# Image region description with LBP

**Objectives**:

- To learn how to apply the LBP operator.
- To learn how an LBP image looks like.
- To learn how to compute an LBP descriptor at image region level.

For validating your code, use the (texture) images provided in Moodle.

## 1. Minimum (Up to 7/10 points)

Write and generate a main program named "`test_lbp.exe`" to test the following functions.

The following functions will be implemented in the files "*lbp.cpp*" and "*lbp.hpp*". Note that template code is provided in Moodle.

A. (2.5 pts) Given an input image (Mat, cv::8UC1) compute the corresponding LBP for each pixel. How will you handle the image limits? Use the following function header:

```
void fsiv_lbp(const cv::Mat & img, cv::Mat & lbp, const
bool uLBP=false);
```

Help: see usage of bitwise operators in C++ (*shift*)

B. (0.5 pt) Display an LBP matrix as an image. Use the following function header:

```
void fsiv_lbp_disp(const cv::Mat & lbpmat, const
std::string & winname);
```

C. (2 pts) Compute the corresponding LBP histogram (region descriptor) from an LBP matrix. To normalize the histogram will be the default option. HINT: check the use of cv::calcHist(). Use the following function header:

```
void fsiv_lbp_hist(const cv::Mat & lbp, cv::Mat &
lbp_hist, const bool normalize=true, const bool
uLBP=false);
//! \param lbphist [out]: row vector with 256/59
dimensions
//! \param normalize: return a normalized histogram.
Default, true.
```

D. (2 pts) Write a function (in file "*metrics.cpp*") to compute the Chi-squared distance between two histograms *x* and *y*:

$$d(x,y) = 0.5\sum_i \frac{(x_i - y_i)^2}{(x_i + y_i)}$$

Warning: avoid dividing by zero; histograms will be matrices with rows=1 and cols=*N*; and do not assume *N*=256 always.

Use the following function header:

```
float fsiv_chisquared_dist(const cv::Mat & h1, const
cv::Mat & h2);
```

```
//! \param h1, h2 Row vectors with the same dimensions.
```

(* Late delivery of this assignment will imply a penalty in the score of it. The score of each part has to be understood as “up to”).

## 2. Optional (Up to 3/10 points)

Add to the test program “`test_lbp.exe`” sample calls for testing the following features:

A. (+1.5 pts) Given a grid configuration, split the input image into *M*x*N* regions, then, compute an LBP histogram per region and, finally, concatenate the resulting histograms into a single one. HINT: check the use of function cv::hconcat(). Use the following header function:

```
void fsiv_lbp_desc(const cv::Mat & lbpmat, cv::Mat &
lbp_desc, const int *ncells, bool normalize=true, bool
uLBP=false);
//! \param lbp_desc [out] Row vector containing the
compound LBP descriptor.
//! \param ncells [in] [rows x cols] E.g. {2,2}
```

B. (+1.5 pts) Either write a function to compute the U-LBP (uniform LBP) matrix or extend the previous function `fsiv_lbp`. In the main program, display as an image both the standard LBP matrix and the U-LBP. Add a new flag “-uLBP” to the test program to select this option. You may use the following header function:

```
void fsiv_ulbp(const cv::Mat & imagem, cv::Mat &
ulbpmat);
```

HINT: in functions `fsiv_lbp_desc` and `fsiv_lbp_hist`, pay attention to parameter `uLBP` used to indicate the type of LBP descriptor that will be computed.

The recommended code assignment for U-LBP is the following:

```
static int uniform[256] =
{
0,1,2,3,4,58,5,6,7,58,58,58,8,58,9,10,11,58,58,58,58,58,58,58,12,58,58,58,13,58,
14,15,16,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,17,58,58,58,58,58,58,58,18,
58,58,58,19,58,20,21,22,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,
58,58,58,58,58,58,58,58,58,58,58,58,23,58,58,58,58,58,58,58,58,58,58,58,58,58,
58,58,24,58,58,58,58,58,58,58,25,58,58,58,26,58,27,28,29,30,58,31,58,58,58,32,58,
58,58,58,58,58,58,33,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,34,58,58,58,58,
58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,
58,35,36,37,58,38,58,58,58,39,58,58,58,58,58,58,58,40,58,58,58,58,58,58,58,58,58,
58,58,58,58,58,58,41,42,43,58,44,58,58,58,45,58,58,58,58,58,58,58,46,47,48,58,49,
58,58,58,50,51,52,58,53,54,55,56,57
};
```