Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Automatická anotace obrázku

Plzeň 2016 Miroslav Liška

Místo této strany bude zadání práce.

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Ladislavu Lencovi, Ph.D. za jeho podporu, cenné rady a vstřícný přístup během celé mojí práce. V neposlední řadě také děkuji mojí rodině za podporu v průběhu celého studia.

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 5. května 2016

Miroslav Liška

Abstract

Automatic image annotation

This thesis deals with the problem of automatic image annotation (AIA). The goal of AIA is to assign one or more annotations (keywords) to an image. The key task in solving AIA is obtaining image features. The thesis provides an overview of methods used for this task. Subsequently, two such methods LBP and SIFT are described in detail. These methods were chosen for implementation because they achieve excellent results both in AIA and other areas of pattern recognition. The thesis also describes methods for classification of descriptors. For classification, the method k-nearest vectors was chosen and implemented. The resulting application is a system capable of testing methods for automated annotation on commonly available databases. Implemented methods were tested on several standard data sets and obtained results are then compared with results from literature.

Abstrakt

Automatická anotace obrázku

Tato bakalářská práce se zabývá automatickou anotací obrázků (AIA). Cílem této úlohy je přiřadit obrázku jednu nebo více anotací (klíčových slov). Jednou z nejdůležitějších částí problematiky AIA je získání příznaků z obrázku. Metody pro získání příznaků, používaných v oblasti AIA, jsou na úvod teoreticky shrnuty. Detailně práce popisuje metody LBP a SIFT. Tyto metody dosahují vynikajících výsledků i v dalších oblastech rozpoznávání vzorů, a proto byly vybrány pro implementaci. Práce dále popisuje možnosti klasifikace získaných deskriptorů. Pro klasifikaci byla vybrána a implementována metoda k-nejbližších vektorů. V rámci práce byl vytvořen systém, který umožní testování metod pro automatickou anotaci na běžně dostupných databázích. Implementované metody byly otestovány na několika standardních datasetech a jejich výsledky byly srovnány s literaturou.

Obsah

1 Úvod

Vzhledem k velkému rozkvětu digitálních technologií vzniká stále větší množství vizuálních dat. Ta jsou dnes stejně běžná jako data textová. Z tohoto důvodu je zde urgentní potřeba nalezení nástrojů, které umožní efektivní správu vizuálních dat a rovněž poskytnou snadné a přesné vyhledávání. Díky tomu vznikly úlohy jako je indexování obrázků, vyhledávání obrázků na základě obsahu a také automatická anotace.

Počítače nemají schopnost vnímat okolní svět stejně jako lidé. Přesněji řečeno, počítač vidí obrázek jako množinu binárních informací. Cílem člověka je naprogramovat počítač tak, aby na základě těchto informací byl schopen určit, jaké objekty se na obrázku vyskytují.

Automatická anotace obrázku je proces, ve kterém jsou k obrázku automaticky přiřazena metadata, která obsahují klíčová slova, například (dům, zahrada, strom, plot).

Proces automatické anotace obrázku se skládá ze dvou částí. Jedním z nejdůležitějších kroků sémantického porozumění je extrakce klíčových příznaků z obrázku. Správně zvolená reprezentace obsahu výrazně zlepšuje výkon sémantického učení. Po extrakci klíčových příznaků může přijít samotná anotace. V prvním kroku probíhá natrénování klasifikátoru, kdy je použita množina již anotovaných obrázků. Díky tomu se k extrahovaným příznakům obrázku přiřadí slova popisující jejich význam. V druhé fázi probíhá anotace nových obrázků. Nový obrázek je nejprve zpracován a jsou z něho extrahovány příznaky. Na základě příznaků jsou pak pomocí natrénovaného klasifikátoru přiřazena klíčová slova.

Cílem práce je navržení a implementace softwaru umožňujícího automatickou anotaci obrázků. V rámci toho budou prostudovany metody využívané pro AIA. Práce nebude využívat nejjednodušší metody extrakce příznaků, ale budou použity metody LBP a SIFT, které se již osvědčily v rámci jiné problematiky, například rozpoznávání obličejů. Důležitou částí je také vhodná volba testovacích dat. Výběr dat bude volen tak, aby byl svojí charakteristikou co nejvíce podobný datům od ČTK a aby zároveň data byla použita pro srovnání v jiných pracích. Na základě experimentů na testovacích datech bude vybraná nejvhodnější metoda, která se použije na poskytnutá data od ČTK.

2 Extrakce příznaků a reprezentace obrázku

Protože obrázek je nestrukturované pole pixelů, prvním krokem sémantického porozumění je extrakce příznaků z těchto pixelů. Správně zvolená reprezentace obsahu obrázku výrazně zlepšuje výkon sémantického učení.

Příznaky mohou být nízkoúrovňové nebo vysokoúrovňové. Při klasifikaci obrázků jsou většinou používany nízkoúrovňové techniky. V současné době jsou v metodách pro získávání příznaků využívány dva přístupy – globálně nebo lokálně tvořené příznaky. Podle [13] se trend ubírá směrem k využívání lokálních příznaků. Použití lokálních příznaků předpokládá segmentaci obrázku na jednotlivé regiony. Globální příznak je vypočítáván z celého obrázku.

2.1 Segmentace

Segmentace obrázku je obvykle prvním krokem k extrakci lokálních příznaků.U problému automatické anotace obrázků se běžně využívají následující metody: mřízkově založená, shlukově založená, obrysově založená, modelově založená, grafově založená a metoda založená na postupném růstu regionů. Nyní bude každá z nich krátce popsána.

V mřížkově založené metodě je obrázek rozdělen do bloků, ze kterých jsou následně vypočítávány příznaky. Blokově založené postupy jsou charakteristické nízkou výpočetní složitostí, nicméně bloky obvykle obsahují i část vizuálně odlišných objektů a proto taková segmentace nemůže vyhovovat vždy.

Shlukově založené metody jsou používány ke shlukování pixelů do skupin, kdy každá skupina identifikuje region. Běžnou praxí je rozdělení obrázku do bloků pixelů 4x4. Z každého bloku jsou extrahovány příznaky a ty jsou následně využity pro shlukování. Každý shluk obsahuje pouze objekty, které jsou si podobnější, než objekty z jiné skupiny. Hlavním problémem tohoto přístupu je nutnost předdefinování počtu segmentů. Nevhodná volba počtu segmentů může zapříčinit nesprávné výsledky. Volba počtu segmentů se provádí heuristicky.

Hlavní myšlenkou obrysově založené segmentace je rozvoj křivek kolem objektu, jejichž cílem je shoda s hranicemi objektu. Základním problémem

v tomto postupu je závislost na přesnosti detekce hran, pokud je kolem objektu šum. Z tohoto důvodu je postup aplikovatelný pouze na specifické domény, jako například nástroje pro zpracování obrázku.

Segmentačně založené algoritmy staví na statistickém modelu. Široce využíván je systém BlobWorld [2], kdy je každý pixel reprezentován osmidimensionálním vektorem barev, textu a pozic. Pixel obrázku je následně modelován jako náhodná proměnná s Gaussovým rozdělením. Počet regionů a parametry Gaussova rozdělení jsou vypočítávány s využitím algoritmu Maximalizace očekávání (EM). Jakmile je model parametrů nalezen, je s využítím pravděpodobností vypočítáván vztah pixel-region. Vztah pixel-region je využíván k určení obrázkové segmentace. Hlavním problémem je vysoká výpočetní náročnost.

Grafově založený algoritmus segmentace, známý také jako normalizovaný řez (NCut), reprezentuje obrázek jako graf, kde vrcholy jsou pixely a hrany jsou váhy reprezentující obsahové podobnosti mezi pixely. Myšlenkou je rozdělit vrcholy grafu do disjunktních množin tak, že celková podobnost mezi různými množinami je minimalizována. Každá množina je chápána jako region. Nevýhodou je výpočetní náročnost pro získání optimálního rozdělení.

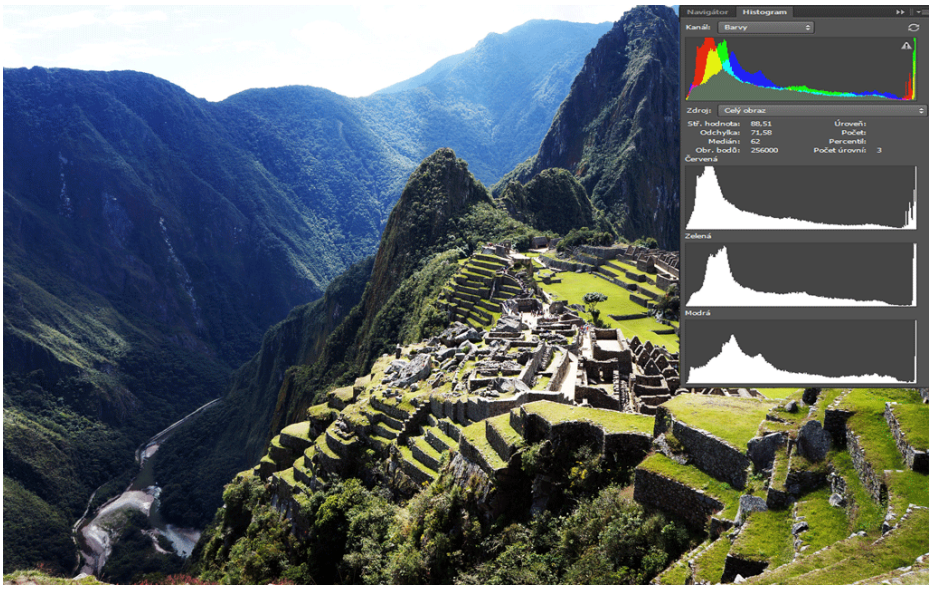
Široce využíván je algoritmus s postupným růstem regionů. Prvním krokem je vybrání množiny bodů (seed), které jsou vybrány na základě nějakého kritéria (například stupeň šedi). Následně je z těchto bodů procházeno okolí a do regionu přidávany pouze ty body, které splňují kritérium pro náležitost v daném regionu.

2.2 Barevné příznaky

Barva je jednou z nejdůležitějších součástí obrázku. Barevné příznaky jsou informace získané z barevného prostoru či modelu. Barevnými prostory jsou například RGB, LUV, HSV, HMMD.

Jakmile je barevný prostor specifikován, jsou z obrázku či regionu extrahovány barevné příznaky. Podle [13] patří mezi důležité barevné příznaky barevný histogram, barevný moment, vektor koherence barev a korelogram. Také MPGE-7 standardizuje počet barevných příznaků zahrnující deskriptor dominantní barvy a deskriptor škálovatelnosti barev.

V tabulce 2.1 na stránce 14 jsou srovnány výhody a nevýhody zmíněných metod.



Obrázek 2.1: Histogram

2.2.1 Barevný histogram

Barevný histogram popisuje barevné rozložení obrázku. Kvantizuje barevný prostor do různých množin a zaznamenává, kolik pixelů spadá do určité množiny. Tento přístup je robustní vůči translaci a rotaci. Avšak barevný histogram nepostkytuje prostorové informace a dimenze histogramu bývá obvykle vysoká.

Na obrázku 2.1 je příklad histogramu. Z grafů můžeme vyčíst, že odstíny červené, zelené a modré jsou na obrázku především v tmavém odstínu.

2.2.2 Barevné momenty

Barevné momenty jsou jeden z nejjednodušších příznaků. Základem je předpoklad, že rozložení barev v obrázku může být reprezentováno jako rozdělení pravděpodobnosti. Momenty běžně počítají s průměrnou barvou v obrázku, standartní odchylkou a šikmostí. Vlastnosti se získávají buď z regionu, nebo z celého obrázku.

2.2.3 Vektor koherence barev

Vektor koherence barev (CCV) zahrnuje prostorovou informaci do základního barevného histogramu. Množiny histogramu rozděluje na dvě komponenty – koherentní a nekoherentní. Koherentí komponenta zahrnuje ty pixely, které jsou prostorově spojené. Naopak nekoherentní komponenty zahrnují

izolované pixely. Obvykle mívá větší výkon než barevný histogram, nicméně dimenze je dvakrát větší než u konvenčního histogramu.

2.2.4 Barevný korelogram

Barevný korelogram maticově charakterizuje rozdělení barevných párů v obrázku. Na korelogram můžeme nahlížet jako na 3D histogram, kde první dvě dimenze reprezentují barvu nějakého páru pixelů a třetí dimenzí je jejich vzdálenost. Výkon korelogramu je lepší než u histogramu a vektoru koherence barev, nicméně za cenu větší výpočetní náročnosti.

2.2.5 MPEG-7 deskriptory

Mezi MPEG-7 barevné deskriptory patří deskriptor škálovatelnosti barev (SCD). SCD je v podstatě histogram v HSV barvách a navíc má lepší škálovatelnost. Stejně jako histogram, SCD také nezahrnuje prostorové informace.

Deskriptor dominantní barvy (DCD) je také variací histogramu. DCD vybere malý počet barev z nejvyššího zásobníku histogramu. Doporučený počet barev pro reprezentaci regionu je dle MPGE-7 1–8. Oproti tradičnímu histogramu je DCD používan na regiony namísto celého obrázku.

2.3 Příznaky tvarů

Tvary jsou důležité pro člověka k identifikaci a rozpoznávání objektů reálného světa. Zhang a Lu [13] klasifikují techniky pro extrakci tvarů na dvě majoritní skupiny: Obrysově založené a regionově založené metody.

Obrysově založené metody určují příznaky tvarů pouze z hranic daného tvaru, zatímco regionově založené metody extrahují vlastnosti z celého regionu. Techniky založené pouze na tvaru jsou velmi náchylné na jakýkoliv šum, tedy jakýkoliv šum znatelně poškodí tvar objektu. Z tohoto důvodu se častěji používají regionově založené příznaky tvarů. Příznaky obvykle zahrnují momenty, kruhovitost a výstřednost. Jednotlivé příznaky nejsou příliš robustní a proto se většinou skládají pro vytvoření efektivnějšího deskriptoru.

Metoda	Výhody	Nevýhody
Histogram	Jednoduchý k vypočítání, intuitivní	Vysoká dimenze, žádné prostorové informace, citlivý na šum
Barevné momenty	Kompaktní, robustní	Nedostatečné pro popis všech barev, žádné prostorové informace
Vektor koherence barev	Prostorové informace	Vysoká dimenze, velká výpočetní náročnost
Korelogram	Prostorové informace	Velmi vysoká výpočetní náročnost. Citlivé na šum, rotaci a škálování
Desktriptor dominantní barvy	Kompaktní, robustní. Vnímání smyslu	Potřebuje post-processing pro prostorové informace
Deskriptor škálovatelnosti barev	Kompaktní dle potřeby, škálovatelnost	Žádne prostorové informace

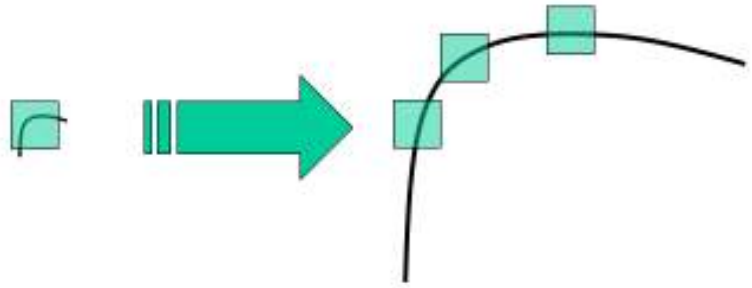
Tabulka 2.1: Srovnání metod pro extrakci příznaků z barev

2.4 Scale-invariant feature transform

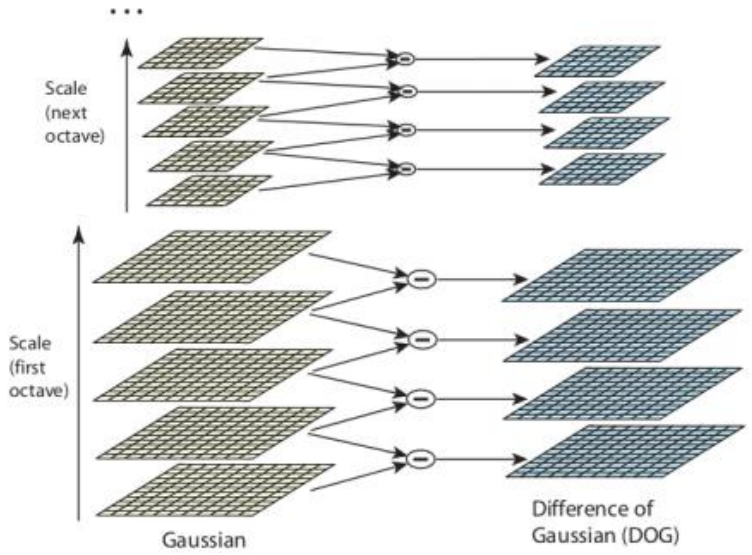
Scale-invariant feature transform (SIFT) je algoritmus využívaný pro nalezení významných bodů v obrázku a vytvoření příznaků v těchto bodech. Algoritmus je popsán v článku [9]. SIFT algoritmus se skládá z hlavních 4 kroků.

Prvním krokem je nalezení extrémů v tzv. „scale-space“. Na obrázku 2.2 je znázorněno, jak se hledají klíčové body pro různé škály. Na každé úrovni škálování je aplikován Laplacián Gaussiánu (LoG) s různou směrodatnou odchylkou σ u Gaussova rozdělení. Tímto způsobem můžeme získat lokální maxima napříč škálami a prostor, který nám dá množinu (x, y, σ) hodnot, které reprezentují potenciální klíčové body na pozici (x, y) se škálou σ.

Tento přístup je vypočetně náročný. Z tohoto důvodu SIFT algoritmus pouzívá rozdíl Gaussiánů (DoG), jenž je aproximací LoG. Rozdíl Gaussiánů je získán jako rozdíl rozmazání obrázku pomocí Gussova filtru s rozdílným σ. Tento proces se provede pro různé oktávy Gaussovy pyramidy obrázku, viz obrázek 2.3.

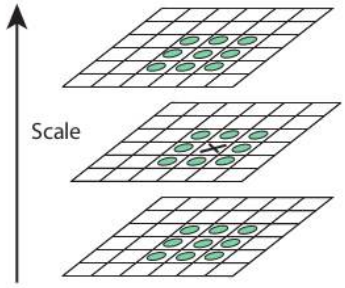


Obrázek 2.2: Na obrázku je znázorněn postup hledání potenciálních klíčových bodů pro algoritmus SIFT. Zdroj: readthedocs.org



Obrázek 2.3: Vytvoření rozdílu Gauusiánů napříč oktávami.
Zdroj: readthedocs.org

Jakmile je nalezen DoG, obrázky jsou prohledávány pro lokální extrémy napříč škálami a prostorem. Například jeden pixel v obrázku je porovnán s 8 sousedy ve stejné úrovni a odpovídajícími 9 pixely v předchozí i následující vrstvě (celkem 26 bodů). Pokud se jedná o lokální extrém, našli jsme potenciální klíčový bod. Znamená to, že klíčový bod je nejlépe reprezentovaný v dané škále, viz obrázek 2.4.



Obrázek 2.4: Zdroj: readthedocs.org

Jakmile jsou nalezeny potenciální klíčové body, musí být eliminovány nejméně kvalitní z nich pro získání přesnějších výsledků. Pro získání přesnějšího umístění extrému je využit Taylorův rozvoj škálovaného prostoru a pokud je intenzita v tomto extrému menší než prahová hodnota, je zamítnut.

DoG také častěji nachází hrany, které také musí být odebrány. Konceptem je to stejný problém jako u Harrisova detektoru rohů. Je použit Hessián (H), který je využit pro vypočet hlavního zakřívení. Z Harrisova detektoru rohů je známo, že pro hrany je jedno vlastní číslo větší než ostatní. Pokud je poměr vlastních čísel větší než je prahová hodnota, klíčový bod je zamítnut. Nyní zůstaly pouze klíčové body s velkou mírou informace.

Následně je potřeba určit orientaci každého klíčového bodu. Tím se získá invariance vůči otočení obrázku. Podle škály se projdou sousední body kolem klíčového bodu a v této oblasti se spočítá gradient a směr. Je vytvořen histogram orientace s 36 úrovněmi, které pokrývají 360 stupňů. Histogram je ohodnocen gradientem a váhován pomocí Gaussova rozdělení s σ rovnému 1.5 krát škále klíčového bodu. Je vybrán nejvyšší prvek v histogramu a další prvky nad 80% pro výpočet orientace.

Nyní může být vytvořen deskriptor. Vezme se 16x16 okolí klíčového bodu, které je rozdělené na 16 bloků velikosti 4x4. Pro každý pod blok je vytvořen histogram orientací. Ve výsledku tedy vznikne 128 hodnot. Ty tvoří vektor, který je použit jako deskriptor klíčového bodu.

2.5 Local binary pattern

Local binary pattern (LBP) je také jedním z algoritmů pro získání či popsání příznaků obrázku.

Podle [5], prvotním krokem je převod barev obrázku na stupeň šedi, tedy každý pixel je reprezentován svojí intenzitou. Následně se pro každý pixel obrázku porovná jeho intenzita s intenzitou sousedních pixelů. Jejich způsob označení je demonstrován na obrázku 2.5. Označíme-li tedy $p_0, p_1, p_2 \dots p_7$ jako intezity sousedních bodů a p_c jako intenzitu centrálního pixelu, pak pro funkci $LBP(p_c)$ platí 2.1. Na výsledné hodnoty z 2.1 se ještě aplikuje funkce 2.2. Nejlépe celý proces demonstruje obrázek 2.6

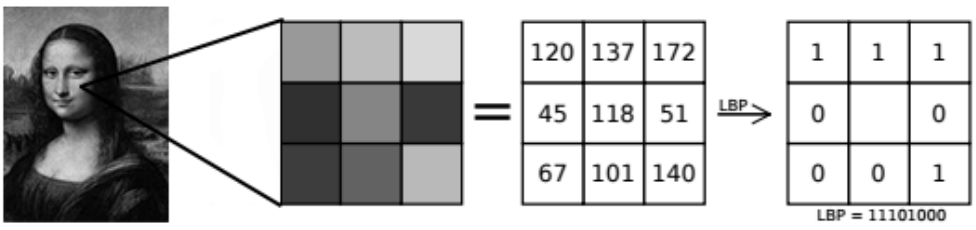
$$LBP(p_c) = \sum_{i=0}^{7} s(p_i - p_c)2^i \tag{2.1}$$

$$s(x) = \begin{cases} 1, & x \leq n \\ 0, & x < 0 \end{cases} \tag{2.2}$$

p_0	p_1	p_2
p_7	p_c	p_3
p_6	p_5	p_4

Obrázek 2.5: Ukázka označení sousedních pixelů $p_0, p_1, p_2 \dots p_7$ centrálního pixelu p_c. Zdroj: [5]

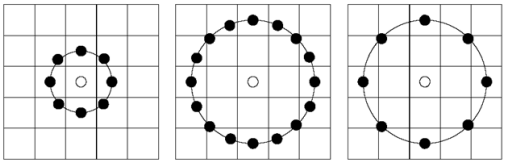
.



Obrázek 2.6: Ukázka procesu LBP. Zdroj: [5]

.

Pro invarianci LBP vůči škálování byla navržena úprava. Jedná se o zobecněnou kruhovou formu $LBP(N, R)$, kde N značí počet sousedních bodů a R představuje poloměr kružnice, na kterém jsou tyto sousední body rovnoměrně rozpostřeny, viz. obrázek 2.9.



Obrázek 2.7: Příklad zobecněného kruhového operátoru $LBP(8,1)$, $LBP(16,2)$ a $LBP(8,2)$. Zdroj: what-when-how.com

Vektor reprezentující obrázek se následně získá spočítáním histogramu, kdy se pro všechny pixely obrázku určí LBP hodnota. Ta při klasickém LBP nabývá hodnot 0 – 255. Histogram má v tomto případě šířku intervalu 256. Sloupce histogramu reprezentují četnost LBP hodnoty. Výsledný histogram se bere jako deskriptor obrázku. Histogram lze počítat buď z celého obrázku nebo lze obrázek rozdělit na regiony a pro každý region spočítat histogram zvlášť. Pokud se obrázek rozdělí na 2 x 2 regiony, bude reprezentován čtyřmi vektory.

2.6 Bag of Features

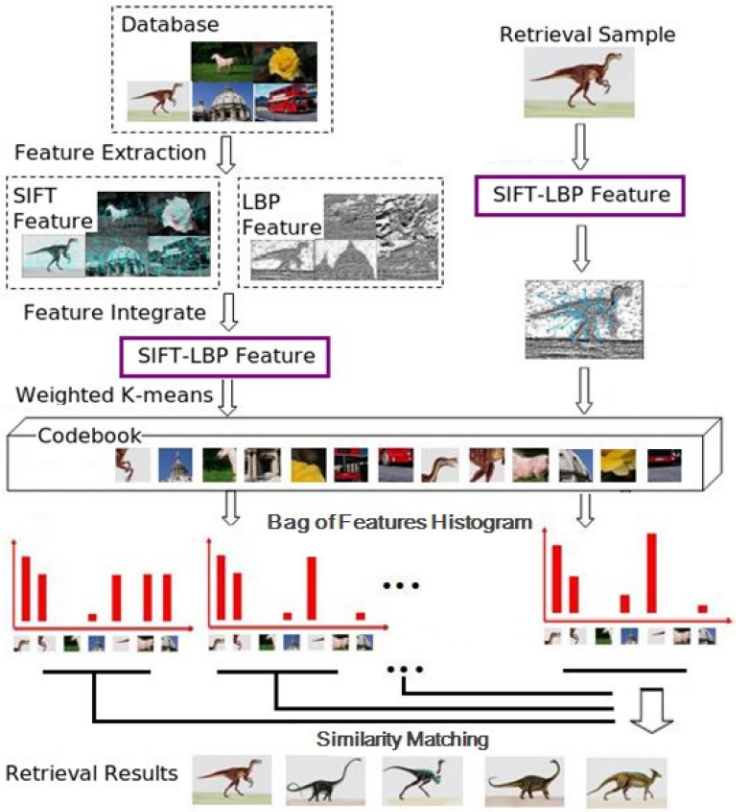
Bag of Features (BoF) je podle [12] metoda inspirována konceptem Bag of Words (BoW), která se využívá pro zpracování textu. Při klasifikaci dokumentu je výsledkem histogram výskytu daných slov.

Pro reprezentaci obrázku za použití BoF modelu může být s obrázkem zacházeno jako s dokumentem. Nejprve se z obrázku extrahují příznaky a převedou se na vektory. V dalším kroku se tyto vektory převedou na tzv. slovník. To je možné například za využití k-means algoritmu přes všechny extrahované vektory. Slova jsou následně definovány jako středy získaných clusterů. Následně je každý příznak obrázku namapován na určité slovo. Obrázek je následně reprezentován jako histogram slov.

Příliš málo clusterů může zapříčinit to, že některé příznaky budou popsány nesprávným slovem. Na druhou stranu příliš mnoho slov může zapříčinit přeučení.

2.7 Bag of SIFT Features

V tomto přístupu se z trénovací množiny vypočítají pro každý obrázek jeho deskriptory. Jak již bylo zmíněno v 2.4, deskriptor je vektor o 128 hodnotách. Ze všech vektorů ze všech obrázků trénovací množiny se provede shlukování



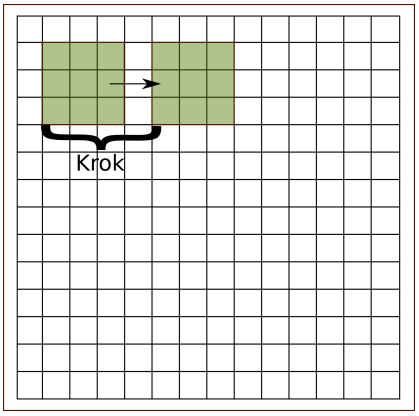
Obrázek 2.8: Proces extrakce příznaků s využitím LBP a SIFT a jejich následné využití pro hledání podobných obrázků. Zdroj: [12].

například metodou k-means o počtu shluků n. Výsledkem je n vektorů o 128 hodnotách reprezentující středy jednotlivých shluků.

Nyní se vytvoří Bag of Features. Pro aktuální obrázek se spočítají spočítají SIFT deskriptory. Následně se testuje pro každý deskriptor, do kterého shluku patří. Tyto informace se ukládají v podobě histogramu o velikosti n, tedy pokud deskriptor spadá například do shluku 1, inkrementujeme hodnotu histogramu na této pozici o 1. Tímto způsobem je možné projít jak testovací, tak i trénovací množiny a získat tak histogramy reprezentující jednotlivé obrázky. Výsledný histogram je deskriptorem obrázku.

2.8 Bag of LBP Features

Jak již zbylo zmíněno v klasickém LBP (viz sekce 2.5), výsledné histogramy lze získat buď z celého obrázku, nebo lze obrázek rozdělit na regiony. V metodě Bag of LBP Features se vždy vezme oblast o velikosti x a y. Pro oblast se spočítá histogram a okénko se posune o určitý krok k nejprve vodorovně a pokud se dojede na konec řádku, je okénko posunuto o krok k i svisle a řádek se zpracovává znovu od začátku. Tímto způsobem se získá z obrázku množina vektorů.



Obrázek 2.9: Ukázka posunu okénka

Tento proces se aplikuje na všechny obrázky z trénovací množiny. Ze všech vektorů ze všech obrázků trénovací množiny se provede shlukování s počtem shluků n. Výsledkem je n vektorů o 256 hodnotách reprezentující středy jednotlivých shluků.

Nyní se vytvoří Bag of Features. Pro aktuálně zpracovávaný obrázek se spočítají výše zmíněné deskriptory (histogramy). Následně se testuje pro každý deskriptor, do kterého shluku patří. Tyto informace se ukládají v podobě histogramu o velikosti n, tedy pokud deskriptor spadá například do shluku

1, inkrementujeme hodnotu histogramu na této pozici o 1. Tímto způsobem je možné projít jak testovací, tak i trénovací množiny a získat tak histogramy reprezentující jednotlivé obrázky. Výsledný histogram je deskriptorem obrázku. Obrázek je tedy reprezentován histogramem příslušností histogramů k patřičnému shluku.

3 Automatická anotace

Jakmile jsou obrázky reprezentovány pomocí příznaků získaných buď z globálních nebo lokálních metod, může být naučen klasifikační model na množině anotovaných obrázků. Všeobecně se používají 2 typy AIA přístupů. První přístup je klasifikace do jedné třídy využívající konvenční klasifikační metody. Druhým přístupem je klasifikace do více tříd využívající Bayesovské metody. Tyto přístupy budou nyní popsány v následujících kapitolách.

3.1 Klasifikace do jedné třídy s využitím binární klasifikace

V tomto přístupu jsou příznaky obrázku přivedeny do konvenčního binárního klasifikátoru. Výstupem klasifikátoru je sémantický koncept, který je využíván pro obrázkovou anotaci. Myšlenka klasifikace do jedné třídy je ekvivalentem anotace pomocí shlukování. Tato metoda provede shlukování a následně je každému obrázku z příslušného shluku přiřazena stejná anotace. V tomto přístupu jsou obvyklými nástroji support vector machines (SVM), umělé neuronové sítě (ANN) nebo rozhodovací strom (DT).

3.1.1 Anotace s využitím support vector machines

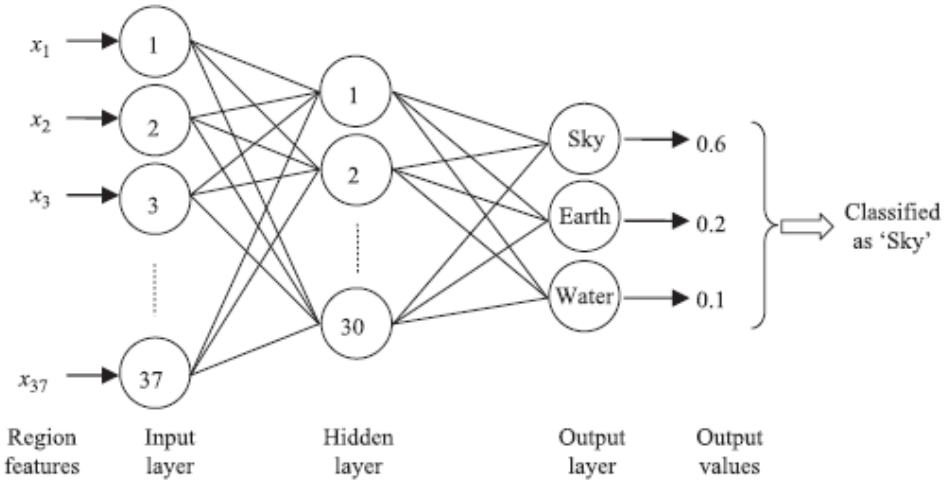
SVM klasifikátor hledá nadrovinu z trénovací množiny vzorků, tak aby je rozdělil. Každý trénovací vzorek je reprezentován vektorem příznaků a anotací. Nadrovina je určena takovým způsobem, aby rozdělila největší část vzorků stejné třídy od ostatních.

3.1.2 Anotace s využitím umělých neuronových sítí

Umělé neuronové sítě (ANN), které mají schopnost se naučit znalosti z trénovací množiny a následně je využít pro klasifikaci nového vzorku. ANN se skládá z několika vrstev vnitřně propojených uzlů (neurony). První vrstva je vstupní vrstvou, která má počet neuronů rovný dimenzi vstupního vzorku. Počet neuronů ve výstupní vrstvě je roven počtu tříd. Výběr počtu skrytých vrstev a počtu neuronů v každé skryté vrstvě jsou běžnými problémy v ANN přístupech. Spojovací hrany mezi neurony jiných vrstev jsou spojeny s určitými váhami. Každý neuron pracuje jako procesový element a je řízený

aktivační funkcí, která generuje výstupy neuronů na základě váhy spojovacích hran a výstupy neuronů z předchozí vrstvy. Během tréninku se ANN učí váhy hran, aby byla celková učící chyba minimalizovaná.

Obecně řečeno, do neuronové sítě vstoupí nějaký vektor a výstupem ze sítě je třída, do které byl vektor klasifikován.



Obrázek 3.1: Neuronová síť. Vstupní vektor má 37 parametrů a bude klasifikován do jedné ze 3 tříd – obloha, země, voda. Zdroj: [13].

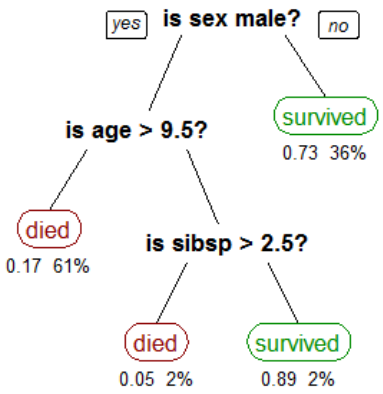
3.1.3 Anotace s využitím rozhodovacího stromu

Rozhodovací strom (DT) je vícestupňové rozhodování. V závislosti na počtu rozhodnutí v každém vnitřním uzlu, může být nazýván binárním stromem nebo n-nárním. Na rozdíl od odstatních modelů klasifikace, pro které je obtížné popsat vztah mezi vstupem a výstupem, tak v DT lze vstupní a výstupní vtah vyjádřit s pomocí lidsky srozumitelných pravidel, například „pokud-tak“ pravidlo.

Cílem je vytvoření modelu, který předpovídá hodnotu výsledné proměnné na základě několika vstupních proměnných.

DT je natrénován s využítím množiny anotovaných vzorků. Vzorky jsou zastoupeny řadou atributů. Během učení je DT vytvářen rekurzivně rozdělením trénovacích vzorků do disjunktních množin a pokaždé, když se vzorky rozdělí, atribut používáný pro dělení se odstraní. Postup se opakuje, dokud vzorky nepatří do jedné skupiny, nebo pokud strom dosáhl své maximální hloubky, když nezbyl žádný atribut, podle kterého by šlo dále dělit.

Pro označení nového vzorku, strom je procházen od kořenového uzlu



Obrázek 3.2: Strom demonstruje přežití pasažerů na Titanicu („sibsb“ je počet na příbuzných na palubě). Čísla pod listem představují pravděpodobnost jevu a procento výskytu tohoto jevu. Zdroj: en.wikipedia.org

k listům pomocí hodnot atributů nového vzorku. Výsledkem je list, kam daný vzorek dosáhne. Příklad DT je ukázán v 3.2

3.2 Klasifikace do více tříd s využitím Bayesovských metod

Rozdílem oproti binárním klasifikačním přístupům je, že metody anotují snímek s využitím více sémantických kategorií. Pojem klasifikace do více tříd souvisí s tzv. „multi-instance“ učením. Toto spojení se označuje multi-instance multi-label učení (MIML).

Obrázek je anotován určitým výrazem, pokud některý z příznaků obrázku je s významem spojený. V důsledku toho může být obrázek anotován více výrazy. Typický MIML je získán použitím pravděpodobnostních nástrojů, jako jsou například Bayesovy metody. Bayesovy metody pracují na základě hledání posteriorní pravděpodobnosti, zda obrázek patří do jakékoli konkrétní koncepce, vzhledem k pozorování určitých znaků z obrázku či regionu. Nechť $I_1, I_2, ..., I_n$ je množina obrázků z množiny sémantických tříd $c_1, c_2, ..., c_n$. Bayesovy modely se snaží určit zadní pravděpodobnost z podmíněné apriorní pravděpodobnosti. Obrázek je definován příznakovým vektorem x. Mějme apriorní pravděpodobnosti $p(c_i)$ a podmíněnou pravděpodobnost $p(x \mid c_i)$, pak se pravděpodobnost, že obrázek I náleží do třídy c_i se spočítá jako 3.1

$$p(c_i \mid x) = \frac{p(c_i \mid x)}{p(x)} \tag{3.1}$$

Protože rozdělení $p(x)$ je obvykle uniformní pro všechny třídy, třída obrázku I může být určena využitím kritéria maximalizace posteriorní pravděpodobnosti 3.3

$$\hat{c} = \arg\max_{c_i} p(c_i|\mathbf{x}) \approx \arg\max_{c_i} \{p(\mathbf{x}|c_i)p(c_i)\}$$

Obrázek 3.3: Maximalizace a výpočet podmíněné pravděpodobnosti.
Zdroj: [13].

Varianty Bayesových modelů se liší podle toho, jak modelují podmíněnou pravděpodobnost $p(x \mid c_i)$. Obecně existují dva typy přístupů: Bezparametrický přístup a parametrický přístup.

3.2.1 Bezparametrický přístup

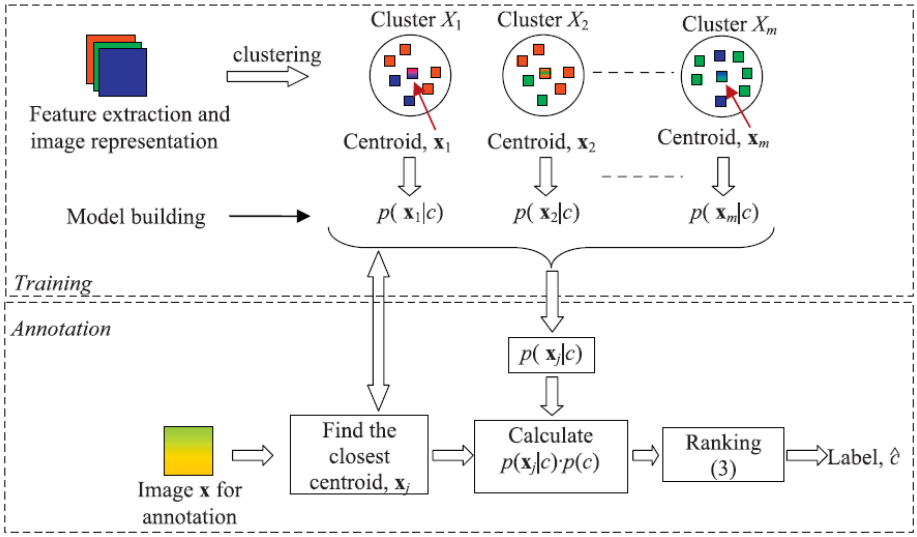
V tomto přístupu jsou podmíněné pravděpodobnosti vypočteny bez předchozího předpokladu o rozdělení příznaků obrázku. Skutečné rozdělení příznaků je naučeno z příznaků trénovací množiny. V praxi jsou příznaky obrázku nejprve kvantizovány do shluků. Následně jsou příznaky nahrazeny centroidem shluku. Nalezením frekvence vzorků náležící třídě je spočítaná podmíněná pravděpodobnost pro každou z tříd. Například pokud je x_j nejbližší centroid přiznakového vektoru x, pak platí 3.2

$$p(x \mid c) \approx p(x_j \mid c) = \frac{\text{Počet vzorků v } x_j, \text{ které jsou z třídy } c}{\text{Počet vzorků z třídy} c} \tag{3.2}$$

Kompletní anotační proces této metody je ukázán na obrázku 3.4. Z daného obrázku jsou extrahovány příznaky a jsou porovnány s hodnotami shluků. Podmíněné pravděpodobnostní modely odpovídající vybranému centru clusteru jsou pak použity k výpočtu posterior pravděpodobností. MAP kritérium 3.3 se pak používá k anotaci nového snímku.

3.2.2 Parametrický přístup

V tomto přístupu je předpokládáno, že prostor příznaků bude následovat určitý typ známého kontinualního rozdělení. Proto je podmíněná pravděpodobnost $p(x \mid c)$ modelovaná za užití přiznakového rozdělení. Hlavní proces



Obrázek 3.4: Bayesův anotační model. Zdroj [13]

je podobný tomu, ukázanému v obrázku 3.4. Příznaky nebo regiony jsou nejprve shlukovány a kvantizovány. Model podmíněné pravděpodobnosti je následně spočítán pro každý shluk. Podmíněná pravděpodobnost $p(x \mid c)$ je obykle modelovaná jako mnohovariantní Gassovo rozdělení, jako je ukázano v obrázku 3.5, kde d je dimenze přiznaků, $\bar{x}$ a $\sum$ jsou průměry a dále se zde pracuje s kovariantní maticí, která ke spočítaná z trénovacích vektorů přiznaků, naležící tříde c.

Paremetrický přístup s sebou nese řadu problémů. Tento přístup obvykle předpokládá konkrétní rozdělení příznaků, což nemusí být vždy pravdou. Navíc parametr modelu jsou obvykle počítány optimalizačními metodami, které většinou nemusí konvergovat a navíc je to velmi výpočetně náročné.

$$p(\mathbf{x}|c) = \frac{1}{\sqrt{2^d \times \pi^d \times |\mathbf{\Sigma}|}} e^{-(\mathbf{x}-\bar{\mathbf{x}})^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\bar{\mathbf{x}})}$$

Obrázek 3.5: Podmíněná pravděpodobnost při parametrickém přístupu. Zdroj: [13]

3.2.3 Hledání k nejbližších vektorů

Dalším přístupem je metoda hledání k nejbližších vektorů. Tato metoda je principialně velmi podobná metodě k-nejbližších sousedů ([3]). Předpokládejme, že existuje testovací množina vektorů, kdy každý vektor má přiřazené

anotace. Za těchto předpokladů se vezme trénovací vektor. Pro tento vektor se najde k nejbližších vektorů (v rámci jejich vzdálenosti. Lze použít například metriku uvedenou v 4). Z těchto k nejbližších vektorů se použije jejich anotace a následně přiřadí k testovanému vektoru.

Při tomto přístupu můžeme chtít odfiltrovat anotace, jejichž výskyt nebyl u k nejbližších sousedů tak častý, jako u jiných anotací. Pro tento případ je nutné zavést práh.

Nechť je práh p, aktuálně zpracovávaná anotace a_x, počet výskytů aktuálně zpracovávané anotace c_x, celkový počet anotací *total* a množina přiřazených anotací k testovanému vektoru A.

Pak platí:

$$\frac{c_x}{total} \begin{cases} > p, & a_x \in A \\ < p, & a_x \notin A \end{cases} \tag{3.3}$$

4 Porovnávání histogramů

Pro určení podobnosti histogramů existuje několik metrik. Za předpokladu, že H_i je je značení histogramu a N je počet prvků histogramu, lze použít následující metody:

Korelace:

$$\mathrm{d}(\mathbf{H_1}, \mathbf{H_2}) = \frac{\sum_I (H_1(I) - \bar{H_1})(H_2(I) - \bar{H}_2)}{\sqrt{\sum_I (H_1(I) - \bar{H_1})^2 \sum_I (H_2(I) - \bar{H}_2)^2}} \tag{4.1}$$

kde

$$\bar{H_k} = \frac{1}{N} \sum_J H_k(J) \tag{4.2}$$

X^2 vzdálenost:

$$\mathrm{d}(\mathbf{H_1}, \mathbf{H_2}) = \sum_I \frac{(H_1(I) - H_2(I))^2}{H_1(I)} \tag{4.3}$$

Intesekce:

$$\mathrm{d}(\mathbf{H_1}, \mathbf{H_2}) = \sum_I \min(H_1(I), H_2(I)) \tag{4.4}$$

Vzdálenost Bhattacharyya:

$$\mathrm{d}(\mathbf{H_1}, \mathbf{H_2}) = \sqrt{1 - \frac{1}{\sqrt{\bar{H_1}\bar{H}_2 N^2}} \sum_I \sqrt{H_1(I) \cdot H_2(I)}} \tag{4.5}$$

Cosinová vzdálenost:

$$\mathrm{d}(\mathbf{H_1}, \mathbf{H_2}) = \frac{\sum_I H_1(I) \times H_2 I}{\sqrt{\sum_I H_1(I)^2} \times \sqrt{\sum_I H_2(I)^2}} \tag{4.6}$$

5 Vyhodnocení úspěšnosti anotace

Podle [1], úspěšnost automatické anotace se hodnotí porovnáním automaticky generovaných anotací s anotacemi přidělenými člověkem.

5.1 Per word precision a recall

Pojem Precision a Recall je běžně využíván pro zhodnocení výsledků klasifikace do více tříd [1]. Označme W_h jako počet výskytu konkrétní anotace přidelené člověkem v testovací množině, W_auto jako počet počet přiřazení konkrétní anotace automatickým anotování, W_c jako počet správně přidělených automatických anotací. Pak platí:

$$Precision = \frac{W_c}{W_{auto}} \tag{5.1}$$

$$Recall = \frac{W_c}{W_h} \tag{5.2}$$

Precision – česky přesnost – představuje poměr mezi počtem správně přidělené, konkrétní anotace a počtem, kolikrát konkrétní anotace měla být přidělena.
Recall – lze přeložit jako úplnost – představuje poměr mezi počtem správně přidělené, konkrétní anotace a počtem, kolikrát konkrétní anotace byla přidělena.

V případě, že vyjde vysoká hodnota pro *recall* a nízká hodnota pro precision, znamená to, že se anotováním získalo mnoho výsledků, ale málo z nich je korektních. Pokud hodnoty vyjdou opačně, znamená to, že se anotováním získálo málo, avšak korektních výsledků. Ideálním případem je mít obě hodnoty vysoké.

Pro zvýšení hodnoty *recall* je možné například snížit práh u klasifikátorů tak, aby vracel více výsledků.

5.2 Accuracy

Accuracy neboli přesnost, se využívá u vyhodnocení klasifikace do jedné třídy. U automatické anotace se předpokládá, že má objekt přiřazené jedno

slovo přiřazené programem a jedno slovo přiřazené člověkem. Označme aktuálně vyhodnocovaný objekt *current*, slovo získané anotace $current_a$ a slovo přiřazené člověkem $current_h$. Pak pro výpočet pro jeden konkrétní objekt platí:

$$accuracy(current) = \begin{cases} 1, & current_a = current_h \\ 0, & current_a ! = current_h \end{cases} \tag{5.3}$$

Označme si celkový počet testovaných objektů jako N. Celková přesnost pro všechna data lze pak vypočítat jako:

$$totalAccuracy = \frac{\sum_{n=1}^{N} accuracy(n)}{N} \tag{5.4}$$

6 Metody rozdělení dat

Jak již bylo zmíněno, počítač se potřebuje z nějakých dat nejprve natrénovat a posléze je možné získat výsledky pro neznámá data. Je zde tedy potřeba rozdělit data na trénovací a testovací množinu. Podle [6] existuje několik metod. Nyní budou krátce popsány.

6.1 Holdout

Metoda Holdout je z metod nejjednodušší. Množina dat se rozdělí na dvě exkluzivní množiny – testovací a trénovací. Běžně používané rozdělení bývá 2/3 z dat je přiřazena množině trénovací a 1/3 množině testovací.

6.2 Křížová validace

Při křížové validaci se vstupní množina dat rozdělí na jednu exkluzivní trénovací podmnožinu a zbylá data tvoří trénovací data. Můžeme si například představit, že každý desátý prvek bude patřit pouze do trénovací množiny a zbytek do testovací. Rozdílem oproti Holdout metodě (6.1) je, že se tento proces opakuje několikrát, pokaždé s jiným výběrem testovací a trénovací množiny. Podle [6] lze metody křížové validace rozdělit na k-fold, kompletní.

K-fold křížová validace rozdělí množinu dat do k exkluzivních podmnožin $D_1, D_2, \ldots D_k$ stejné velikosti. Počítač je trénován k krát. Nechť tedy platí $t \in \{1, 2, \ldots, k\}$. Pak je pro každé trénování brána množina $D \backslash D_t$ a každé testování množina D_t.

Kompletní křížová validace vytváří všechny kombinace testovacích a trénovacích množin. Tato metoda je však velmi náročná.

7 Data

Jak již bylo zmíněno, počítač se musí nejprve nějaké znalosti naučit. K tomu je potřeba mít množinu obrázků a k nim patřičná metadata obsahující anotaci.

7.1 Data od ČTK

Pro práci nám jsou propůjčeny obrázky od ČTK. Balík obsahuje 3383 obrázků. Obrázek má vždy metadata, která se jmenují stejně jako obrázek a jsou ve formátu XML. Pro naši prácí jsou klíčové elementy jako `Keyword` a `Category` Veškeré informace v metadatech jsou psané v českém jazyce. Elementy `Keyword` a `Category` jsou vyplňovány redaktorem, či autorem snímku. Jak můžeme vidět na obrázku 7.1, element `Keyword` je velmi konkrétní a často popisuje až příliš abstraktní věci jako například USA. Pro naší práci se tedy lépe hodí informace z elementu `Category`, který odpovídá kategorií používaným v ČTK.

Obrázek 7.1: Ukázka obrázku z ČTK datasetu. Tento obrázek byl anotován slovy z elementu Keyword – USA, vesmír, ISS, 2. Z elementu Category je k obrázku přiřazeno označení – vat

7.2 Corel 1000 dataset

Pro rychlé otestování programu je potřeba užití nějaké menší databáze, přesto s relevatními daty. Tuto podmínku splňuje Corel 1000 dataset, jenž je podmnožinou Corel5K databáze. Tato databáze obrázků byla využita v ([8],[11]) a pro porovnání algoritmů pro vyhledávání obrázků na základě

obsahu. Databáze obsahuje 1000 obrázků ve formátu JPG. K obrázkům není přiložen žádný soubor s anotacemi. obrázky jsou rozdělený po stovkách do kategorií. Tedy například 0 – 99 je první kategorie, 100 – 199 druhá kategorie atd. Z toho plyne, že každý obrázek bude mít právě jednu anotaci.

7.3 Li dataset

Tato databáze obrázků byla vytvořena pro výzkum v oblasti vyhledávání obrázků a anotace ([7]). Databáze obsahuje 2360 obrázků a jsou uloženy ve formátů JPEG. K databázi je přiložený soubor annotation.txt. V něm jsou uloženy informace jako pořadí obrázku, relativní umístění obrázku vůči souboru a jeho anotace. Informace o každém souboru jsou odděleny tabulátorem.

7.4 Iapr-tc dataset

Databáze obsahuje kolekci 20 000 obrázků ve formátu JPG[4]. K datasetu je přiložen soubor iaprtc12dictionary.txt, ve kterém je slovník všech slov, které se využívají při anotaci. Tento slovník slouží pro získání klíčových slov z textu přiloženému ke každému obrázku. Jak již bylo zmíněno, každý obrázek má svůj textový soubor o formátu připomínající XML Pro naší práci obsahuje klíčové elementy jako `DESCRIPTION` a `IMAGE`. Element `DESCRIPTION` představuje textový popis obrázku. Z něj je potřeba získat klíčová slova pro anotaci. Element `IMAGE` obsahuje relativní cestu k obrázku od kořenové složky.

8 Volba metod pro implementaci

Klíčovým aspektem pro výběr metod jsou data, která se budou anotovat. Práce je především zaměřena na testovací data od ČTK. Obrázky u poskytnutých dat jsou fotografie, z čehož plyne, že obrázky jsou velmi proměnlivé, budou mít například různé nasvícení, budou různě velké, nebo objekty na obrázku budou různě pootočené. Z tohoto důvodu a po konzultaci s vedoucím práce jsem se rozhodl pro využití metod pro extrakci příznaků – Bag of LBP Features a Bag of SIFT Features, neboť obě metody jsou invariantní vůči škálování. Dále jsem se rozhodl pro využití základní verze LBP, která je jednodušší než dvě předchozí zmíněné a bude tak využita pro porovnání ke složitějším metodám. Metoda SIFT je navíc invariantní vůči rotaci a po lehkých úpravách může být i odolná vůči různému nasvícení. Z tohoto důvodu od SIFT očekávám přesnější výsledky, avšak za cenu větší výpočetní náročnosti. Metody pro extrakci přiznaků – LBP a SIFT jsou součástí softwarové knihovny OpenCV 8.2.

K porovnání dosažených výsledků, budou využity databáze Iapr-tc dataset a Li dataset, které již mnoho prací v oblasti AIA použilo k otestování svých výsledků, například v práci [10]. Databáze Corel bude využita pro otestování klasifikaci snímků do jedné třídy.

Při anotaci jsem se rozhodl pro jednoduchost využít vzdálenosti histogramů popsané v kapitole 4. Tento přístup umožňuje jak klasifikaci do jedné třídy, tak i do více tříd. Práce je navržena tak, aby výstup z předzpracovaných dat šel snadno použít na další metody, jako jsou například neuronové sítě.

8.1 Požadavky na software

Program byl navržen na platformu Windows 7 a vyšší, 64-bitové verze. Program využívá knihovnu Dirent, která slouží k procházení složek. V programu je dále používaná knihovna OpenCV, jejíž instalace je popsaná v uživatelské příručce. Pro přeložení programu je nutné mít nainstalovaný překladač Microsoft Visual C++ 2012 (x64). Přeložení a spuštění programu je popsáno v uživatelské příručce.

8.2 OpenCV

OpenCV je otevřená, multiplatformní, softwarová knihovna vydávaná pod licencí BSD. Knihovna má rozhraní pro programovací jazyky jako je C++, C, Python a Java. Knihovna byla navržena jako výpočetně efektivní knihovna se silným důrazem na výpočty prováděné v reálném čase.

9 Architektura aplikace

Jak již bylo zmíněno v předchozích kapitolách, proces automatické anotace zahrnuje dva problémy – extrakce příznaků a samotná automatická anotace. Dalším problémem je vyhodnocení správnosti výsledků. Cílem návrhu architektury programu je modulárnost každého z problému. Architektura je tedy rozdělena na tři části.

9.1 Předzpracování dat

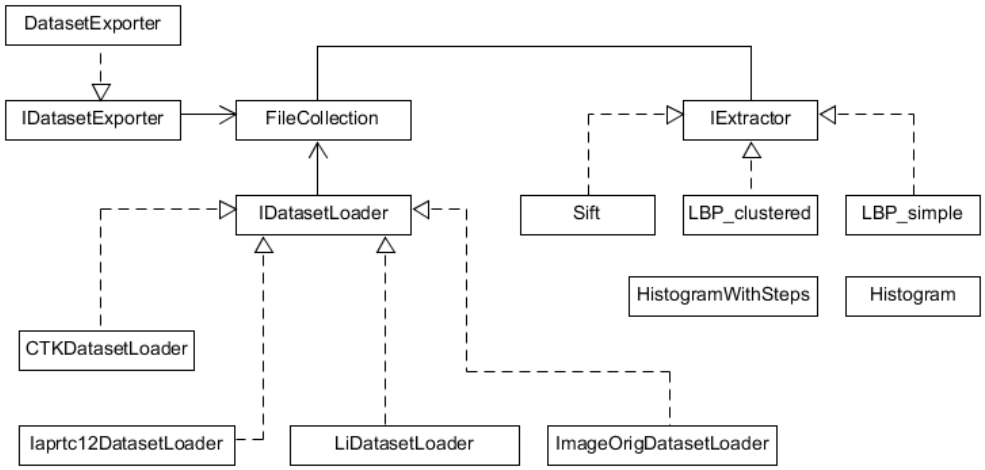
Vzhledem k tomu, že se v práci využívá několik databází obrázků, které nemají stejnou strukturu, je nutné je pro budoucí výpočty sjednotit. Pro načtení obrázků a jejich metadat je zde rozhraní `IDatasetLoader`. Rozhraní obsahuje čistě virtuální metodu `GetDataset()`, jenž vrací mapu `map<string, FileCollection>`. Klíčem mapy je název souboru bez suffixu a hodnotou je objekt `FileCollection`. Ten obsahuje klíčové údaje o souboru jako například celá testa k souboru či přiřazené anotace. Každá databáze má svou vlastní třídu pro načítání, která implementuje právě od `IDatasetLoader`. Jsou to třídy `CTKDatasetLoader`, `Iaprtc12DatasetLoader`, `ImageOrigDatasetLoader` a `LiDatasetLoader`. Tento návrh umožňuje snadné přidání tříd pro zpracování dalších databází.

Kolekce načtená s využitím rozhraní `IDatasetLoader`, je následně zpracovaná s pomocí jedné z tříd implementující rozhraní `IExtractor`. Rozhraní obsahuje čistě virtuální funkci `ProcessDataSet`. Mezi třídy implementující rozhraní `IExtractor` patří: `LBP_simple` (2.5), `LBP_clustered` (2.8) a `Sift` (2.7).

Vzhledem k výpočetní náročnosti shlukované metody Bag of SIFT features a Bag of LBP features, jsou již při extrakci rozděleny data na testovací a trénovací množiny – foldy (6.2).

Předzpracované výsledky jsou následně exportovány do souboru s využitím rozhraní `IDatasetExporter`. Toto rozhraní má čistě virtuální metodu `ExportDataSet`, která vyexportuje zpracovaná data do souboru. Dále je zde třída `DatasetExporter`, která právě toto rozhraní implementuje. Využití rozhraní umožňuje v budoucnu snadnou implementaci další třídy pro export předzpracovaných dat.

Strukturu modulu pro předzpracování popisuje diagram na obrázku (9.1).



Obrázek 9.1: Architektura aplikace – Načítání a zpracování databází obrázků

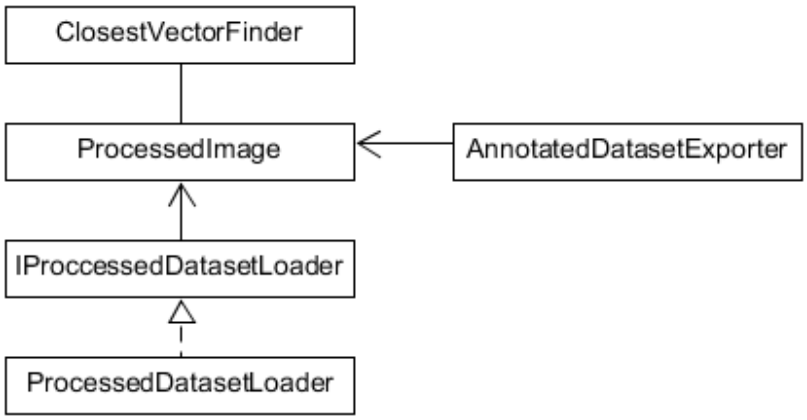
9.2 Anotace obrázků

Tento modul se zabývá samotným anotováním předzpracovaných dat.

V prvním kroku je nutné načíst předzpracovaná data. O to se stará rozhraní `IProccessedDatasetLoader`, od kterého implementuje třída `ProcessedDatasetLoader`. Rozhraní obsahuje čistě virtuální metody `void LoadProccessedDataSet` a `LoadFoldsProccessedDataSet`. První zmíněná metoda slouží k načtení předzpracovaných dat, které následně rozřadí na testovací a trénovací množiny (foldy). Druhá metoda slouží pro načtení předpřipravených množin ze souborů. Užitím tohoto rozhraní je budoucí snadná rozšiřitelnost programu o další předzpracovaná data, například v jiném formátu. Výsledkem rozhraní `IProccessedDatasetLoader` je kolekce objektů `ProcessedImage` rozdělených na trénovací a testovací. Třída `ProcessedImage` slouží jako přepravka pro předzpracovaná data a zároveň v sobě uchovává data pro anotace.

V dalším kroku se hledají pro každý objekt z testovací množiny, přesněji k jeho vektorové reprezentaci (deskriptorům), nejbližší vektory z trénovací množiny. O to se stará třída `ClosestVectorFinder`. Třída obsahuje metodu `FindIt`, která má parametry jako metoda metriky vzdálenosti (4), práh 3.2.3 a parametr, zda brát pouze nejlepší anotaci nebo všechny. Parametr `methodId` může mít hodnot:

- 0 – Korelace
- 1 – X^2 vzdálenost
- 2 – Vzdálenost Bhattacharyya



Obrázek 9.2: Architektura aplikace – Anotace obrázků

- 3 – Cosinová vzdálenost

Následně jsou výsledky anotace z testovací množiny zapsány do souboru s využitím třídy `AnnotatedDatasetExporter`.

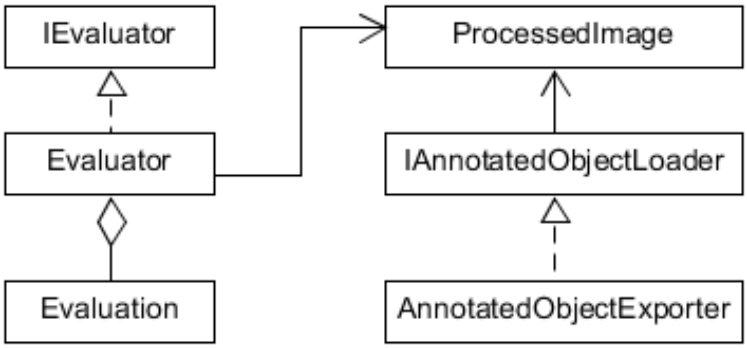
Strukturu modulu pro anotování popisuje diagram na obrázku 9.2

9.3 Vyhodnocení anotovaných dat

Tento modul slouží k vyhodnocení anotovaných dat.

V prvním kroku je nutné načíst anotovaná data. To zajišťuje rozhraní `IAnnotatedObjectLoader`, od kterého implementuje třída `AnnotatedObjectLoader`. Rozhraní obsahuje čistě virtuální metodu `LoadAnnotatedObjects`. Výsledkem této metody je kolekce objektů `ProcessedImage` rozdělených na trénovací a testovací. V této fázi objekty obsahují i přiřazené anotace. Načtená kolekce je následně zpracovaná třídou `Evaluator`, která implementuje rozhraní `IEvaluator`. Rozhraní obsahuje čistě virtuální metody `EvaluateAndExportResultToFile` a `EvaluateAccuracyAndExportResult`. První metoda slouží k vypočítání preccision a recall. Teoreticky byly tyto hodnoty popsané v sekci 5.1. Druhá metoda slouží pro vypočítání a export accuracy. Výpočet accuracy je popsán v sekci 5.2.

Strukturu modulu pro anotování popisuje diagram na obrázku 9.3



Obrázek 9.3: Architektura aplikace – Vyhodnocení výsledků

10 Výsledky

Pro přehlednost jednotlivých sekcí této kapitoly je nutné si nadefinovat, jaké metody s jakými parametry budou vyhodnocovány. Vzhledem k tomu, že nastavení parametrů se liší problém od problému, je vždy nutné nějakým testováním získat optimální hodnoty. Z toho důvodu jsou všechny parametry voleny čistě experimentálně. Začněme definicí parametrů pro metody extrakce příznaků.

LBP 25

- metoda – Bag of LBP features,
- výška okénka – 50,
- šířka okénka – 50,
- krok okénka – 25,
- výsledná velikost deskriptoru – 50

LBP 50

- metoda – Bag of LBP features,
- výška okénka – 100,
- šířka okénka – 100,
- krok okénka – 50,
- výsledná velikost deskriptoru – 50

SIFT 25

- metoda – Bag of SIFT features,
- výsledná velikost deskriptoru – 25

SIFT 50

- metoda – Bag of SIFT features,
- výsledná velikost deskriptoru – 50

LBP 1x1

- metoda – LBP,
- počet sloupců – 1,
- počet řádek – 1,

LBP 2x2

- metoda – LBP,
- počet sloupců – 2,
- počet řádek – 2,

LBP 4x4

- metoda – LBP,
- počet sloupců – 4,
- počet řádek – 4,

Nyní definujme parametry pro anotování.

Parametry anotování

- práh – 0,1,
- počet nejbližších vektorů – 5,
- počet množin křížové validace – 10 (vyjma Iaprtc-12),
- metrika porovnání histogramů – Bhattacharrya (zdůvodnění popsané v následující sekci 10.1)

V případě dat Iaprtc-12 je testovací a trénovací množina rozdělena podle testovacího protokolu použitého v dokumentu [1]. V ostatních případech jsou data rozdělena pomocí křížové validace do deseti množiny.

10.1 Volba nejlepší metriky pro porovnání histogramů

Jak bylo zmíněno v kapitole 4, pro porovnání vzdálenosti vektorů existuje několik metrik. Vzhledem k výpočetní náročnosti předzpracování a následného anotování dat jsem se rozhodl použít k porovnání výsledků pouze jednu metriku. Zmíněné metriky byly experimentálně otestované při anotaci předzpracovaných dat, získaných z datasetu Li. Jelikož platí, že čím větší je trénovací množina, tím je anotování testovaných objektů náročnější, je volba tohoto datasetu kompromisem mezi dostatečným počtem dat a výpočetní náročností.

Metoda	**Korelace**	X^2	**Intersekce**	**Bhattacharyya**	**Cosin**
LBP 1x1	0,077	0,099	0,097	0,109	0,085
LBP 2x2	0,093	0,095	0,114	0,132	0,096
LBP 4x4	0,102	0,089	0,136	0,139	0,119
LBPC 25	0,072	0,116	0,095	0,119	0,073
LBPC 50	0,073	0.114	0,094	0,112	0,073
SIFT 25	0,073	0,086	0,073	0,086	0,078
SIFT 50	0,082	0,092	0,092	0,103	0,09
$\sum$	0,572	0,691	0,701	0,8	0,614

Tabulka 10.1: Naměřené výsledky metrik provnání histogramů. Jednotlivé sloupce reprezentují vypočítanou hodnotu *precision* pro určitou metodu.

Z tabulky 10.1 vyplývá, že nejlepší metrikou pro porovnání histogramů bude zvolena metoda Bhattacharyya, která dosáhla nejlepších výsledků ve všech případech.

10.2 Dosažené výsledky

Výsledky představují aritmetický průměr ze všech testovacích množin konkrétních dat. V tabulce 10.2 můžeme pozorovat vliv předzpracování na výsledek klasifikace.

V případě LBP se ukázalo, že čím větší počet regionů obrázek má, tím lepších výsledků pak klasifikace dosahuje. V případě metod SIFT se prokázalo, že delší deskriptor lépe popisuje obrázek a tím metoda dosahuje lepších výsledků. V případě metod LBPC můžeme pozorovat, že menší velikost posuvného okénka a kroku dosahuje vyšších hodnot precision, nicméně rozdíl není většinou moc markantní.

Metoda	ČTK p%	ČTK r%	Iaprtc p%	Iaprtc r%	Li p%	Li r%
LBP 1x1	10,2	28,9	11,7	5.4	10,9	27,0
LBP 2x2	10,6	29,6	15,8	6,9	13,2	29,8
LBP 4x4	11,2	28,9	17,1	8.3	13,6	29,0
LBPC 25	9,2	28,3	12,7	3,8	11,9	28,7
LBPC 50	9,2	28,3	11,5	3,6	11,2	28,0
SIFT 25	8,2	24,3	9,5	3,6	8,6	23,8
SIFT 50	9,3	26,4	13,0	4,4	10,3	24.8

Tabulka 10.2: Dosažené výsledky. Písmena p a r jsou zkratkami pro precission a recall. Dosažené hodnoty jsou uváděny v procentech.

Zajímavým úkazem u dat Iaprtc je nízká hodnota recall vůči precision. To nám říká, že klasifikátor přiřadil málo anotací, ale mnoho z nich přiřadil správně. Nízkou hodnotu recall lze vysvětlit tím, že data od Iaprtc mají mnoho kategorií a snímky stejné kategorie si jsou málo podobné.

Nejlepších hodnot precision dosahují metody LBP, úplně nejlepších hodnot konkrétně metoda LBP 4x4. Metody LBP také dosahují nejvyšších hodnot recall. Metody LBPC, které extrahují příznaky stejně jako LBP, ale navíc je pak shlukují, takových výsledků nedosahují. Princip shlukování využívají i metody SIFT. Metody SIFT dosahují podobných výsledků. Pro data od ČTK a Iaprtc dosahuje SIFT 50 lepších výsledků než metody LBPC, naopak tomu je však u dat Li. Nelze tak obecně říct, která z těchto dvou metod je lepší.

10.3 Doby běhu

10.3.1 Doba běhu přezpracování

V tabulce 10.3 můžeme pozorovat, že se doba běhu předzpracování konkrétních metod může velmi výrazně lišit. U metody LBP sledujeme, že s přibývajícím počtem regionů roste i doba předzpracování. Nejnáročnější na předzpracování byla metoda LBPC 25, kdy posouvání okénka o malý krok vygeneruje velké množství vektorů, ze kterých se následně napříč celou tré-

Data	LBP 1x1	LBP 2x2	LBP 4x4	LBPC 25	LBPC 50	SIFT 25	SIFT 50
ČTK	31,3	31,6	32,6	22632,1	5063,1	647,1	880,7
LI	7,3	7,7	8,6	5717,4	1299,6	251,5	359,1
Ia-prtc	127,4	129,2	269,4	3881,2	2421,9	2238,8	2609,6
Co-dak	3,2	3,7	3,8	2770,9	522,2	111,2	173,6

Tabulka 10.3: Naměřené doby běhu předzpracování. Hodnoty jsou uváděny v sekundách. Doba byla měřena od začátku extrakce příznaků z konkrétních dat až po ukončení extrakce.

novací množinou vytváří Bag of Features. Metoda LBPC 50, která svými parametry vygeneruje podstatně méně vektorů, je podstatně rychlejší. V případě metod SIFT 25 a SIFT 50 pozorujeme, že i rozdílná velikost výsledného vektoru má vliv na výkon – tedy čím větší je počet shluků (tedy i velikost výsledného vektoru), tím déle předzpracování trvá.

10.3.2 Doba běhu anotování

Při pohledu na tabulku 10.4 můžeme říct, že na dobu běhu má vliv charakter předzpracovaných dat. Jelikož v metodě k-nejbližších vektorů, která je při anotaci využita, platí, že čím více vektorů se musí porovnat k nalezení nejbližších, tím je metoda náročnější. Toto tvrzení se potvrzuje při pohledu na hodnoty dat, získaných z metod LBPC a SIFT.

Lze také tvrdit, že na rychlost má vliv velikost vektoru. Pokud se podíváme na doby běhu u LBP 1 x 1, kde je obrázek reprezentován vektorem o velikosti 255 a srovnáme je s výsledky LBPC 25 a 50 a SIFT 50, které mají velikost 50, LBP 1 x 1 vcelku zaostává. Navíc nejkratší dobu má metoda SIFT 25, která je reprezentovaná nejkratším vektorem – 25.

Z logiky věci pak vyplývá, že nejdelší doby anotování bude dosahovat metoda LBP 4x4, kde je obrázek reprezentován 8 vektory o délce 255.

10.4 Porovnání s výsledky dosaženými v literatuře

Při výběru datasetů, které v práci slouží k otestování navržených algoritmů, bylo klíčovým parametrem najít takový, který byl využit i v jiných pracích

Data	LBP 1x1	LBP 2x2	LBP 4x4	LBPC 25	LBPC 50	SIFT 25	SIFT 50
ČTK	52,8	166,9	657,8	47,01	46,6	46,0	46,9
LI	19,7	96,7	334,1	18,8	19,2	19,0	19,2
Iaprtc	3893,6	10375,4	12561,2	2205,1	2168,6	2001,8	2099,4
Codak	3,7	12,8	56,6	3,4	3,4	3,4	3,5

Tabulka 10.4: Naměřené doby běhu anotování. Hodnoty jsou uváděny v sekundách. Doba byla měřena od začátku anotování až po jeho ukončení.

Metoda	p%	r%
Lasso	28	29
JEC	28	29
MBRM	24	23
RGB	24	24
LAB	24	25
Haar	24	25
HSV	20	20
HaarQ	19	16
Gabor	15	15
GaborQ	8	9

Tabulka 10.5: Dosažené výsledky v literatuře. Písmena p a r jsou zkratkami pro precission a recall

z oblasti AIA a měl přesně nadefinovaný testovací protokol. Tyto parametry splňují data Iaprtc-12. Výsledky dosažené na těchto datech jsou zmíněny v článku [1].

Výsledky pro precision v tabulce 10.5 dosahují hodnot v rozmezí od 8% do 28%. Hodnoty precision a recall jsou si většině případech velmi podobné. Lze sledovat, že s rostoucí hodnotou recall roste i precision.

V tabulce 10.6 jsou pro přehlednost uvedeny výsledky z metod navržených v této práci. Pro dosažené výsledky také platí, že s rostoucí hodnotou recall roste i hodnota precision, avšak zajímavým rozdílem oproti výsledkům z literatury jsou nízké hodnoty pro recall.

Přesto ale výsledky dosahují v porovnání s výsledky v literatuře uspkojivých hodnot. Pro výsledky precision u všech navržených metod platí, že jsou lepší, než nejhorší hodnoty dosažené v literatuře.

Metoda	p%	r%
LBP 4x4	17	8
LBP 2x2	16	7
SIFT 50	13	4
LBPC 25	13	4
LBP 1x1	12	5
LBPC 50	12	4
SIFT 25	9,5	4

Tabulka 10.6: Dosažené výsledky v literatuře. Písmena p a r jsou zkratkami pro precission a recall

11 Závěr

V teoretické části byla popsaná problematika extrakce příznaků. Podrobněji byly popsané metody LBP a SIFT, které se osvědčily v jiných pracích. Dále je v práci popsaná problematika anotace, přesněji klasifikace do jedné a více tříd.

Po analýze byl navržen a implementován software pro automatickou anotaci dat a následné vyhodnocení dosažených výsledků. Pro extrakci příznaků byly implemetovány metody LBP, Bag of LBP Features a Bag of SIFT Features. Pro samotnou anotaci byla implementovaná metoda k-nejbližích sousedů. Při vývoji byl kladen důraz na modulární architekturu programu a snadnou rozšiřitelnost. Data jsou vždy exportovaná po jednotlivých krocích, kdy vstupem jednoho modulu je výstup druhého modulu. Moduly ale na sobě mohou pracovat nezávisle.

Funkčnost naimplementovaných metod byla otestovaná na několika databázích. Při práci bylo důležité sjednotit různorodou strukturu dat na data stejného formátu. Deskriptory předzpracovaných obrázků jsou vektorového typu a tudíž je možné práci v budoucnu rozšířit o další typy klasifikátorů, například neuronové sítě. Podstatnou část práce zabralo vyhledání testovacích dat. Cílem bylo vyhledat data charakteristikou podobná datům od ČTK, tedy aby byla dostatečně rozsáhlá a měla jasně definovaný testovací protokol. Dalším důležitým aspektem navíc bylo, aby vybraná data byla již použita v ostatních pracích z oblasti AIA Byla snaha použít Corel5K data, neboť jsou nejčastěji využívaná. Tato data jsou volně dostupná ke stažení, nicméně není k nim přiložený soubor s anotacemi. Ten byl následně nalezen na internetu, nicméně po implementaci načítání dat bylo zjištěno, že anotace k obrázkům neodpovídají. Nakonec byla pro porovnání výsledků použita data Iaprtc-12, která také splňovala všechny výše zmíněné požadavky.

12 Použité zkratky

AIA Automatic image annotation.

LBP Local binary pattern.

SIFT Scale invariant feature transform.

ČTK Česká tisková kancelář.

EM Expectation maximization.

nCut Normalizovaný řez.

CCV Vektor koherence barev.

SIFT Scale invariant feature transform.

LoG Laplacián Gaussiánů.

DoG Rozdíl Gaussiánů.

BoF Bag of Features.

BoW Bag of Words.

SVM Support vector machine.

ANN Umělé neuronové sítě.

DT Rozhodovací strom.

MIML Multi-instance multi-label.

Literatura

[1] Carneiro, G. et al. Supervised learning of semantic classes for image annotation and retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2007, 29, 3, s. 394–410.

[2] Carson, C. et al. Blobworld: A System for Region-Based Image Indexing and Retrieval. , 3.

[3] Dudani, S. A. The distance-weighted k-nearest-neighbor rule. *Systems, Man and Cybernetics, IEEE Transactions on*. 1976, , 4, s. 325–327.

[4] Grubinger, M. et al. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International Workshop OntoImage*, 5, s. 10, 2006.

[5] Košař, V. Srovnání deskriptorů pro reprezentaci obrazu. Master's thesis, Západočeská univerzita, Plzeň, 2015.

[6] Kohavi, R. – others. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, 14, s. 1137–1145, 1995.

[7] Li, J. *Jia Li dataset* [online]. Jia Li, 2016. [cit. 2016/27/04]. Dostupné z: `http://sites.stat.psu.edu/~jiali/index.download.html`.

[8] Li, J. – Wang, J. Z. Automatic linguistic indexing of pictures by a statistical modeling approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2003, 25, 9, s. 1075–1088.

[9] Lowe, D. G. Object Recognition from Local Scale-Invariant Features. , 4.

[10] Makadia, A. – Pavlovic, V. – Kumar, S. A new baseline for image annotation. In *Computer Vision–ECCV 2008*. Springer, 2008. s. 316–329.

[11] Wang, J. Z. – Li, J. – Wiederhold, G. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2001, 23, 9, s. 947–963.

[12] Yuan, X. et al. A SIFT-LBP image retrieval model based on bag of features. In *IEEE International Conference on Image Processing*, 2011.

[13] Zhang, D. – Islam, M. M. – Lu, G. A review on automatic image annotation techniques. *Pattern Recognition*. 2012, 45, 1, s. 346–362.

A Uživatelská příručka

Pro spuštění programu je nutné mít nainstalovanou knihovnu OpenCV. V následující části bude popsáno, jak program spustit pro operační systém Windows 10 s využitím vývojového přestředí Visual Studio 2012.

A.1 Instalace OpenCV

Tato část čerpá téměř celý postup z diplomové práce [5] z části – Instalace na operačním systému Windows.

1. stáhněte instalační soubor OpenCV for Windows VERSION 2.4.9. z adresy
   `http://opencv.org/downloads.html`.

2. po stažení spust'te samorozbalovací archiv s právy administrátora (spustit jako správce). Zvolite vhodnou cestu, například `C:\OpenCv\`.

3. Nyní je potřeba přidat systémovou proměnnou. Následující postup předpokládá umístění rozbaleného archivu do `C:\OpenCv\`. Spust'te příkazovou řádku s právy administrátora a zadejte text:
   `setx -m OPENCVDIR C:\OpenCv\build\x64\vc11`

4. Nyní je potřeba nastavit cesty k dynamickým knihovnám. Klikněte pravým tlačítkem myši na „Tento počítač“ a vyberte „Vlastnosti“, následně klikněte na „Upřesnit nastavení systému“. V otevřeném dialogu „Vlastnosti systému“ klikněte na tlačítko „Promměnné prostředí“. V části„Uživatelské proměnné“ do proměnné „PATH“ přidejte nakonec řetězec `%OPENCVDIR%\bin`. V části „Systémové proměnné“ přidejte novou proměnnou s názvem `OPENCVDIR` s hodnotou:
   `C:\OpenCv\build\x64\vc11`.

A.2 Přeložení a spuštění programu ve vývojovém prostředí Visual Studio 2012

1. V tomto kroku se předpokládá, že máte nainstalované Visual Studio 2012. Otevřete si projekt spuštením souboru `BPAIA.sln`.

2. Pokud jste správně postupovali při konfiguraci proměnných pro OpenCV, projekt jde přeložit. To se dělá následovně:
   V horním menu vyberte `Build` -> `Build solution`.

3. Program se spouští s parametry popsaných v sekci A.3. Ty se nastavují následovně:
   V horním menu vyberte `Project` -> `Properties`. Z výběru `Configuration` vyberte možnost `Release` a z výběru `Platform` vyberte možnost `x64`. Dále v `Configuration properties` zvolte `Debugging`. Zde se do řádku `Command Arguments` vkládají příkazy popsané v A.3.

4. Pokud se zadají neplatné vstupní parametry, je uživateli vypsaná nápověda a program je ukončen.

A.3 Ovládání programu

Parametry musí být vždy zadávané v definovaném pořadí.

A.3.1 Zpracování databáze obrázků

Příkaz

`-p <korenová slozka> <typ databaze> <zpracovani> <vysledek>`

Popis parametrů

- `<korenová slozka>` – jedná se o cestu ke kořenové složce zvolené databáze obrázků
- `<typ databaze>` – typ databáze obrázků:

  `iaprtc` - Iaprtc-12 databáze

  `ctk` – ČTK databáze

  `corel` – Corel databáze

  `li` – Li databáze
- `<zpracovani>` – výběr metody zpracování:

  `lbp` – LBP

  `lbpc` – shlukované LBP s posuvným okénkem

  `sift` – shlukovaný SIFT
- <vysledek> – cesta k výslednému souboru

Další parametry

Na další parametry je uživatel dotazován programem až po spuštění. Parametr se píše do konzole a odesílá tlačítkem na klávesnice – `Enter`.

Při výběru metody `lbp` je uživatel dotázán, zda chce výsledný soubor rozdělit na trénovací a testovací množinu a na počet řádků a sloupců regionu.

Při výběru metody `lbpc` je uživatel dotázán, zda chce rozdělení trénovacích množin nechat na programu, nebo zda má uživatel připravené soubory pro testovací a trénovací množinu s názvy snímků pro identifikaci, do které množiny patří. Dále je uživatel dotazován na parametry posuvného okénka, krok posunutí a velikost výsledného vektoru.

U metody `sift` je uživatel dotazován na rozdělení množin jako u `lbpc`. Dále je uživatel dotazován na požadovanou velikost výsledného vektoru.

Příklad

```
-p d:\pictures\ctk_image_text\ctk lbp d:\results\ctk.txt
```

A.3.2 Anotování předzpracovaných dat s tvorbou množin

Příkaz

```
-a <zdroj> <foldy> <nejblizsi> <prah> <metrika> <vysledek>
```

Popis parametrů

- `zdroj` – soubor s přezpracovanými daty ve tvaru `<název snímku>; <anotace oddělené čárkou>;<deskriptory oddělené čárkou>`
- `foldy` – počet testovacích a trénovacích množin
- `nejblizsi` – počet hledaných nejbližších vektorů, ze kterých je braná anotace
- `prah` – práh, s jakým budou anotace přijatá. Parametr je v intervalu <0,1>
- `metrika` – zvolená metrika porovnání deskriptorů:
  - 0 – Korelace
  - 1 – X^2 vzdálenost
  - 2 – Vzdálenost Bhattacharyya
  - 3 – Cosinová vzdálenost
- `<vysledek>` – cesta k výslednému souboru

Další parametry

Na další parametry je uživatel dotazován programem až po spuštění. Parametr se píše do konzole a odesílá tlačítkem na klávesnice – `Enter`.

Po spuštění příkazu je uživatel dotázán, zda chce vybrat právě jeden výsledek (ten nejlepší). Tato možnost slouží pro klasifikaci do jedné třídy. V opačném případě je vybráno více výsledků.

Příklad

```
-a d:\results\ctk_lbp.txt 10 5 0.1 2 d:\results\ctk_anot.txt
```

A.3.3 Anotace předzpracovaných dat se zadáním množin

Příkaz

`-f <zdroj> <nejblizsi> <prah> <metrika> <vysledek>`

Popis parametrů

- `zdroj` – cesta k souboru, který obsahuje cesty k souborům reprezentující jednotlivé množiny. Odkazované soubory obsahují data ve tvaru `<název snímku>;`
  `<anotace oddělené čárkou>;<deskriptory oddělené čárkou>`, kdy testovací a trénovací množina je oddělena řádkou `###`. Každý záznam je na nové řádce.
- `nejblizsi` – počet hledaných nejbližších vektorů, ze kterých je braná anotace
- `prah` – práh, s jakým budou anotace přijatá.
  Parametr je v intervalu <0,1>
- `metrika` – zvolená metrika porovnání deskriptorů:
  0 – Korelace
  1 – X^2 vzdálenost
  2 – Vzdálenost Bhattacharyya
  3 – Cosinová vzdálenost
- `<vysledek>` – cesta k výslednému souboru

Další parametry

Na další parametry je uživatel dotazován programem až po spuštění. Parametr se píše do konzole a odesílá tlačítkem na klávesnice – `Enter`.

Po spuštění příkazu je uživatel dotázán, zda chce vybrat právě jeden výsledek (ten nejlepší). Tato možnost slouží pro klasifikaci do jedné třídy. V opačném případě je vybráno více výsledků.

Příklad

`-f d:\results\ctk_sift.txt 5 0.1 2 d:\results\ctk_anot.txt`

A.3.4 Vyhodnocení anotovaných dat

Příkaz

`-e <zdroj> <vysledek>`

Popis parametrů

- `zdroj` – cesta k souboru s anotovanými daty ve tvaru: `<název snímku>; <přiřazené anotace oddělené čárkou>;<původní anotace>; <správně určené anotace jsou reprezentovány číslem 1, špatně určené číslem 2. Hodnoty jsou oddělelny čárkou>.` Každý záznam je na nové řádce.
- `<vysledek>` – cesta k výslednému souboru

Další parametry

Na další parametry je uživatel dotazován programem až po spuštění. Parametr se píše do konzole a odesílá tlačítkem na klávesnice – `Enter`.

Po spuštění příkazu je uživatel dotázán, zda zdrojová data obsahují pouze nejlepší výsledek. Tato možnost slouží pro vyhodnocení klasifikace do jedné třídy – program spočítá `accuracy`. V opačném případě program spočítá `precision` a `recall`.

Příklad

`-e: d:\results\ctk_anot.txt 5 0.1 2 d:\results\ctk_eval.txt`