Report of Computer Vision with OpenCV

(Project 1, part 1)[1]

Author: ZHUO Xu (卓旭); Email: zhuoxu@seu.edu.cn

1. Get Image

A photo of the gravel road in our campus is taken.

Figure 1 – The whole photo of the gravel road.

The original size of the photo is very big (> 10 million pixels). To reduce the computation cost, a patch of 512×512 is cropped to be used in the following text.

2. Read the Image

Using OpenCV's Python port with Matplotlib, the image is read and displayed.

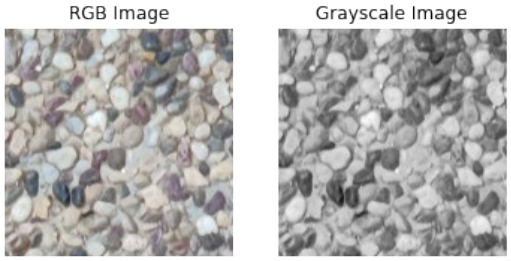

Figure 2 – The image and corresponding grayscale one.

3. Compute One Pixel LBP

The function to calculate LBP for one pixel is implemented in the code *def lbp_pixel(img, r, c)*. Note that I personally prefer using row (r) and column (c) to locate a pixel rather than (*x*, *y*). Using the example in the slide as a test can verify the correctness of the implementation.

```
In [14]: testImg = np.array([
             [100, 101, 103],
             [40, 50, 80],
             [50, 60, 90]
         ], dtype=np.uint8)
         testLbp = lbp_pixel(testImg, 1, 1)
         print(testLbp, bin(testLbp))

         252 0b11111100
```

Figure 3 – The test of the LBP implementation.

[1] Corresponding code: *lbp_part1.ipynb*; Opensource at https://github.com/z0gSh1u/rennes1-seu-cv .

Iterate all pixels in the image, calculate their LBPs, we can obtain another "LBP image" with same shape as the origin.

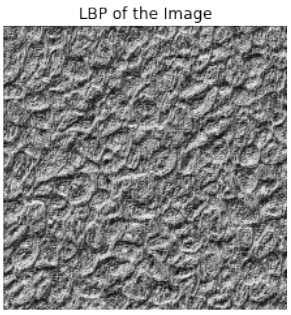

Figure 4 – LBP of the image.

4. Compute LBP Histogram

Using OpenCV's *cv2.calcHist* to get the histogram values, and use Matplotlib's *plt.bar* to plot, we get the LBP Histogram of the image.

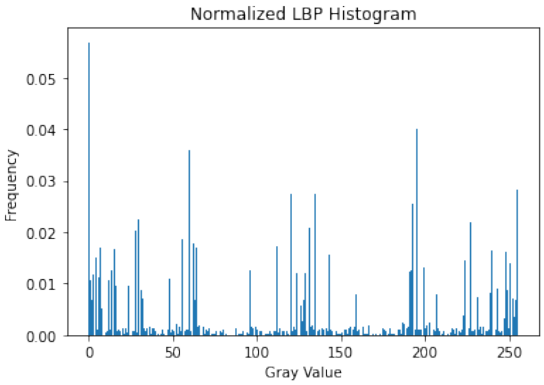

Figure 5 – The LBP Histogram of the image.

5. Evaluate LBP Robustness

We first evaluate the robustness to the rotation. The rotation axis is the "approximate" center of the image ($\lfloor height/2 \rfloor, \lfloor width/2 \rfloor$, image is 512×512 so the real center locates at a sub-pixel). Here are some examples of rotated images.

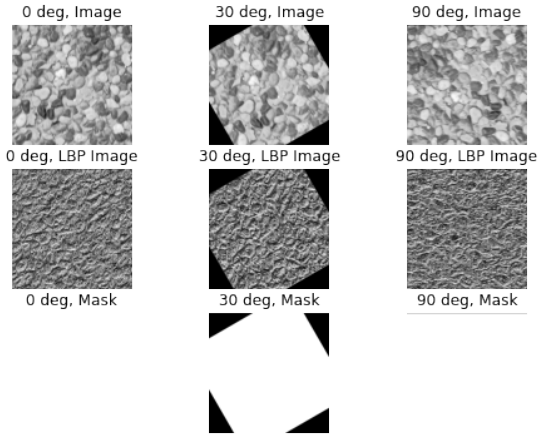

Figure 6 – Transformed images and corresponding LBP image with different rotation angles.

Note that for rotation angles except 0, 90, 180, 270 (degree), there will be blank regions, which shrink the effective region of the image and causes unfair comparison of the LBP

histogram if we use the whole image. To handle this, blank regions caused by the rotation will be ignored in the rotated image and the corresponding region in the original one synchronously. To implement this, we can use a all 1 image as mask and rotate it parallelly with the image. Only those pixels where mask is 1 (white) is effective. This can be implemented by specifying the parameter *mask* of *cv2.calcHist.*

After computation, we get the normalized LBP histogram distance vs. rotation angle as follows in $[0, 360)$ per 1 degree as a plot.

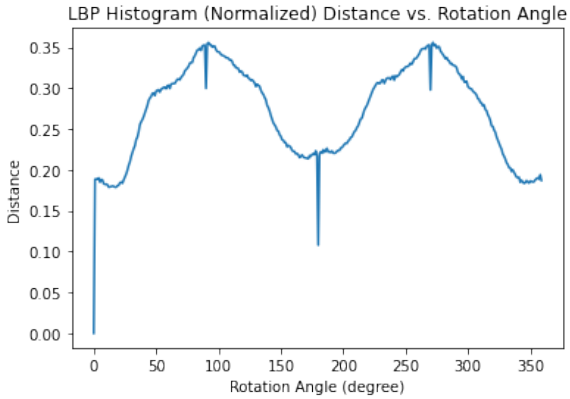

Figure 7 – The distance of normalized LBP histogram with different rotation angles.

We can find that the **LBP feature is not rotation-invariant** (at least when we use LBP Histogram as feature descriptor). An obvious way to think is that when we rotate a square image by multiple of 90 degrees, the binary code of LBP will get circularly shifted, resulting in different decimals, unless it's 00000000, 11111111, 01010101, 10101010 or something like these. Also, there is a dive when rotation angle is 90, 180 or 270 degrees.

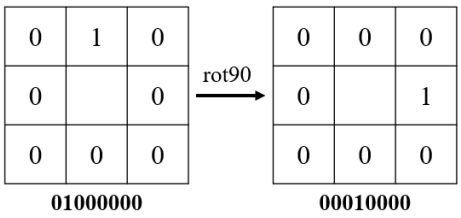

Figure 8 – The example of LBP's rotation variation.

Now, we'll evaluate the robustness to scaling. We use scaling factors 1, 1.25, 1.5 to scale the image and compute corresponding histogram distance then. Scaling means zoom in/out the image and center crop the result to the same shape as the original image. The result is shown below.

Figure 9 – Scaled images.

```
Scale: 1x, Dist: 0.000
Scale: 1.25x, Dist: 0.303
Scale: 1.5x, Dist: 0.413
```

Figure 10 - The distance of normalized LBP histogram with different scales.

We can find that the **LBP feature is not scaling-invariant** either (at least when we use LBP Histogram as feature descriptor). This is obvious because when we scale an image, the 8 neighbors of a pixel changes, resulting in different LBP. [1]

We may be curious about what's invariant in LBP feature? In fact, **LBP feature is insensitive to brightness (light) changes.** Because if we add a same number onto the image, the relative less-than or greater-than relationship between the center pixel and its 8 neighbors doesn't change, making the LBP unchanged.

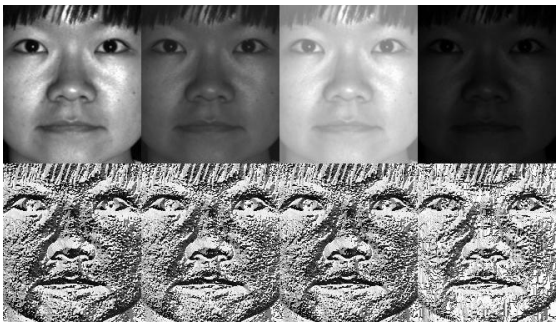

Figure 11 – The example showing LBP feature's (line 2) insensitiveness to brightness (light) changes (line 1). [2]

References

[1] Zhi Li, Guizhong Liu, Yang Yang and Junyong You, "Scale- and Rotation-Invariant Local Binary Pattern Using Scale-Adaptive Texton and Subuniform-Based Circular Shift", *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2130-2140, 2012. Available: 10.1109/tip.2011.2173697 [Accessed 29 June 2022].

[2] "OpenCV: Face Recognition with OpenCV", *Docs.opencv.org*, 2022. [Online]. Available: https://docs.opencv.org/ref/master/da/d60/tutorial_face_main.html#tutorial_face_lbph_algo. [Accessed: 29- Jun- 2022].