

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

## **Bakalářská práce**

# **Automatická anotace obrázků**

Místo této strany bude  
zadání práce.

# Prohlášení

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

V Plzni dne 13. 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

The text of the abstract (in English). It contains the English translation of the thesis title and a short description of the thesis.

## 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 zahrnující 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

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>JEC Joint Equal Contribution</b>	<b>3</b>
2.1	Příznaky . . . . .	3
2.1.1	Barva . . . . .	3
2.1.2	Textura . . . . .	5
2.2	Vzdálenosti . . . . .	9
2.3	Kombinace vzdáleností . . . . .	9
2.4	Přenesení klíčových slov . . . . .	10
2.4.1	Originální algoritmus pro přenesení klíčových slov metody JEC . . . . .	10
2.4.2	Dynamické přenesení klíčových slov pomocí práhování . . . . .	11
<b>3</b>	<b>POEM</b>	<b>12</b>
3.0.1	Výpočet gradientu a magnitudy . . . . .	12
3.0.2	Diskretizace směru gradientu . . . . .	12
3.0.3	Výpočet lokálního histogramu orientace gradientů z okolí . . . . .	13
3.0.4	Zakódování příznaků pomocí LBP . . . . .	13
3.0.5	Konstrukce globálního histogramu . . . . .	14
3.1	Barevný POEM . . . . .	15
<b>4</b>	<b>Testovací databáze</b>	<b>18</b>
4.1	iaprtc12 . . . . .	18
4.2	ESP . . . . .	18
<b>5</b>	<b>Evaluační metriky</b>	<b>20</b>
5.1	Přesnost a úplnost pro celý klasifikátor . . . . .	20
5.2	Přesnost a úplnost - per word . . . . .	21
5.3	F-measure . . . . .	21
<b>6</b>	<b>Návrh systému</b>	<b>22</b>
<b>7</b>	<b>Implementace</b>	<b>23</b>
7.1	Použité programové prostředky . . . . .	23
7.1.1	OpenCV . . . . .	23
7.1.2	Scikit . . . . .	23

7.2	Modulové jednotky programu . . . . .	24
7.2.1	Config . . . . .	24
7.2.2	Load data . . . . .	24
7.2.3	Knn classifier . . . . .	25
7.2.4	Label transfer . . . . .	25
7.2.5	Evaluator . . . . .	26
<b>8</b>	<b>Vyhodnocení výsledků</b>	<b>27</b>
8.1	Srovnání výsledků . . . . .	27
8.1.1	Gabor - porovnání parametrů . . . . .	27
8.1.2	Haar - porovnání parametrů . . . . .	27
8.1.3	Přiřazování klíčových slov pomocí práhování . . . . .	27
8.1.4	Konečné výsledky a srovnání s literaturou . . . . .	28
<b>9</b>	<b>Závěr</b>	<b>30</b>
<b>10</b>	<b>Použité zkratky</b>	<b>31</b>
	<b>Literatura</b>	<b>32</b>
<b>A</b>	<b>Uživatelská dokumentace</b>	<b>33</b>

# 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 nejdříve 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 metadata, která obsahují klíčová slova.

Práce se bude zabývat nízkourovňovými příznaky konkrétně barvou a texturou. Ovšem v případě kdy použijeme barevný příznak ochudíme se o informaci o textuře obrázku, pro změnu 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) [1]). 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 ovlivňovat i protichůdně.

Cílem práce je navrhnout a implementovat software umožňující automatickou anotaci obrázků za použití nízkourovňových příznaků, konkrétně barvy a textury a jejich kombinací. Metody budeme zkoušet na standardních datech IAPRTC12 a ESP, následně výsledky porovnáme mezi sebou a s literaturou a pokusíme se 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 jednotlivých sousedů jsou posuzována odlišně a to právě na základě toho o kolik se s testovaným obrázkem liší. Metoda je postavena na dvou typech příznaků - barevných a texturových. [1]

### 2.1 Příznaky

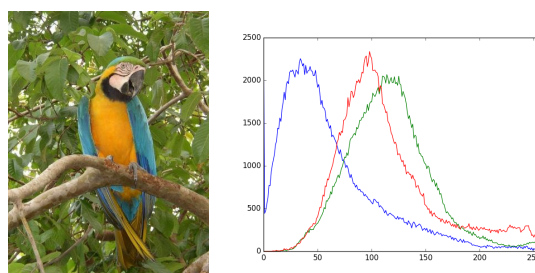
Barva a textura jsou považovány za dva nejdůležitější nízkourovňové příznaky pro obrázkovou reprezentaci. Nejběžnější barevné deskriptory jsou barevné histogramy, které jsou často využívány pro porovnávání a indexování obrázků, zejména z důvodu jejich efektivnosti a snadného výpočtu. K vytvoření texturových příznaků se používají Haarovy a Gaborovy wavelety a to především z důvodu, že jsou efektivní při vytváření řídkých a zároveň diskriminativních obrázkových rysů. Je-li žádoucí omezit vliv a předpoklady jednotlivých funkcí a maximizovat množství získaných informací, že využijeme několik jednoduchých a snadných výpočetních funkcí.

#### 2.1.1 Barva

U digitálního obrazu je barva reprezentovaná  $n$ -rozměrným vektorem. Jeho velikost a význam jednotlivých složek (tzv. barevných kanálů) závisí 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 nejpožívanější barevný prostor pro zachycení obrazu nebo jeho zobrazení. Oproti tomu HSV (Hue, Saturation and Value) se snaží zachytit barevný model tak jak ho vnímá lidské oko, ale zároveň se snaží zůstat jednoduchý na výpočet. Hue znamená odstín barvy (měří se jako poloha na standartním barevném

kole  $0^\circ - 360^\circ$ ), saturation je systost barvy (množství šedi v poměru k odstínu 0% šedá barva - 100% plně sytá barva) a 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 shopen 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.



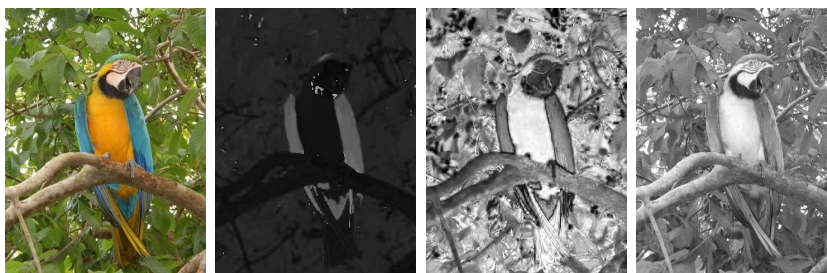
Obrázek 2.1: RGB histogram - zastoupení jednotlivých složek v obrázku



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

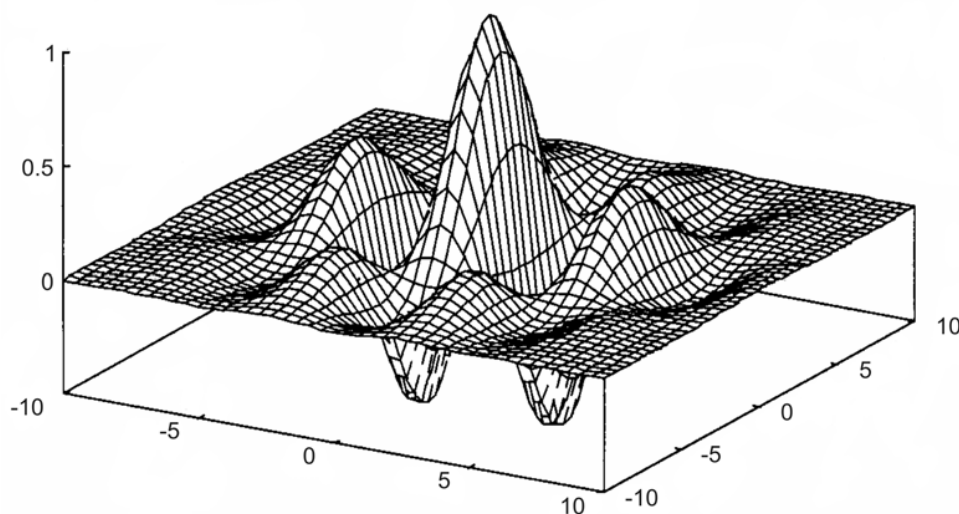
Pro RGB, HSV i LAB je použita barevná hloubka 16 bitů na kanál histogramu v jejich příslušném barevném prostoru. To znamená, že z každého barevného prostoru vzniknou tři šestnácti prvkové histogramy. Tyto histogramy jsou zřetězeny a následně použité jako reprezentace příslušného barevného prostoru.

### 2.1.2 Textura

Jako reprezentace textur a detekci hran budou použity Gaborovi a Haarovi 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 Gaborovi 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 sinusovou rovinnou vlnou jak můžeme vidět v rovnici 2.1.



Obrázek 2.5: Gáborova vlnka je tvořena kombinací dvou cosinových funkcí, s rozdílnou frekvencí pro každou osu, a následně jsou vynásobeny dvourozměrnou Gaussovou funkcí [3].

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

Gáborovi 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 (obvykle 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 hranách a místech, kde se textura mění. [8]

Gaborův filtr reaguje na hrany a změny textury. Když se řekne, že filtr odpovídá na konkrétní funkci, myslí se tím že filtr má rozlišovací hodnotu v prostorové poloze této funkce (když se bude zabývat aplikací konvolučních jader v prostoru - směru. Stejně platí i pro jinou oblast, jako frequency)

U Gaborova filtru máme několik parametrů, které ho ovlivňují.

**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 (jen kvůli jednotnosti). Při různých ksize se velikost konvolučního jádra mění. To také znamená, že konvoluční jádro je měřítko invariantní, protože zmenšení velikosti jádra je analogické k zmenšení velikosti obrazu.

**sigma** označuje standartní odchylka Gaussovi funkce použita v gaborově filtru. Tento parametr kontroluje šířku Gaussovi obalu použité v gabor jádře.

**theta** je orientace normálu na paralelní pruhy Gaborovy funkce. Představuje možná jeden z nejdůležitějších parametrů gabor filtru. Theta rozhoduje jakého druhu funkce (na jaký typ funkce filtr reaguje). Například při nulév 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 sinusovky ve výše uvedené rovnici.

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

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

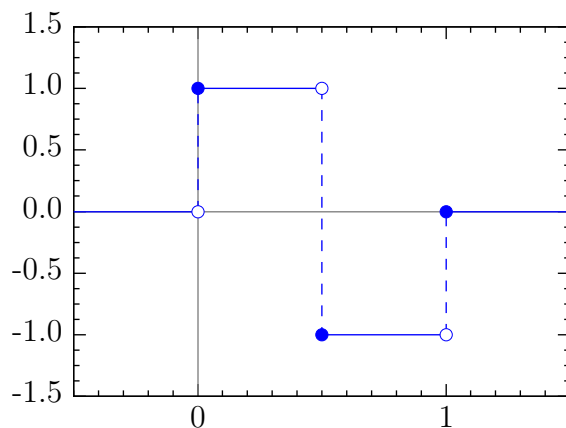
Podle [1] bude každý obrázek filtrován na třech vlnových délkách a čtyřech orientacích. V této závyslosti vyjde 12 filtrovaných obrázku pro jeden původní obrázek.

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

Druhý příznak zachycující fáze, je označen jako GaborQ. Obdobně jako v případě Gabora jsou vzniklé fáze 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 předchodem od kladného k zápornému.



Obrázek 2.6: Haarova vlnka. Převzato z [5]

Předpis Haarovi vlnky:

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

Haarovi 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 vyzdvýžení 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 [1] bude Haarova vlnka generovat konvoluční blok s Haarovými filtry na třech rozdílných orientacích (horizontální, diagonální a vertikální). Použité na obrázky různých velikostí.

-1	-1
1	1

Vertikální

-1	1
-1	1

Horizontální

-1	1
1	-1

Diagonální

Výsledný příznak je možné vytvořit dvěma způsoby. První možností je výslednou matici, z každého velikosti i orientace, převést na 16 prvkový vektor. Tímto vyjde 12 šestnácti prvkových vektorů pro jeden obrázek, které

jsou v konečné fázi zřetězeny. Další možností je udělat z výsledné matice sumu, výsledným vektorem bude v tomto případě 12 prvkový vektor sum.

## 2.2 Vzdálenosti

K určení příslušné vzdálenosti se můžeme setkat se čtyřmi měřítky vzdálenosti pro histogramy a rozdělení Kullaback-Leibler divergence KL - divergence,  $\chi^2$  statistika, L1 - vzdálenost a L2 - vzdálenost. Na RGB a HSV je nejlépejší 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-Leiber divergence:

$$D_{KL}(P||Q) = \sum_i P(i) \log_e \left( \frac{P(i)}{Q(i)} \right) \quad (2.2)$$

L1 (jinak označováno jako Manhattan):

$$L_1 = \sum_{i=1}^N |x_i - y_i| \quad (2.3)$$

L2 (jinak označováno jako Euklidovská vzdálenost)

$$L_2 = \sqrt{\sum_{i=1}^N (x_i - y_i)^2} \quad (2.4)$$

## 2.3 Kombinace vzdáleností

Nejrozumnějším přístupem ke zkombinování vzdáleností od různých deskriptorů je aby jednotlivé vzdálenosti přispívali rovnocenně. Z tohoto důvodu je potřeba vzdálenosti přeškálovat na jednotné měřítko.

$I_i$  představuje  $i$ -tý obrázek s  $N$  příznaky.  $d_{(i,j)}^k$  představuje vzdálenosti mezi příznaky  $f_i^k$  a  $f_j^k$ . Při snaze zkombinovat všechny vzdálenosti příznaků mezi obrázky  $I_i$  a  $I_j$ , tedy  $d_{(i,j)}^k$ ,  $k = 1, \dots, N$  je třeba dbát na to, že v praxi nevyjdou tak aby měli 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  $\tilde{d}_{(i,j)}^k$  následně může být kompletní



vzdálenost mezi obrázky  $I_i$  a  $I_j$  označena jako (2.5) Joint Equal Contribution (JEC).

$$JEC = \sum_{k=1}^N \frac{\tilde{d}_{(i,j)}^k}{N} \quad (2.5)$$

## 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  $I_i, i = 1, \dots, K$ , těchto  $K$  nejbližších 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 algoritmu na přenesení klíčových slov.

1. Klíčová slova z  $I_1$  jsou seřazena podle jejich frekvence výskytu v trénovací sadě.
2. Ze všech  $|I_1|$  klíčových slov z  $I_1$  je přeneseno  $n$  nejvýše umístěná klíčová slova do dotazovaného  $\tilde{I}$ . 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 přenesených v kroku 2
  - (b) místní frekvence (tj. jak často se vyskytují jako klíčová slova u obrázků  $I_2$  až  $I_K$ ). Jsou vybrána nejvíce vyskytující  $n - |I_1|$  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 [1], přinášeli tyto přímé postupy horší výsledky v porovnání s použitým dvoufaktorovým algoritmem pro přenos klíčových slov.

V souhrnu použitá metoda je složenina ze dvou složenin a to obrázkové vzdálenosti (JEC) a výše popsaným algoritmem na přenášení klíčových slov.

### 2.4.2 Dynamické přenesení klíčových slov pomocí práhování

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

Práh (2.6) je vyčíslen jako jedna děleno (počet všech klíčových slov i s jejich redundantními výskyty - 1). Následuje výpočet váhy pro dané klíčové slovo (2.7), které 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.

$$th = \frac{1}{count\_keywords - 1} \quad (2.6)$$

$$vaha = frequency\_keyword / total\_keywords \quad (2.7)$$

## 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. [7]

### 3.0.1 Výpočet gradientu a magnitudy

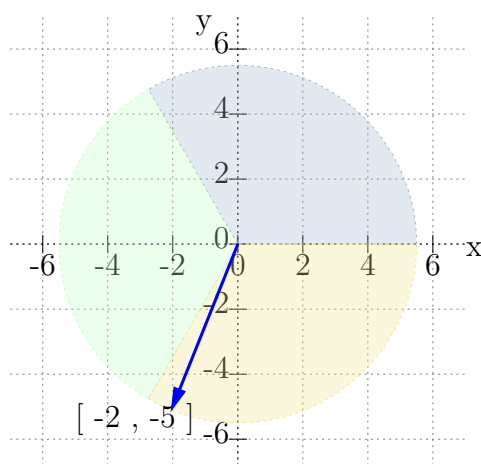
Nejprve je potřeba vypočítat gradient. Gradient je obecně směr růstu. Výpočet může probíhat různými způsoby. Jednou z možností je použít masku, kterou aplikujeme na vstupní obrázek. Podle některých studií jsou nejlepší jednoduché masky jako je např.  $[1, 0, -1]$  a  $[1, 0, -1]^t$ . Okraje obrázku se 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 můžeme spočítat jako velikost 2D vektorů, které jsme dostaly 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  $0 - \pi$  nebo neznaménkovou reprezentaci  $0 - 2\pi$ . 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  $(0 - \frac{2}{3}\pi)$ ,  $(\frac{2}{3}\pi - \frac{4}{3}\pi)$  a  $(\frac{4}{3}\pi - 2\pi)$ . 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.



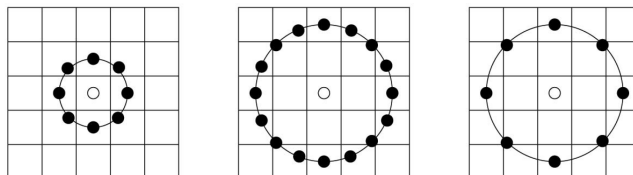
Obrázek 3.1: Diskretizace směru gradientu. Každá barva představuje jeden směr šedá:  $(0 - \frac{2}{3}\pi)$ , zelená:  $(\frac{2}{3}\pi - \frac{4}{3}\pi)$  a žlutá:  $(\frac{4}{3}\pi - 2\pi)$ . Vektor  $[-2, -5]$  směřuje do třetího směru, proto uložíme jeho magnitudu do třetí matice.

### 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

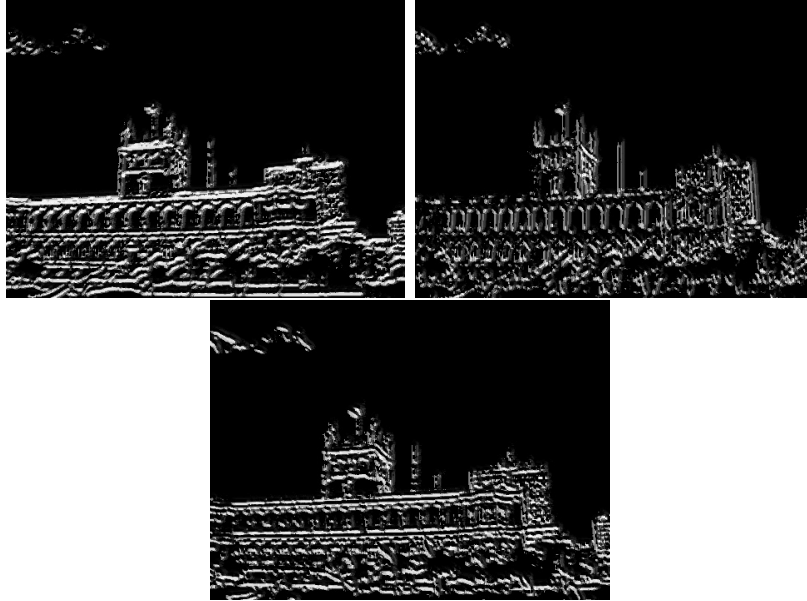
LBP operátor je aplikován na okolí každého pixelu o velikost  $3 \times 3$ . Oproti tomu POEM je možné aplikovat na větší okolí. Toto okolí se nazývá block, zpravidla se jedná o kruhové okolí s poloměrem  $L/2$  ( $L$  představuje velikost bloku). Pro stanovení intenzit okolních hodnot je možné použít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



Obrázek 3.2: Znázornění blocku Převzato z [7]

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  $\tau$ . Každý obrázek představuje jeden směr.

### 3.0.5 Konstrukce globálního histogramu

Obrázky získané z LBP jsou rozděleny 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  $3 \times 2$  matic. Na matice se dá pohlížet jako na 2 vektory o 3 složkách. Vektory jsou sloučeny pomocí součtu vektoru do jednoho 3 složkového vektoru. Magnituda je opět velikost vektoru tentokrát, ale v prostoru.

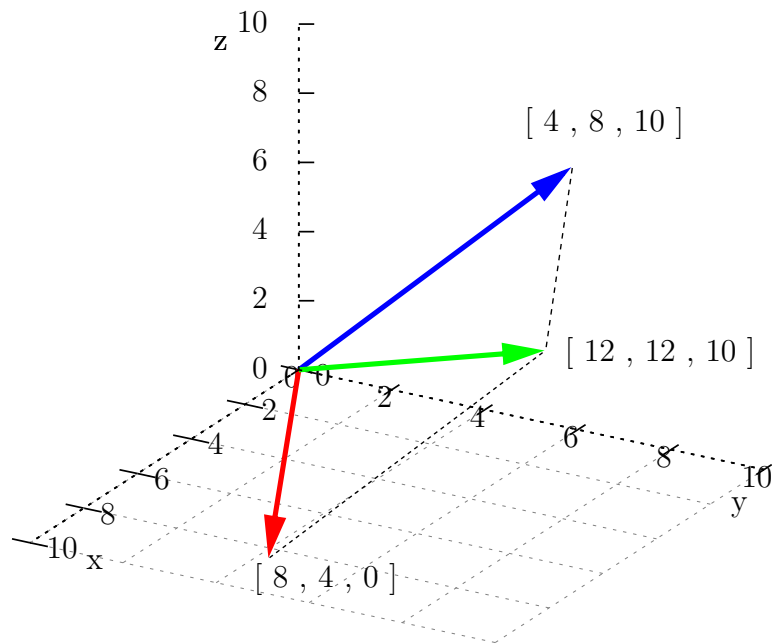
Format vzniklých vektorů

$$u = [blue_x, green_x, red_x] \quad (3.1)$$

$$v = [blue_y, green_y, red_y] \quad (3.2)$$

Pomocí součtu vektorů je získán jeden tříslžkový vektor:

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



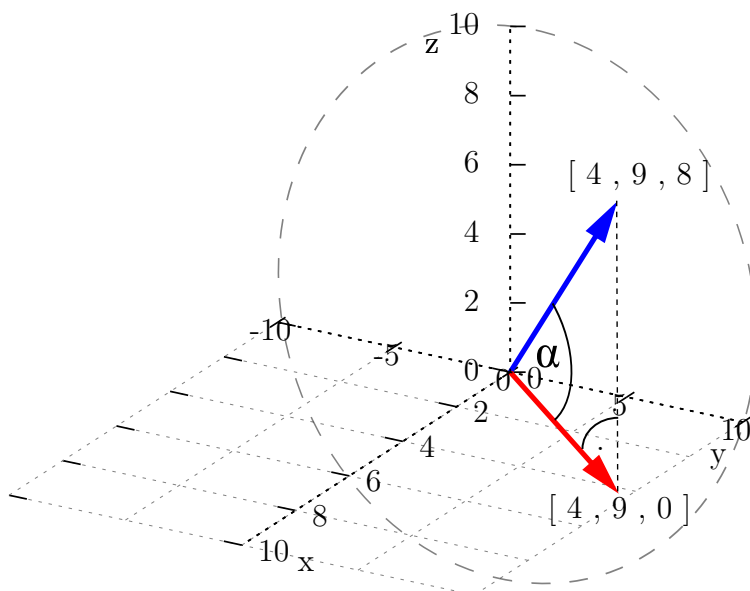
Obrázek 3.4: Grafické znázornění součtu vektorů. Součet je tvořen z vektorů  $[4, 8, 10]$  a  $[8, 4, 0]$ .

## 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  $\alpha$ , ale jeho doplněk  $(180 - \alpha)$ .



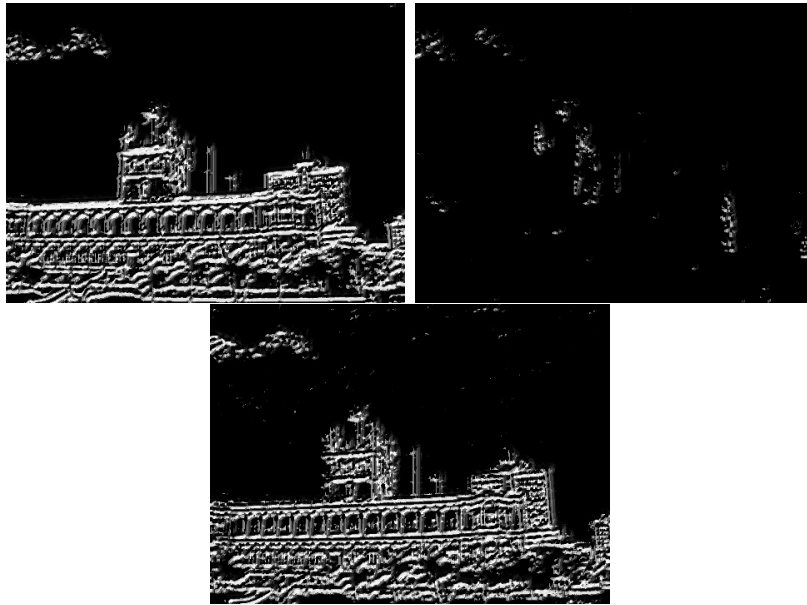
Obrázek 3.5: Grafické znázornění součtu vektorů. Součet je tvořen z vektorů  $[4, 8, 10]$  a  $[8, 4, 0]$ .

## Výpočet lokálního histogramu

Zbývajícím postupem 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 centrálnímu pixelu přičítána malá konstanta  $\tau$



Obrázek 3.6: Obrázky po aplikaci LBP s použitím  $\tau$ . Každý obrázek představuje jeden směr.

## Konstrukce globálního histogramu

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



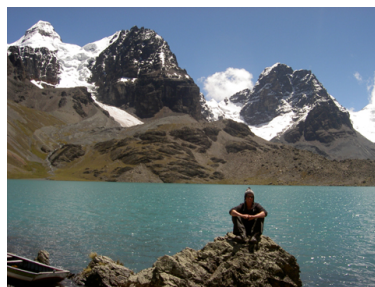
## 4 Testovací databáze

Pro natrénování a následné testování byla použita data z databází IAPRC a ESP. Kolekce obrázků na natrénování musí být pečlivě vybrána aby zahrnovala co možná největší okruh z různých témat.

### 4.1 iaprtc12

Sada iaprtc12 je kolekce obrázků přírodních scén která 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, a v tagu description je například a woman and a child are walking over the square. Spolu s databází jsme získali i klíčová slova která byla z přiložených xml extrahována.

K jednomu obrázku je v průměru přiřazeno 5.7 klíčových slov. Pro trénování bylo použito 17 664 obrázků, na následné testování jich bylo použito 1960.



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

### 4.2 ESP

Sada ESP obsahuje širokou škálu snímků s anotacemi, ze kterých byla použita jen malá část. Konkrétně 18 689 obrázků na trénování a 2061 na testování.

vání. Ke každému obrázku je přiřazen soubor ve formátu *desc*, který obsahuje anglické anotace. Z celkových 269 klíčových slov je k jednomu obrázku 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 obrázku 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

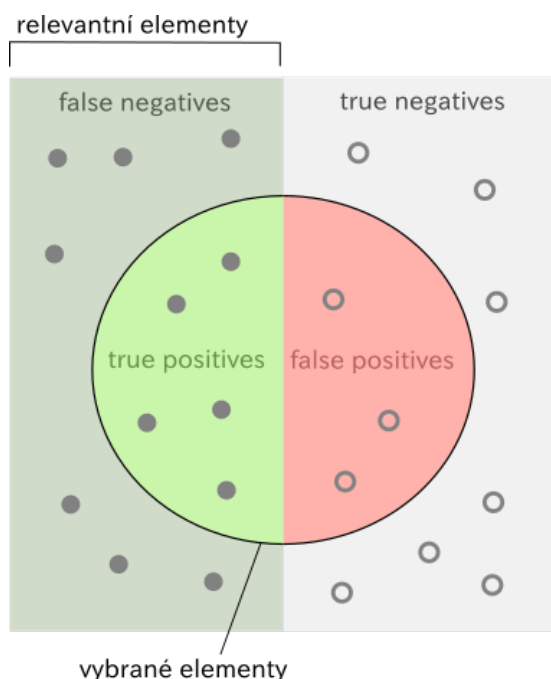
## 5 Evaluační metriky

Kvalitu a úspěšnost klasifikátoru udává přesnost (precision) a úplnost (recall). Přesnost udává jak moc jsou výsledky relevantní a úplnost kolik skutečně relevantních 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. V práci je použitý postup per word. [10]

Počet nenulových slov značí počet slov, které byli při anotaci použity 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 nepřidil.



$$Prec = \frac{TP}{TP + FP} \quad (5.1)$$

$$Rec = \frac{TP}{TP + FN} \quad (5.2)$$

Obrázek 5.1: Znázornění precision a recall.

## 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  $w_{correctly}$  počet obrázků, kterým bylo slovo přiřazeno 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).

$$Rec = \frac{w_c}{w_h} \quad (5.3)$$

$$Prec = \frac{w_c}{w_a} \quad (5.4)$$

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

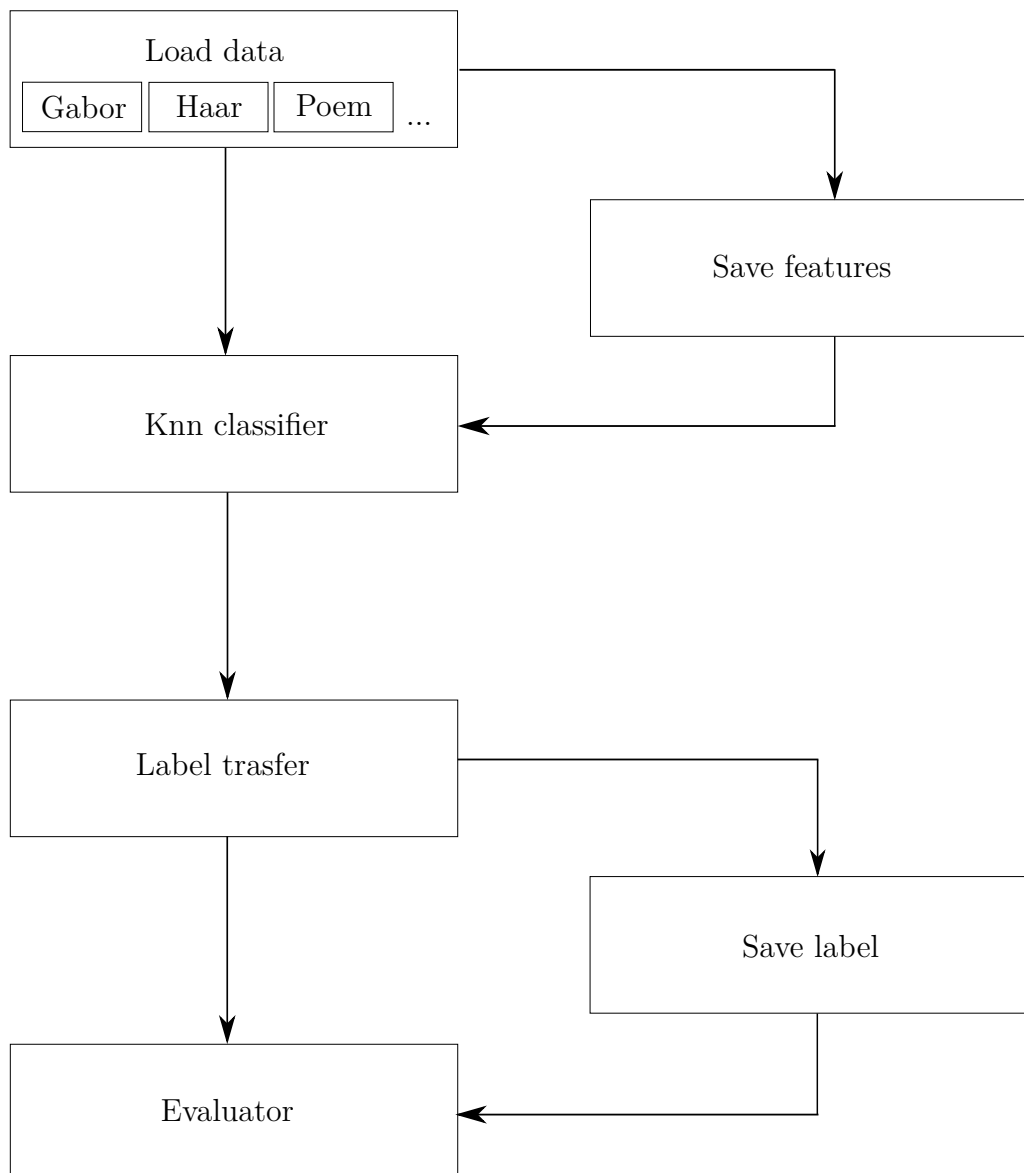
## 5.3 F-measure

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

$$F - measure = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (5.5)$$

## 6 Návrh systému

Systém byl navržen jako modulární z důvodu snadné obměny některé z částí. 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 práci bylo navrženo i ukládání mezivýsledků. Návrh systému je zobrazen na následujícím obrázku.



Obrázek 6.1: Navrh 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.

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 navržena 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. [6]

## 7.2 Modulové jednotky programu

### 7.2.1 Config

Při spuštění programu je nejdříve načten konfigurační soubor, který obsahuje jeho veškerá nastavení. Pro snadné a pohodlné spuštění všech modulů je připraven skript *run.py*. V opačném případě můžeme jednotlivé moduly pouštět postupně.

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 klíčové\_slovo*. Předpokládaný formát *.txt*.
- *TEST\_LIST* - Testovací list, který by měl obsahovat cesty k obrázkům testovací sady a jejich klíčová slova ve formátu *cesta\_k\_obrazku; klíčové\_slovo klíčové\_slovo*. Předpokládaný formát *.txt*.
- *DATAFILE\_TRAIN* - Soubor do kterého budou uloženy příznakové vektory, načtených obrázků. Předpokládaný formát *.py*.
- *PICTURE\_RESULT* - Obrázky a prirazené klíčová slova klasifikátorem.
- *PICTURE\_ALL\_KEYWORDS* - obrázky s prirazenými slovy od klasifikátoru i s se slovy prirazené člověkem.
- *KEYWORDS\_RESULT* - Výsledky klíčových slov, jejich přesnost a úplnost.
- *COUNT\_NEIGHBORS* - Počet sousedů.
- *COUNT\_KEYWORDS* - Počet klíčových slov.

Dále jsou v configu uvedené jednotlivé metody s očekávanou hodnotou *True* nebo *False* v závislosti zda se mají použít nebo ne.

### 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 povolána metoda *load\_features*, která načte příslušné příznaky. Nakonec jsou celé struktury listů testovacích nebo trénovacích obrázků uloženy do souboru, dle configu *DATAFILE\_TRAIN* popřípadě *DATAFILE\_TEST*.

## Extrakce příznaků

Jednotlivé výpočty příznaků jsou rozděleny do zvláštních modulů, aby byla obměna jejich výpočtu snadno nahraditelná. V každém modulu je stěží 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*.

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

### 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 pomoci zjištěného maxima a minima přeškáluje 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 přenos 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 ostatními klíčovými slovy v metodě *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 ukončena v opačném případě pokračuje do funkce *add\_keywords\_from\_neighbors*, která dodá potřebná klíčová slova od dalších k nejbližším sousedům. Nakonec jsou klíčová slova uložena do souboru dle *configu*.



Druhou variantou je dynamické přenesení klíčových slov pomocí práhování. Zde je rovnou spustena 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.

### 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 přenost a úplnost. Tyto hodnoty jsou popsány v sekci Vyhodnocení výsledků.

## 8 Vyhodnocení výsledků

### 8.1 Srovnání výsledků

Přesné parametry se kterými autoři dosahovali nejlepší výsledků nebyli zjištěny, proto bylo třeba je u některých příznaků zkoušet metodou pokus omyl.

#### 8.1.1 Gabor - porovnání parametrů

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

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

Tabulka 8.2: Haar na datech iaprtc12.

#### 8.1.3 Přiřazování klíčových slov pomocí práhování

Metoda	5 sousedů			8 sousedů			10 sousedů		
	$P_{\%}$	$R_{\%}$	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	0	0	0	0	0	0	0	0	0

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

Metoda	5 sousedů			8 sousedů			10 sousedů		
	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N	$P_{\%}$	$R_{\%}$	N
RGB	17.6	8.9	170	13.1	12.6	195	0	0	0
LAB	7.9	4.1	95	7.7	6.5	121	0	0	0
HSV	19	11	192	14	14.8	205	0	0	0
RGB, LAB, HSV	20.4	11	182	14.4	14.8	201	-	-	-
JEC	0	0	0	15.6	15.5	206	0	0	0

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

Metoda	IAPRTC12			ESP		
	$P_{\%}$	$R_{\%}$	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	14.2	10.8	194
GaborQ	6.9	4.8	133	12.1	9.9	187
Haar	5.8	4	114	10.2	8.4	178
HaarQ	5.8	4.4	123	9.4	7.3	169
JEC	0	0	0	19.6	15.1	211
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.

Metoda	IAPRTC12			ESP		
	$P_{\%}$	$R_{\%}$	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 [2].



Obrázek 8.1: Ukázka obrázku s klíčovými slovy ze sady ESP.

Originální klíčová slova: circle coin face head metal money old round silver  
 Klíčová slova přiřazena systémem metodou JEC: man old metal coin money  
 Klíčová slova přiřazena systémem metodou JEC s přenášením klíčových slov pomocí práhování: old money coin

## 9 Závěr

V práci byla řešena automatická anotace obrázků pomocí spojování barevných a texturových příznaků. U prvního řešení s metodou JEC byla použito vypočtení příznaků odděleně následováno spojením jejich vzdáleností do jedné. 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 nízkourovňové příznaky barva a textura. U barvy se zabývalo barevnými modely RGB, LAB a HSV. U textury to byl GABOR, HAAR a POEM. Dále se zabývalo přenášením klíčových slov za pomoci práhu, kdy musela frekvence výskytu u sousedů převýšit stanovený práh.

V praktické 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 prostudována a použita knihovna openCV. Funkčnost programu byla otestována na databázích iaprtc12 a ESP.

Vylepšení programu barevného poemu.

## 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.

# Literatura

- [1] AMEESH MAKADIA, VLADIMIR PAVLOVIC, SANJIV KUMAR. A new baseline for image annotation. *Commun. ACM*. July 1961, 4, 7, s. 321. ISSN 0001-0782. doi: 10.1145/366622.366644. Dostupné z: <http://doi.acm.org/10.1145/366622.366644>.
- [2] AMEESH MAKADIA, VLADIMIR PAVLOVIC, SANJIV KUMAR. Baselines for Image Anotation.
- [3] CRUSE, H. Neural Networks as Cybernetic Systems. *Commun. ACM*. July 1961, 4, 7, s. 321. ISSN 0001-0782. doi: 10.1145/366622.366644. Dostupné z: <http://doi.acm.org/10.1145/366622.366644>.
- [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](https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=117319).
- [5] *Haar wavelet* [online]. Oracle, 2016. [cit. 2016/03/09]. Java SE Documentation. Dostupné z: [https://en.wikipedia.org/wiki/Haar\\_wavelet](https://en.wikipedia.org/wiki/Haar_wavelet).
- [6] *Class Graphics2D* [online]. Oracle, 2016. [cit. 2016/03/09]. Java SE Documentation. Dostupné z: <http://scikit-image.org/>.
- [7] 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>.
- [8] 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/>.
- [9] *Hierarchical classification needs a metric* [online]. 2016. [cit. 2017/05/12]. idelab. Dostupné z: <http://idalab.de/blog/data-science/hierarchical-metrics>.
- [10] V. LAVRENKO, R.MANMATHA, J. JEON. A model for learning the semantics of pictures. *Commun. ACM*. July 1961, 4, 7, s. 321. ISSN 0001-0782. doi: 10.1145/366622.366644. Dostupné z: <http://doi.acm.org/10.1145/366622.366644>.

# 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, které se od postupu na jiném operačním systému mohou 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 klíčové\_slovo*. Předpokládaný formát *.txt*.
- *TEST\_LIST* - Testovací list, který by měl obsahovat cesty k obrázkům testovací sady a jejich klíčová slova ve formátu *cesta\_k\_obrazku; klíčové\_slovo klíčové\_slovo*. Předpokládaný formát *.txt*.
- *DATAFILE\_TRAIN* - Soubor do kterého budou uloženy příznakové vektory, načtených obrázků. Předpokládaný formát *.py*.
- *PICTURE\_RESULT* - Obrázky a přiřazené klíčová slova klasifikátorem.
- *PICTURE\_ALL\_KEYWORDS* - obrázky s přiřazenými slovy od klasifikátoru i s slovy přiřazenými člověkem
- *KEYWORDS\_RESULT* - Výsledky klíčových slov, jejich přesnost a úplnost
- *COUNT\_NEIGHBORS* - Počet sousedů.
- *COUNT\_KEYWORDS* - Počet klíčových slov.

Dále jsou v configu uvedené jednotlivé metody s očekávanou hodnotou *True* nebo *False* v závislosti zda se mají použít nebo ne.



## 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* - spočítání 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 závislosti na nastavení configu.

- *PICTURE\_RESULT* - Obrázky a přiřazené klíčová slova klasifikátorem.
- *PICTURE\_ALL\_KEYWORDS* - obrázky s přiřazenými slovy od klasifikátoru i s slovy přiřazenými člověkem
- *KEYWORDS\_RESULT* - Výsledky klíčových slov, jejich přesnost a úplnost
- *DATAFILE\_TRAIN* - Soubor do kterého budou uloženy příznakové vektory, načtených obrázků.