



# KIV/UIR - Semestrální práce

25. května 2017  
40 hodin

Kateřina Kratochvílová  
A13B0364P

# Obsah

<b>1</b>	<b>Úvod</b>	<b>1</b>
<b>2</b>	<b>Poem</b>	<b>2</b>
2.0.1	Výpočet gradientu a magnitudy . . . . .	2
2.0.2	Diskretizace směru gradientu . . . . .	4
2.0.3	Výpočet lokálního histogramu orientace gradientů z okolí	5
2.0.4	Zakódování příznaků pomocí LBP . . . . .	6
2.0.5	Konstrukce globálního histogramu . . . . .	9
<b>3</b>	<b>Barevný poem</b>	<b>10</b>
3.0.1	Výpočet gradientu a magnitudy u barevneho poemu . . .	10
3.0.2	Diskretizace směru gradientu u barevneho poemu . . . .	13
3.0.3	Výpočet lokálního histogramu u barevneho poemu . . . .	14
3.0.4	Zakódování příznaku pomocí LBP u barevneho poemu . .	15
<b>4</b>	<b>Použité programové prostředky</b>	<b>16</b>
4.1	OpenCV . . . . .	16
<b>5</b>	<b>Testovací data</b>	<b>17</b>
<b>6</b>	<b>Závěr</b>	<b>18</b>
<b>7</b>	<b>Uživatelská dokumentace</b>	<b>19</b>
	<b>Literatura</b>	<b>21</b>

### **Abstrakt**

Tato práce byla vytvořena za účelem vytvoření deskriptoru, který ponese informaci nejen o barvě, ale i o textuře obrázku.

# Kapitola 1

## Úvod

Většina metod používaných na detekci textury pracuje pouze s šedotónovým obrázkem. Takže zanedbávají informaci o barvě a pracují jen s intenzitou obrazu. Cílem této práce však je obě informace zkombinovat a vytvořit tak jeden deskriptor, který ponese jak informaci o barvě tak i informaci o textuře obrázku [1]. Nabízí se několik možností:

**Vytvoření společného příznaku** například rozšíření LBP/POEM na všechny barvené kanály. Musí se však dbát na to, že informace o barvě a textuře se mohou ovlivňovat i protichůdně.

**Vyhodnotit a klasifikovat příznaky odděleně** a pak výslednou klasifikaci spojit z několika částí (například JEC - Joint Equal Contribution [2]). Výhodou tohoto přístupu je zachování vlastností obou původních příznaků. Nevýhodou je náročnější výpočet a úspěšnost přístupu závisí na způsobu kombinace obou informací.

V práci je rozebrán POEM (Patterns of Oriented Edge Magnitudes) pracující jen s texturou následovaný metodou POEM procházející všechny kanály barevného prostoru RGB.

Hotový deskriptor bude použit pro porovnání v bakalářské práci "Automatická anotace obrázků". Kde hlavním tématem je právě spojení příznaků vyhodnocených zvlášť.

# Kapitola 2

## 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. [3]

### 2.0.1 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).

Vstupní obrázek:

8	7	5
1	2	4
3	5	7

Masky:

$$\begin{aligned} maska(x) &= \begin{bmatrix} -1 & 1 \end{bmatrix} \\ maska(y) &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{aligned}$$

Na základě předchozích masek a vstupního obrázku byly vypočteny tyto matice gradientů:

Pro x:

-1	-2	0
1	2	0
2	2	0

Pro y:

-7	-5	-1
2	3	3
0	0	0

Gradient jako 2D vektory:

[-1 ; -7]	[-2 ; -5]	[0 ; -1]
[1 ; 2]	[2 ; 3]	[0 ; 3]
[2 ; 0]	[2 ; 0]	[0 ; 0]

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.

Vzoreček pro výpočet velikosti vektoru v rovině:

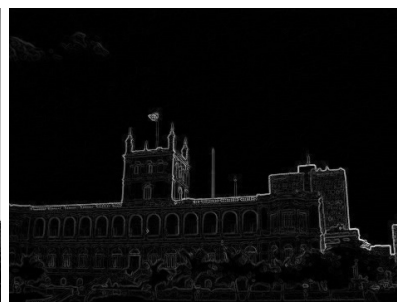
$$|u| = \sqrt{u_1^2 + u_2^2} \quad (2.1)$$

Magnituda na základě výše uvedených gradientů:

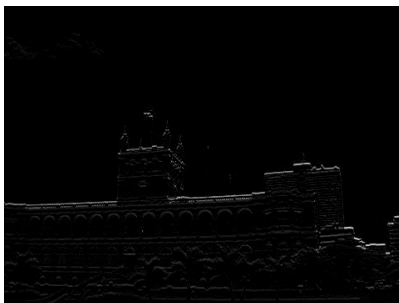
7.1	5.4	1.0
2.2	3.6	3.0
2.0	2.0	0.0



Obrázek 2.1: Vstupní obrázek



Obrázek 2.2: magnituda



Obrázek 2.3: gradient x

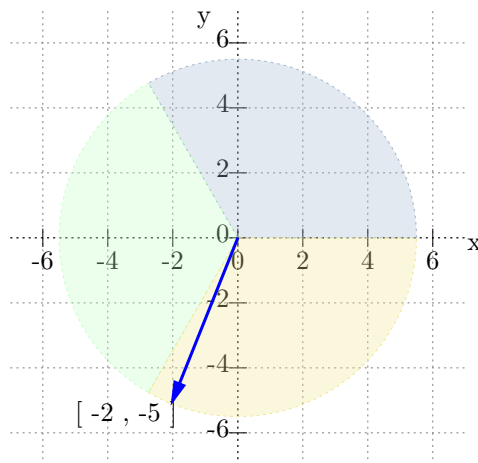


Obrázek 2.4: gradient y

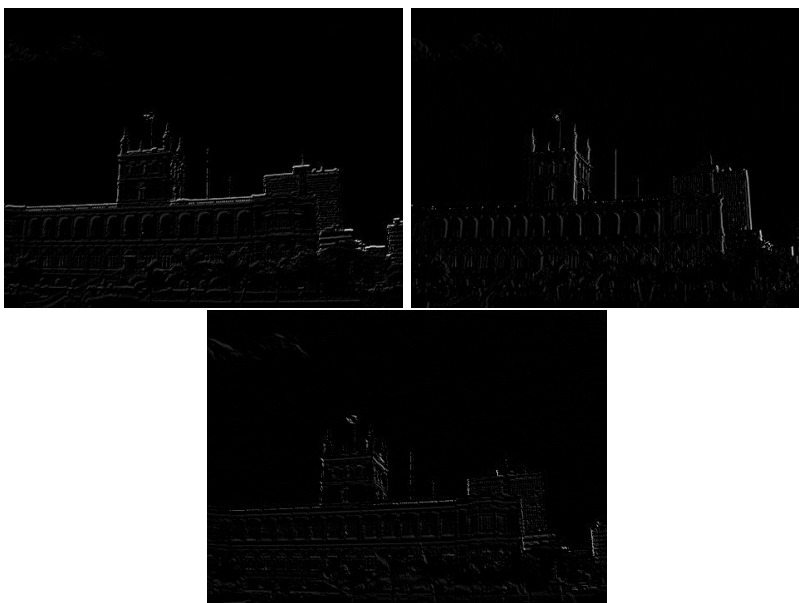
### 2.0.2 Diskretizace směru gradientu

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 2.5: 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.



Obrázek 2.6: Obrázky po diskretizaci. Každý obrázek představuje jeden směr.

### 2.0.3 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.

Výpočet u jednoho směru při  $\text{cell} = 3$ , Oranžově je vyznačená oblast výpočtu pro jeden pixel:

Vybraný směr:

0	0	0	0
0	0	0	0
0	0	0	0
0	3.6	3.6	3
0	2	2	0

Jeho aems:

0	0
0.8	1.1
1.2	1.5

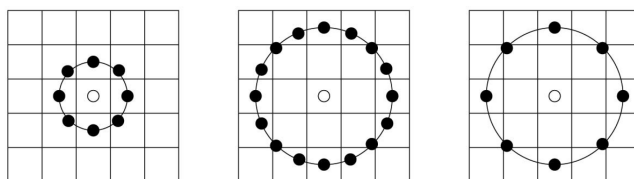




Obrázek 2.7: Obrázky po diskretizaci, oproti předchozím jsou hrany více rozmazané. Každý obrázek představuje jeden směr.

#### 2.0.4 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 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  $\tau$ .

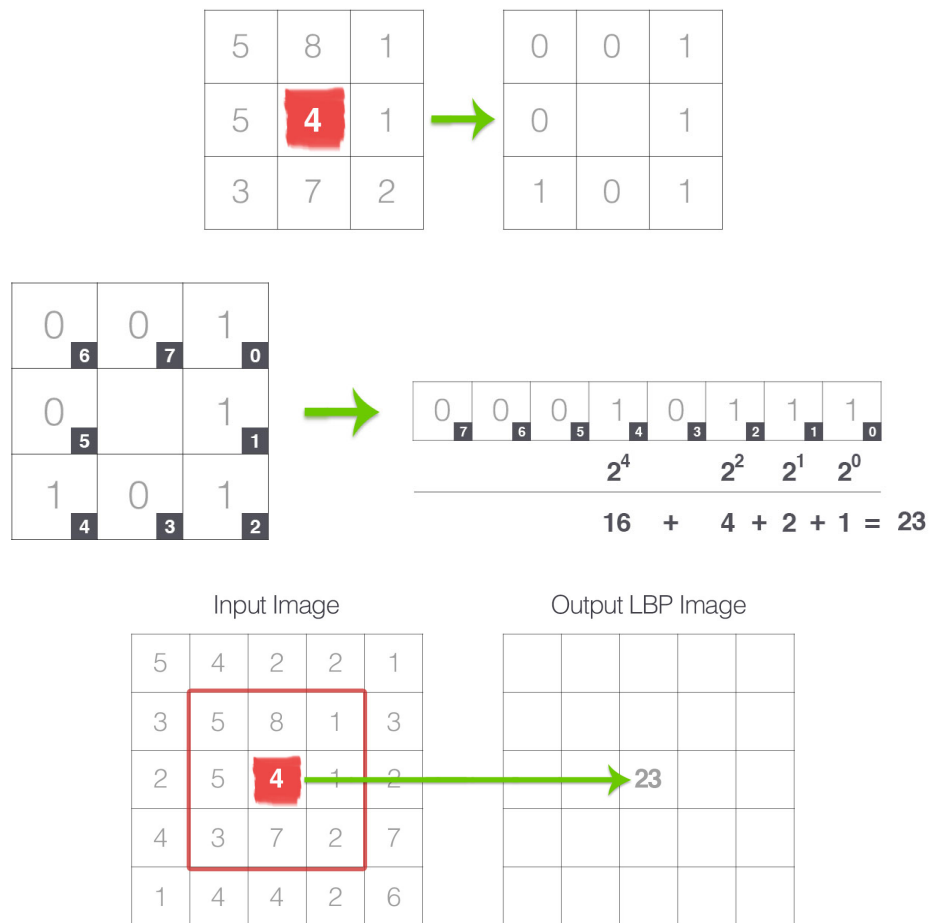


Obrázek 2.8: Znázornění blocku Převzato z [3]

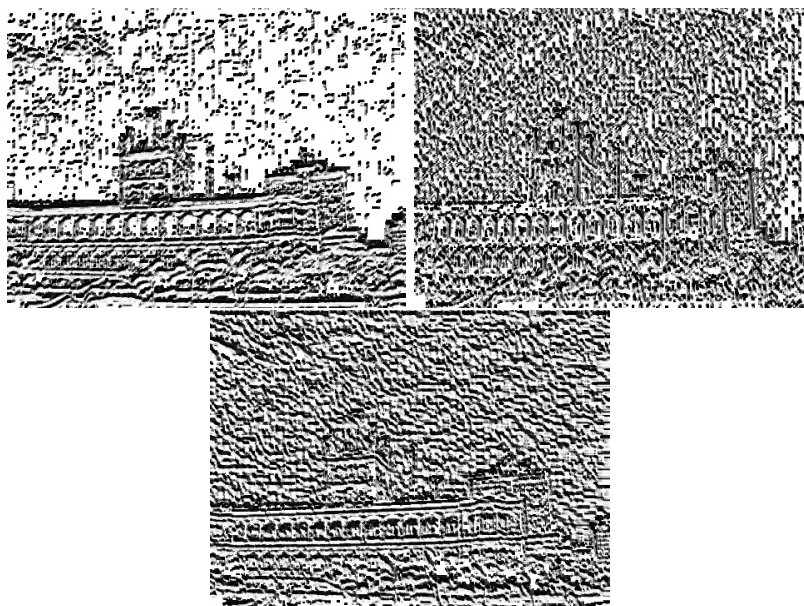
Následující matice představuje block pro střední pixel s hodnotou intenzity 4, označený písmenem  $c$  (centrální). Algoritmus následně prochází všechny okolní pixely, označené písmenem  $x$ . Hodnota daného pixelu je označena jako  $p(x)$  a výsledek tohoto porovnání je označen  $s(x)$ .

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

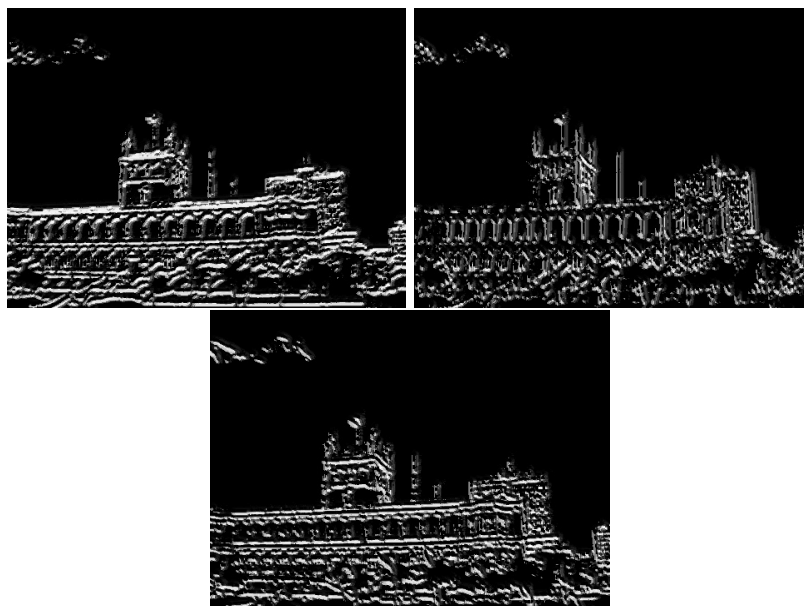
Zobrazení postupu algoritmu LBP:



Obrázek 2.9: Znázornění výpočtu LBP pro jeden pixel. Převzato z [4]



Obrázek 2.10: Obrázky po aplikaci LBP s velikostí bloku 8. Každý obrázek představuje jeden směr.



Obrázek 2.11: Obrázky po aplikaci LBP při použití  $\tau$ . Každý obrázek představuje jeden směr.

### 2.0.5 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.

#### Uniformní vzory

Jelikož histogram, který bude vytvořen je velmi dlouhý, je možné ho zkrátit vybíráním pouze tzv. uniformních vzorů. Některé binární vzory se totiž na běžných obrázcích vyskytují častěji a to až, dle experimentů z 90 %. Jsou to právě výše zmíněné uniformní vzory. Uniformní vzory jsou hodnoty čísla (čísla z pohledu binární reprezentace), kdy dochází k maximálně dvěma přechodům z 0 na 1 a nebo opačně. Například 00011110 je uniformní, oproti tomu 01101111 není. Těchto vzorů je 58, všechny ostatní vzory jsou reprezentovány jediným vzorem. Takto je délka jednoho lokálního histogramu zredukována z 256 na 59 binů.



Obrázek 2.12: Obrázky získané z LBP jsou rozděleny pravidelnou čtvercovou mřížkou. Převzato z [3]

## Kapitola 3

# Barevný poem

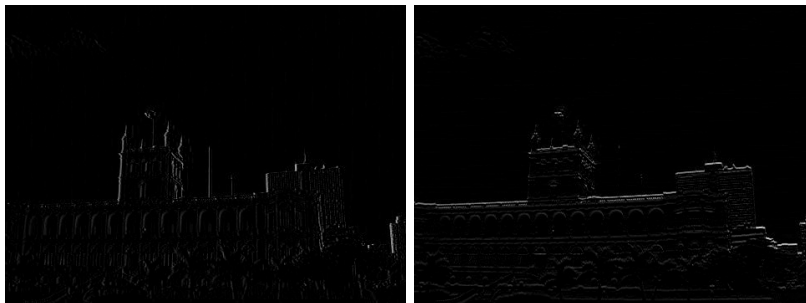
### 3.0.1 Výpočet gradientu a magnitudy u barevneho poemu

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.

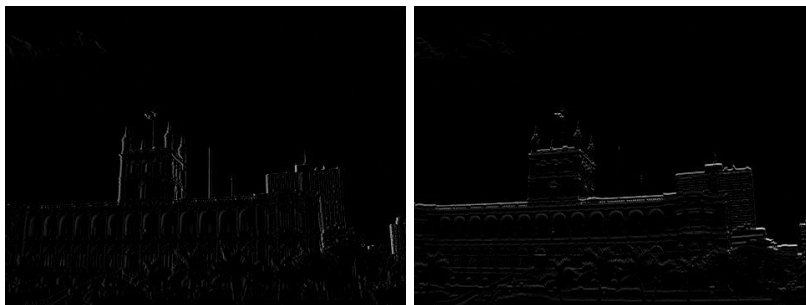
Format vzniklých vektoru

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

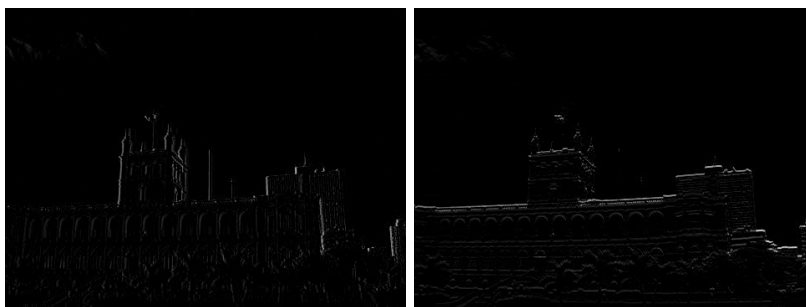
$$v = [blue_y, green_y, red_y] \quad (3.2)$$



Obrázek 3.1: Gradienty pro modrou složku.



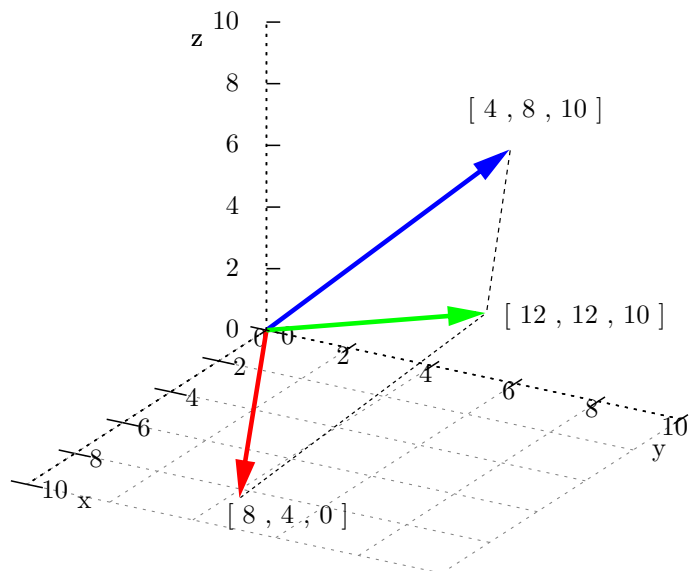
Obrázek 3.2: Gradienty pro červenou složku



Obrázek 3.3: Gradienty pro zelenou složku.

Pomocí součtu vektorů je získán jeden třísloužkový 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 vytvořen z vektoru  $[4, 8, 10]$  a  $[8, 4, 0]$ .

Výpočet velikosti vektoru v prostoru (magnituda):

$$|u| = \sqrt{u_1^2 + u_2^2 + u_3^2} \quad (3.4)$$

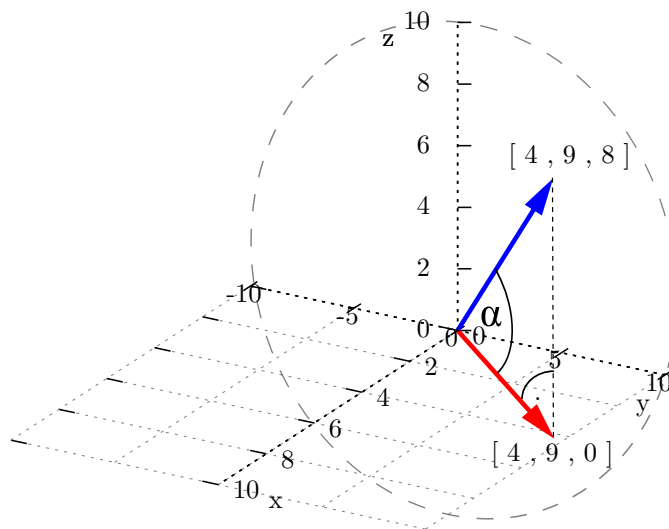


Obrázek 3.5: Porovnání magnitud. Vlevo magnituda z POEMU, vpravo magnituda z barevného POEMU.

### 3.0.2 Diskretizace směru gradientu u barevneho poemu

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  $(0 - \frac{\pi}{3})$ ,  $(\frac{\pi}{3} - \frac{2\pi}{3})$  a  $(\frac{2\pi}{3} - \pi)$ .



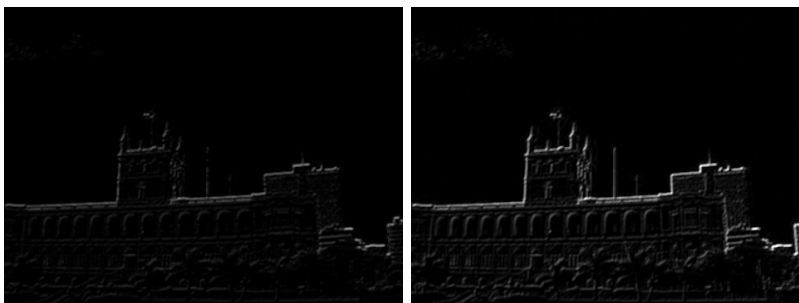
Obrázek 3.6: Diskretizace směru gradientu v neznaménkové reprezentaci. Mějme vektor  $[4, 9, 8]$  a jeho ekvivalentní vektor s vynulovanou složkou  $z$   $[4, 9, 0]$ . Úhel, který mezi sebou vektory svírají je  $39^\circ$ . Zároveň je vektor v prvním kvadrantu tudíž spadá do prvního intervalu.

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  $(180 - \alpha)$ .



### 3.0.3 Výpočet lokálního histogramu u barevneho poemu

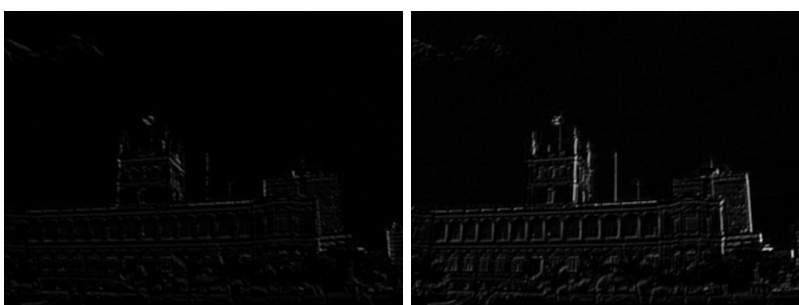
Zbývající postup je již totožný s výše uvedeným POEM. Pro srovnání jsou zde alespoň uvedeny výsledné obrázky.



Obrázek 3.7: Obrázky po diskretizaci první směr. Vlevo původní POEM, vpravo barevný POEM

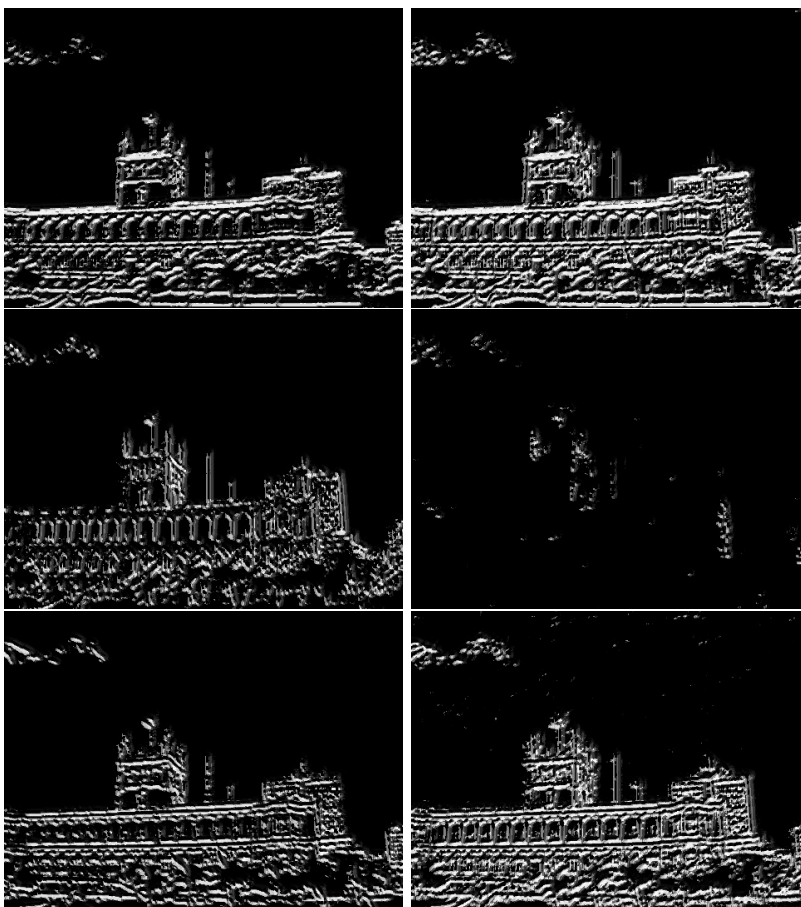


Obrázek 3.8: Obrázky po diskretizaci druhý směr. Vlevo původní POEM, vpravo barevný POEM



Obrázek 3.9: Obrázky po diskretizaci první směr. Vlevo původní POEM, vpravo barevný POEM

### 3.0.4 Zakódování příznaku pomocí LBP u barevneho po- emu



Obrázek 3.10: Obrázky po aplikaci LBP s použitím  $\tau$ . Vlevo původní POEM, vpravo barevný POEM

## Kapitola 4

# Použité programové prostředky

Program byl navržen na operační systému 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. Počítač na kterém byl program vyvíjen a spouštěn pracoval s Python verzí 2.7.12. Program využívá knihovnu OpenCV 3.1.

### 4.1 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++.

Knihovnu OpenCV je možné stáhnout na adrese: <http://opencv.org/>

## Kapitola 5

# Testovací data

Pro natrénování a následné testování byla použita data z databáze iaprc12. Data obsahují 20 000 obrázků ve formátu *jpg* s celkovým počtem 291 klíčových slov. Z toho bylo použito 19 805 obrázků a to 17 664 na trénování a 1960 na testování. K jednomu obrázku je v průměru přiřazeno 5.7 klíčových slov.

Ke každému obrázku jsou přiložena metadata ve formátu XML, která obsahují informace o obrázku v různých jazycích. Kromě angličtiny je zde možné nalést například i španělštinu nebo němčinu. Metadata ovšem neobsahují klíčová slova, ale v jednotlivých elementech jsou roztrženy obsáhlé informace. Například titulek obrázku, který může vypadat *The Plaza de Armas*, a v elementu *description* je například *a woman and a child are walking over the square*. Spolu s databází byla získána i klíčová slova, která byla extrahována právě z přiloženého xml.



Obrázek 5.1: Ukázka obrázku s klíčovými slovy: front lake man mountain rock sky summit

## Kapitola 6

# Závěr

V semestrální práci byl vytvořen skript, který z vloženého obrázku vytvoří POEM deskriptor. Dále bylo vyzkoušeno vytvoření POEM na barevném obrázku, tak aby obsahoval jak informaci o textuře tak informaci o barvě. Vytvořený POEM bude použit pro porovnání v bakalářské práci *Automatická anotace obrázků*.

Dosažené výsledky:

	Přesnost (%)	Úplnost (%)	Nemulových slov
RGB	14.1	9	167
LAB	12.7	7.5	148
HSV	16.7	10.9	181
RGB, LAB, HSV *	17.4	11.1	178
POEM	21.5	12.8	189
RGB, LAB, HSV, POEM *	21.8	13.8	187
Barevný POEM	21	12.4	184

\* Při spojování bylo použito Joint Equal Contribution. [2] s KL divergencí pro LAB a L1 vzdálenosti pro ostatní

## Kapitola 7

# Uživatelská dokumentace

Pro spuštění programu je třeba mít nainstalovaný python (2.7.12) a knihovnu openCV verzi 3.1. Následné postupy jsou uvedeny pro operační systém Linux.

### Spuštění programu

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

- *python poem.py obrazek.jpg* - v případě poemu pracujícím s šedotónovým obrázkem
- *python color\_poem.py obrazek.jpg* - v případě poemu pracujícím s barevným obrázkem

### Výstupy programu

V případě zájmu se můžeme podívat na mezivýstupy programu.

- *gradientX.jpg* - obrázek po použití filtru x, analogicky *gradientY.jpg*
- *gradientX.txt* - matice po použití filtru x, analogicky *gradientY.txt*
- *magnitude.jpg* - obrázek vzniklý po spočtení magnitud
- *magnitude.txt* - matice magnitud
- *phase.txt* - matice magnitud
- *directional1.jpg* - obrázek magnitud po rozdělení do smeru, smer 1 (analogicky *directional2.jpg*, atd)
- *directional1.txt* - magnitudy po rozdělení do smeru, smer 1 (analogicky *directional2.txt*, atd)
- *ames1.jpg* - obrázek po výpočtu lokálních histogramů orientace gradientů, smer 1 (analogicky *aems2.jpg*, atd)

- *ames1.txt* - matice po výpočtu lokálních histogramů orientace gradientů, smer 1 (analogicky aems2.txt, atd)
- *lbp1.jpg* - obrázek výsledného LBP, smer 1 (analogicky lbp2.jpg, atd)
- *lbp1.txt* - výsledného LBP, smer 1 (analogicky lbp2.txt, atd)

# Literatura

- [1] B. J. HUTÁREK, “Klasifikace objektu v obraze podle textury,” Master’s thesis, Vysoké učení technické v Brně, Brno, 2010. [Online]. Available: [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)
- [2] S. K. Ameesh Makadia, Vladimir Pavlovic, “A new baseline for image annotation,” *A computer Vision*, 2008. [Online]. Available: [https://link.springer.com/chapter/10.1007/978-3-540-88690-7\\_24](https://link.springer.com/chapter/10.1007/978-3-540-88690-7_24)
- [3] V. Košář, “Srovnání deskriptorů pro reprezentaci obrazu,” Master’s thesis, Západočeská univerzita v Plzni, Plzeň, 2015. [Online]. Available: <https://dspace5.zcu.cz/bitstream/11025/17883/1/A13N0110P.pdf>
- [4] A. Rosebrock. (2015) Local binary patterns with python & opencv. Pyimagesearch. [Online]. Available: <http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/>