Kryo` pro náhradu standardní serializace tréningového datasetu.

## 4.4 Trénovací množina jako náhodný strom

Po dokončení kontrukce s hashovací tabulkou byla rozpracována základní metoda MSE, bohužel rozpoznání trvalo příliš dlouho. Jasným důvodem je to, že při každém testovaném obrazu musela struktura projít všechny klíče hashovací tabulky, aby určila pravděpodobnost a podobnost. Proto bylo nutné zvolit datovou strukturu s podstatně nižší algoritmickou složitostí. Jako správná struktura byl zprvu zvolen seznam náhodných binárních rozhodovacích stromů, avšak následně se ukázalo, že je nutné z konstrukce stromu vyřadit prvek náhody. Byl začleněn výpočet informačního zisku na základě výpočtů s tabulkami `_anotationCounter` a `_anotationPropability`.

Jako vlastnost obrazu byl v prvotních pokusech použit naivní velmi jednoduchý barevný histogram jako jediná vlastnost obrazu trénovacího i testovacího obrazu, avšak vzhledem k mal rozsahu této vlastnosti, byly použity následně objekty knihovny `LIRE` implementující rozhraní `LireFeature`. Toto rozhraní poskytuje poměrně jednoduchou funkci `extract()` pro extrakci zvolené vlastnosti z obrazu. Vytěžené vlastnosti jsou ukládány do objektu `ImagePropertyCreator`. Samotný `ImagePropertyCreator` slouží pouze jako sjednocená paměť pro nastavení parametrů podobnosti a návratu vlastností obrazu pro testovací i trénovanou množinu, aby se nestalo, že v při trénování budou extrahovány jiné vlastnosti než pro testovaný obraz. Samotná tvorba těchto stromů je časově značně náročnější než vytváření hash tabulky, ale následný průchod danými stromy je otázkou sekund. Při průchodu je testovaný obraz je porovnáván se všemy stromy trénovacího datasetu, průchod každým binárním stromem pro jednoduchý histogram je popsán v v algoritmu 7. Průchod stromem, který využívá vlastnosti dosažené pomocí knihovny Lire je popsán v algoritmu 8.

V prvním případě byla použita metoda, kdy se k testovanému obrazu přidávali vždy anotace z každého otevřeného uzlu, to se však ukázalo jako velmi neproduktivní, proto se v konečném provedení přidávají k testovanému obrazu pouze anotace, které obsahují obrazy trénovací množiny vyhodnocené jako podobné. Dále nejsou k obrazu řazeny anotace, které nesplňují podmínku minimálního výskytu v rámci celé trénovací množiny. Tímto je docíleno toho, že se řadí pouze významné anotace podobných obrazů a anotační množina testovacího obrazu není zanášena nepotřebnými informacemi. Speciálním případem je použití databáze `SUN2012` a nastavení

**Algoritmus 7:** Průchod stromem za použití histogramu

**Vstup** : Testovaná obraz (*testImage*), náhodný strom *randomTree*, procenta podobnosti *proc*

```
1  aktualniUzel = randomTree.Kořen;
2  testovanyHistogram = testImage.Kořen.dejHistogram()
3  while aktualniUzel != null do
4  |   aktualniHistogram = aktualniUzel.dejHistogram();
5  |   početStejnychHistogramHodnot = porovnejHistogram(testovanyHistogram,
   |   aktualniHistogram);
6  |   if početStejnychHistogramHodnot > (početVšechHodnotHistogram * proc)
   |   then
7  |   |   aktualniUzel = aktualniUzel.LevyPodstrom();
8  |   end
9  |   else
10 |   |   aktualniUzel = aktualniUzel.PravyPodstrom();
11 |   end
12 end
```

`\\root\additionalSettings\Setting (@Name="SUN2012_anotationFromDirectories",@Value="false")`, v tomto případě se jako anotace pro daný obrázek bere pouze název adresáře - scény, ve které je daný obrázek z trénovací množiny uložen.

Trénovací dataset může být načten ze souboru s připravenými hodnotami, aby se urychlila fáze přípravy. V základním nastavení aplikace se načítá soubor `.\sandbox\trDataset1.dat`, avšak aplikaci lze nastavit na libovolný soubor adresáře sandbox pomocí nastavení uloženého v proměnné `\\root\Datasets\trainingDataset(@LoadFileName)`.

## 4.5 Testovací množina obrázků

Vytvoření testovací množiny obrázků není velmi složitý úkol. Pro reprezentaci testovací množiny vzniklo rozhraní `ifaceTestDataset`. Rozhraní reprezentuje všechny dostupné funkce, které by měla trénovací množina mít. Zatím jedinou implemntací rozhraní je třída `JPGTestDataset`. Tato třída načítá jednoduchý adresář obrázků ve formátu JPG nebo JPEG. Hlavními úkoly této třídy je načtení obrazů do paměti a výpočet jejich podobnostních vlastností (buď histogramu nebo LIRE vlastností) a následný zápis výsledků porovnání do umístění testovací množiny.

**Algoritmus 8:** Průchod stromem s použitím LIRE

**Vstup** : Testovaná obraz (*testImage*), náhodný strom *randomTree*, tabulka vlastností a prahů podobností *property*[*vlastnost*][*prah*] , celkový podobnostní prah *SimilarityLevel*

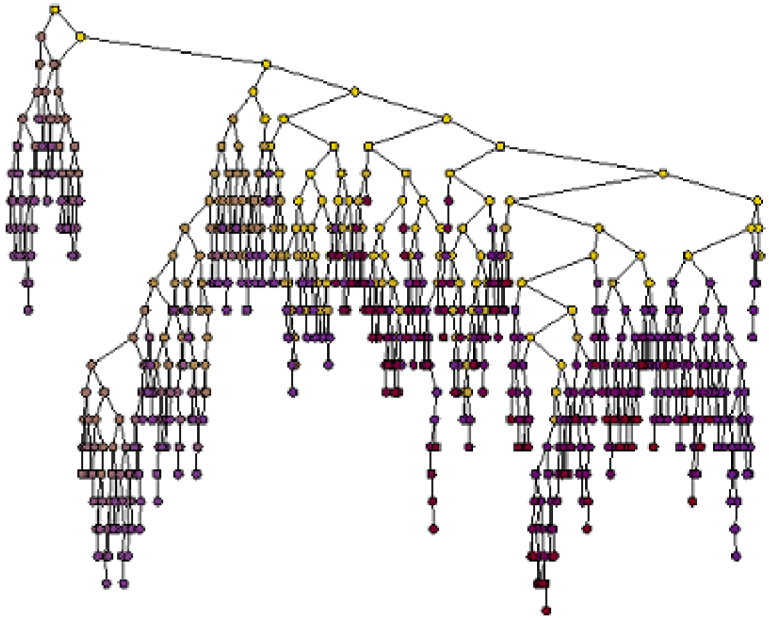
```
1  aktualniUzel = randomTree.Kořen;
2  testVlastnosti[][] = testImage.Kořen.dejVlastnosti()
3  while aktualniUzel != null do
4  |   aktualniVlastnosti[][] = aktualniUzel.dejVlastnosti();
5  |   početStejnychVlastnosti = porovnejVlastnosti(testVlastnosti[][],
   |   aktualniVlastnosti[][],property[][] );
6  |   if početStejnýchVlastností > (SimilarityLevel) then
7  |   |   aktualniUzel = aktualniUzel.LevyPodstrom();
8  |   end
9  |   else
10 |   |   aktualniUzel = aktualniUzel.PravyPodstrom();
11 |   end
12 end
```

Testovací dataset může být načten ze souboru s připravenými hodnotami. V základním nastavení aplikace se načítá soubor `.\sandbox\testDataset.dat`, avšak aplikaci lze nastavit na libovolný soubor adresáře sandbox pomocí nastavení uloženého v proměnné `\\root\Datasets\testDataset(@LoadFileName)`.

Rozhranní `ifaceTrainingDataset` a `ifaceTestDataset` jsou nyní přichystány na to, že v budoucnu poskytnou podporu různým datům i formátům. Tzn. bude možno porovnávat různé formátově i datově odlišné datasety.

## 4.6 Vyhodnocení podobnosti, podobnostní test, zápis výsledků

O porovnání a vyhodnocení podobnosti je zodpovědné hlavně rozhraní `ifaceSimilarityTest`. Díky tomuto rozhraní je možno provádět typově různé porovnání nad stejnými daty. Toto rozhraní obsahuje hlavně metody pro provedení podobnostního testu mezi trénovací a testovací množinou, zápis výsledků na disk a vyhodnocení přesnosti odhadu anotace, přesněji metody `processTDSandTS()`, `writeResultIntoResultFile()` a `determineTheAccurracy()`.



Obr. 4.6: Grafické zobrazení náhodného stromu [15]

Hlavní použitou implementací rozhraní `ifaceSimilarityTest` je třída `SimilarityTestProperties`. Rřída realizuje porovnání testovaných obrázků s trénovacím datasetem na základě vlastností knihovny LIRE. Další z vlastností, kterou obsahuje rozhraní `ifaceSimilarityTest,` je nastavení způsobu vyhodnocení pravděpodobných anotací z množiny nalezených anotací. Implicitně je v aplikaci nastaveno, že se bere N nejpravděpodobnějších (s nejčastějším výskytem) anotací nalezených v rámci obrázku, tento způsob je změnitelný v rámci konfigurace v uzlu `\\root\SimilarityTest\AccuracyType`. N je nastaveno na 10, ale je možno ho také v rámci konfigurace aplikace změnit v uzlu `\\root\SimilarityTest\AccuracyLevel`.

Výsledek rozpoznání je zapsán na disk nebo zobrazen na základě konkrétní implementace rozhraní `ifaceTestDataset`. Při použití třídy `JPGTestDataset` je proveden zápis anotačních informací pod sebe do jednoduchého textového souboru do adresáře `RESULTS`, popřípadě je umožněno uživateli měnit výstupní adresář v konfiguraci aplikace pomocí `\\root\additionalSettings\Setting(@Name="ResultPath", @Value="CESTA")`. Pokud je v rámci testovací množiny dostupný i soubor s anotačními informacemi testovaných obrázků, může aplikace provést porovnání přesnosti odhadu anotací. Výsledek tohoto porovnání je zapisován do souboru `.\sandbox\*TestAccuracy.csv`. Název se může měnit v rámci různých použitých testovacích množin. Soubor obsahuje velmi jednoduchou strukturu názvu souboru obrázku a hodnoty od 0 do 1. Kdy 0 znamená, že nebyla odhadnuta žádná anotace, 1 znamená, že byly nalezeny všechny anotace obsažené v původním anotačním

souboru.

## 4.7 Popis struktury a použití aplikace

### 4.7.1 Souborová struktura

Aplikace pro automatickou anotaci obrazu obsahuje tyto soubory a adresáře:

- `config` - adresář obsahující konfigurace a možná nastavení aplikace
- `images` - tento adresář obsahuje možné trénovací a testovací databáze. Obsahuje tyto podadresáře:
  - `TEST_DB` - adresář, který slouží k načtení testovaných obrazů do aplikace, jsou zde uloženy obrazy, možné porovnávací anotace pro vyhodnocení výsledku a úspěšnosti a zde jsou zapisovány odhadnuté anotace. Adresářů s testovanými obrazy může být opět více. Slouží k tomu, aby si mohl uživatel vybrat, ze které množiny chce testovat obrazy. Přesná cesta k konkrétní testované množině se dá nastavit pomocí atributu `DirectoryPath` uzlu `testDataset`.
  - `TRAINING_DB` - tento adresář obsahuje trénovací množiny pro volbu trénování aplikace. Každá trénovací databáze je ve zvláštním adresáři a své vlastní struktuře (adresáře ESP-ImageSet, SUN2012 atd.)
  - `RESULTS` - adresář pro implicitní ukládání výsledných odhadů
- `libs` - adresář knihoven, které aplikace využívá
- `logs` - výstupní adresář logovacích záznamů
- `sandbox` - adresář, ve které jsou uloženy předpřipravené a předpočítané trénovací i testovací datasety, mezivýsledky a CSV soubory s výstupními informacemi o přesnostech a množinách anotací
- soubor `DP.jar` - samotná aplikace pro rozpoznání anotace obrazu
- soubory `*.cmd` - spouštěcí soubory aplikace pro různá nastavení aplikace a JVM[3]

### 4.7.2 Použití aplikace

V rámci práce s aplikací pro automatickou anotaci obrazu je umožněno uživateli, aby spouštěl aplikace s různými konfiguračními soubory a tím zcela zásadně ovlivnil její chování:

- `run.cmd` - příkazový soubor spouštící aplikaci s nastavením v implicitním config.xml, ponechává na uživateli, jaké chce použít nastavení

[3]JAVA Virtual Machine

- `run_ESP_build_only.cmd` - vytvoření pouze souboru `ESPGAMEtr.dat`, který v sobě má uložený a zpracovaný trénovací dataset ESP GAME - tato úloha může na slabším PC běžet i několik dní
- `run_ESP_build_recogn.cmd` - vytvoření stejného trénovacího datasetu jako v `run_ESP_build_only.cmd` a okamžité rozpoznávání testovaných obrázků - tato úloha může na slabém PC běžet i několik dní
- `run_ESP_propability.cmd` - pouhé spočítání pravděpodobnostních tabulek a čítačů anotací v rámci databáze ESP GAME
- `run_ESP_recognition1000.cmd` - načtení zpracovaného trénovacího z předpočítaného trénovacího datasetu ESP GAME ze souboru `ESPGAMEtr1000.dat` a následné rozpoznání testovaných obrázků. `ESPGAMEtr1000.dat` je dataset vytvořený z náhodného výběru cca 1000 obrázků z množiny ESP GAME
- `run_ESP_recognition5000.cmd` - načtení zpracovaného trénovacího z předpočítaného trénovacího datasetu ESP GAME ze souboru `ESPGAMEtr5000.dat` a následné rozpoznání testovaných obrázků. `ESPGAMEtr5000.dat` je dataset vytvořený z náhodného výběru cca 5000 obrázků z množiny ESP GAM
- `run_ESP_recognition9000.cmd` - načtení zpracovaného trénovacího z předpočítaného trénovacího datasetu ESP GAME ze souboru `ESPGAMEtr9000.dat` a následné rozpoznání testovaných obrázků. `ESPGAMEtr9000.dat` je dataset vytvořený z náhodného výběru cca 9000 obrázků z množiny ESP GAME
- `run_SUN_build_only.cmd` - vytvoření pouze souboru `SUN2012tr.dat`, který v sobě má uložený a zpracovaný trénovací dataset SUN2012 - tato úloha může na slabším PC běžet i několik dní
- `run_SUN_build_recogn.cmd` - vytvoření stejného trénovacího datasetu, jako v `run_SUN_build_only.cmd` a okamžité rozpoznávání testovaných obrázků - tato úloha může na slabém PC běžet i několik dní
- `run_SUN_propability.cmd` - pouhé spočítání pravděpodobnostních tabulek a čítačů anotací v rámci databáze SUN 2012
- `run_SUN_recognition.cmd` - načtení zpracovaného trénovacího z předpočítaného trénovacího datasetu SUN2012 ze souboru `SUN2012.dat` a následné rozpoznání testovaných obrázků

Z uvedených příkladů je vidět, že základním způsobem použití vzniklé aplikace by mělo být spočítání trénovací množiny, uložení do souboru, a pak následně opakované použití již takto zpracovaných dat pro různé testované obrázky.

## 4.8 Pomocné výpisy aplikace

Pomocné výpisy aplikace, nebo-li "logování"jsou prováděny pomocí knihovny `log4j.jar`. Tato knihovna standardně zapisuje několik úrovní informací od DEBUG - informační a vývojové informace - po ERROR - informace o vzniku chyby nebo výjimky. Dostupné výpisy jsou v rámci aplikace ukládány v adresáři `logs`.

# 5 DOSAŽENÉ VÝSLEDKY

V této kapitole je pojednáno o dosažených výsledcích automatické anotace z pohledu úspěšnosti odhadu, ale také z pohledu výpočetní náročnosti a času.

## 5.1 Časová a výpočetní náročnost aplikace

Časová a výpočetní náročnost aplikace pro automatickou anotaci se dá rozdělit na dvě hlavní časové údobí. Prvním z nich je tvorba a příprava trénovacích a testovacích množin. Při tvorbě trénovací množiny velmi silně závisí na počtu a velikosti obrazů, ze kterých se celá trénovací množina staví.

Dalším velmi důležitým aspektem je i forma poskytnutí anotace. Je samozřejmé, že zpracování plochého textového souboru (jako například v databázi ESP GAME) zabere podstatně méně času i výpočetních prostředků než vyčítání ze složitých binárních nebo XML souborů (například SUN2012 DB). Byly provedeny pokusy a zjištěny tyto doby přípravy tréningových množin[1]:

- ESPGAME - náhodně vybráno cca 1000 obrazů s jednoduchou textovou anotací - zpracování 10 minut
- ESPGAME - náhodně vybráno cca 5000 obrazů s jednoduchou textovou anotací - zpracování 90 minut
- ESPGAME - náhodně vybráno cca 9000 obrazů s jednoduchou textovou anotací - zpracování 144 minut
- ESPGAME - zpracování všech 67796 obrazů bylo ukončeno po 5 dnech nekompletní. Na testovaném operačním systému se již nedostávalo prostředků.
- SUN2012 - zpracována celá množina 16tisíc obrazů za 5 dní

Vytvořené datasety se ukládají do souborů o velikosti od 100MB do 2GB.

Druhou částí je samotné vyhodnocení podobnosti, to je možno provádět i z předem připravených datasetů uložených v adresáři `sandbox`. Při opětovných testech s předpřipraveným datasetem `SUN2012.dat` s testovanou množinou 1000 obrazů bylo rozpoznání dokončeno do deseti minut. V případě rozpoznávání tvoří opět nejdelší časové údobí načtení a zpracování testovaných obrazů. Následné vyhodnocení je vzhledem ke konečné stromové struktuře poměrně rychlé a je otázkou sekund.

[1] výpočty byly prováděny na osobním počítači s OS Windows 7 64bit s procesorem Intel® Core™ i7-2677M CPU @ 1.8GHz a 4 GB RAM

## 5.2 Přesnost odhadu anotací

Aplikace je postavena tak, aby se pomocí konfigurace dala nastavit k různé přesnosti odhadu a využitím různých metod knihovny LIRE se dá nastavit i přesnost na základě různých parametrů. Při pokusech s testovací množinou SUN2012 bylo testováno 1000 obrazů vyčleněných z této množiny (podadresář `misc`). Prvním krokem bylo sledování pokusů s nastavením globálního prahu podobnosti, tzn. za podobné bylo považováno pokud X % z celkového počtu testů bylo úspěšných (nastavení v konfiguraci `\\root\TestLevels(@PassLevel)`[2]).

Vzhledem k tomu, že přesnost rozpoznání se vyhodnocuje na základě vzdáleností konkrétních vlastností knihovny LIRE, byly provedeny první testy s těmito vzdálenostmi:

- AutoColorCorrelogram - vzdálenost 70
- CEDD - vzdálenost 30
- FCTH - vzdálenost 50
- Gabor - vzdálenost 100
- GeneralColorLayout - vzdálenost 200
- HSVColorHistogram - vzdálenost 4000
- JCD - vzdálenost 60
- JpegCoefficientHistogram - vzdálenost 50
- SimpleColorHistogram - vzdálenost 3
- Tamura - vzdálenost 100000

Na základě určených vzdáleností bylo dosaženo těchto úspěšností v rámci odhadu anotací pro celou testovanou množinu.

Tab. 5.1: Přehled přesnosti rozpoznání anotace na základě prahu

| Hodnota prahu | Průměr podobnosti | Střední hodnota podobnosti |
|---|---|---|
| 0,1 | 28,8% | 27,2% |
| 0,2 | 31,5% | 30% |
| 0,4 | 27,4% | 25% |
| 0,5 | 20,7% | 16,7% |
| 0,6 | 12,4% | 0% |

Dalším způsobem, kterým se dá ovlivňovat přesnost a objektivnost hledání je změna úrovní jednotlivých vlastností obrazu. Při použití polovičních minimálních

[2]použité nastavení je uvedeno v desetinných číslech - 10% = 0.1, 20% = 0.2 atd.

vzdáleností pro vyhodnocení podobnosti dostáváme nepříliš dobré výsledky. V případě nastavení hlavního prahu na hodnotu 0.5 se nedaří nalézt vhodné anotace pro většinu obrazů a je anotováno pouze 12 souborů. Ale tyto anotace se neshodují s předloženými anotacemi. Při nastavení prahu na hodnotu 0.1 sice anotuje značně větší počet obrazů, ale pořád je výsledek úplně mimo zadané anotace

Z tohoto příkladu je vidět, že se obecně obrazy v testované a trénovací množině SUN2012 velmi liší a případná podobnost mezi nimi je velmi nízká. Pokud postupujeme druhým směrem, tzn. posuneme prahy daných vlastností na dvojnásobnou úroveň o proti původní, dostáváme hodnoty pokřivené chybným rozpoznáváním podobnosti. Při nastavení prahu na 0.9, tzn., 90% hodnot musí být podobných, dostáváme neuspokojivé 1% společných anotací. Jedná se o to, že použití takto velké vzdálenosti již vyhodnotí skoro každé dva obrazy jako stejné, a proto jsou všechny testované obrazy anotovány stejně nebo nevhodně pro jejich skutečný význam. Příklad správně a nesprávně anotovaného obrázku je možno nalézt na obrázcích 5.1 a 5.2.

## 5.3 Porovnání s podobnými pracemi

V rámci této práce je nunto provést porovnání výsledků s jinými pracemi, které se specializovali na podobné téma. Práce [4] vypracovaná na University of Nottingham z roku 2012 vykazuje poměrně podobné výsledky nad různými testovacími databázemi avšak z vypracování je vidět, že výsledky jsou více stabilní a u všech databází podobné. Při vypracování této práce použili autoři 3 různé metody. V této práci jsou uvedeny pouze výsledky společně použité DB ESP Game.

Tab. 5.2: Přehled přesnosti rozpoznání anotace porovnávané práce [4]

| Metoda | ESP GAME |
|---|---|
| RF_Count | 38,6% |
| RF_Count$^2$ | 38,1% |
| RF_optimize | 38,8% |

Obr. 5.1: Příklad správného odhadu anotace - Obrázek obsahoval anotace: sky, building. Odhadnuto: wall, window, floor, ceiling, door, sky, ceiling lamp, building, curtain, plant

Obr. 5.2: Příklad nesprávného odhadu anotace - Obrázek obsahoval anotace: ceiling, curtains, wall, table occluded, sofa, floor, plants, table. Odhadnuto:sky, building, window, door, trees, tree, person, person crop, grass, ground

# 6 ZÁVĚR

V této práci byla vytvořena aplikace, která využívá měření podobnosti na základě rozličných metod porovnání podobnosti obrazu pro úkon označování obrázků anotacemi. Aplikace nyní provádí přiřazení s velkým rozsahem přesnosti přiřazených anotací. Tato přesnost je založena na použité trénovací množině i její velikosti. Výsledná přesnost je také ovlivněna i konkrétním nastavením aplikace. První použitá metoda, metoda střední kvadratické chyby - MSE, byla implementována, ale shledána absolutně nevhodnou pro svou velmi velkou algoritmickou složitost, časovou náročnost a obecně nevhodnost pro daný účel. Dalším problémem, který byl shledán při použití této metody, byla velké zátěž operační paměti RAM počítače. Na standardním domácím PC[1] by trvalo samotné rozpoznání jednoho obrázku několik hodin, ne - li dní.

Dalším krokem bylo tedy přepracování do struktury úplně náhodných binárních rozhodovacích stromů, které se zprvu zdály jako vhodná struktura. Rozpoznání obrazu testovací struktury se provádí v rámci sekund. Pro rozpoznání podobnosti byla po MSE použita hlavně metoda porovnání histogramů. V téro realizaci hrála velkou roli náhoda, protože obrazy byly do stromu skládány zcela nahodile. Proto byla zvolena třetí, značně úspěšnější metoda - tvorba stromů na základě informačního zisku. Výpočet tohoto údaje nijak zásadně nezatížil celý algoritmus, ale přinesl značný řád do tvorby binárního rozhodovacího stromu.

V této práci probíhal i vývoj různých podobnostních metod použitých pro rozpoznání podobnosti. První byla zvolena již výše uvedená MSE, další metoda počítala s jednoduchými barevnými histogramy obrázku, avšak ani ta se neukázala dostačující. Posledním a hlavním krokem bylo využití podobnosti na základě knihovny LIRE, implementace rozhodování a budování rozhodovacích stromů na základě pravděbodnosti a informačního zisku.

Hlavním přínosem této práce je vytvoření funkční aplikace, která je schopna automaticky anotovat obrazy. Bylo dosaženo maximálně 32% úspěšnosti nad danou testovací množinou o velikosti 1000 obrazů. Druhotným přínosem této práce je rozšíření znalostí v oboru image processingu, prozkoumání různých algoritmů pro rozpoznání porobnosti a samotná implementace podobnostních algoritmů. V průběhu této práce byla vyhodnocena úspěšnost rozpoznání na základě databáze SUN2012. Vzhledem k rozdílnosti a odlišnosti obrazů v byla provedeno přidělení anotace relativně úspěšně, avšak v porovnání s původními anotace jsou výsledky nedokonalé, avšak použitelné.

Další rozvoj vzniklé aplikace by mohl obsahovat paměťové optimalizace apli-

[1] Personal Computer - osobní počítač

kace, optimalizace načítání testovací množiny popřípadě rozšíření aplikace za využití paralelních algoritmů. Možným rozšířením by určitě mohlo být i doplnění grafického rozhraní pro aplikaci, popřípadě napojení na výkonější vyhodnocovací nástroj. Vzhledem k tomu, že pro rozpoznání anotací je založeno na zpracování velké množiny obrazů, doporučoval bych použití aplikace na serverových stanicích nebo výkonějších desktopových stanicích. Provozování aplikace na standardním notebooku se ukázalo jako poměrně zdlouhavé a né úplně vhodné.

# LITERATURA


[1] Seghouane, A.-K. *"A Note on Image Restoration Using and MSE,"Signal Processing Letters, IEEE , vol.15, no., pp.61-64, 2008* [online]. [cit. 6. 5. 2001]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4418408&isnumber=4418381l>.

[2] Novak, C.L.; Shafer, S.A. *"Anatomy of a color histogram,"Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92* [online]. [cit. 15. 7. 1992]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=223129&isnumber=5817l>.

[3] Grassi, G.; Di Sciascio, E *"A new learning algorithm for pattern classification using cellular neural networks,"Circuits and Systems, 2001* [online]. [cit. 6. 5. 2001]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=921395&isnumber=19927l>.

[4] Hao Fu, Qian Zhang, and Guoping Qiu *"Random Forest for Image Annotation,"ECCV Conference, 2012* [online]. Dostupné z URL: <http://www.viplab.cs.nott.ac.uk/l>.

[5] Boiman, O.; Shechtman, E.; Irani, M *"In defense of Nearest-Neighbor based image classification,"Computer Vision and Pattern Recognition, 2008. CVPR 2008* [online]. [cit. 23. 7. 2008]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4587598&isnumber=4587335l>.

[6] Sampat, M.P.; Zhou Wang; Gupta, S.; Bovik, A.C.; Markey, M.K. *"Complex Wavelet Structural Similarity: A New Image Similarity Index,"Image Processing, IEEE Transactions on* [online]. [cit. 11. 2009]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5109651&isnumber=5286712>.

[7] Cortes, Corinna; and Vapnik, Vladimir N. *"Support-Vector Networks", Machine Learning, 20, 1995* [online]. [cit. 1995]. Dostupné z URL: <http://www.springerlink.com/content/k238jx04hm87j80g/>.

[8] Hearst, M.A.; Dumais, S.T.; Osman, E.; Platt, J.; Scholkopf, B. *"Support vector machines,"Intelligent Systems and their Applications, IEEE , vol.13, no.4* [online]. [cit. 1995]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=708428&isnumber=15361>.

[9] Jianxin Wu *"Efficient HIK SVM Learning for Image Classification,"Image Processing, IEEE Transactions on , vol.21, no.10* [online]. [cit. 11. 2012]. Dostupné

z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=708428&isnumber=15361>.

[10] Jovic M. and Thomas Seidl and Stejic Z. and Ira Assent *"Image Clustering and Retrieval Combining fixed/adaptive-binned Histograms and various Distance Functions", 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS 2004)* [online]. [cit. 3. 12. 2004]. Dostupné z URL: <http://dme.rwth-aachen.de/en/publications/2046>.

[11] Bai, C.; Zou, W.; Kpalma, K.; Ronsin, J. *"Efficient colour texture image retrieval by combination of colour and texture features in wavelet domain,"Electronics Letters , vol.48, no.23, pp.1463-1465* [online]. [cit. 8. 11. 2012]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6354230&isnumber=6354212>.

[12] Sean Moran and Victor Lavrenko *Optimal Tag Sets for Automatic Image Annotation. In Jesse Hoey, Stephen McKenna and Emanuele Trucco, Proceedings of the British Machine Vision Conference, pages 1.1-1.11. BMVA Press* [online]. [cit. 9. 2011]. Dostupné z URL: <http://dx.doi.org/10.5244/C.25.1>.

[13] Jianhua Li; Mingsheng Liu; Yan Cheng *"User Interest Model-based Image Retrieval Technique,"Automation and Logistics, 2007 IEEE International Conference on , vol., no., pp.2265-2269* [online]. [cit. 18. 9. 2011]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4338953&isnumber=4338503>.

[14] Ju Han; Kai-Kuang Ma *"Fuzzy color histogram and its use in color image retrieval,"Image Processing, IEEE Transactions on , vol.11, no.8, pp. 944- 952* [online]. [cit. 8. 2002]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1025168&isnumber=22043>.

[15] *Random trees* [online]. Dostupné z URL: <http://luc.devroye.org/trees.html>.

[16] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba *"SUN Database: Large-scale Scene Recognition from Abbey to Zoo", IEEE Conference on Computer Vision and Pattern Recognition, 2010.* [online]. [cit. 2010]. Dostupné z URL: <http://groups.csail.mit.edu/vision/SUN/>.

[17] A. Barriuso and A. Torralba *"Notes on image annotation"* [online]. [cit. 2010]. Dostupné z URL: <http://groups.csail.mit.edu/vision/SUN/>.

[18] *"An Open Source Java Content Based Image Retrieval Library "* [online]. [cit. 2013]. Dostupné z URL: <http://www.semanticmetadata.net/lire>.

[19] Jing Huang; Kumar, S.R.; Mitra, M.; Wei-Jing Zhu; Zabih, R. *"Image indexing using color correlograms,"Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on , vol., no., pp.762,768, 17-19 Jun 1997* [online]. [cit. 1997]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=609412&isnumber=13322>.

[20] Chatzichristofis, S.A.; Boutalis, Y.S. *"FCTH: Fuzzy Color and Texture Histogram - A Low Level Feature for Accurate Image Retrieval,"Image Analysis for Multimedia Interactive Services, 2008. WIAMIS '08. Ninth International Workshop on , vol., no., pp.191,196, 7-9 May 2008* [online]. [cit. 2008]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4556917&isnumber=4556857>.

[21] Savvas A. Chatzichristofis; Yiannis S. Boutalis *"CEDD: Color and Edge Directivity Descriptor:A Compact Descriptor for Image Indexing and Retrieval"* [online]. [cit. 2008]. Dostupné z URL: <http://link.springer.com/content/pdf/10.1007%2F978-3-540-79547-6_30.pdf>.

[22] Tamura, Hideyuki; Mori, Shunji; Yamawaki, Takashi *"Textural Features Corresponding to Visual Perception,"Systems, Man and Cybernetics, IEEE Transactions on , vol.8, no.6, pp.460,473, June 1978* [online]. [cit. 1978]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4309999&isnumber=4309992>.

[23] Zagoris, K.; Chatzichristofis, S.A.; Papamarkos, N.; Boutalis, Y.S., *"Automatic Image Annotation and Retrieval Using the Joint Composite Descriptor,"Informatics (PCI), 2010 14th Panhellenic Conference on , vol., no., pp.143,147, 10-12 Sept. 2010* [online]. [cit. 2010]. Dostupné z URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5600451>.

# SEZNAM PŘÍLOH

- DVD 1 - Aplikace, dokumentace, zdrojové soubory
  - Elektronická příloha 1 - Vygenerovaná dokumentace aplikace ve fomátu PDF - adresář DOKUMENTACE
  - Elektronická příloha 2 - Vygenerovaná dokumentace aplikace ve fomátu HTML (spouštěcí soubor index.html) - adresář DOKUMENTACE/HTML
  - Elektronická příloha 3 - Zkompilovaná aplikace DP - obsahuje připravené datasety, trénovací DB ESP_GAME a testovací množiny pro ESP_GAME a SUN2012 - adresář DP
  - Elektronická příloha 4 - zdrojové soubory aplikace - adresář DP12PodobnostObrazu
- DVD 2 - trénovací DB SUN 2012