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

Bakalářská práce

Automatická anotace obrázků

Plzeň 2017 Kateřina Kratochvílová

Místo této strany bude zadání práce.

Prohlášení

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

V Plzni dne 16. června 2017

Kateřina Kratochvílová

Poděkování

Ráda bych poděkovala Ing. Ladislavu Lencovi, Ph.D. za cenné rady, věcné připomínky, trpělivost a ochotu, kterou mi v průběhu zpracování této práce věnoval.

Abstract

This thesis deals with automatic image annotation (AIA). Purpose is to check working of choosen functions and try to find their improvements. One of these methods is Joint Equal Contribution and it's modification where were changed trasnporting key words. One of other methods included is Patterns of Oriented Edge Magnitudes metod and implementation of it's extensions including color and texture of image. Methos are described in theoretical part and implemented later. Achieved resoults are compared to literature in final phase. Testing run on data sets iaprtc12 and ESP.

Abstrakt

Bakalářská práce se zabývá automatickou anotací obrázků (AIA). Cílem práce je prověřit funkčnost vybraných metod z literatury a pokusit se o jejich vylepšení. V práci byla vyzkoušena metoda Joint Equal Contribution (JEC) a její modifikace, kdy bylo pozměněno přenášení klíčových slov. Dále byla otestována metoda Patterns of Oriented Edge Magnitudes (POEM) a implementováno její rozšíření zahranující barvu i texturu. Metody jsou v teoretické části rozebrány a následně byly implementovány. V konečné fázi byly dosažené výsledky porovnány s literaturou. Testování probíhalo na datasetech iaprtc12 a ESP.

Obsah

1 Úvod

V dnešní době, kdy je svět přesycen obrázky v digitální podobě, není vůbec snadné nalézt obrázek zobrazující požadovaný obsah. Naneštěstí počítače nedokáží vnímat obraz jako lidé, vnímají totiž obrazy jako sérii binárních informací. Přitom počítače a jejich práce s obrazy by se dala využít v mnoha oborech jako je lékařství nebo doprava. Na základě toho vyplouvá na povrch problém jak spravovat digitální obrázky a efektivně mezi nimi vyhledávat. Prostřednictvím klíčových slov přiřazených k obrázkům se dá problém vyhledávání zjednodušit. Přiřazení klíčových slov probíhá pomocí procesu automatické anotace obrázků. Klíčová slova přiřazená k obrázku by měla vyjadřovat jeho obsah (například les, strom). Při reálném použití můžeme ovšem narazit na problém při zadávání abstraktních slov, například šťastná rodina.

Pro automatickou anotaci obrázků se používá strojové učení. Můžeme ji rozdělit na dvě části. V první části získáme klíčové příznaky, ve druhé už je samotná anotace, tedy přidělení klíčových slov. Abychom tento postup mohli provést v praxi, musíme nejdřív klasifikátor natrénovat pomocí trénovací množiny. Trénovací množina je množina obrázků, která již má ke každému obrázku přidána metadata s klíčovými slovy připravenými od lidí. Vybrané obrázky v trénovací množině by měly být různorodé, aby anotace probíhala správně. Pojem automatická anotace obrázků je jednoduše řečeno proces, při kterém jsou k obrázku automaticky přiřazena metada, která obsahují klíčová slova.

Práce se bude zabývat nízkoúrovňovými příznaky, konkrétně barvou a texturou. Ovšem v případě kdy požijeme barevný příznak ochudíme se o informaci o textuře obrázku, prozměnu když použijeme texturový příznak (který pracuje s šedotónovým obrázkem) zanedbáme informaci o barvě. Jako možnost zpřesnění klasifikátoru by se tedy dalo použít jejich zkombinování. Nabízí se několik řešení [4]:

Vyhodnotit a klasifikovat příznaky odděleně a pak výslednou klasifikaci spojit z několika částí (například Joint Equal Contribution (JEC) [7]). Výhodou tohoto přístupu je zachování vlastností obou původních příznaků. Nevýhodou je náročnější výpočet a úspěšnost přístupu závisí na způsobu kombinace obou informací.

Vytvoření společného příznaku. Například rozšíření Patterns of Oriented Edge Magnitudes (POEM) na všechny barvené kanály. Musí se

však dbát na to, že informace o barvě a textuře se mohou ovliňovat i protichůdně.

Cílem práce je navrhnout a implementovat software umožňující automatickou anotaci obrázků za použití nízkoúrovňových příznaků, konkrétně barvy a textury a jejich kombinací. Metody budou testovány na standardních datech IAPRTC12 a ESP. Výsledky dosažené použitím jednotlivých příznaků budou porovnány s literaturou a s nově vytvořeným deskriptorem. V konečné fázi se pokusíme o jejich vylepšení.

# 2 JEC Joint Equal Contribution

Tato metoda je založena na hypotéze, že podobné obrázky mají podobná klíčová slova. Pomocí metody hledání nejbližších sousedů (dále jen KNN) je nalezeno K nejpodobnějších obrázků. Přičemž klíčová slova od nejbližšího souseda jsou posuzována odlišným způsobem než klíčová slova od k nejbližších sousedů mimo nejbližšího. Metoda je postavena na dvou typech příznaků - barevných a texturových. [7]

2.1 Příznaky

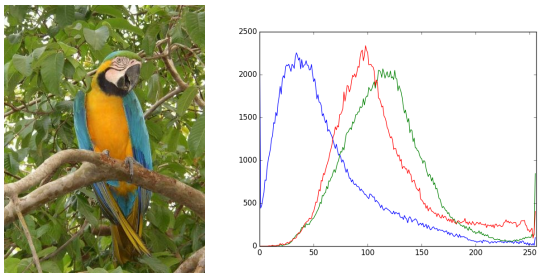
Barva a textura jsou považovány za dva nejdůležitější nízkoúrovňové příznaky pro obrázkovou reprezentaci. Nejběžnější barevné deskriptory jsou barevné histogramy, které jsou často využívány pro porovnávání a indexování obrázků, zejména z důvodu jejich efektivnosti a snadného výpočtu. K vytvoření texturových příznaků se používají Haarovy a Gaborovy wavelety a to především z důvodu efektivity při vytváření řídkých a zároveň diskriminativních obrázkových rysů. Při snaze omezit vliv předpokladů jednotlivých funkcí a maximalizovat množství získaných informací, je použito několik jednoduchých a snadno sestavitelných funkcí. [7]

2.1.1 Barva

U digitálního obrazu je barva reprezentovaná n-rozměrným vektorem. Jeho velikost a význam jednotlivých složek (tzv. barevných kanálů) zavisí na příslušném barevném prostoru. Počet bitů použitých k uložení buď celého vektoru nebo jeho jednotlivých složek se nazývá barevná hloubka (totožně bitová hloubka). Obvykle se můžeme setkat s hodnotami 8, 12, 14 a 16 bitů na kanál.

V použité metodě jsou získány vlastnosti z obrázků ve třech rozdílných barevných prostorech: RGB, HSV a LAB. RGB (Red, Green, Blue) je nejpoužívanější barevný prostor pro zachycení obrázu nebo jeho zobrazení. Oproti tomu HSV (Hue, Saturation and Value) se snaží zachytit barevný model tak, jak ho vnímá lidské oko. Zároveň se snaží zůstat jednoduchý na výpočet. Hue znamená odstín barvy (měří se jako poloha na standardním barevném kole

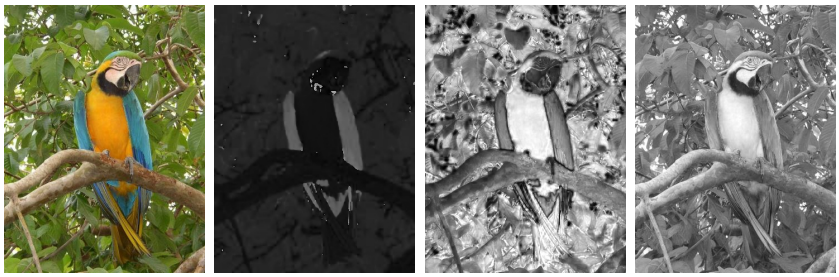
$0° - 360°$), saturation je systost barvy (množství šedi v poměru k odstínu 0% šedá barva - 100% plně sytá barva). Value je hodnota jasu nebo také množství bílého světla (relativní světlost nebo tmavost barvy). Některé kombinace hodnot H, S a V mohou dávat nesmyslné výsledky. RGB je závislý na konkrétním zařízení, nemůže dosáhnout celého rozsahu barev, které vidí lidské oko, zatímco barevný model LAB je schopen obsáhnout celé viditelné spektrum a navíc je nezávislý na zařízení. L (ve zkratce LAB) značí luminanci (jas dosahuje hodnot 0 - 100, kde 0 je černá a 100 je bílá). Zbylé A a B jsou dvě barvonosné složky, kdy A je ve směru červeno/zeleném a B se pohybuje ve směru modro/žlutém.



Obrázek 2.1: RGB histogram - zastoupení jednotlivých složek v obrázku

Obrázek 2.2: Barevný prostor RGB a jeho jednotlivé složky v pořadí R, G, B

Obrázek 2.3: Barevný prostor LAB a jeho jednotlivé složky v pořadí L, A, B



Obrázek 2.4: Barevný prostor HSV a jeho jednotlivé složky v pořadí H, S, V

Pro RGB, HSV i LAB je použito 16 binů na kanál histogramu v jejich příslušném barevném prostoru. To znamená, že z každého barevného prostoru vzniknou tři šestnácti prvkové histogramy. Tyto histogramy jsou zřetězeny a následně použité jako reprezentace příslušného barevného prostoru.

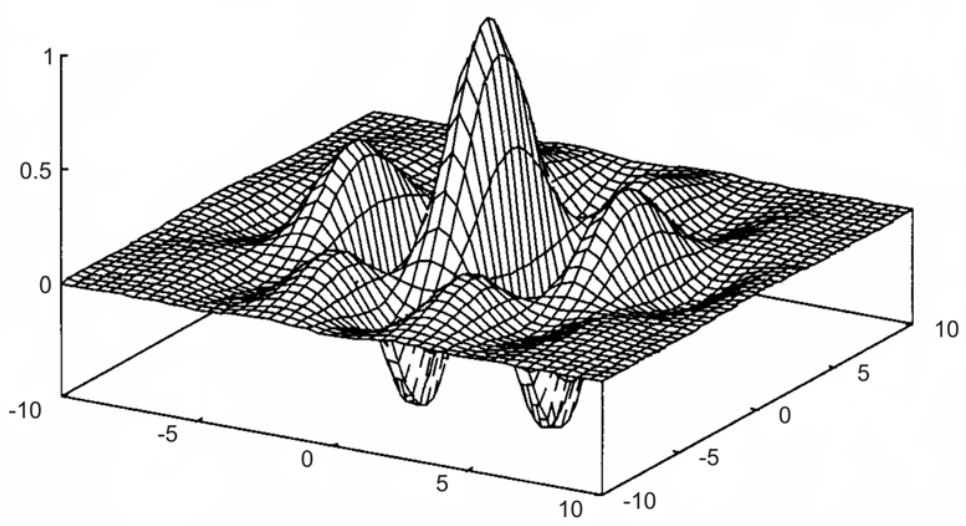
Algoritmus pro výpočet histogramu jednoho kanálu je následující. Jako první bude připraven vektoru nul představující histogram. Při výpočtu histogramu algoritmus prochází postupně jednotlivé hodnoty pixelu kanálu vstupního obrázku. Tato hodnota je vydělena 16. Výsledek určuje pozici (index) hodnoty ve vektoru, která má být inkrementována.

2.1.2 Textura

Jako reprezentace textur budou použity Gaborovy a Haarovy vlnky (v originále Gabor a Haar wavelet).

Gabor

Gaborův filtr je lineární filtr používaný pro analýzu textury, což znamená, že v podstatě zkoumá, zda existuje nějaký specifický frekvenční obsah v obraze ve specifických směrech v lokalizované oblasti kolem oblasti analýzy. Frekvence a orientace reprezentující Gaborovy filtry je podobná lidskému vnímání a proto je jejich použití zvláště vhodné při reprezentaci textury a detekci hran. V prostoru je 2D Gaborův filtr funkcí Gausova jádra modulovaného sinusouvou rovinnou vlnou. Gaboruv filtr je definován v rovnici 2.1.



Obrázek 2.5: Gaborova vlnka je tvořena kombinací dvou cosinových funkcí, s rozdílnou frekvencí pro každou osu, a následně jsou vynásobeny dvourozměrnou Gaussovou funkcí [2].

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = exp\left(- \frac{x^2 + \gamma^2 y^2}{2\sigma^2} \right) exp\left(i\left(2\pi \frac{x}{\lambda} + \psi \right) \right) \tag{2.1}$$

Gaborovy filtry jsou aplikovány na obrázky stejnou cestou jako běžné filtry. Základ tvoří maska (přesnější termín je konvoluční jádro), která reprezentuje filtr. Maskou je myšleno pole (obyvykle 2D protože se jedná o 2D obrázky) pixelů ve kterém každý pixel má přiřazenou hodnotu (váhu). Toto pole je přesunuto na každý pixel obrazu a je provedena konvoluční operace. Když je na obrázek aplikován Gaborův filtr, poskytuje nejvyšší odezvu na hranach a místech, kde se textura mění. [9]

Gaborův filtr reaguje na hrany a změny textury. Když se řekne, že filtr odpovídá na konkrétní funkci, myslí se tím, že podle parametrů reaguje na změny v obraze, které mají určitou frekvenci a orientaci.

U Gaborova filtru máme několik parametrů, které ovlivňují jeho vlastnosti.

ksize určuje velikost Gabor jádra. Když je ksize (a, b), je získáno jádro velikostí $a \times b$ pixelů. Jako u mnoha jiných konvolučních jader je preferován rozměr čtverce o lichých hranách, z důvodu umístění středu filtru na pixel. Pří různých ksize se velikost konvolučního jádra mění.

sigma označuje směrodatnou odchylku Gaussovy funkce použitou v Gaborově filtru. Tento parametr kontroluje šířku Gaussovy obálky použité v Gabor jádře.

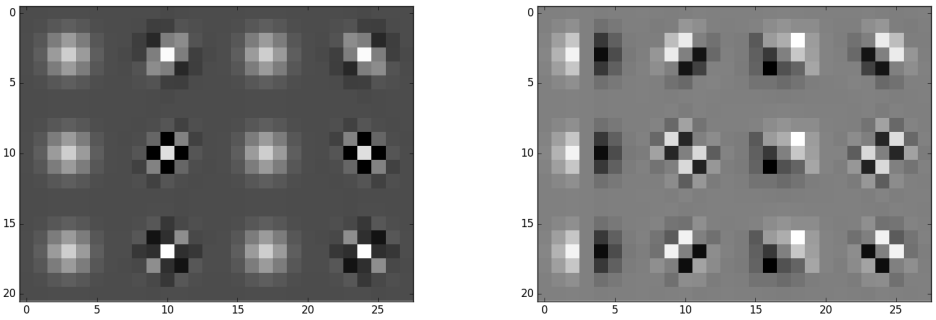
theta je orientace normály na paralelní pruhy Gaborovy funkce. Představuje možná jeden z nejdůležitějších parametrů Gaborova filtru. Theta rozhoduje jakého druhu funkce je tz. na jaký typ funkce filtr reaguje. Například při nulové thetě bude filtr reagovat pouze na vodorovné příznaky. Proto abychom získali vlastnosti v různých úhlech obrazu, rozdělíme interval mezi 0-180 na několik stejných částí a vypočítáme Gaborovo jádro pro každou takto získanou hodnotu theta.

lambda udává vlnovou délku sinusové funkce ve výše uvedené rovnici.

gamma určuje prostorový poměr stran. Kontroluju elipsicitu Gausovy funkce. Když je gamma = 1 je Gauss do kruhu (obalen kruhem)

psí je fázový posun (určuje jestli nám vrátí reálnou nebo imaginární část).

Podle [7] bude každý obrázek filtrován na třech vlnových délkách a čtyřech orientacích. Výsledkem bude 12 filtrovaných obrázku pro jeden původní obrázek.



Obrázek 2.6: Podoba použítých Gaborových filtrů. Vlevo reálná část, vpravo imaginární část.

Po použití filtru na obrázek vzniknou dvě matice, jedna pro reálnou část a jedna pro část imaginární. Prvky matic se vybírají po dvojicích, jeden prvek z reální části a jeden z části imaginární, vždy se stejnými souřadnicemi a je na ně pohlíženo jako na vektory. Pro vzniklé vektory je spočtena jejich

velikost, označena jinak jako magnituda. Tato část je opakována pro každý filtr, tedy 12krát.

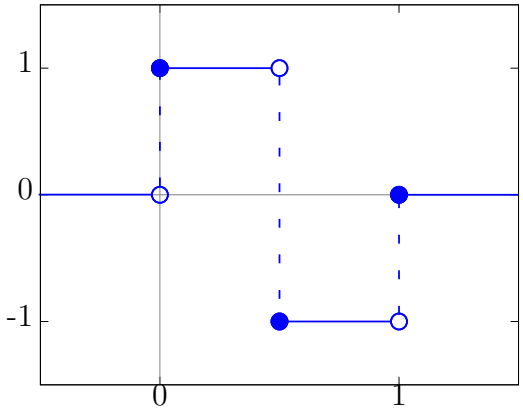
Z každého z dvanácti obrázků bude postaven 16 binový histogram skrze získané magnitudy. Vzniklé histogramy magnitud jsou zřetězeny a označeny jako příznak Gabor.

Druhý příznak zachycující fáze, je označen jako GaborQ. Opět je jsou prvky matice vybírány po dvojicích, jeden prvek z reálné části a jeden z části imaginární, vždy se stejnými souřadnicemi. U vzniklých vektorů je tentokrát pohlíženo na velikost úhlu, který svýrají s osou x. Jinak se dá tento úhel označit jako fáze.

Vzniklé fáze jsou převedeny na 16 binové histogramy pro každý z dvanácti obrázků a v konečné fázi zřetězeny.

Haar

Haarova vlnka je nejjednodušší vlnka, jejíž výhodou je především rychlý výpočet. Vlnka je realizována dvěma jednotkovými skoky, z čehož vzniknou dva obdelníkové pulzy s přechodem od kladného k zápornému.



Obrázek 2.7: Haarova vlnka.

Předpis Haarovy vlnky:

$$\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2}, \\ -1 & \frac{1}{2} \leq x < 1, \\ 0 & jinde. \end{cases}$$

Haarovy vlnové filtry jsou schopny extrahovat charakteristiky daných vlastností obrázku, jako jsou například hrany nebo změny v textuře. Při

zpracovávání průměrné intenzity oblastí je snížena citlivost na šum a změny jasu. Velká množina Haarových filtrů se skládá z filtrů s různým počtem obdelníkových oblastí a s různými orientacemi vzhledem k vyzdvyžení různorodých texturových informací obrázku. Haarův vlnový filtr nabízí jednoduché a efektivní získávání informací z obrázků.

Základní Haarův vlnový filtr bere v potaz přilehlé obdelníkové oblasti v dané části obrázku a počítá rozdíl intenzit mezi nimi.

Podle [7] bude Haarova vlnka generovat konvoluční blok s Haarovými filtry na třech rozdílných orientacích (horizontální, diagonální a vertikální) použité na obrázky čtyř různých velikostí. Různé velikosti obrázku jsou získány postupným zmenšováním podle vzorce 2.2. Vzorec je přízpůsoben, aby v případě nejmenšího rozměru obrázku byla velikost nejmenší ze stran 64 pixelů.

Je definovan *factor_resize* jako faktor změny velikosti obrázku, *min_size* jako velikost menší strany obrázku, *scales* jako požadovaný počet různých velikostí obrázků.

$$factor\_resize = \left(\frac{64}{min\_size} \right)^{\frac{1}{scales}} \tag{2.2}$$

Při velikosti vstupního obrázku 480×360 , vyjde faktor změny velikosti 0.562. Další rozměry obrázku tedy budou 270×202, 152×114 a 85×64.

Haarovy filtry:

-1	-1
1	1

Vertikální

-1	1
-1	1

Horizontální

-1	1
1	-1

Diagonální

Z článku [7] není přesně zřejmé jak autoři příznak vypočítali. V práci je zaměřeno na dva možné postupy. První možností je matici, vzniklou po aplikaci filtru z každé velikosti i orientace, převést na 16ti prvkový vektor. Tímto vyjde 12 šestnácti prvkových vektorů pro jeden obrázek, které jsou v konečné fázi zřetězeny.

Dalsí možností je udělat skrz výslednou matici její průměr. Výsledným vektorem bude v tomto případě 12ti prvkový vektor průměrů.

Příznak HaarQ používá stejný filtr jako Haar s tím rozdílem, že po aplikaci filtru jsou přepočítány jednotlivé prvky matice. Při hodnotě menší než 0 je prvek nahrazen -1, a při hodnotě větší než 0 je nahrazen 1. Z výsledné

matice je utvořen její průměr. Opět nám pro jeden obrázek vznikne 12ti prvkový vektor průměrů.

2.2 Vzdálenosti

Pro výpočet vzdáleností vektorů se používají následující metody Kullaback-Leibler divergence dále jen KL - divergence, χ^2 statistika, L1 - vzdálenost a L2 - vzdálenost. Na RGB a HSV je nejlépší použít L1 zatímco pro LAB je nejvhodnější KL - divergence.

Problém s KL - divergencí nastává pouze tehdy, když se histogramy neshodují v nulách. Jeden předpoklad pro fungování tohoto vzorce je totiž, že když je $Q(i) = 0$ tak zároveň musí být i $P(i) = 0$.

Kullaback-Leibler divergence:

$$D_{KL}(P||Q) = \sum_i P(i) \log_e \left(\frac{P(i)}{Q(i)} \right) \tag{2.3}$$

L1 (jinak označováno jako Manhattan):

$$L_1 = \sum_{i=1}^{N} |x_i - y_i| \tag{2.4}$$

L2 (jinak označováno jako Euklidovská vzdálenost)

$$L_2 = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{2.5}$$

2.3 Kombinace vzdáleností

Nejjednodušším přístupem ke zkombinování vzdáleností od různých deskriptorů je, aby jednotlivé vzdálenosti přispívaly rovnoceně. Z tohoto důvodu je potřeba vzdálenosti přeškálovat na jednotné měřítko.

I_i představuje i-tý obrázek s N příznaky. $d^k_{(i,j)}$ představuje vzdálenosti mezi příznaky f^k_i a f^k_j. Při snaze zkombinovat všechny vzdálenosti příznaků mezi obrázky I_i a I_j, tedy $d^k_{(i,j)}$, $k = 1, ..., N$ je třeba dbát na to, že v praxi nevyjdou tak, aby měly stejný poměr na výsledku. Z tohoto důvodu, předtím než jsou vzdálenosti zkombinovány, je třeba je normalizovat do jednotné formy. Na základě získaných maximálních a minimálních hodnot pro každý příznak jsou vzdálenosti přeškálovány na interval od 0 do 1. Jestliže je přeškálována vzdálenost označena jako $\tilde{d}^k_{(i,j)}$ následně může být kompletní

vzdálenost mezi obrázky I_i a I_j označena jako (2.6) Joint Equal Contribution (JEC).

$$JEC = \sum_{k=1}^{N} \frac{\tilde{d}^k_{(i,j)}}{N} \tag{2.6}$$

2.4 Přenesení klíčových slov

2.4.1 Originální algoritmus pro přenesení klíčových slov metody JEC

Pro přenesení klíčových slov je použita metoda, kdy je přeneseno n klíčových slov k dotazovanému obrázku $\tilde{I}$ od K nejbližších sousedů z trénovací sady. Je nadefinováno $I_i, i = 1, ..., K$, těchto K nejbližší sousedů je seřazeno podle vzrůstající vzdálenosti (tzn. že I_1 je nejvíce podobný obrázek). Počet klíčových slov k danému I_i je označen jako $|I_i|$. Dále jsou popsány jednotlivé kroky alogoritmu na přenesení klíčových slov.

1. Klíčová slova z I_1 jsou seřazeno podle jejich frekvence výskytu v trénovací sadě.

2. Ze všech $|I_1|$ klíčových slov z I_1 je přeneseno n nejvíce vyskytujících se v trénovací sadě k dotazovanému $\tilde{I}$. Když $|I_1| < n$ algoritmus pokračuje na krok 3.

3. Klíčová slova sousedů od I_2 do I_K jsou seřazena podle dvou faktorů

   (a) výskytu v trénovací sadě s klíčovými slovy přenesenými v kroku 2

   (b) místní frekvence (tj. jak často se vyskytují jako klíčová slova u obrázků I_2 až I_K). Jsou vybrána nejčetnější $n - |I_1|$ klíčových slov převedených do $\tilde{I}$.

Tento algoritmus pro přenos klíčových slov je poněkud odlišný od algoritmů, které se běžně používají. Jeden z běžně užívaných funguje na principu, že klíčová slova jsou vybrána od všech sousedů (se všemi sousedy je zacházeno stejně bez ohledu na to, jak jsou danému obrázku podobní). Jiný užívaný algoritmus k sousedům přistupuje váženě (každý soused má jinou váhu) a to na základě jejich vzdálenosti od testovaného obrázku. Podle testů v článku [7], přinášely tyto přímé postupy horší výsledky v porování s použitým dvoufaktorovým algoritmem pro přenos klíčových slov.

2.4.2 Dynamické přenesení klíčových slov pomocí prahování

Pro přenesení klíčových slov lze použít algoritmus kdy přeneseme pouze ta klíčová slova, která svými výskyty přesahují předepsaný práh. Je definováno $total\_keywords$ jako počet všech klíčových slov i s jejich redundantními výskyty, $frequence\_keyword$ - počet výskytů daného slova v k nejbližších sousedech a $count\_keywords$ jako počet jedinečných klíčových slov (bez redundantních výskytů).

Práh (2.7) je vyčíslen jako jedna děleno počtem všech všech klíčových slov i s jejich redundantními výskyty - 1. Následuje výpočet váhy pro dané klíčové slovo (2.8), který probíhá jako počet výskytů daného slova děleno počet všech klíčových slov i s jejich redundantními výskyty. Pokud je tato hodnota vyšší než práh, je klíčové slovo přeneseno.

$$th = \frac{1}{count\_keywords - 1} \tag{2.7}$$

$$váha = frequence\_keyword/total\_keywords \tag{2.8}$$

3 POEM

POEM (Patterns of Oriented Edge Magnitudes). Vstupem algoritmu se předpokládá šedotónový obrázek o rozměrech $m \times n$. Jelikož většinou je vložený barevný obrázek, musí být po načtení převeden na šedotónový. [5]

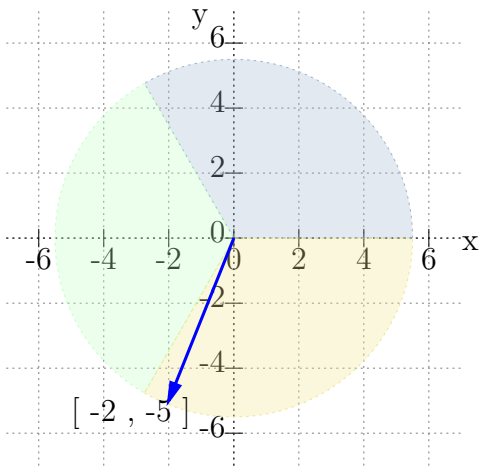
3.0.1 Výpočet gradientu a magnitudy

Nejprve je potřeba vypočítat gradient. Gradient je obecně směr růstu. Výpočet může probíhat různými způsoby. Jednou z možností je použít masku, kterou aplikujeme na vstupní obrázek. Podle některých studii jsou nejlepší jednoduché masky, jako je např. $[1, 0, -1]$ a $[1, 0, -1]^t$. Okraje obrázku se buď vypouštějí nebo se dají doplnit (opět existuje více způsobů). Výstupem jsou dva obrázky o rozměrech $m \times n$.
Na výstup se dá pohlížet také jako na vektory, kdy každý bod původního obrázku je reprezentován právě 2D vektorem. Analogicky pokud si vektory rozložíme na x a y složku dostaneme dva obrázky. Jeden který reprezentuje obrázek po použití x-ového filtru a druhý který reprezentuje obrázek po použití y-filtru. Přičemž použití y filtru by nám mělo zvýraznit hrany v y směru (svislé) a x zvýrazní hrany v x směru (vodorovné).

Magnituda je velikost směru růstu, lze si ji představit jako velikost směru růstu pro každý pixel (počítá se tedy pro každý pixel). Z toho vyplývá, že ji je možné spočítat jako velikost 2D vektorů, které byly získány při výpočtu gradientu. Zjednodušeně magnituda představuje velikost vektoru gradientu.

3.0.2 Diskretizace směru gradientu

Pokud se na gradienty bude pohlížet jako na 2D vektory je možné určit nejen jejich velikost (magnitudu) ale i jejich směr. Při výpočtu lze použít znaménkovou reprezentaci $0 - 2\pi$ nebo neznaménkovou reprezentaci $0 - \pi$. V praxi je kružnice rovnoměrně rozdělena na několik dílů (dle počtu požadovaných směrů). Počet dílů je označen písmenem d. Pro $d = 3$ znaménkovou reprezentaci to tedy bude $\left(0 - \frac{2}{3}\pi\right)$, $\left(\frac{2}{3}\pi - \frac{4}{3}\pi\right)$ a $\left(\frac{4}{3}\pi - 2\pi\right)$. Je připraveno d matic (pro každý směr jedna) a podle toho kam vektor směřuje, je umístěna jeho magnituda na souřadnice, kde se nachází v původní matici.



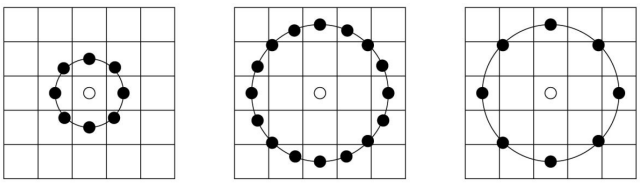
Obrázek 3.1: Diskretizace směru gradientu. Každá barva představuje jeden směr šedá: $\left(0 - \frac{2}{3}\pi\right)$, zelená:$\left(\frac{2}{3}\pi - \frac{4}{3}\pi\right)$ a žlutá: $\left(\frac{4}{3}\pi - 2\pi\right)$. Vektor $[-2, -5]$ směřuje do třetího směru, proto uložíme jeho magnitudu do třetí matice.

3.0.3 Výpočet lokálního histogramu orientace gradientů z okolí

U každého směru se vezmou jednotlivé pixely s jejich okolím a zprůměrují se jejich hodnoty. Toto okolí se nazývá cell.

3.0.4 Zakódování příznaků pomocí LBP

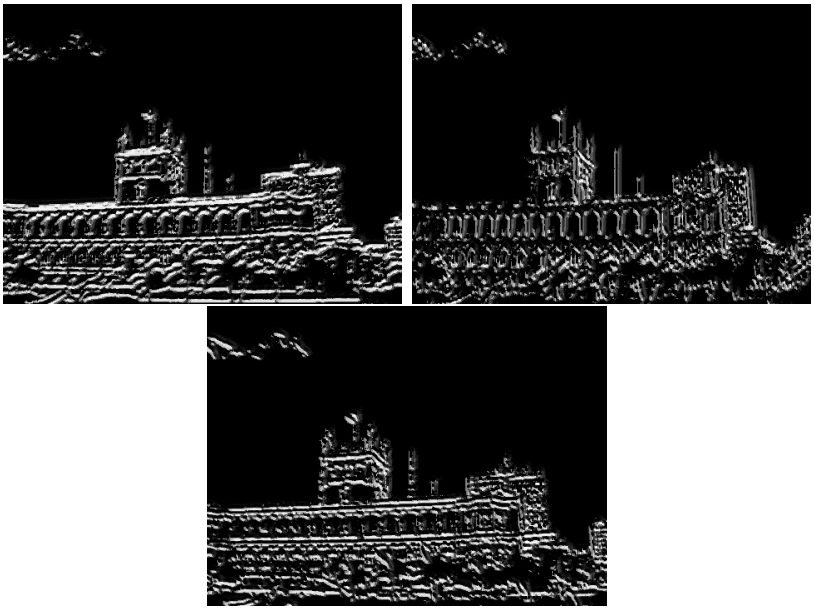
LBP operátor je aplikován na okolí každého pixelu o velikost 3×3. Oproti tomu POEM je možné aplikovat na větší okolí. Toto okolí se nazývá block, zpravidla se jedná o kruhové okolí s poloměrem L/2 (L představuje velikost blocku). Pro stanovení intenzit okolních hodnot je možné použít bilineární interpolaci. Pro zvýšení stability v téměř konstantní oblasti lze k centrálnímu pixelu přičítat malou konstantu τ.



Obrázek 3.2: Znázornění blocku Převzato z [5]

Výpočet LBP probíhá podle následujícího vzorce, kde je pixel pro který se hodnoty počítají označen písmenem c (centrální). Algoritmus následně prochází všechny okolní pixely označené písmenem x. Hodnota daného pixelu je označena jako $p(x)$ a výsledek tohoto porovnání je označen $s(x)$.

$$s(x) = \begin{cases} 1, & p(x) \geq h(c) \\ 0, & p(x) < h(c) \end{cases}$$



Obrázek 3.3: Obrázky po aplikaci LBP s použitím τ. Každý obrázek představuje jeden směr.

3.0.5 Konstrukce globálního histogramu

Obrázky získané z LBP jsou rozdeleny pravidelnou čtvercovou mřížkou. Pro každou vzniklou oblast je vypočten lokální histogram. Vzniklé histogramy jsou zřetězeny. Díky tomu jsou získány tři histogramy, pro každý směr jeden, které jsou opět zřetězeny.
Rozdělení obrázků a určování lokálních histogramů se dělá za účelem zachování informace o prostorovém rozložení jednotlivých příznaků.

3.1 Barevný POEM

Výpočet gradientu a magnitudy

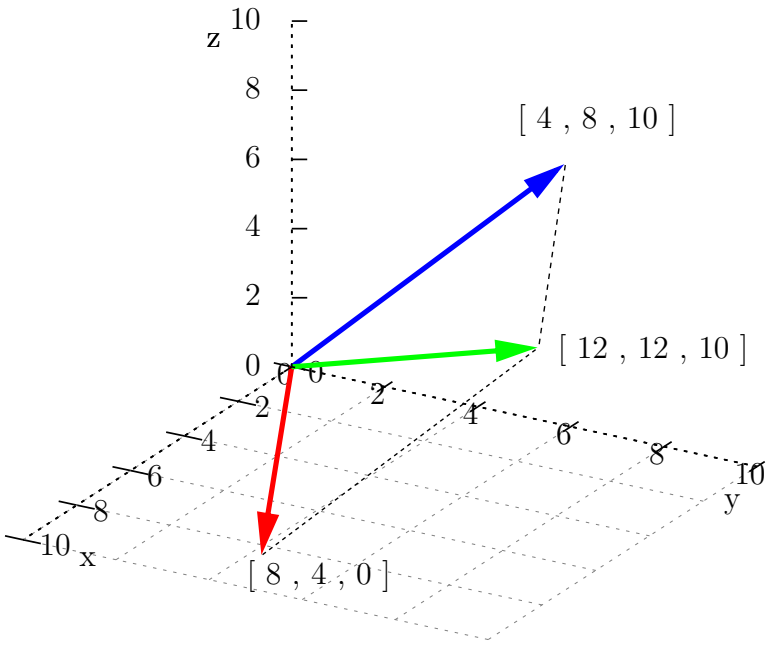
Výpočet gradientu probíhá obdobně jako u nebarevného obrázku. Pro každou ze tří složek jsou získány dvě matice filtrované maskami. Celkem bude 3×2 matic. Na matice se dá pohlížet jako na 2 vektory o 3 složkách. Vektory jsou sloučeny pomocí součtu vektorů do jednoho 3 složkového vektoru. Magnituda je opět velikost vektoru tentokrát ale v prostoru.

Formát vzniklých vektorů

$$u = [blue_x, green_x, red_x] \tag{3.1}$$
$$v = [blue_y, green_y, red_y] \tag{3.2}$$

Pomocí součtu vektorů je získán jeden třísložkový vektor:

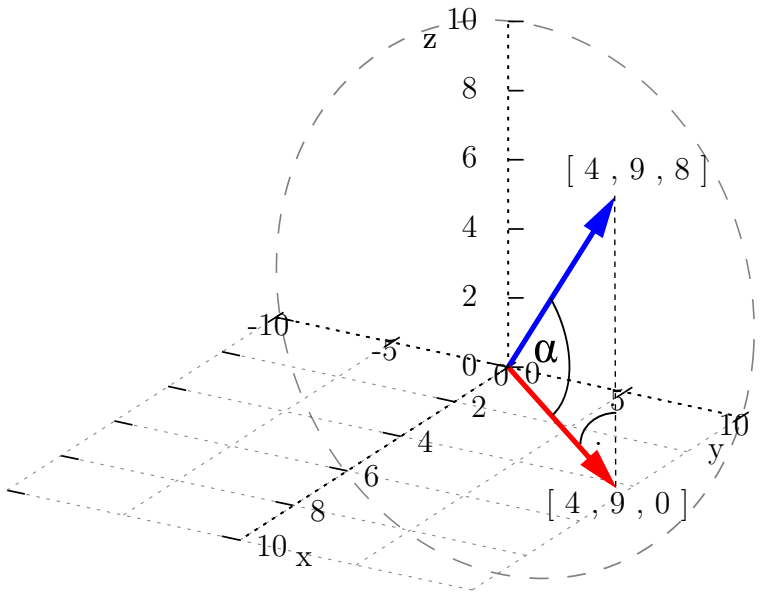
$$\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3) \tag{3.3}$$



Obrázek 3.4: Grafické znázornění součtu vektorů. Součet je tvořen z vektorů [4, 8, 10] a [8, 4, 0].

Diskretizace směru gradientu

U vektorů získaných v předchozím kroku je určena velikost úhlu mezi vektorem a ekvivalentní vektorem s vynulovanou složkou z. Následně je spočítáno do které části kružnice vektor směřuje. Pro znaménkovou reprezentaci je celkový rozsah $0 - \pi$, pro neznaménkovou reprezentaci $0 - 2\pi$.
Při neznaménkové reprezentaci a počtu směrů $d = 3$, jsou následující intervaly $\left(0 - \frac{\pi}{3}\right)$, $\left(\frac{\pi}{3} - \frac{2\pi}{3}\right)$ a $\left(\frac{2\pi}{3} - \pi\right)$.
Pro výpočet diskretizace směru při neznaménkové reprezentaci je y složka rozdělena na kladnou a zápornou část. To hraje velkou roli, pokud je y složka vektoru záporná. V tom případě je nutné nebrat úhel α, ale jeho doplněk $(180 - \alpha)$.



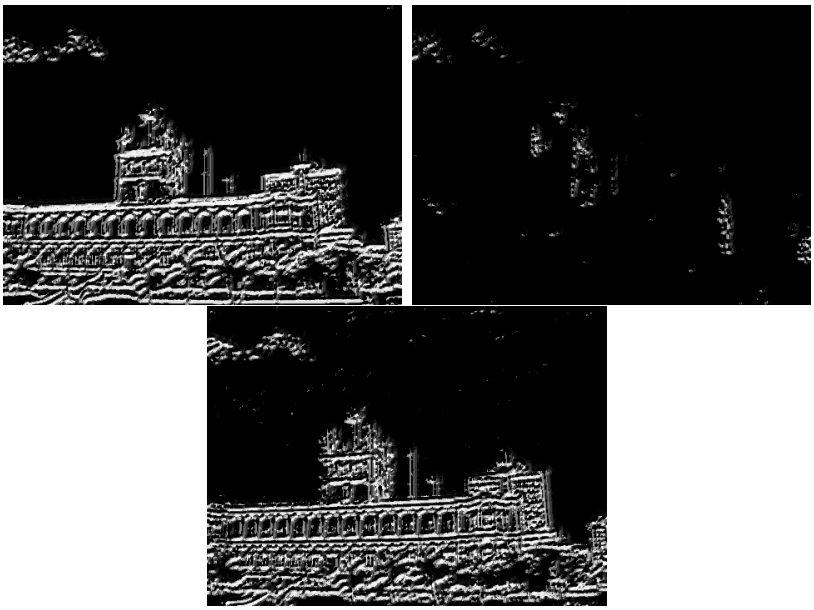
Obrázek 3.5: Grafické znázorněnní součtu vektorů. Součet je tvořen z vektorů [4, 8, 10] a [8 4, 0].

Výpočet lokálního histogramu

Zbývající postup je totožný s POEMEM. U každého směru se vezmou jednotlivé pixely s jejich okolím a zprůměrují se jejich hodnoty.

Zakódování příznaků pomocí LBP

LBP operátor se aplikuje na kruhové okolí s poloměrem L/2 označeném jako block. Pro zvýšení stability je k centrálními pixelu přičítána malá konstanta τ.



Obrázek 3.6: Obrázky po aplikaci LBP s použitím τ. Každý obrázek představuje jeden směr.

Konstrukce globálního histogramu

Obrázky získané z LBP jsou rozdeleny pravidelnou čtvercovou mřížkou. Pro každou vzniklou oblast je vypočten lokální histogram. Vzniklé histogramy jsou zřetězeny přes všechny tři směry.

4 Testovací databáze

Pro natrénování a následné testování byla použita data z databází IAPRC a ESP. Obě databáze jsou standardně používané v automatické anotaci obrázků a jejich použít umožní srovnání s literaturou. Kolekce obrázků na natrénování musí být pečlivě vybrána, aby zahrnovala co možná největši okruh z různých témat.

4.1 iaprtc12

Sada iaprtc12 je kolekce obrázků přírodních scén, které zahrnují různé sporty a akce, fotografie lidí, zvířat, měst, krajin a mnoho jiných aspektů současného života. Data obsahují 20 000 obrázků ve formátu *jpg* s celkovým počtem 291 klíčových slov. Ke každému obrázku jsou přiložena metadata ve formátu *XML*, která obsahují informace o obrázku v různých jazycích. Kromě angličtiny je tam i například španělština nebo němčina. V metadatatech ovšem nenajdeme klíčová slova tak, jak bychom si je představovali, ale v různých tagách nalezneme například titulek obrázku, který může vypadat například The Plaza de Armas. V tagu description je například a woman and a child are walking over the square. Spolu s databází jsme zíkali i klíčová slova která byla z přiložených xml extrahována.
K jednomu obrázku je v průměru přiřazeno 5.7 klíčových slov. Pro trénovaní bylo použito 17 664 obrázků, na následné testování jich bylo použito 1960 [12].

Obrázek 4.1: Ukázka obrázku s klíčovými slovy: front lake man mountain rock sky summit

4.2 ESP

Sada ESP obsahuje širokou škálů snímku s anotacemi, ze kterých byla použita jen malá část. Konkrétně 18 689 obrázků na trénování a 2061 na testování. Ke každému obrázku je přiřazen soubor ve formátu *desc*, který obsahuje anglické anotace. Z celkových 269 klíčových slov je k jednomu obrázků přiřazeno v průměru 4.6 slov.

Obrázky získaly svá klíčová slova pomocí ESP game, což je hra, která funguje pouze online. V principu spojí dva hráče, kteří nemají možnost spolu komunikovat. Následně je oběma hráčům zobrazen stejný obrázek, který musí popsat co nejvíce různými výrazy v angličtině. V případě, že se hráči shodnou, počítač předpokládá že mu poskytli pravdivou informaci o tom, co se na obrázků nachází. Tak si tuto anotaci uloží do databáze a hráči získají body.

Obrázek 4.2: Ukázka obrázku s klíčovými slovy: brown chart country map old orange ship white

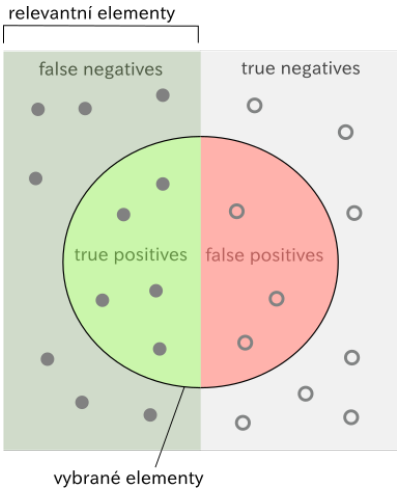
5 Evaluační metriky

Kvalitu a úspěšnost klasifikátoru udává přesnost (precision) a úplnost (recall). Přesnost udává jak moc jsou výsledky relevantní a úplnost kolik skutečně relevantní výsledků bylo přiřazeno. V případě že přesnost převyšuje úplnost, jsou klíčová slova sice korektní, ale je jich málo. V opačném případě při převyšující úplnosti bylo získáno hodně klíčových slov, ale málo z nich je korektních. Proto je snaha získat obě čísla co nejvyšší. Níže jsou uvedeny dva postupy pro výpočet přesnosti a úplnosti [6]. V práci je použitý postup per word, stejně jako je uvedeno v článku [7].

Počet nenulových slov značí počet slov, která byla při anotaci použita alespoň jednou.

5.1 Přesnost a úplnost pro celý klasifikátor

TP (True Positive) značí klíčová slova, která měla být k obrázku klasifikátorem přiřazena a skutečně mu přiřazena byla. *FP* (False Positive) určuje klíčová slova, která k danému obrázku nepatří, avšak klasifikátor je přiřadil. *FN* označuje klíčová slova, která k obrázku patří a klasifikátor je nepřiřadil.



Obrázek 5.1: Znázornění precision a recall.

$$Prec = \frac{TP}{TP + FP} \tag{5.1}$$

$$Rec = \frac{TP}{TP + FN} \tag{5.2}$$

5.2 Přesnost a úplnost - per word

Zpracování precision a recall probíhá pro každé slovo v testovací sadě (proto také per word). Výpočet probíhá jako porovnání anotací přidělených člověkem s anotacemi přidělenými klasifikátorem. w_{auto} představuje počet obrázků, kterým bylo dané slovo přiřazeno klasifikátorem, w_{human} počet obrázků, kterým bylo dané slovo přiřazeno člověkem a $w_{corrrectly}$ počet obrázků, kterým bylo slovo přiřazené správně.

Recall (5.3) je počet obrázků správně anotovaných s daným slovem děleno počtem obrázků, kterým bylo toto slovo přiděleno v anotaci člověkem. Precision (5.4) je počet správně anotovaných obrázků s tímto slovem děleno celkovým počtem anotovaných obrázků s tímto slovem (správně nebo ne).

$$Rec = \frac{w_c}{w_h} \tag{5.3}$$

$$Prec = \frac{w_c}{w_a} \tag{5.4}$$

Výsledná precision a recall se počítá jako průměr dosažených výsledků pro jednotlivá slova.

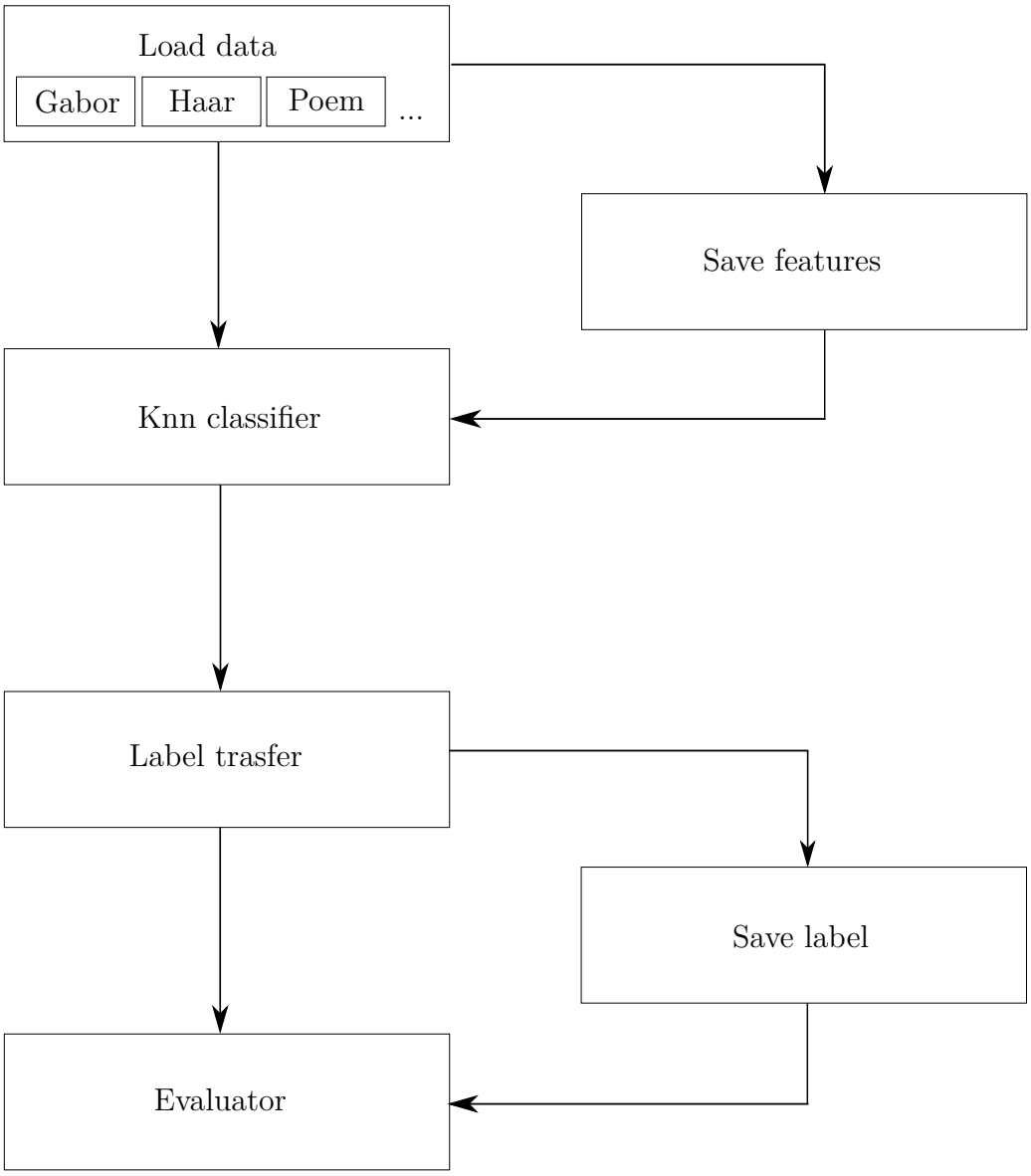
5.3 F-measure

F-measure je měřítkem úspěšnosti klasifikace a je definována jako harmonický průměr precision a recall. (5.5) Jak již bylo výše zmíněno, klasifikátor dosahuje nejlepších výsledků právě tehdy, když precision a recall dosahují co nejvyších hodnot, avšak zároveň jsou vybalancovány. Pokud tedy bude klasifikátor optimalizován pouze pro jednu z techto hodnot, a tím znevýhodněna druhá, harmonický průměr se rychle sníží. V literatuře na anotaci obrázku se však F-measure běžně neobjevuje.

$$F - measure = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.5}$$

6 Návrh systému

Systém byl navržen jako modulární z důvodu snadné obměny některé z častí. Problém automatické anotace obrázků se dá rozdělit do několika částí. Extrahování příznaků z dat, samotná klasifikace a v neposlední řadě vyhodnocení úspěšností klasifikace. Pro efektivnější použití bylo navrženo i ukládání mezivýsledků. Návrh systému je zobrazen na následujícím obrázku.



Obrázek 6.1: Návrh systému.

7 Implementace

7.1 Použité programové prostředky

Program byl navržen na operační systém Linux. Jako programovací jazyk byl zvolen Python a to z důvodu jeho jednoduchého použití, což je na prototyp, jako je tento velice výhodné na časovou náročnost a je pro něj k dispozi hodně knihoven.

Pro spuštění programu je třeba mít nainstalovaný Python ve verzi 2.7.12 s NumPy verze 9, knihovnu openCV verzi 3.1 a vědeckou knihovnu scipy verze 0.17.0. Vzhledem k náročnosti programu je pro jeho spuštění nutné mít v počítači alespoň 16 GB RAM paměti a to především z důvodů ukládání mezivýsledků pomocí modulu *pickle*, což je modul pro serializaci objektů. Následující postupy jsou uvedeny pro operační systém Linux, a tak se mohou od postupu na jiném operačním systému lišit.

7.1.1 OpenCV

OpenCV (Open source computer vision) je knihovna vydávána pod licencí BSD a je volně k dispozici jak pro akademické účely, tak pro komerční použití. Je vhodná pro použití v C++, C, Python a Javě. Podporuje operační systémy Windows, Linux, Mac OS, iOS a Android.

Knihovna byla navrhnuta pro výpočetní efektivitu v oblasti počítačového vidění a zpracování obrazu se zaměřením na zpracování obrazu v reálném čase. Z důvodu optimalizace byla napsána v C/C++.

Knihovna OpenCV je dostupná na adrese: http://opencv.org/

7.1.2 Scikit

Scikit-image je vědecká knihovna algoritmů pro zpracování obrazu. Je k dispozici zdarma a bez omezení s licencí BSD. Poskytuje dobře zdokumentované API v programovacím jazyce Python a je vyvíjena aktivním mezinárodním týmem spolupracovníků. [11]

7.2 Modulové jednotky programu

Pro snadné spuštění programu je pro uživatele připraven skript *run.py*, který postupně spouští jednotlivé moduly. Moduly je možné spouštět i odděleně.

7.2.1 Config

Při spuštění programu je nejdříve načten konfigurační soubor, který obsahuje jeho veškerá nastavení. Mimo jiné zde najdeme cesty k souborům ze kterých program načítá data nebo cesty k soubourům do kterých data naopak ukládá. Dále jsou v cofingu uvedené jednotlivé metody s očekávanou hodnotou *True* nebo *False* v závislosti zda se mají použít nebo ne a nastavení vzdáleností které se mají na jednotlivé příznaky použít.

7.2.2 Load data

Modul pomocí funkce *load_pictures* načte obrázky z listů uvedených v configu (*TRAIN_LIST* a *TEST_LIST*). Následně je na všechny obrázky zavolána metoda *load_features*, která načte příslušné příznaky. Nakonec jsou celé stuktury listů testovacích nebo trenovacích obrázku uloženy do souboru, dle configu *DATAFILE_TRAIN* popřípadě *DATAFILE_TEST*.

Extrakce příznaků

Jednotlivé výpočty příznaků jsou rozděleneny do zvláštních modulů, aby byla obměna jejich výpočtu snadno nahraditelná. V každém modulu je stěžejní pouze funkce *count_* a název příslušné metody (např. *count_haarq*), která je volána právě z modulu load data.

Při počítání barevných histogramů byl zjištěn překvapivý poznatek. V případě kdy je histogram jako datová struktura *list* a až výsledný histogram převeden do *numpy array* je rychlost programu nesrovnatelně větší oproti tomu, když jsou histogramy vytvořeny rovnou jako *numpy array*.

Za zmíňku také stojí převádění výsledných matic na vektor, kdy v případě převádění vektoru do 16 prvkové podoby výsledné číslo může dosahovat i nejvyší hodnoty je tato nejvyšší hodnota posunuta na poslední index vektoru. Toto řešení se mimo jiné zobrazuje v případě příznaku HaarQ.

V případě získávání příznaku z POEMU a barevného POEMU byl obrázek před výpočtem zmenšen na polovinu z důvodu rychlejšího průběhu.

U většiny příznaků byla použita knihovna OpenCV. U Gabora byla použita knihovna scikit, která si ksize určuje podle parametrů vlny tz. tento parametr nezadáváme.

7.2.3 Knn classifier

V tomto modulu probíhá počítání vzdáleností mezi jednotlivými příznaky (vektory). Ve funkci *count_all_distance* je v jednom běhu cyklu zjištěna

jak vzdálenost mezi příznaky testovaného obrázku se všemi obrázky z trénovací sady, tak i určeno jejich maximum a minimum. Dále je volána metoda *count_jec*, která vzdálenosti za pomocí zjištěného maxima a minima přeškálujuje na interval 0 až 1. Naškálovanou hodnotu přičte do celkové sumy vzdáleností. V konečné fázi je suma vzdáleností podělena počtem příznaků. Vznikne tak výsledná vzdálenost JEC.

7.2.4 Label transfer

Modul pro přenost klíčových slov má dvě modifikace. Mezi modifikacemi je možno přepínat pomocí parametru *LABEL_TRANSFER* v configu.

Při variantě kdy je přenesení provedeno pomocí originálního algoritmu pro přenesení klíčových slov, jsou předpočteny četnosti klíčových slov v trénovacích datech pomocí funkce *count_keyword_frequency_train_set*, následováno předpočtením frekvence výskytu klíčových slov s ostaními klíčovými slovy ve funkci *frequency_word_with_other_word_dictionary*. Pro každý obrázek následuje funkce *label_transfer*. V této funkci se již přiřazují samotná klíčová slova od prvního souseda. Pokud je zjištěno, že od prvního souseda je dostatek klíčových slov, je funkce je ukončena. V opačném případě pokračuje do funkce *add_keywords_from_neighbors*, která dodá potřebná klíčová slova od dalších k nejbližších sousedů. Nakonec jsou klíčová slova uložena do souboru dle configu.

Druhou variantou je dynamické přenesení klíčových slov pomocí prahování. Zde je rovnou spuštěna funkce *label_transfer* pro každý obrázek a jsou přenesena klíčová slova přesahující hodnotu práhu. V konečné fázi jsou klíčová slova uložena do souboru dle configu.

Při přenášení klíčových slov pomocí prahování na databázi ESP při pěti nejbližších sousedech byl objeven problém při počtání prahu. V případě kdy má K nejbližších sousedů pouze jedno totožné klíčové slovo, ve vzorečku pro práh by jmenovatel vyšel nula. Proto je doplněna podmínka a v této situaci je jediné klíčové slovo přeneseno i bez výpočtu prahu.

7.2.5 Evaluator

Tento modul vyhodnotí úspěšnosti anotace. Jako první získáme všechna klíčová slova a to pomocí funkce *getKeywords*. Pokračujeme získáním anotovaných dat ve funkci *read_data_from_file*. Následně je pro každé klíčové slovo spočítána přesnost a úplnost. Tyto hodnoty jsou popsány v sekci Vyhodnocení výsledků.

7.3 Doby běhu programu

Doby běhu programu jsou v řádu hodin. S nejdelším časem je možné se setkat u modulu Load data. Jeho doba běhu závisí na načítaných příznacích a může přesahovat i 12 hodin. Za zmíňku stojí i modul *Knn classifier*, který následuje *label transfer* a jejich doba běhu se může pohybovat kolem 1 hodiny. Jak již bylo výše zmíněno, doba běhu se dá ovlivnit použitými konstrukcemi. Například rozdíl použití *listu* a *numpy array*. Pokud by se tento prototyp překlápěl do efektivnějšího programovacího jazyku jako je například C, byla by doba běhu optimálnější. V tomto případě byla důležitější rychlost vývoje.

8 Vyhodnocení výsledků

8.1 Srovnání výsledků

Přesné parametry se kterými autoři článku [8] dosahovali nejlepší výsledků nebyly zjištěny, proto bylo třeba je u některých příznaků zkoušet metodou pokus omyl.

8.1.1 Gabor - porovnání parametrů

Gabor patří mezi příznaky u kterých nebyl uveden přesný postup zpracování příznaků. Byly pouze zmíněny tři vlnové délky a čtyři orientace. Z tabulky 8.1 je zřejmé, že algoritmus dosahoval nejlepší výsledků při použití filtrů s vlnovými délkami 0.25, 0.5, 1.0 a orientacemi 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3}{4}\pi$. Tyto nejúspěšnější parametry byly také použity v metodě JEC.

Parametry	$P_{\%}$	$R_{\%}$	N
lambda 0.25, 0.5, 1.0 sigma 1 theta 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3}{4}\pi$	9.9	6.8	151
lambda 2, $2\sqrt{2}$, 4 sigma 1 theta 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3}{4}\pi$	8.5	5.7	143

Tabulka 8.1: Gabor s knihovnou scikit na datech iaprtc12.

8.1.2 Haar - porovnání parametrů

U Haara byly vyzkoušeny dvě možnosti vytváření deskriptoru. Deskriptor jako vektor 12x16 prvků je označena metoda, kdy byla výsledná matice, z každé velikosti i orientace převedena na 16ti prvkový vektor.

Dalsí možností je udělat skrz výslednou matici její průměr. Výsledným vektorem bude v tomto případě 12ti prvkový vektor průměrů. Z následující tabulky je vidět že jednodušší vektor průměrů je úspěšnější, proto také bude použit u metody JEC.

Parametry	$P_{\%}$	$R_{\%}$	N
Desktiptor jako vektor 12×16 prvků	2.9	2.1	63
Deskriptor jako vektor průměrů	5.8	4	114

Tabulka 8.2: Haar na datech iaprtc12.

8.1.3 Přiřazování klíčových slov pomocí prahování

U této metody vyšlo vyhodnocení klasifikace nejlépe pro 8 sousedů.

	5 sousedů			8 sousedů			10 sousedů		
Metoda	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N
RGB	20.1	9	178	15.7	12.9	195	14.3	15.3	205
LAB	17.1	8.8	156	14.3	12.4	172	14.4	14.7	185
HSV	21	11.4	191	17.2	16.4	213	14.9	18.6	221
RGB, LAB, HSV	21.9	11.2	188	18.8	16.9	218	16.1	19	221
JEC	23.8	12.3	189	19.7	17.7	212	17.1	19.9	215

Tabulka 8.3: Výsledky získané přiřazování klíčových slov s práhem na datech iaprtc12. P značí přesnost, R úplnost a N počet nenulových klíčových slov.

	5 sousedů			8 sousedů			10 sousedů		
Metoda	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N
RGB	17.6	8.9	170	13.1	12.6	195	11.1	14.4	202
LAB	7.9	4.1	95	7.7	6.5	121	7.4	8.2	133
HSV	19	11	192	14	14.8	205	11.9	16.4	207
RGB, LAB, HSV	20.4	11	182	14.4	14.8	201	12.1	16.9	206
JEC	21.6	11.6	195	15.3	15.3	206	13.1	17.2	210

Tabulka 8.4: Výsledky získané přiřazování klíčových slov s práhem na datech esp. P značí přesnost, R úplnost a N počet nenulových klíčových slov.

8.1.4 Konečné výsledky a srovnání s literaturou

Při konečném srovnání výsledků z článku [8] je možné si povšimnout, že ačkoliv byl u barevných příznaků přesně určen postup vytvoření histogramu, nebylo dosaženo stejně úspěšných výsledků. Je ovšem možné, že autoři obrázky před klasifikací předzpracovávali (změna velikostí obrázků atd.). Následuje výpis parametrů, díky kterým byly výsledky získány.

klasifikátor

- Počet sousedů: 5
- Počet klíčových slov: 5

- Přenášení klíčových slov pomocí originálního algoritmu uvedeného u metody JEC

RGB, HSV

- 3×16 binový histogram
- Vzdálenost: L1

LAB

- 3×16 binový histogram
- Vzdálenost: KL

Gabor, GaborQ

- 12×16 prvkový vektor
- Vzdálenost: L1
- Lambda: 0.25, 0.5, 1.0
- Sigma: 1
- Theta: 0, $\frac{\pi}{4}$, $\frac{\pi}{2}$, $\frac{3}{4}\pi$

Haar, HaarQ

- 12 prvkový vektor
- Vzdálenost: L1
- Filtry: [1.0, 1.0], [-1.0,-1.0]; [-1.0, 1.0], [-1.0, 1.0]; [1.0, -1.0], [-1.0, 1.0]

POEM

- Vzdálenost: L1
- Počet směrů: 3
- Velikost cellu: 3
- Velikost blocku: 8
- Tau: 4

Barevný POEM

- Vzdálenost: L1
- Počet směrů: 3
- Velikost cellu: 3
- Velikost blocku: 8
- Tau: 4

	IAPRTC12			ESP		
Metoda	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N
RGB	14.1	9	167	17	13.2	209
LAB	12.7	7.5	148	6.1	5.2	117
HSV	16.7	10.9	181	18.2	14.8	211
RGB, LAB, HSV	17.4	11.1	178	18.7	14.8	209
Gabor	8.1	4.7	126	14.4	11.4	194
GaborQ	6.9	4.8	133	11.6	9.1	187
Haar	5.8	4	114	10.2	8.4	178
HaarQ	5.8	4.4	123	9.4	7.3	169
JEC	17.3	11.6	182	19.6	14.8	210
POEM	21.5	12.8	189	0	0	0
RGB, LAB, HSV, POEM	21.8	13.8	187	3.3	3.3	85
Barevný POEM	21	12.4	184	17.4	13	200

Tabulka 8.5: Výsledky získané v rámci práce. P značí přesnost, R úplnost a N počet nenulových klíčových slov. V případě kombinace RGB, LAB, HSV a POEMU bylo použito JEC.

	IAPRTC12			ESP		
Metoda	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N
RGB	20	13	189	21	17	221
LAB	22	14	194	20	17	221
HSV	18	12	190	18	15	217
Haar	17	8	161	21	14	210
HaarQ	16	10	173	19	14	210
Gabor	14	9	169	16	12	199
GaborQ	8	6	137	14	11	205
JEC	25	16	196	23	19	227

Tabulka 8.6: Výsledky z literatury [8].

Obrázek 8.1: Ukázka obrázku ze sady ESP.

Získání klíčových slov	**Klíčová slova**
Originální klíčová slova	circle coin face head metal money old round silver
JEC s deskriptory RGB, HSV, LAB, Gabor, GaborQ, Haar a HaarQ	man old metal coin money
JEC s deskriptory POEM, RGB, HSV a LAB	
POEM rozšířenou na všechny barevné kanály	
JEC s deskriptory RGB, HSV, LAB, Gabor, GaborQ, Haar a HaarQ s přenášením klíčových slov pomocí prahování	old money coin
JEC s deskriptory POEM, RGB, HSV a LAB s přenášením klíčových slov pomocí prahování	
POEM rozšířenou na všechny barevné kanály s přenášením klíčových slov pomocí prahování	

Tabulka 8.7: Klíčová slova k obrázku 8.1

9 Závěr

V práci byla řešena automatická anotace obrázků pomocí spojování barevných a texturových příznaků. U prvního řešení s metodou JEC probíhalo vyhodnocení a klasifikace příznaků odděleně a pak byla výsledná klasifikace spojena z několika částí. V druhém případě bylo spojení příznaků interpretováno jako jeden společný příznak, kdy byl POEM rozšířen na všechny barevné kanály.

V teoretické části byly popsány a rozebrány nízkoúrovňové příznaky barva a textura. U barvy se zabývalo barevnými modely RGB, LAB a HSV. U textury to byl Gabor, Haar a POEM. Dále byla pozornost kladena na přenášení klíčových slov za pomocí práhu, kdy musela frekvence výskytu u sousedů převýšit stanovený práh.

V realizační části byl navržen a implementován program v jazyce Python pro automatickou anotaci obrázků, který využívá právě vyše zmíněné barevné prostory a reprezentace textur. Na základě naměřených výsledků byly laděny optimální parametry. V rámci práce byla prostudovaná a použita knihovna openCV. Funkčnost programu byla otestována na databázích iaprtc12 a ESP.

Po vyhodnocení dosažených výsledků a porovnání z literaturou bylo zjištěno, že nebylo dosaženo tak vysokých hodnot přenosti a úplnosti klasifikátoru jako uvádějí autoři článku [8]. V některých případech se dá předpokládat, že to může být z důvodu neznalosti přesných parametrů, avšak u barevných příznaku (RGB, HSV a LAB), kdy byly uvedené přesné parametry a postup sestavení příznaku je nesouhlasnost výsledků vysoce zvláštní.

Do budoucna jako vylepšení se nabízí úprava barevného POEMU, kdy by se diskretizace směru gradientu nezaměřovala ve třírozměrném prostoru pouze na 3 směry v rovině, ale brala by v potaz celé postavení vektoru v rovině, tím pádem by vznikla kupole o 9 částech (směrech). Co se týká JEC za vylepšení by určitě stálo použít další deskriptory.

10 Použité zkratky

AIA Automatic image anotation.

JEC Joint equal contribution

RGB Barevný model Red, Green, Blue (červená, zelená, modrá).

LAB Barevný model.

HSV Barevný model.

POEM Patterns of oriented edge magnitudes.

LBP Local binary pattern.

OpenCV Open source computer vision.

BSD Licence pro svobodný software, umožňující volné šíření softwaru.

KL - divergence Kullaback-Leibler divergence.

Literatura

[1] Carneiro, G. et al. Supervised Learning of Semantic Classes for Image Annotation and Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. March 2007, 29, 3, s. 394–410. ISSN 0162-8828. doi: 10.1109/TPAMI.2007.61. Dostupné z: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4069257.

[2] Cruse, H. *Neural Networks as Cybernetic Systems*. Thieme Medical Publishers, Incorporated, 1996. ISBN 0865776725.

[3] Gong, Y. et al. Deep Convolutional Ranking for Multilabel Image Annotation. *CoRR*. 2013, abs/1312.4894. Dostupné z: http://arxiv.org/abs/1312.4894.

[4] HUTÁREK, B. J. Klasifikace objektu v obraze podle textury. Master's thesis, Vysoké učení technické v Brně, Brno, 2010. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117319.

[5] Košař, V. Srovnání deskriptorů pro reprezentaci obrazu. Master's thesis, Západočeská univerzita v Plzni, Plzeň, 2015. Dostupné z: https://dspace5.zcu.cz/bitstream/11025/17883/1/A13N0110P.pdf.

[6] Lavrenko, V. – R.Manmatha – Jeon, J. A Model for Learning the Semantics of Pictures. 2004, s. 553–560. Dostupné z: http://papers.nips.cc/paper/2474-a-model-for-learning-the-semantics-of-pictures.pdf.

[7] Makadia, A. – Pavlovic, V. – Kumar, S. *A new baseline for image annotation*. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-88690-7_24. Dostupné z: http://dx.doi.org/10.1007/978-3-540-88690-7_24. ISBN 978-3-540-88690-7.

[8] Makadia, A. – Pavlovic, V. – Kumar, S. Baselines for Image Annotation. *International Journal of Computer Vision*. 2010, 90, 1, s. 88–105. ISSN 1573-1405. doi: 10.1007/s11263-010-0338-6. Dostupné z: http://dx.doi.org/10.1007/s11263-010-0338-6.

[9] Murthy, K. *GABOR FILTERS : A PRACTICAL OVERVIEW* [online]. 2014. [cit. 2017/05/12]. wordpress. Dostupné z: https://cvtuts.wordpress.com/2014/04/27/gabor-filters-a-practical-overview/.

[10] Rosebrock, A. *Local Binary Patterns with Python & OpenCV* [online]. 2015. [cit. 2017/05/12]. pyimagesearch. Dostupné z: http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/.

[11] Walt, S. et al. *scikit-image: image processing in Python* [online]. 6 2014. [cit. 2016/03/09]. Dostupné z: http://dx.doi.org/10.7717/peerj.453.

[12] Von Ahn, L. – Dabbish, L. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, s. 319–326. ACM, 2004.

[13] Zhang, D. – Islam, M. M. – Lu, G. A review on automatic image annotation techniques. *Pattern Recognition*. 2012, 45, 1, s. 346 – 362. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2011.05.013. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0031320311002391.

A Uživatelská dokumentace

Pro spuštění programu je třeba mít nainstalovaný python ve verzi 2.7.12 s NumPy verze 9, knihovnu openCV verzi 3.1 a vědeckou knihovnu scipy verze 0.17.0. Vzhledem k náročnosti programu je pro jeho spuštění nutné mít v počítači alespoň 16 GB RAM paměti. Následující postupy jsou uvedeny pro operační systém Linux. Na jiném operačním systému se mohou postupy lišit.

Veškeré nastavení aplikace probíhá pomocí souboru *config.py*

Parametry configu:

- *TRAIN_LIST* - Trénovací list obrázků, který by měl obsahovat cesty k obrázkům trénovací sady a jejich klíčová slova ve formátu *cesta_k_obrazku; klíčové_slovo další_klíčové_slovo*. Předpokládaný formát je *.txt*.
- *TEST_LIST* - Testovací list, který by měl obsahovat cesty k obrázkům testovací sady a jejich klíčová slova ve formátu *cesta_k_obrázku; klíčové_slovo klíčové_slovo*. Předpokládaný formát je *.txt*.
- *DATAFILE_TRAIN* - Soubor do kterého budou uloženy příznakovové vektory, načtených obrázků z trénovací sady. Předpokládaný formát *.pyc*.
- *DATAFILE_TEST* - Soubor do kterého budou uloženy příznakovové vektory z načtených obrázků z testovací sady. Předpokládaný formát *.pyc*.
- *PICTURE_RESULT* - Obsahuje název obrázku a klíčová slova přiřazena klasifikátorem.
- *PICTURE_ALL_KEYWORDS* - Obsahuje názvy obrázku s přiřazenými slovy od klasifikatoru i s se slovy přiřazené člověkem. Ve formátu *cesta_k_obrázku;originální_klíčová_slova;klíčová_slova_přiřazena_klasifikátorem*
- *KEYWORDS_RESULT* - Cesta k souboru, který obsahuje výsledky klíčových slov, jejich přesnost a úplnost.
- *COUNT_NEIGHBORS* - Počet sousedů.
- *COUNT_KEYWORDS* - Počet klíčových slov.
- *LABEL_TRANSFER* - Určuje způsob přiřezování klíčových slov. Při hodnotě *TH* bude použíté prahování, při jakékoliv jiné hodnotě bude použit originální algoritmus metody JEC.

Dále jsou v cofingu uvedené jednotlivé metody s očekávanou hodnotou *True* nebo *False* v závislosti zda se mají použít nebo ne a jejich parametr deskriptor_DISTANCE (například *RGB_DISTANCE*), který určuje jaká vzdále-

nost pro porovnání vektorů konkrétního desktriptoru se má použít. Na výběr je z *L1*, která představuje Manhattnovskou vzdálenost, *L2* představující Euklidovskou vzdálenost a *KL* představující Kullaback Leibler divergenci.

Spuštění programu

Program spustíme z příkazové řádky zadáním příkazu *python nazevskriptu.py*.

- *python run.py* - v případě spuštění všech skriptů postupně.
- *python load_data.py* - v případě načtení dat, získání příznaků z načtených obrázků a následné uložení do souboru uvedeného v configu.
- *python count_distance_jec.py* - spočítá vzdáleností a přiřadí klíčová slova.
- *python count_count_result.py* -vyhodnotí úspěšnost klasifikace mimo jiné přesnost a úplnost.

Výstupy programu

Názvy výstupných souborů se mohou lišit v závyslosti na nastavení configu.

- *PICTURE_RESULT* - Cesty k obrázkům s klíčovými slovy, přiřazenými klasifikátorem.
- *PICTURE_ALL_KEYWORDS* - Obsahuje názvy obrázku s přiřazenými slovy od klasifikatoru i s se slovy přiřazené člověkem. Ve formátu *cesta_k_obrázku;originální_klíčová_slova;klíčová_slova_přiřazena_klasifikátorem*
- *KEYWORDS_RESULT* - Soubor, který obsahuje vysledky klíčových slov, jejich přesnost a úplnost.
- *DATAFILE_TRAIN* - Soubor obsahující příznakovové vektory načtených obrázků.