School of Automation Science and Electrical Engineering
160324, 16711094, 李翰韬

Experiment of pattern recognition II

——Local Binary Patterns

I. LBP image Calculating

i). Original LBP Operations

The original LBP Operator is defined in a 3×3 window, with the center pixel of the window as the threshold value.

$$LBP(x_c, y_c) = \sum_{p=1}^{8} s(I(p) - I(c)) \times 2^p, s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & otherwise \end{cases}$$

In the formula, *p* represents the p^{th} pixel in the 3 × 3 window except for the center pixel; *I(c)* represents the gray value of the center pixel, and *I(p)* represents the gray value of the p^{th} pixel in the field.

The Python code is as follows:

```
def LBP(src):
    height = src.shape[0]
    width = src.shape[1]
    dst = src.copy()
    lbp_value = np.zeros((1,8), dtype=np.uint8)
    neighbours = np.zeros((1,8), dtype=np.uint8)
    for x in range(1, width-1):
        for y in range(1, height-1):
            neighbours[0, 0] = src[y + 1, x - 1]
            neighbours[0, 1] = src[y + 1, x]
            neighbours[0, 2] = src[y + 1, x + 1]
            neighbours[0, 3] = src[y, x + 1]
            neighbours[0, 4] = src[y - 1, x + 1]
            neighbours[0, 5] = src[y - 1, x]
            neighbours[0, 6] = src[y - 1, x - 1]
            neighbours[0, 7] = src[y, x - 1]
            center = src[y, x]
            for i in range(8):
                if neighbours[0, i] < center:
                    lbp_value[0, i] = 0
                else:
                    lbp_value[0, i] = 1
            lbp = lbp_value[0, 0] * 1 + lbp_value[0, 1] *
2 + lbp_value[0, 2] * 4 + lbp_value[0, 3] * 8 \
                + lbp_value[0, 4] * 16 + lbp_value[0, 5]
* 32 + lbp_value[0, 6] * 64 + lbp_value[0, 7] * 128
            dst[y, x] = lbp
    return dst
```

All image processing results will displayed in the final group.

ii). Circular LBP Operations

The original LBP Operator only covers a small area of 3×3, which obviously can not meet the requirements of extracting texture features of different sizes.So the 3 × 3 neighborhood is extended to any neighborhood, while the square neighborhood is replaced by the circle neighborhood.

$$LBP_{P,R}(x_c, y_c) = \sum_{p=1}^{P} s(I(p) - I(c)) \times 2^p, \begin{cases} x_p = x_c + R \times cos(\frac{2\pi p}{P}) \\ y_p = y_c + R \times sin(\frac{2\pi p}{P}) \end{cases}$$

In the formula, *P* is the number of sampling points; *R* is the radius of sampling circle area; *s(x)* is the same as in the original LBP.

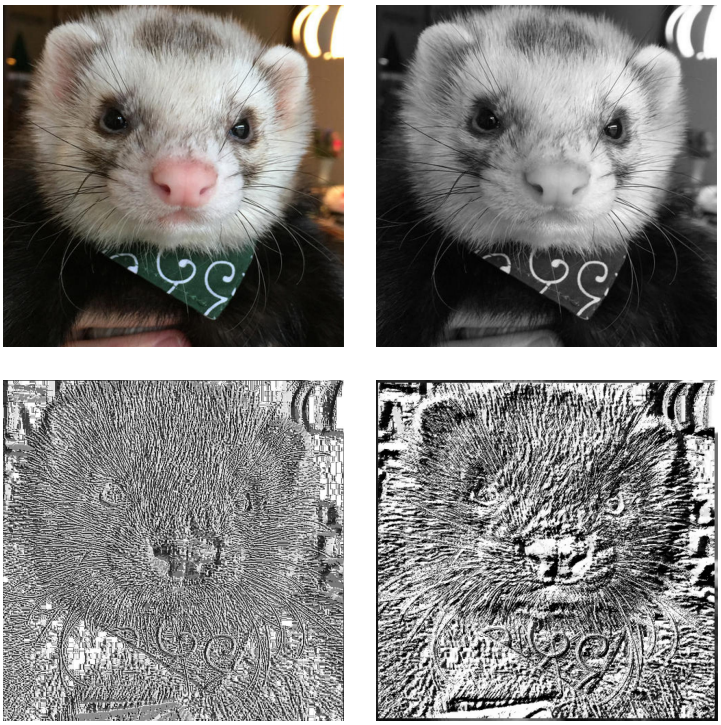
The Python code is as follows:

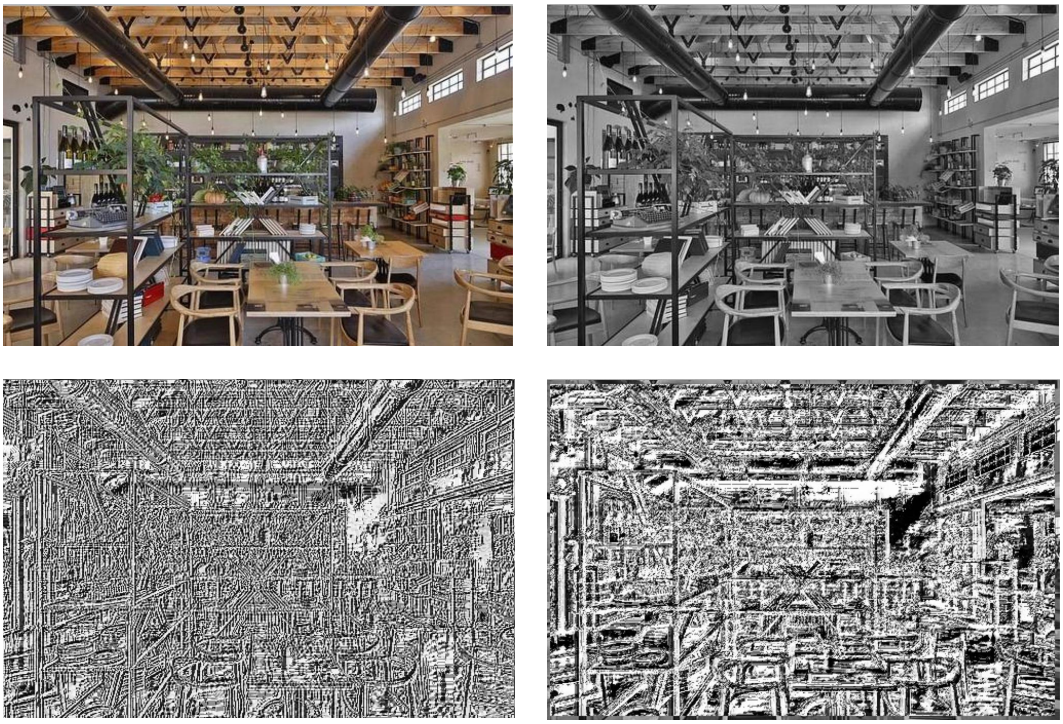
```
def circular_LBP(src, radius, n_points):
    height = src.shape[0]
    width = src.shape[1]
    dst = src.copy()
    src.astype(dtype=np.float32)
    dst.astype(dtype=np.float32)
    neighbours = np.zeros((1, n_points), dtype=np.uint8)
    lbp_value = np.zeros((1, n_points), dtype=np.uint8)
    for x in range(radius, width - radius - 1):
        for y in range(radius, height - radius - 1):
            lbp = 0.
            for n in range(n_points):
                theta = float(2 * np.pi * n) / n_points
                x_n = x + radius * np.cos(theta)
                y_n = y - radius * np.sin(theta)
                x1 = int(math.floor(x_n))
                y1 = int(math.floor(y_n))
                x2 = int(math.ceil(x_n))
                y2 = int(math.ceil(y_n))
                tx = np.abs(x - x1)
                ty = np.abs(y - y1)
                w1 = (1 - tx) * (1 - ty)
                w2 = tx * (1 - ty)
                w3 = (1 - tx) * ty
                w4 = tx * ty
                neighbour = src[y1, x1] * w1 + src[y2,
x1] * w2 + src[y1, x2] * w3 + src[y2, x2] * w4
                neighbours[0, n] = neighbour
            center = src[y, x]
            for n in range(n_points):
                if neighbours[0, n] < center:
                    lbp_value[0, n] = 0
                else:
                    lbp_value[0, n] = 1
            for n in range(n_points):
                lbp += lbp_value[0, n] * 2**n
                # print('lbp_value[0, n] * 2**n :
{}'.format(lbp_value[0, n] * 2**n))
            dst[y, x] = int(lbp / (2**n_points-1) * 255)
    return dst
```

iii). Results

In this experiment, two pictures are tested, one is a picture of the cafe and the other is a picture of a ferret.

The original image, gray-scale image, original LBP calculation results and the circular LBP calculation results of the two pictures are given below.





The above Circular LBP parameters are $R=4$, $P=16$.

It is worth noting that when $R=1$, $P=8$, the Circular LBP is basically the same as the original LBP with the theoretical analysis.

II. LBP histogram Calculating

By calling the *matplotlib.pyplot* library to draw a simple histogram, the *hist* function can draw histogram directly.

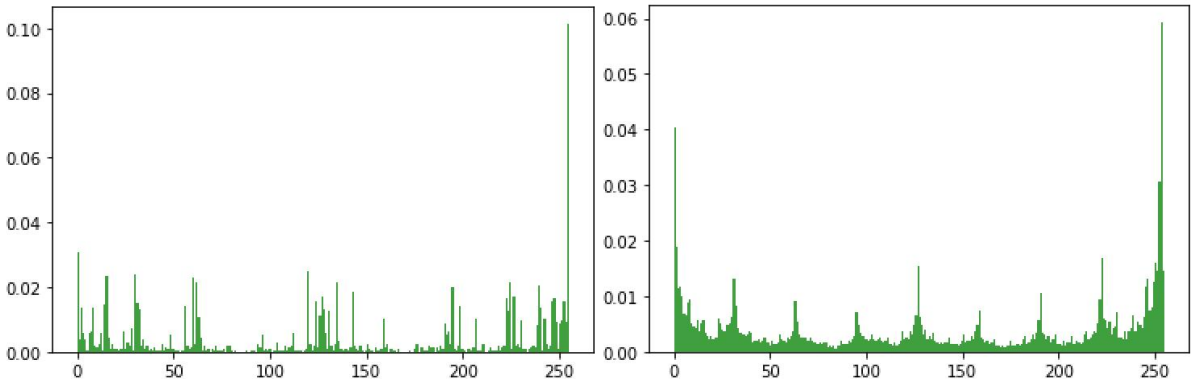
Here is the main function part of the whole function:

The Python code is as follows:

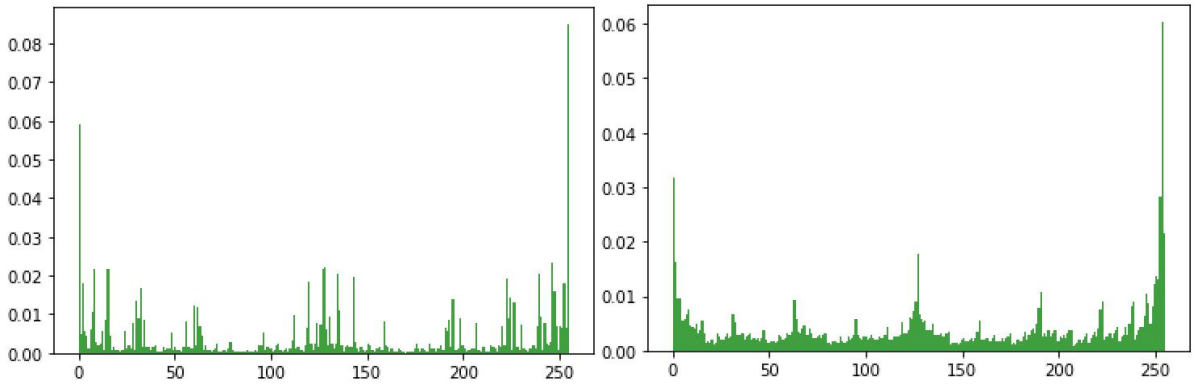
```
if __name__ == '__main__':
    img = cv2.imread('/Users/hantao.li/Documents/lbp/pic1.jpg')
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    dst1 = LBP(gray)
    dst2 = circular_LBP(gray, radius=4, n_points=16)
    # disp_test_result(img, gray, dst1, mode=0)
    # disp_test_result(img, gray, dst2, mode=0)
    arr1=dst1.flatten()
    n, bins, patches = plt.hist(arr1, bins=256, normed=1, facecolor='green', alpha=0.75)
    plt.show()
    arr2=dst2.flatten()
    n, bins, patches = plt.hist(arr2, bins=256, normed=1, facecolor='green', alpha=0.75)
    plt.show()
```

The LBP histogram and Circular LBP histogram of the results of the above two images will shown below respectively:

i). Ferret



ii). Cafe



It can be seen from the experimental results that no matter which picture or LBP operation, 255 patterns are always has the highest occurrence rate.

According to the theoretical analysis, when the gray value of the center pixel is the same as the surroundings', the output LBP pattern will be 255, which is obviously the most common circumstance in general pictures.

III. Options—Different parameters

The following is the result of Circular LBP calculation with different parameters. From left to right are *R=1, P=8;* *R=2, P=8;* *R=2, P=16.*