KIV/UIR - Semestrální práce

Kateřina Kratochvílová

dtwok8@students.zcu.cz

8. května 2017

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 textury pracuje pouze s šedotónovým obrázkem. Takže zahazuje informaci o barvě a pracuje 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. 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). 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.

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.

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 vektor, 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.

Vstupní obrázek:

$$\begin{pmatrix} 8 & 7 & 5 \\ 1 & 2 & 4 \\ 3 & 5 & 7 \end{pmatrix}$$

Masky:

$$maska(x) = \begin{pmatrix} -1 & 1 \end{pmatrix} \tag{2.1}$$

$$maska(y) = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \tag{2.2}$$

Na základě předchozích masek byly vypočteny tyto matice gradientů:

$$gradient(x) = \begin{pmatrix} -1 & -2 & 0 \\ 1 & 2 & 0 \\ 2 & 2 & 0 \end{pmatrix} \tag{2.3}$$

$$gradient(y) = \begin{pmatrix} -7 & -5 & -1 \\ 2 & 3 & 3 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.4}$$

Gradient jako 2D vektory:

$$\begin{pmatrix} [-1;-7] & [-2;-5] & [0;-1] \\ [1;2] & [2;3] & [0;3] \\ [2;0] & [2;0] & [0;0] \end{pmatrix} \tag{2.5}$$

Vzoreček pro výpočet velikosti vektoru v rovině:

$$|u| = \sqrt{u_1^2 + u_2^2} \tag{2.6}$$

Obrázek 2.1: Vstupní obrázek

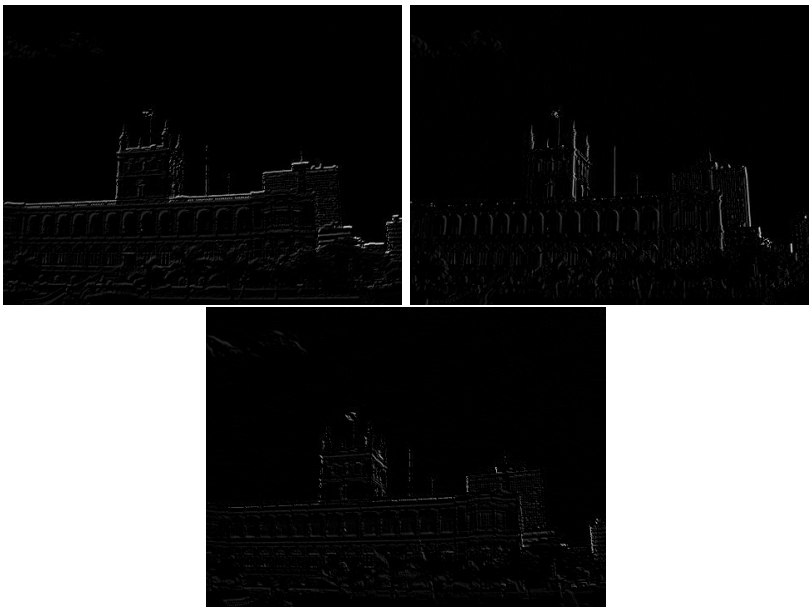
Obrázek 2.2: magnituda

Obrázek 2.3: gradient x

Obrázek 2.4: gradient y

2.0.2 Diskretizace směru gradientu

Pokud se na gradienty budeme koukat 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 (podle toho kolik chceme směrů). Označme si počet dílů d. Pro $d = 3$ znaménkovou reprezentaci to tedy bude $0-\frac{2}{3}\pi$, $\frac{2}{3}\pi-\frac{4}{3}\pi$ a $\frac{3}{3}\pi - 2\pi$



Obrázek 2.5: Obrázky po diskretizaci. Každý obrázek představuje jeden směr.

2.0.3 Výpočet lokálního histogramu orientace gradientů z okolí

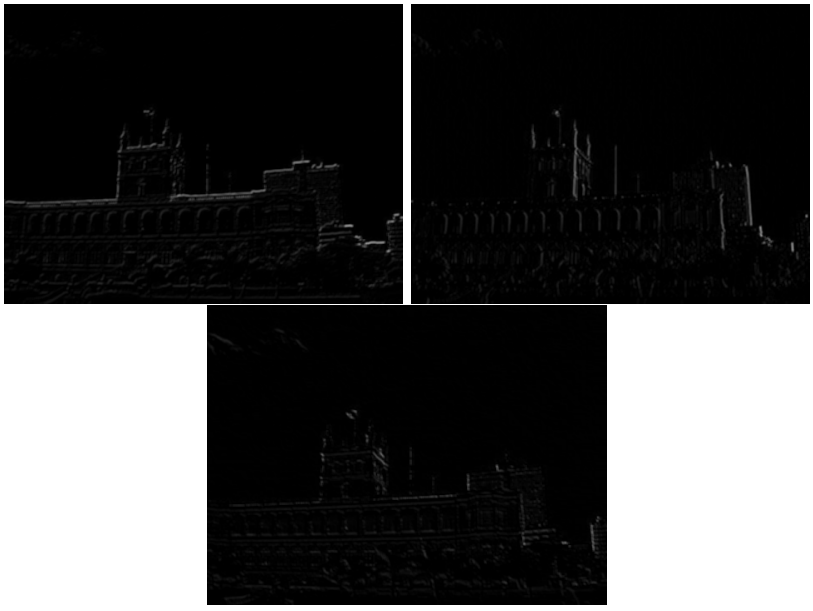
U každého směru projdeme jednotlivé pixely s jejich okolí a zprůměrujeme jejich hodnoty. Toto okolí se nazývá cell. Ukázka výpočtu u jednoho směru při velikosti cell 3.

Výpočet u jednoho směru při cell = 3:

$$smer = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3.6 & 3.6 & 3 \\ 0 & 2 & 2 & 0 \end{pmatrix} aems = \begin{pmatrix} 0 & 0 \\ 0.8 & 1.1 \\ 1.2 & 1.5 \end{pmatrix} \tag{2.7}$$

Výpočet pro jeden pixel jednoho směru při cell = 3:

$$cell = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.6 & 3.6 \\ 0 & 2 & 2 \end{pmatrix} aems = \begin{pmatrix} 0.8 \end{pmatrix} \tag{2.8}$$

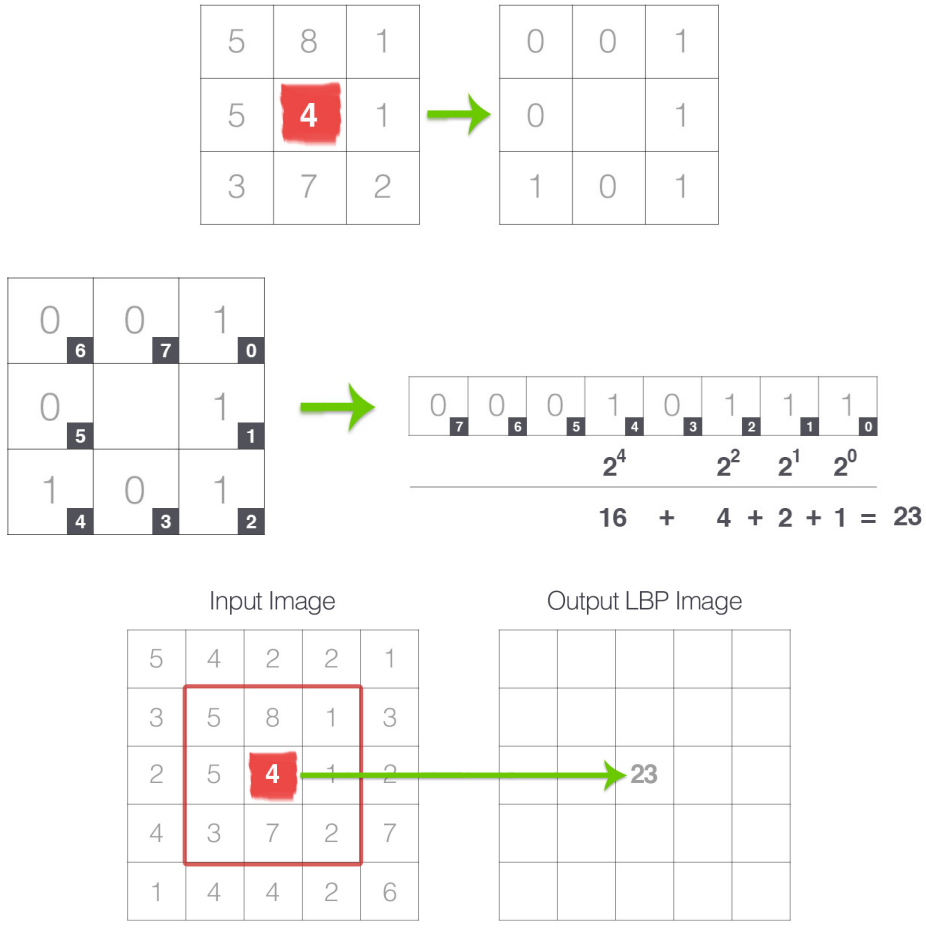


Obrázek 2.6: Obrázky po výpočtu lokálních histogramů orientace gradientů. Obrázky by měli mít rozmazanější hrany. Každý obrázek představuje jeden směr.

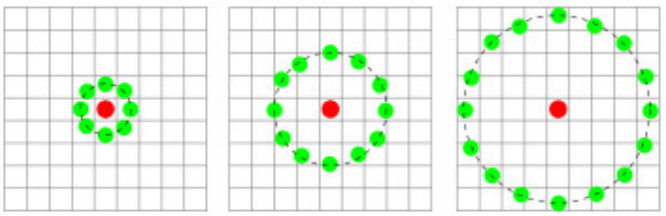
2.0.4 Zakódování příznaků pomocí LBP

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

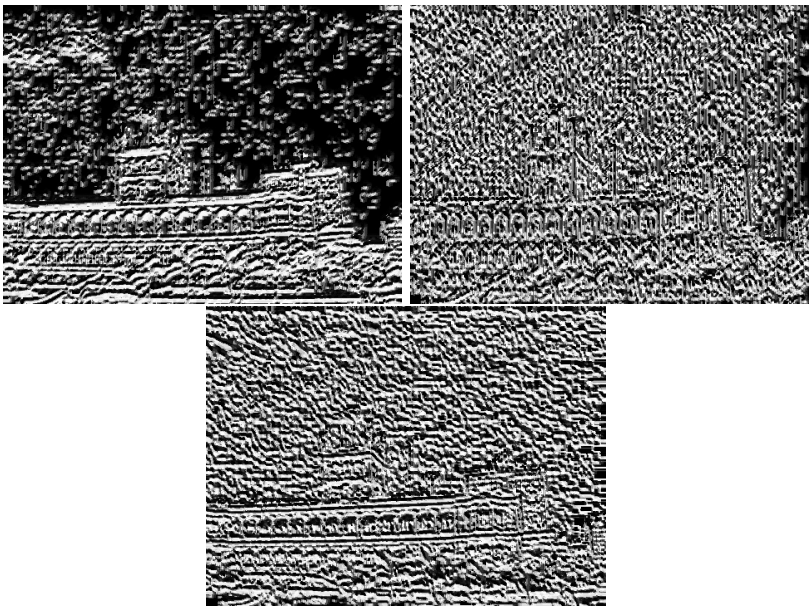
se jedná o kruhové okolí s poloměrem L/2 (L představuje velikost blocku). Pro stanovení intenzit okolních hodnot je možné použít bilineární interpolaci. Pokud bychom chtěli zvýšit stabilitu v téměř konstantní oblasti lze k centrálnímu pixelu přičítat malou konstantu τ.



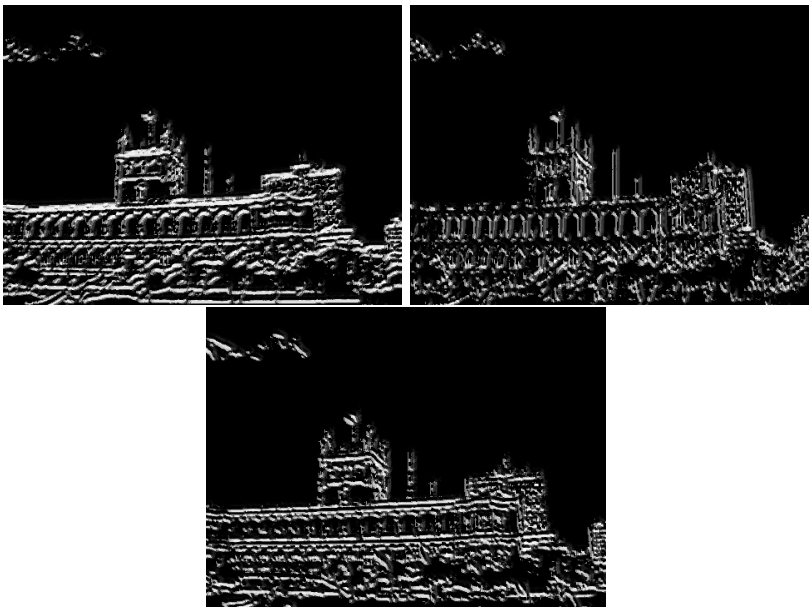
Obrázek 2.7: Znázornění LBP



Obrázek 2.8: Znázornění LBP



Obrázek 2.9: Obrázky po aplikaci LBP. Každý obrázek představuje jeden směr.



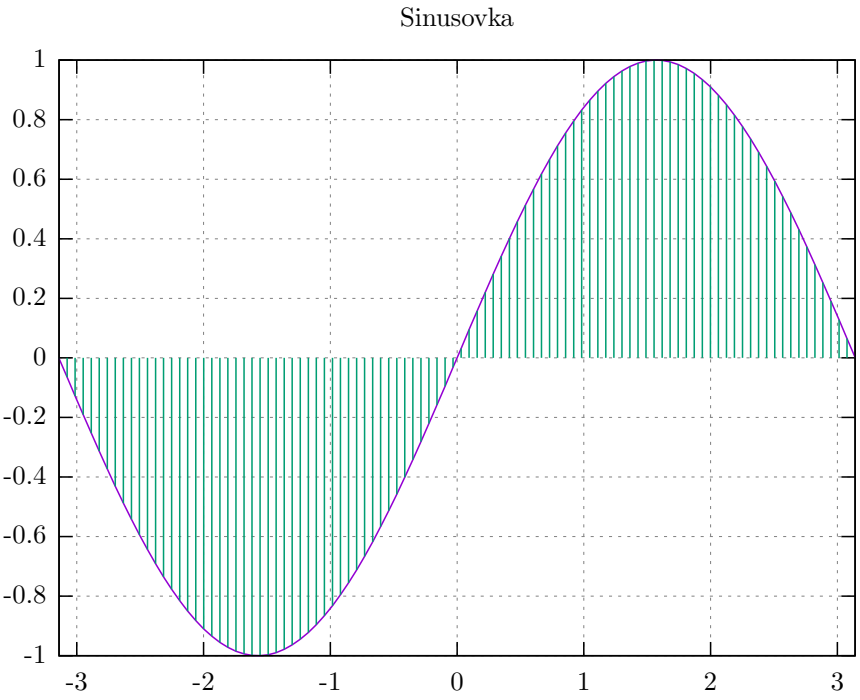
Obrázek 2.10: Obrázky po aplikaci LBP při použití τ. Každý obrázek představuje jeden směr.

2.0.5 Konstrukce globálního histogramu

256 binu pro každý směr a zřetězí se.

Kapitola 3

Barevný poem



Obrázek 3.1: Your image caption

3.0.1 Výpočet gradientu a magnitudy

Výpočet gradientu probíhá obdobně jako u nebarevného obrázku. Pro každou složku získáme 2 matice takže celkem budeme mít 3*2 matice. Na které se dá pohlížet jako na 2 vektory o 3 složkách. Tyto vektory sloučíme pomocí součtu vektoru do jednoho 3 složkového vektoru. Magnituda je opět velikost vektoru tentokrát, ale v prostoru.

Součet vektoru:

$$\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3) \tag{3.1}$$

Výpočet velikosti vektoru v prostoru:

$$|u| = \sqrt{u_1^2 + u_2^2 + u_3^2} \tag{3.2}$$

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

Závěr

V semestrální práci jsem vytvořila skript, který z vloženého obrázku vytvoří POEM deskriptor.

Kapitola 6

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* - v případě poemu pracujícím s šedotónovým obrázkem
- *python color_poem.py* - 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
- *directional0.jpg* - obrázek magnitud po rozdeleni do smeru, smer 0 (analogicky directional1.jpg, atd)
- *directional0.txt* - magnitudy po rozdeleni do smeru, smer 0 (analogicky directional1.txt, atd)
- *ames0.jpg* - obrázek po výpočtu lokálních histogramů orientace gradientů, smer 0 (analogicky aems1.jpg, atd)
- *ames0.txt* - matice po výpočtu lokálních histogramů orientace gradientů, smer 0 (analogicky aems1.txt, atd)

- *lbp0.jpg* - obrázek výsledného LBP, smer 0 (analogicky lbp1.jpg, atd)
- *lbp0.txt* - výsledného LBP, smer 0 (analogicky lbp1.txt, atd)

Kapitola 7

Zdroje

https://dspace5.zcu.cz/bitstream/11025/17883/1/A13N0110P.pdf
https://en.wikipedia.org/wiki/Local_binary_patterns
http://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/
https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=
117319