

KIV/UIR - Semestrální práce

Kateřina Kratochvílová

dtwok8@students.zcu.cz

7. května 2017

Obsah

1	Poem	1
1.0.1	Výpočet gradientu a magnitudy	1
1.0.2	Diskretizace směru gradientu	2
1.0.3	Výpočet lokálního histogramu orientace gradientů z okolí	3
1.0.4	Zakódování příznaků pomocí LBP	5
1.0.5	Konstrukce globálního histogramu	8
2	Barevný poem	9
3	Použité programové prostředky	11
3.1	OpenCV	11
4	Závěr	12
5	Uživatelská dokumentace	13
6	Zdroje	14

Abstrakt

abstrakt

Kapitola 1

Poem

POEM (Patterns of Oriented Edge Magnitudes). Vstup algoritmu se předpokládá šedotónový obrázek o rozměrech $m \times n$.

1.0.1 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 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.

Ukázka vstupního obrázku:

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

Ukázka masky:

$$maska(x) = \begin{pmatrix} -1 & 1 \end{pmatrix} \quad (1.1)$$

$$maska(y) = \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad (1.2)$$

Ukázka gradientů:

$$gradient(x) = \begin{pmatrix} -1 & -2 & 0 \\ 1 & 2 & 0 \\ 2 & 2 & 0 \end{pmatrix} \quad (1.3)$$

$$gradient(y) = \begin{pmatrix} -7 & -5 & -1 \\ 2 & 3 & 3 \\ 0 & 0 & 0 \end{pmatrix} \quad (1.4)$$

Gradient jako 2D vektory

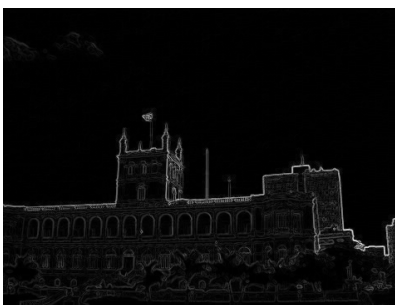
$$\begin{pmatrix} [-1; -7] & [-2; -5] & [0; -1] \\ [1; 2] & [2; 3] & [0; 3] \\ [2; 0] & [2; 0] & [0; 0] \end{pmatrix} \quad (1.5)$$

Vzoreček pro výpočet velikosti vektoru v rovině

$$|u| = \sqrt{u_1^2 + u_2^2} \quad (1.6)$$



Obrázek 1.1: Vstupní obrázek



Obrázek 1.2: magnituda

1.0.2 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 $\frac{4}{3}\pi - 2\pi$



Obrázek 1.3: gradient x



Obrázek 1.4: gradient y

1.0.3 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.

Ukázka výpočtu u jednoho směru při velikosti cell 3

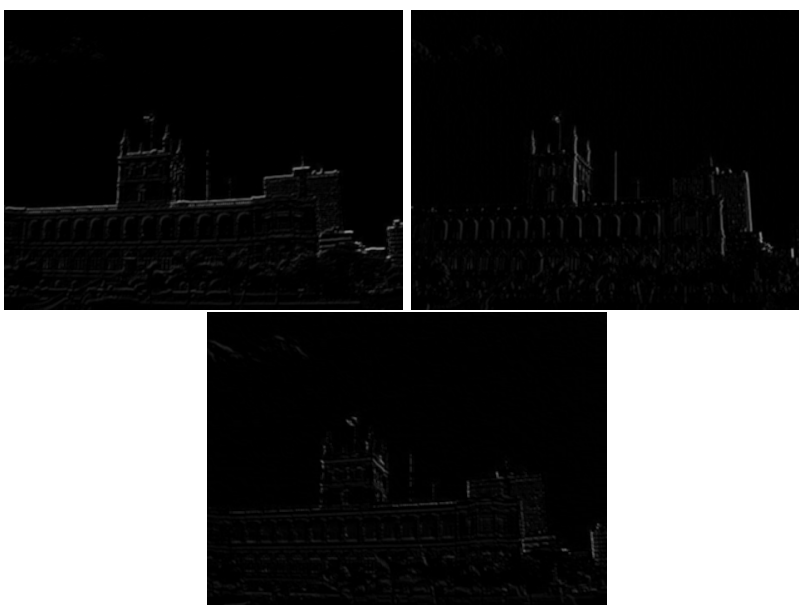
$$smer = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 3.6 & 3.6 & 3 \\ 0 & 2 & 2 & 0 \end{pmatrix} aems = \begin{pmatrix} 0 & 0 \\ 0.8 & 1.1 \\ 1.2 & 1.5 \end{pmatrix} \quad (1.7)$$

Ukázka výpočtu pro jeden pixel při velikosti cell 3

$$cell = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 3.6 & 3.6 \\ 0 & 2 & 2 \end{pmatrix} aems = (0.8) \quad (1.8)$$



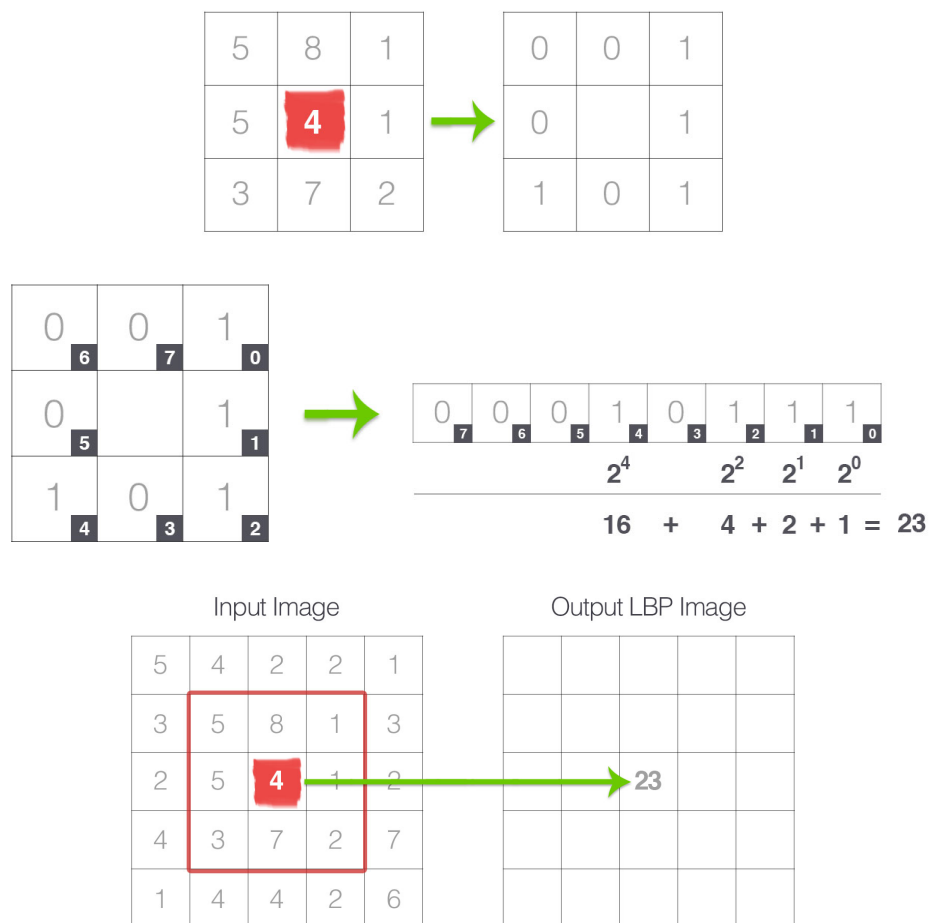
Obrázek 1.5: Obrázky po diskretizaci. Každý obrázek představuje jeden směr.



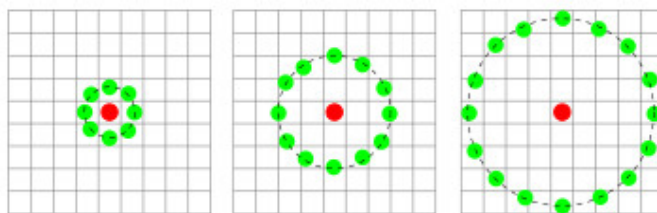
Obrázek 1.6: Obrázky po výpočtu lokálních histogramů orientace gradientů . Každý obrázek představuje jeden směr.

1.0.4 Zakódování příznaků pomocí LBP

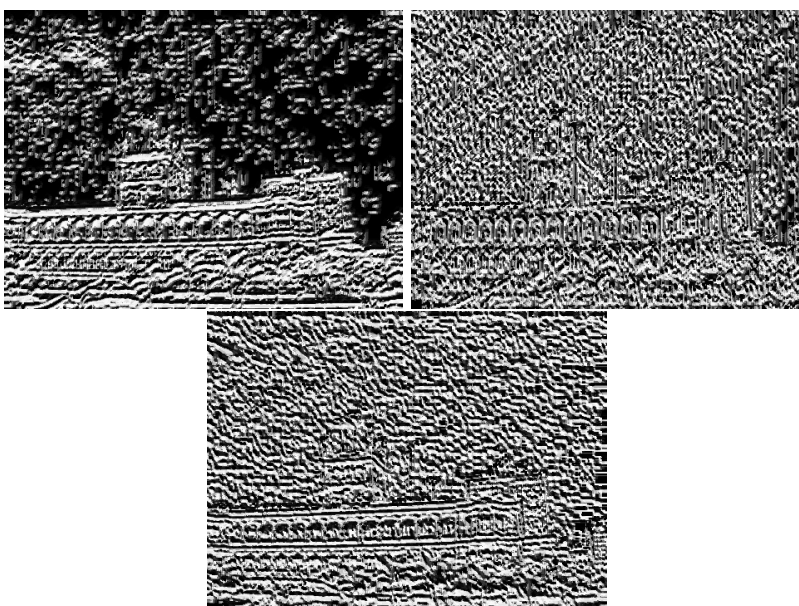
LBP operátor je aplikován na okolí každého pixelu o velikost 3×3 . Oproti tomu POEM můžeme aplikovat na větší okolí. Toto okolí nazýváme block, zpravidla se jedná o kruhové okolí s poloměrem $L/2$. Pro stanovení intenzit okolních hodnot je možné použít bilineární interpolaci. Pokud bychom chtěli zvýšit stabilitu v téměř konstantní oblasti lze k centrálnímu pixelu přičítat malou konstantu τ .



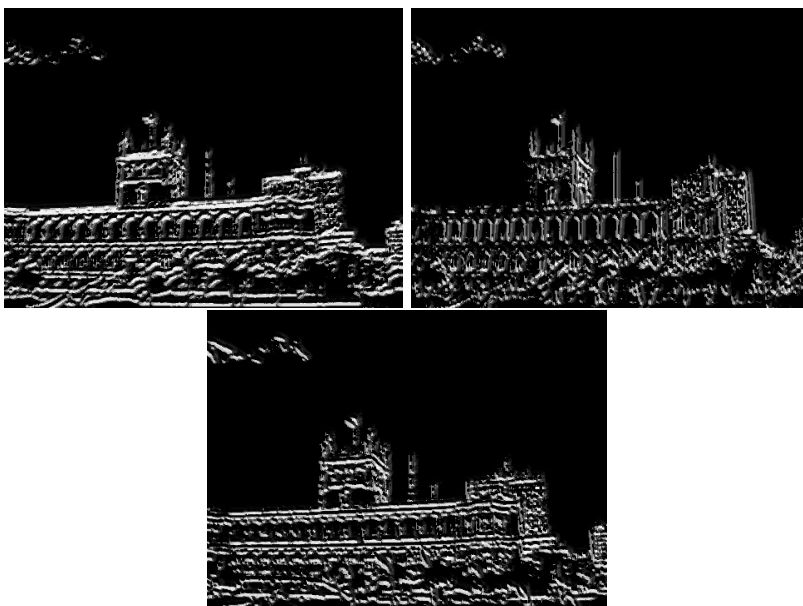
Obrázek 1.7: Znázornění LBP



Obrázek 1.8: Znázornění POEM



Obrázek 1.9: Obrázky po aplikaci POEM. Každý obrázek představuje jeden směr.



Obrázek 1.10: Obrázky po aplikaci POEM při použití τ . Každý obrázek představuje jeden směr.

1.0.5 Konstrukce globálního histogramu

256 binu pro každý směr a zřetězí se.

Kapitola 2

Barevný poem

LBP i Gabor pracují s informací o intenzitě obrazu. Detekce hran. Obyčejné LBP problém s rotací.

kombinace textur a barevne informace 1. Vytvoření společného příznaku, například rozšíření LBP na všechny barvené kanály informace o barvě a textuře se mohou obliňovat protichůdně

2. Vyhodnotit a klasifikovat příznaky odděleně a pak výslednou klasifikaci nějak spojit z několika částí to je například JEC

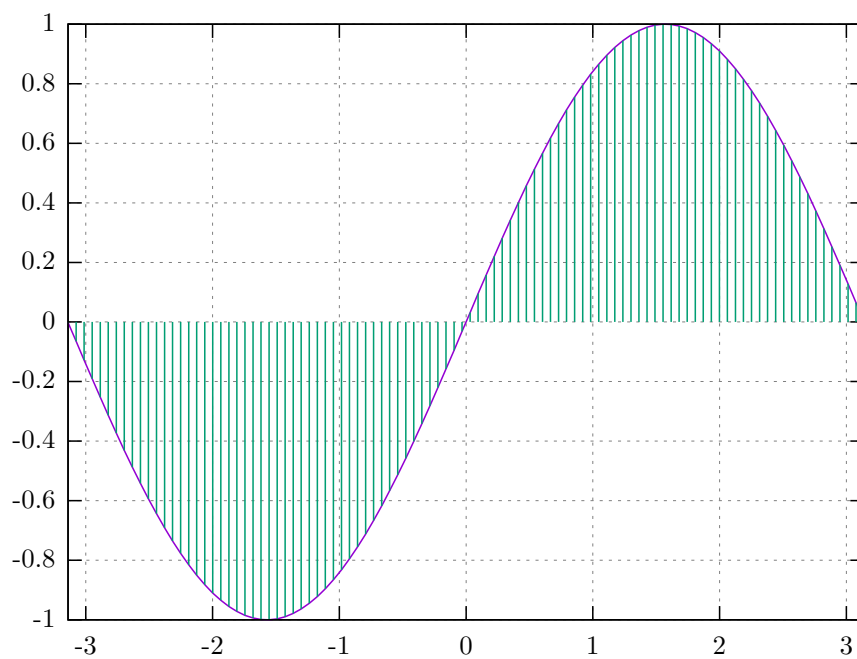
výhoda zachovává vlastnosti obou původních příznaků výpočetně náročnější a jeho úspěšnost je přímo závislá na způsobu kombinace obou informací

vutbrno 117319 10 stranka

$$|u| = \sqrt{u_1^2 + u_2^2 + u_3^2} \quad (2.1)$$

$$\vec{u} + \vec{v} = (u_1 + v_1, u_2 + v_2, u_3 + v_3) \quad (2.2)$$

Sinusovka



Obrázek 2.1: Your image caption

Kapitola 3

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. Program využívá knihovnu OpenCV 3.1.

3.1 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 4

Závěr

V teoretické části byly popsány nízkoúrovňové příznaky barva a textura. Byla rozebrána metoda JEC, která bude v bakalářské práci implementována. Seznámili jsme se s knihovnou OpenCV, prostudovali obrázky y apřiložená metadata.

Kapitola 5

Uživatelská dokumentace

popsani jak vypada zdrojovej soubor kterej to zere, nejdriv cesta k souboru a pak jeho klicovy slova

Kapitola 6

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/>