Local Derivative Pattern (LDP)

Antara Chansarkar 0801CS171013
Jahanvi Sisodiya 0801CS171029

Contents

Chapter 1

Introduction

1. Feature Extraction

Feature extraction is a process that identifies important features or attributes of the data. It increases the accuracy of learned models by extracting features from the input data. This phase of the general framework reduces the dimensionality of data by removing the redundant data.

2. Local Derivative Pattern

An LDP operator is proposed, in which the (n-1)th-order derivative direction variations based on a binary coding function. In this scheme, LBP is conceptually regarded as the nondirectional first-order local pattern operator, because LBP encodes all-direction first-order derivative binary result while LDP encodes the higher-order derivative information which contains more detailed discriminative features that the first-order local pattern (LBP) can not obtain from an image.

The proposed LDP operator labels the pixels of an image by comparing two derivative directions at two neighbouring pixels and concatenating the results as a 32-bit binary sequence. The derivative direction comparisons defined in are performed on 16 templates reflecting various distinctive spatial relationships in a local region. Different from LBP encoding the binary derivative gradient directions, the second-order LDP encodes the change of the neighbourhood derivative directions, which represents the second-order pattern information in the local region.

The high-order local patterns provide a stronger discriminative capability in describing detailed texture information than the first-order local pattern as used in LBP. However, they tend to be sensitive to noise when the order n becomes high. In the design of the proposed approach, the last-order operation only preserves the coarse gradient direction transition information instead of conventional difference information. This can alleviate the noise sensitivity problem in the high-order LDP representation

3. Advantages of LDP over LBP

The advantages of the high-order LDP over LBP can be briefly summarised in the following aspects.

1. LBP cannot provide a detailed description for faces by encoding the binary gradient directions. However, the nth -order LDP can provide more detailed description by coding the (n-1)th-order derivative direction variations.

2. LBP encodes the relationship between the central point and it's neighbours, but LDP encodes the various distinctive spatial relationships in a local region and, therefore, contains more spatial information.

.

Chapter 2

Mathematical Formulation

1. Formulation

Given an image *I(Z)*, the first-order derivatives along 0^0 , 45^0 , 90^0 and 135^0 directions are denoted as $I'_{\alpha}(Z)$ where $\alpha = 0^0$,45^0 ,90^0 and135^0 .Let Z_0 be a point in *I(Z)*, and Z_i, i = 1,…,8 be the neighbouring point around Z_0. The four first order derivatives at $Z = Z_0$ can be written as

$$I'_{0^\circ}(Z_0) = I(Z_0) - I(Z_4) \quad (2)$$

$$I'_{45^\circ}(Z_0) = I(Z_0) - I(Z_3) \quad (3)$$

$$I'_{90^\circ}(Z_0) = I(Z_0) - I(Z_2) \quad (4)$$

$$I'_{135^\circ}(Z_0) = I(Z_0) - I(Z_1). \quad (5)$$

The second order directional LDP $^2{}_{\alpha}(Z_0)$, in α direction at $Z = Z_0$ is defined as

$$LDP^2_{\alpha}(Z_0) = \{f\left(I'_{\alpha}(Z_0), I'_{\alpha}(Z_1)\right), f\left(I'_{\alpha}(Z_0), I'_{\alpha}(Z_2)\right) \cdots, f\left(I'_{\alpha}(Z_0), I'_{\alpha}(Z_8)\right)\} \quad (6)$$

where f(.,.)is a binary coding function determining the types of local pattern transitions. It encodes the co-occurrence of two derivative directions at different neighbouring pixels as

$$f\left(I'_{\alpha}(Z_0), I'_{\alpha}(Z_i)\right) = \begin{cases} 0, & if\ I'_{\alpha}(Z_i) \cdot I'_{\alpha}(Z_0) > 0 \\ 1, & if\ I'_{\alpha}(Z_i) \cdot I'_{\alpha}(Z_0) \leq 0 \end{cases} \quad i = 1, 2, \cdots, 8. \quad (7)$$

Finally, the second-order Local Derivative Pattern, LDP $^2(Z)$, is defined as the concatenation of the four 8-bit directional LDPs

$$LDP^2(Z) = \left\{LDP^2_{\alpha}(Z) | \alpha = 0^\circ, 45^\circ, 90^\circ, 135^\circ\right\}. \quad (8)$$

Chapter 3

Algorithm

In this section, we give an overview of the LDP algorithm:

1. LDP algorithm

Algorithm:

Input : image in .jpg format

Output : LDP matrix of the image

1. Convert image into grid of pixels, represented by a matrix.

2. Select a centre pixel and examine its neighbouring pixels.

3. Take centre pixel and a neighbouring pixel, and compare them with pixels in different directions with them, that is, 0^0, 45^0, 90^0 and 135^0.

4. Assign bits to each neighbouring pixel for each direction.

5. If both centre pixel and neighbouring pixel are monotonically increasing or decreasing for a direction, then assign 0 to that neighbouring pixel for that direction.

6. While, if one is increasing and one is decreasing, that is, there is a turning point, then assign 1 to that neighbouring pixel for that direction.

7. Take neighbouring pixels clockwise or anti-clockwise and form an 8-bit binary number for each angle.

8. Concatenate all four (0^0, 45^0, 90^0 and $135^{0)}$ 8-bit binary numbers and form a 32-bit binary number.

9. Convert it into a decimal number.

10. Replace the centre pixel value by the decimal number.

11. Repeat the above steps for every pixel.

Chapter 4

Documentation of API

1. Package organization

from local_derivative_pattern import ldp

2. Methods

def ldp(photo)

Returns LDP of an image

Parameters:

Photo**:** Input image

def assign_bit(picture, x, y, c_1, c_2, d)

To assign bits to the neighbouring pixels.

Parameters:

picture: input image
x: row co-ordinate of neighbouring pixel
y: column co-ordinate of neighbouring pixel
c_1: row coordinate of centre pixel
c_2: column coordinate of centre pixel
d: degree

def local_der_val(picture, x, y)

To calculate LDP value of a pixel.

Parameters:

picture: input image
x: row co-ordinate of centre pixel
y: column co-ordinate of centre pixel

Chapter 5

Example

1. Example 1

```
photo = cv2.imread("/content/barney-stinson-1.webp")
l=ldp(photo)
```

Output:

```
array([[ 64, 128,   0, ..., 128,   0,  56],
       [  0,   0,   0, ...,   0,   0,  56],
       [  0,   0,   0, ...,   0,   0,  56],
       ...,
       [187, 131,   0, ..., 131,   0,  56],
       [187, 131,   0, ..., 131,   0,  56],
       [177, 143,  14, ..., 143,  14,  62]], dtype=uint8)
```

2. Example 2

```
photo = cv2.imread("/content/Ashton_kutcher.jpg")
l=ldp(photo)
```

Output:

```
array([[ 26, 195, 128, ..., 206, 142,   6],
       [ 59, 131,   0, ..., 224, 224, 248],
       [179, 143,  14, ...,   0,   0,  56],
       ...,
       [128,  40,  76, ...,  24, 227, 192],
       [  0,  24, 227, ...,  56, 131,   0],
       [  0,  48, 143, ...,  62, 129,   0]], dtype=uint8)
```

Chapter 6

Learning Outcome

- Compared high order local pattern with first order local pattern and concluded that the former provide a stronger discriminative capability in describing detailed texture information than the later.
- Found that high order local pattern tend to be sensitive to noise when the order n becomes high.
- Capacity to integrate practical and theoretical knowledge to use local derivative pattern.
- Analysed and evaluated higher mathematical concepts with the ability to clearly implement and present the conclusions and the knowledge behind it.

Appendix A

References

- Baochang Zhang, Yongsheng Gao, *Senior Member, IEEE*, Sanqiang Zhao, and Jianzhuang Liu, *Senior Member,* IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 19, NO. 2, FEBRUARY 2010