KIV/UIR - Semestrální práce

14. května 2017
40 hodin

Kateřina Kratochvílová
A13B0364P

Obsah

Abstrakt

Tato práce byla vytvořena za účelem vytvoření deskriptoru, který ponese informaci nejeno 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. My bychom ovšem chtěli 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. Musíme si být ale vědomi, že informace o barvě a textuře se mohou ovlinovat i protichůdně.

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

V práci rozebereme 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). Vstup algorimutmu se předpokládá šedotónový obrázek o rozměrech $m \times n$. Proto každý obrázek musí být po načtení převeden na šedotóny, protože většinou vkládáme obrázek barevný. [2]

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 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 pohlíže 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:

$maska(x) =$

-1	1

$maska(y) =$

-1
1

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:

$$|u| = \sqrt{u_1^2 + u_2^2} \tag{2.1}$$

Magnituda na základě vyše 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

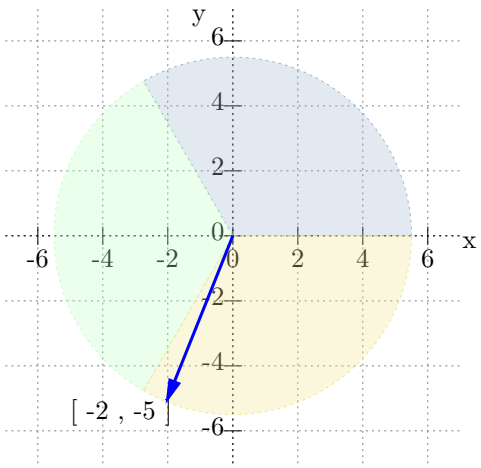
Obrázek 2.2: magnituda

Obrázek 2.3: gradient x

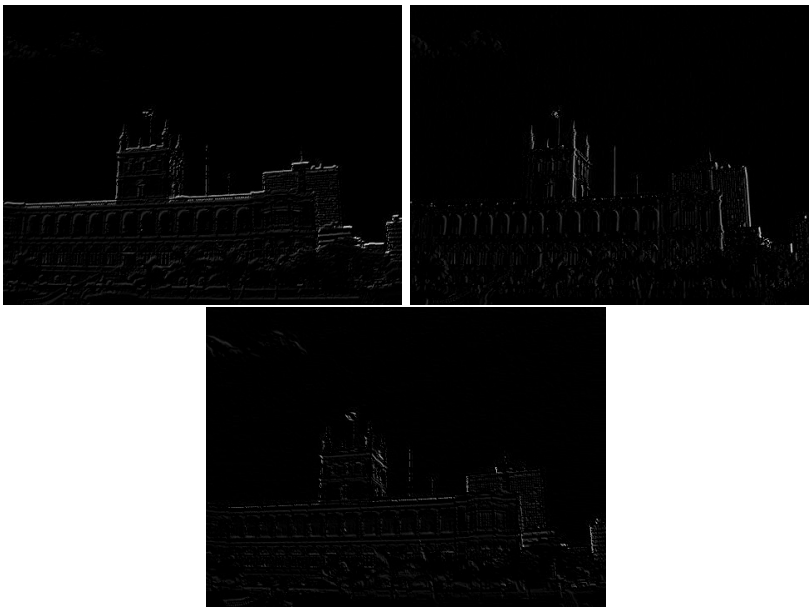
Obrázek 2.4: gradient y

2.0.2 Diskretizace směru gradientu

Pokud na gradienty budeme pohlížet jako na 2D vektory můžeme určit nejen jejich velikost(magnitudu) ale i jejich směr. Je možné použít znaménkovou reprezentaci $0 - \pi$ nebo neznaménkovou reprezentaci $0 - 2\pi$.
V praxi si rovnoměrně rozdělíme kružnici na několik dílů (dle počtu požadovaných směrů). Označme si počet dílů 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 $\left(\frac{4}{3}\pi - 2\pi\right)$. Máme připraveno d matic (pro každý směr jednu) a podle toho kam vektor směřuje, umístíme jeho magnitudu na souřadnice kde se nachází v původní matici.



Obrázek 2.5: 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.



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

2.0.3 Výpočet lokálního histogramu orientace gradientů z okolí

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

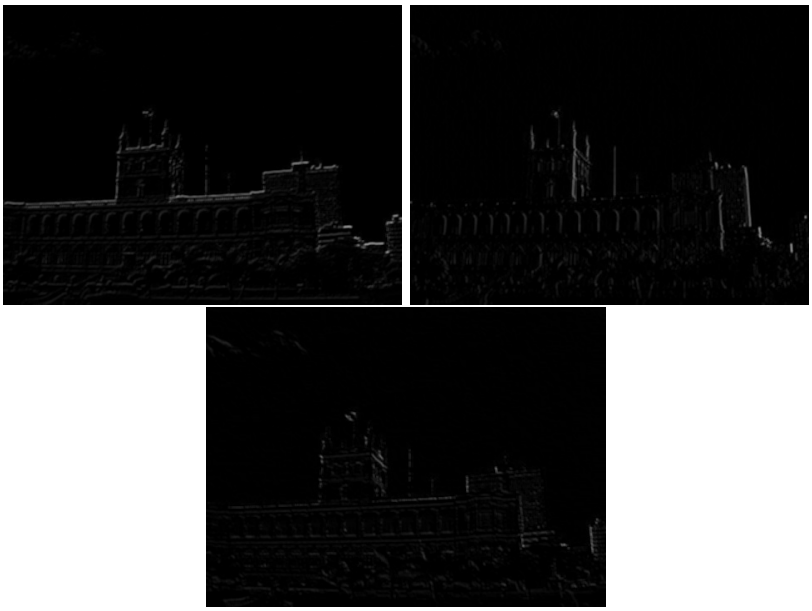
Výpočet u jednoho směru při cell = 3, Oranžově je vyznačená oblast výpočtu pro jeden pixel:

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:

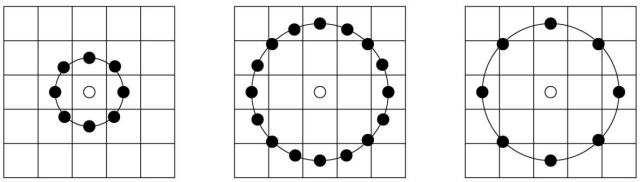
0	0
0.8	1.1
1.2	1.5



Obrázek 2.7: Obrázky po diskretizaci, oproti předchozím by měli být 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 3×3. Oproti tomu POEM můžeme aplikovat na větší okolí. Toto okolí nazýváme 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. Pokud bychom chtěli zvýšit stabilitu v téměř konstantní oblasti lze k centrálnímu pixelu přičítat malou konstantu τ.

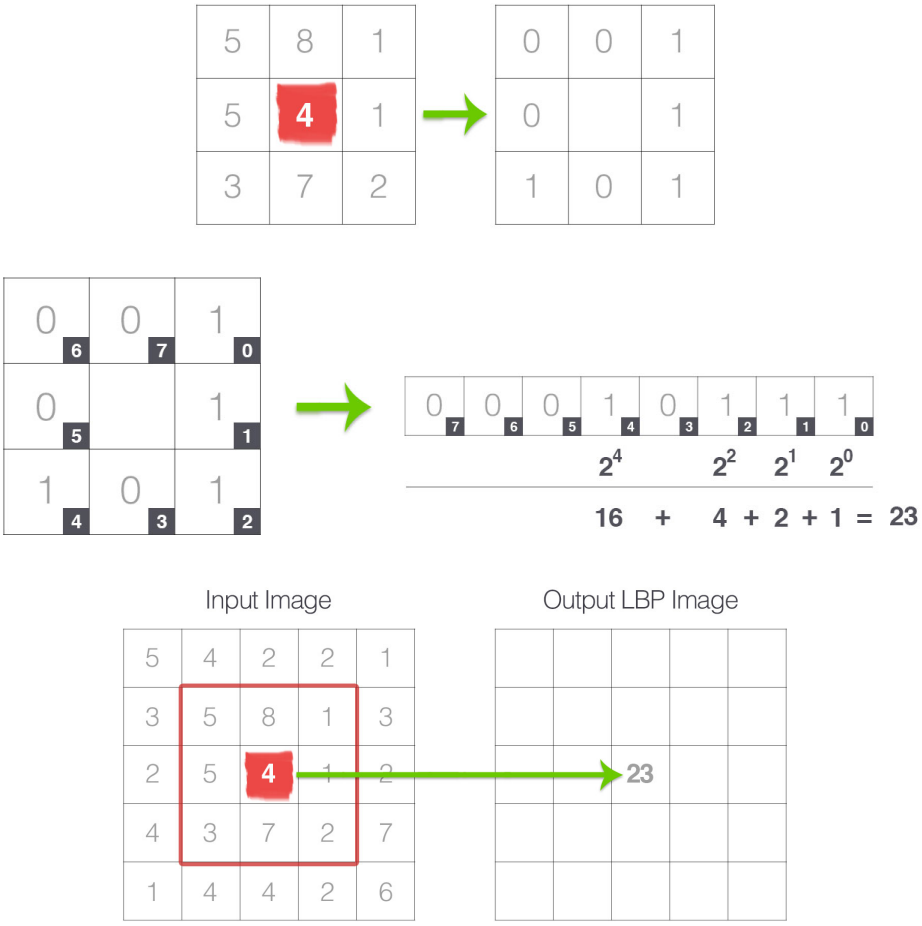


Obrázek 2.8: Znázornění blocku Převzato z [2]

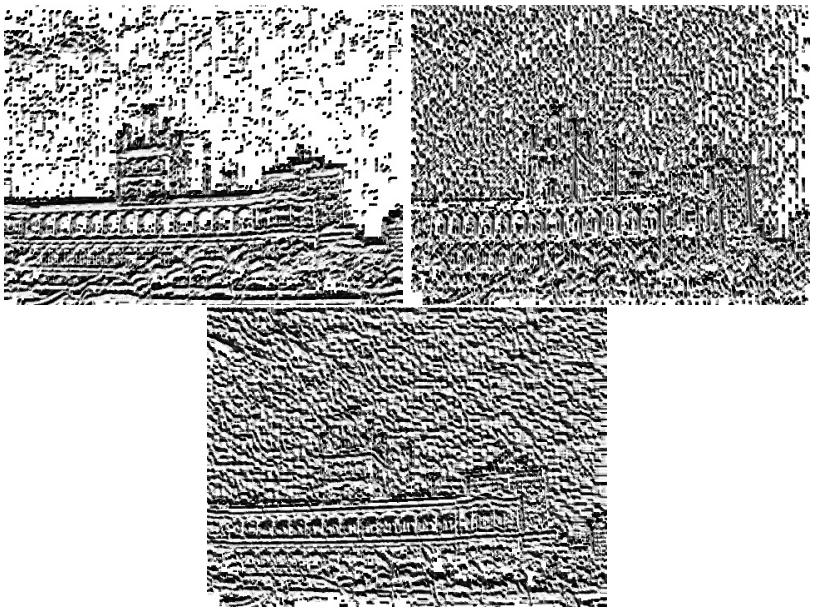
Následující matice představuje block pro střední pixel s hodnotou intenzity 4, označme si ho c (centrální). Následně procházíme všechny okolní pixely, označme si je x, hodnotu daného pixelu jako $p(x)$. $s(x)$ představuje výsledek tohoto porovnání.

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

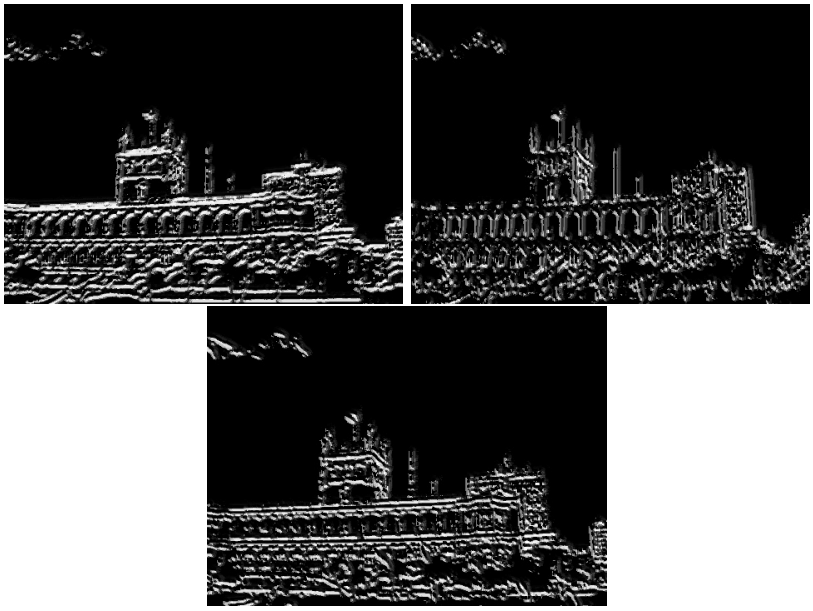
Zobrazení postupu algoritmu LBP:



Obrázek 2.9: Znázornění výpočtu LBP pro jeden pixel. Převzato z [3]



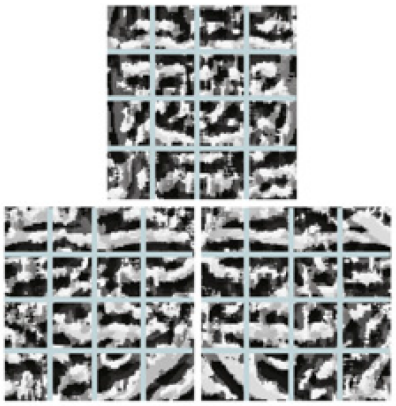
Obrázek 2.10: Obrázky po aplikaci LBP s velikostí blocku 8. Každý obrázek představuje jeden směr.



Obrázek 2.11: Obrázky po aplikaci LBP při použití τ. Každý obrázek představuje jeden směr.

2.0.5 Konstrukce globálního histogramu

Obrázky získané z LBP rozdělíme pravidelnou čtvercovou mřížkou. Pro každou vzniklou oblast vypočteme lokální histogram. Vzniklé histogramy zřetězíme. Díky tomu získáme tři histogramy pro každý směr jeden, které opět zřetězíme.



Obrázek 2.12: Obrázky získané z LBP jsou rozdeleny pravidelnou čtvercovou mřížkou. Převzato z [2]

Kapitola 3

Barevný poem

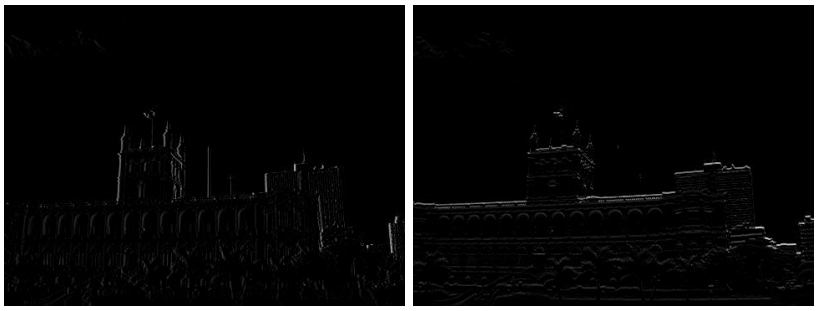
3.0.1 Výpočet gradientu a magnitudy u barevneho poemu

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

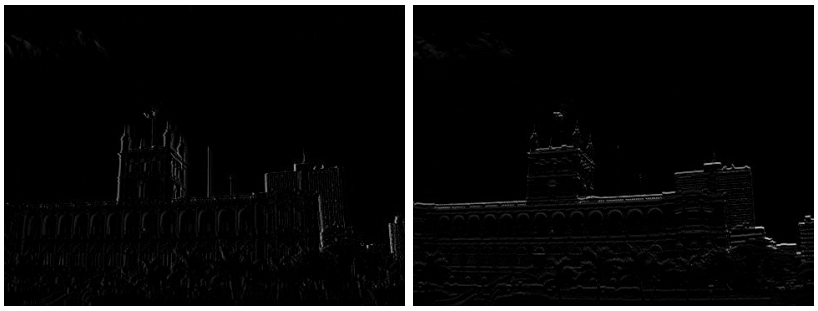
Format vzniklych vektoru

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

Obrázek 3.1: Gradienty pro modrou složku.



Obrázek 3.2: Gradienty pro červenou složku



Obrázek 3.3: Gradienty pro zelenou složku.

Pomocí součtu vektorú získáme jeden třísložkový vektor:

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

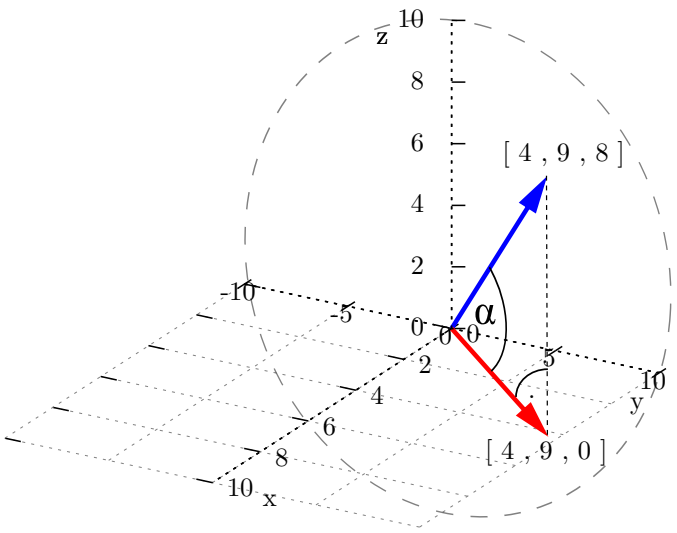
Výpočet velikosti vektoru v prostoru (magnituda):

$$|u| = \sqrt{u_1^2 + u_2^2 + u_3^2} \tag{3.4}$$

Obrázek 3.4: Porovnání magnitud. Vlevo magnituda z POEMU, vpravo magnituda z barevného POEMU.

3.0.2 Diskretizace směru gradientu u barevneho poemu

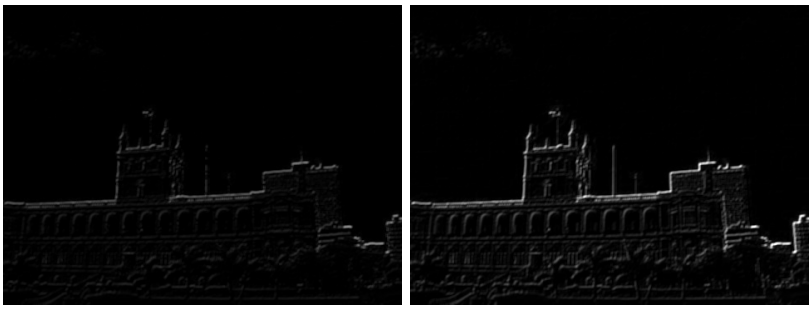
U vektorů získaných v předchozím kroku určíme velikost úhlu mezi vektorem a equivalentní vektorem s vynulovanou složkou z. Následně počítáme do které části naší kružnice vektor směřuje. Pro znaménkovou reprezentaci je celkový rozsah $0 - \pi$, pro neznaménkovou reprezentaci $0 - 2\pi$
Při neznaménkové 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 $\left(\frac{2\pi}{3} - \pi\right)$.



Obrázek 3.5: Diskretizace směru gradientu v neznamenkové 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°. Zároveň je vektor v prvním kvadrantu tudíž spadá do prvního intervalu. Pokud by ovšem byl vektor například $[4, -9, 8]$, tedy spadající do druhého kvadrantu, pak bychom jeho úhel počítali jako $180-$ úhel mezi $[4, -9, 8]$ a $[4, -9, 0]$. A vektor by připadl do třetího intervalu.

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.

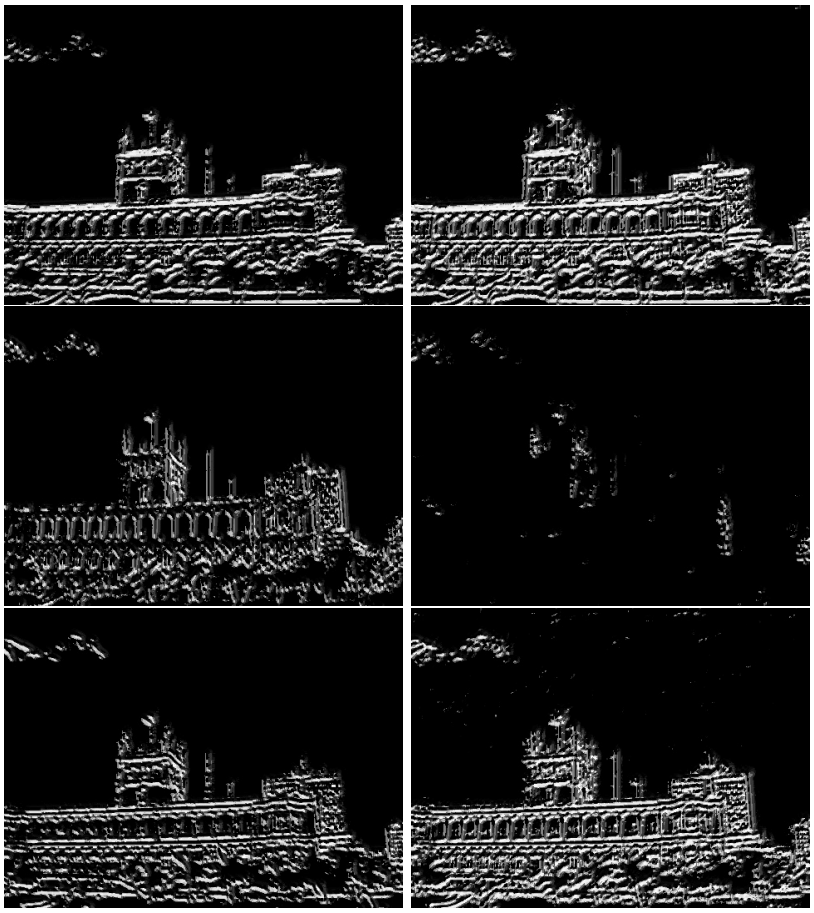


Obrázek 3.6: Obrázky po diskretizaci první směr. Vlevo původní POEM, vpravo barevný POEM

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

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

3.0.4 Zakódování příznaku pomocí LBP u barevneho poemu



Obrázek 3.9: Obrázky po aplikaci LBP s použitím τ. 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 kterým 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 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++.
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. Ke každému obrázku jsou přiložena metadata ve formátu XML, která obsahují informace o obrázku v různých jazycích. Kromě angličtiny zde nalezneme například i španělštinu nebo němčinu. Metadatata ovšem neobsahují klíčová slova tak jak bychom si je představovali. V jednotlivých elementech jsou roztřízené obsáhle 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 jsme zíkali klíčová slova které byla extrahována právě z přiloženého xml. Z celkových 20 000 obrázků jich bylo použitou 19 805 a to 17 664 na trénování a 1960 na testování. K jednomu obrázku je v průměru přiřazeno 5.7 klíčových slov.

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

Kapitola 6

Závěr

V semestrální práci jsem vytvořila skript, který z vloženého obrázku vytvoří POEM deskriptor. Dále jsme vyzkoušeli vytvoření POEM na barevný obrázek, tak aby obsahoval jak informaci o textuře tak informaci o barvě. Použitý POEM byl použit pro porovnání v bakalářské práci *Automatická anotace obrázků*.

Dosažené výsledky:

	Přesnost (%)	Úplnost (%)	Nenulových slov
RGB	0	0	0
LAB	12	7	148
HSV	0	0	0
RGB, LAB, HSV	0	0	0
POEM	0	0	0
RGB, LAB, HSV, POEM	0	0	0
Barevný POEM	0	0	0

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 rozdeleni do smeru, smer 1 (analogicky directional2.jpg, atd)
- *directional1.txt* - magnitudy po rozdeleni 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

[2] V. Košař, "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

[3] 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/