SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Diplomski sveučilišni studij računarstva

Obrada slike i računalni vid

Local Binary Patterns

Seminarski rad

Ivana Puljić

Osijek, 2018.

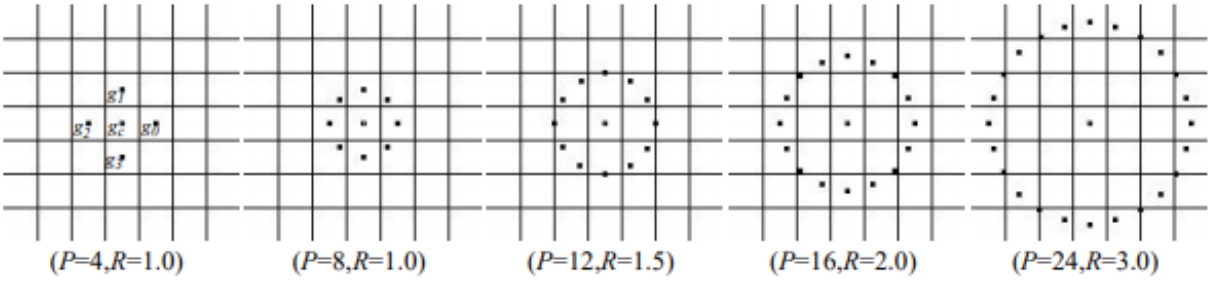
Sadržaj

1. UVOD

Lokalni binarni uzorak (engl. Local Binary Patterns, LBP) je metoda koja je zasnovana na uspoređivanju središnjeg piksela sa susjednim pikselima. Cilj ove metode je sažeti sliku na njezinu lokalnu strukturu. Uspješno se koristi u mnogim područjima obrade slike i računalnog vida: detekcija lica, prepoznavanje lica, prepoznavanje izraza lica, detekcija tekstura i pokreta.

2. PREGLED PODRUČJA I PROBLEMATIKE

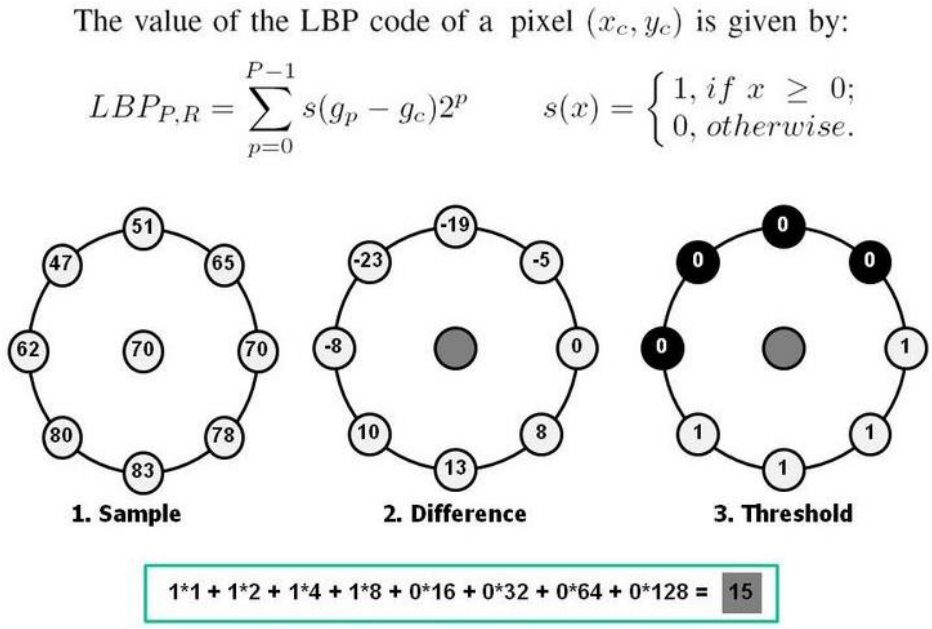
Local binary pattern, ili skraćeno LBP, pojavljuje se 1993. godine, a postaje popularan nakon objavljivanja članka *Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns* 2002. godine. LBP je jednostavan i učinkovit operator koji označava vrijednosti piksela slike tako što provodi operaciju threshold na susjednim pikselima odabranog piksela. LBP operator označava piksele slike decimalnim brojevima, te svaki piksel uspoređuje sa susjednim pikselima. Za svaki piksel slike, odabire se broj susjednih piksela P i radijus R koji predstavlja udaljenost središnjeg piksela od susjednih piksela, što je vidljivo na slici 2.1.



Slika 2.1. Prikaz središnjeg piksela i susjednih piksela ovisno o radijusu R i broju susjednih piksela P

Nakon odabira radijusa R i broja susjednih piksela P, računa se LBP vrijednost središnjeg piksela. Gleda se vrijednost središnjeg piksela i vrijednost susjednih piksela se uspoređuje sa središnjim pikselom. Ukoliko je vrijednost susjednog piksela veća od središnjeg piksela, susjednom pikselu se pridaje vrijednost 1, dok se susjednom pikselu koji ima manju vrijednost od središnjeg piksela pridaje vrijednost 0. Zatim se zapisuje binarni zapis susjednih piksela te se dobivena binarna vrijednost pretvara u dekadsku vrijednost koja predstavlja LBP

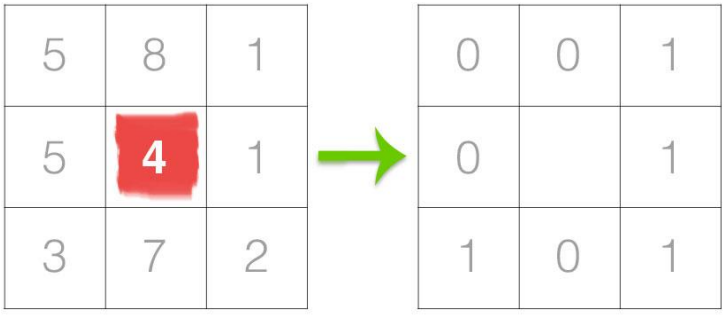
vrijednost središnjeg piksela. Moguć broj kombinacija LBP vrijednosti središnjeg piksela iznosi 2^P, gdje P predstavlja broj susjednih piksela. Postupak se ponavlja za sve piksele slike.



Slika 2.2. Primjer dobivanja vrijednosti središnjeg piksela

3. OPIS ZADATKA S DOBIVENIM REZULTATIMA

Za realiziranje Local Binary Pattern operatora korišten je Python programski jezik sa biblotekama Numpy i OpenCv. Kao P, odnosno broj susjednih piksela, odabrana je vrijednost 8, dok je za R radijus odabrana vrijednost 1. Na slici 3.1. prikazan je izgled središnjeg i susjednih piksela s odabranim parametrima.



Slika 3.1. Prikaz središnjeg piksela sa 8 P susjednih piksela i radijusom R koji iznosi 1

Prvi korak u realizaciji bio je učitati sliku za obradu. Zatim je potrebno napisati funkciju koja dohvaća središnji piksel slike (x,y) i njegovih 8 susjednih piksela. Funkcija kao parametre prima sliku za obradu te x i y vrijednost promatranog piksela.

```
#loading and converting image to grayscale
img = cv2.imread("/home/osirv/Osirv projekt/lenna.bmp", cv2.IMREAD_GRAYSCALE)
img_after = cv2.imread("/home/osirv/Osirv projekt/lenna.bmp", cv2.IMREAD_GRAYSCALE)

#selecting pixels, for each pixel in the image, neighbor pixels surrounding the center pixel are selected
def get_pixel(img, x, y):
    try:
        return img[x,y]
    except:
        pass
```

Slika 3.2. Prikaz učitavanja slike i funkcije za dohvaćanje piksela

Nakon toga slijedi funkcija za uspoređivanje središnjeg i susjednih piksela. Funkcija kao parametre prima središnji piksel i 8 susjednih piksela. Zatim uspoređuje svaki susjedni piksel sa središnjim i vraća rezultat. Ako je središnji piksel veći od susjednog, funkcija vraća vrijednost 0, a ako je središnji piksel manji od susjednog, funkcija vraća vrijednost 1.

```
#comparing neighbor pixels values with center pixel value
#if center pixel has larger value than neighbor pixel, neighbor pixel value is set to 0
#if center pixel has smaller value than neighbor pixel, neighbor pixel value is set to 1;
def calc_pixel_value(center_pixel, neighbor_pixels):
    value=[]
    for pixel in neighbor_pixels:
        if pixel >= center_pixel:
            value.append(1)
        else:
            value.append(0)
    return value
```

Slika 3.3. Prikaz funkcije za uspoređivanje središnjeg sa susjednim pikselima

Nakon toga, slijedi glavni dio koda koji se izvršava za svaki piksel slike. Prvo se dohvaćaju svi stupci i redci slike. Zatim je potrebno odrediti položaje središnjeg i susjednih piksela. Za to se koristi funkcija get_pixel prikazana na slici 3.2. Središnji piksel ima položaj (x,y), dok se ostalim pikselima mijenja položaj ovisno o tome gdje se nalaze od središnjeg piksela. Definiranje položaja susjednih piksela vidljivo je na slici 3.4.

```
for x in range(0, len(img)): #select all rows
    for y in range(0, len(img)): #select all columns
        center_pixel     = img[x,y] #center pixel
        above_pixel      =get_pixel(img, x, y-1) #pixel above center pixel
        down_pixel       =get_pixel(img, x, y+1) #pixel under center pixel
        right_pixel      =get_pixel(img, x+1, y) #right pixel
        left_pixel       =get_pixel(img, x-1, y) #left pixel
        aboveleft_pixel  =get_pixel(img, x-1, y-1) #above left pixel
        aboveright_pixel =get_pixel(img, x+1, y-1 ) #above right pixel
        downleft_pixel   =get_pixel(img, x-1, y+1) #down left pixel
        downright_pixel  =get_pixel(img, x+1, y+1) #down right pixel
```

Slika 3.4. Prikaz definiranja položaja susjednih piksela

Zatim slijedi pozivanje funkcije za uspoređivanje središnjeg piksela sa susjednim pikselima i pridavanje vrijednosti susjednim pikselima. Funkcija prima središnji piksel i susjedne piksele poredane u smjeru kazaljke na satu počevši od gornjeg lijevog piksela. Rezultat je binarni zapis vrijednosti središnjeg piksela. Nakon toga su navedene vrijednosti potencija broja dva od 2^0 do 2^8. Te vrijednosti su potrebne da bi se binarna vrijednost središnjeg piksela pretvorila u dekadsku vrijednost središnjeg piksela. Dobivena binarna vrijednost se množi sa potencijama broja 2, te se tako dobiva dekadska vrijednost središnjeg piksela.

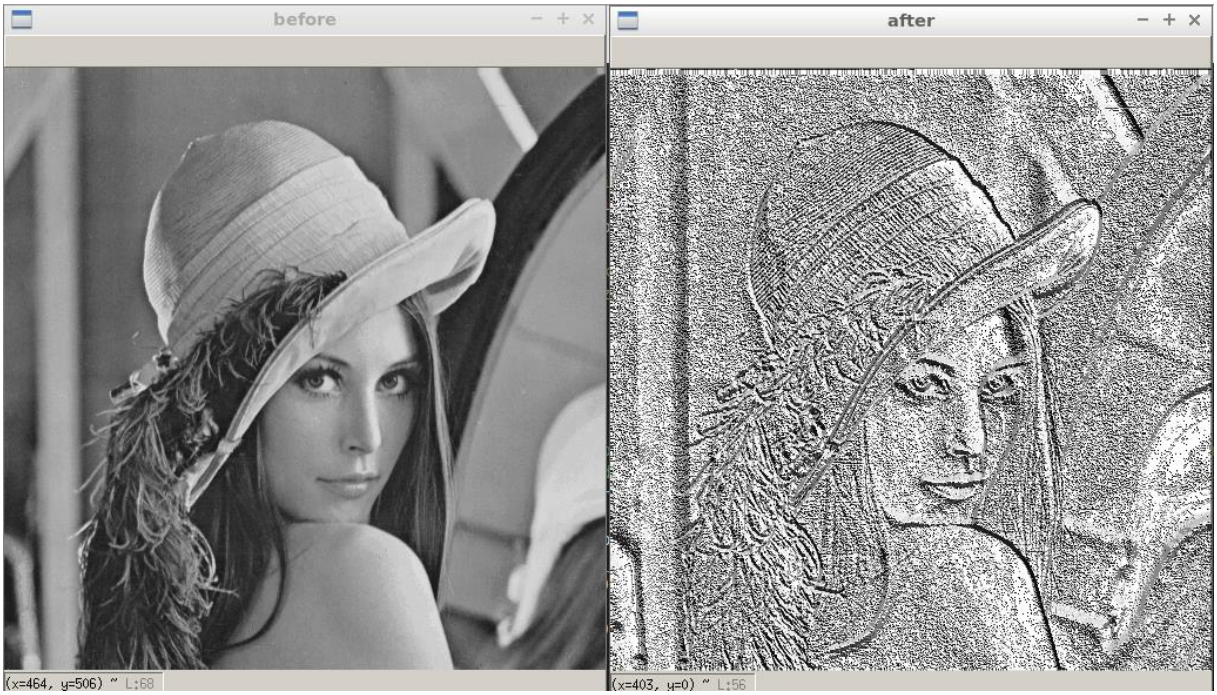
```
#selecting neighbour pixels clockwise, starting from the top left pixel and getting their value depending on the value of center pixel
#store pixels in 8 bit array
lbp_pixels = calc_pixel_value(center_pixel, [aboveleft_pixel, above_pixel, aboveright_pixel, right_pixel, downright_pixel,
                              down_pixel, downleft_pixel, left_pixel])

lbp_combinations = [1, 2, 4, 8, 16, 32, 64, 128]
lbp_pixel_value = 0
for pixel in range(0, len(lbp_pixels)):
    #calculating final pixel value
    lbp_pixel_value = lbp_pixel_value+ (lbp_combinations[pixel]* lbp_pixels [pixel])

img_after.itemset((x,y), lbp_pixel_value)
```

Slika 3.4. Prikaz dobivanja binarnog zapisa središnjeg piksela i pretvaranje binarne vrijednosti u dekadsku vrijednost središnjeg piksela

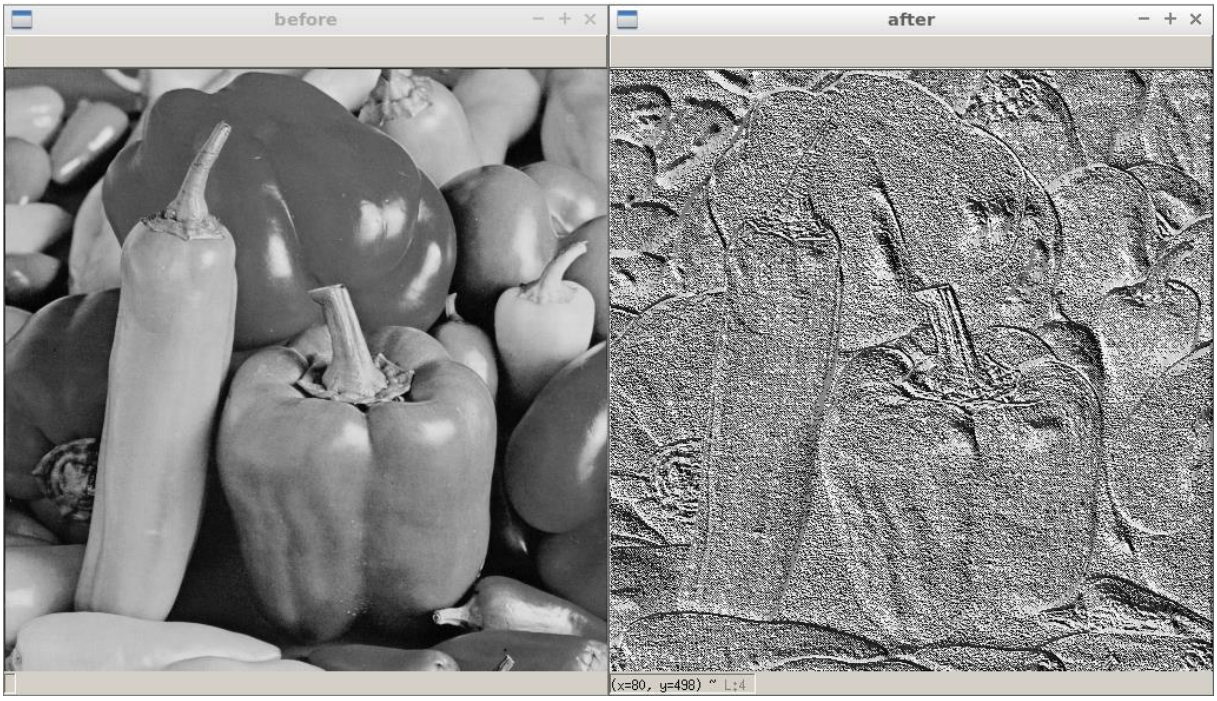
Postupak se ponavlja za svaki piksel slike te se kao rezultat dobije slika sažeta na svoju lokalnu strukturu.

```
cv2.imwrite ( "/home/osirv/Osirv projekt/" +'lbp2.png', img_after)
cv2.imshow('before', img)
cv2.imshow('after', img_after)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Slika 3.5. Spremanje i prikaz dobivenih rezultata



Slika 3.6. Prikaz dobivenih rezultata prije i poslije obrade



Slika 3.7. Prikaz dobivenih rezultata prije i poslije obrade

4. ZAKLJUČAK

Local binary patterns je korisna i moćna metoda za predstavljanje lokalne strukture slike. Zbog toga i zbog jednostavnosti realizacije, uspješno se koristi u mnogim područjima obrade slike i računalnog vida.

5. LITERATURA

1. Di Huang, Caifeng Shan, Mohsen Ardebilian, Yunhong Wang, and Liming Chen: Local Binary Patterns and Its Application to Facial Image Analysis: A Survey
2. Timo Ojala, Matti Pietikäinen and Topi Mäenpää: Multiresolution Gray Scale and Rotation Invariant Texture Classification with Local Binary Patterns, 2002.
3. https://www.pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/
4. http://www.scholarpedia.org/article/Local_Binary_Patterns

6. PROGRAMSKO RJEŠENJE

```
7. import cv2
8. import numpy as np
9.
10.#loading and converting image to grayscale
11.img = cv2.imread("/home/osirv/Osirv projekt/lenna.bmp",
   cv2.IMREAD_GRAYSCALE)
12.img_after = cv2.imread("/home/osirv/Osirv projekt/lenna.bmp",
   cv2.IMREAD_GRAYSCALE)
13.
14.#selecting pixels, for each pixel in the image, neighbor pixels
   surrounding the center pixel are selected
15.def get_pixel(img, x, y):
16.    try:
17.        return img[x,y]
18.    except:
19.        pass
20.
21.#comparing neighbor pixels values with center pixel value
22.#if center pixel has larger value than neighbor pixel, neighbor pixel
   value is set to 0
23.#if center pixel has smaller value than neighbor pixel, neighbor pixel
   value is set to 1;
24.def calc_pixel_value(center_pixel, neighbor_pixels):
25.    value=[]
26.    for pixel in neighbor_pixels:
27.        if pixel >= center_pixel:
28.            value.append(1)
29.        else:
30.            value.append(0)
31.    return value
32.
33.
34.for x in range(0, len(img)): #select all rows
35.    for y in range(0, len(img)): #select all columns
36.        center_pixel     = img[x,y] #center pixel
37.        above_pixel      =get_pixel(img, x, y-1) #pixel above center
   pixel
38.        down_pixel       =get_pixel(img, x, y+1) #pixel under center
   pixel
39.        right_pixel      =get_pixel(img, x+1, y) #right pixel
40.        left_pixel       =get_pixel(img, x-1, y) #left pixel
41.        aboveleft_pixel  =get_pixel(img, x-1, y-1) #above left pixel
42.        aboveright_pixel =get_pixel(img, x+1, y-1 ) #above right pixel
43.        downleft_pixel   =get_pixel(img, x-1, y+1) #down left pixel
44.        downright_pixel  =get_pixel(img, x+1, y+1) #down right pixel
45.
```

```
46.
47.         #selecting neighbour pixels clockwise, starting from the top
    left pixel and getting their value depending on the value of center
    pixel
48.         #store pixels in 8 bit array
49.         lbp_pixels = calc_pixel_value(center_pixel, [aboveleft_pixel,
    above_pixel, aboveright_pixel, right_pixel, downright_pixel, down_pixel,
    downleft_pixel, left_pixel])
50.
51.
52.         lbp_combinations = [1, 2, 4, 8, 16, 32, 64, 128]
53.         lbp_pixel_value = 0
54.         for pixel in range(0, len(lbp_pixels)):
55.             #calculating final pixel value
56.             lbp_pixel_value = lbp_pixel_value+ (lbp_combinations[pixel]*
    lbp_pixels [pixel])
57.
58.         img_after.itemset((x,y), lbp_pixel_value)
59.
60.
61.cv2.imwrite ( "/home/osirv/Osirv projekt/" +'lbp2.png', img_after)
62.cv2.imshow('before', img)
63.cv2.imshow('after', img_after)
64.
65.cv2.waitKey(0)
66.
67.cv2.destroyAllWindows()
68.
```