Opencv seminar

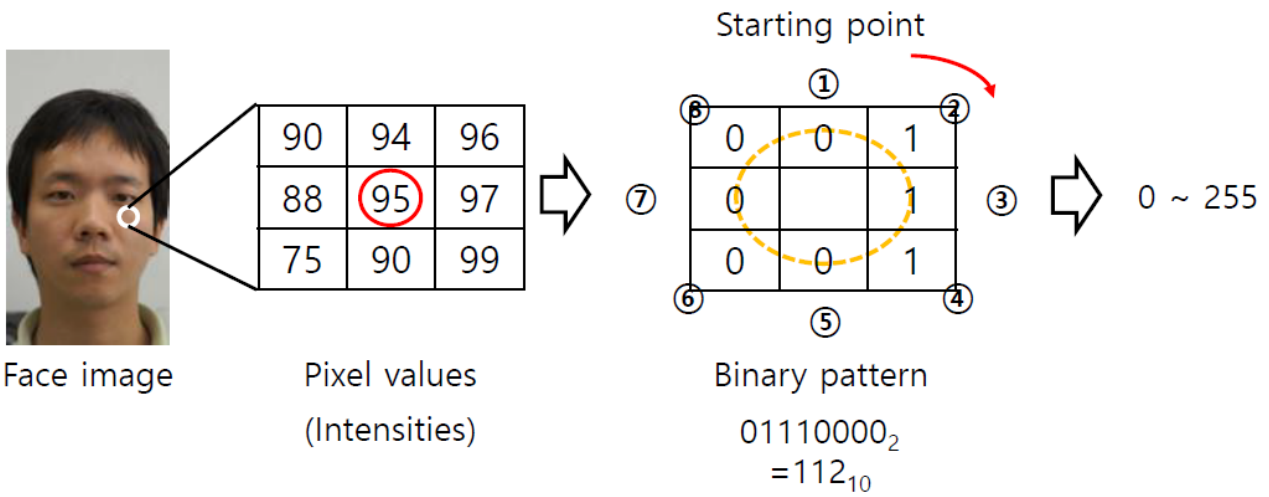
4. Face Identification

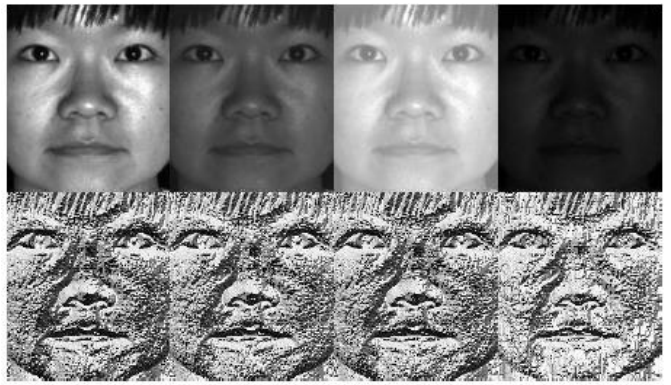
2018. 02 .05

LBP

- **Local binary pattern**
- **LBP works robust to the illumination change!**

$$f_{\mathrm{LBP}}(x,y)=\sum_{1\le i\le n}2^{i-1}\,\mathrm{LBP}^{\,i}(x,y) \quad \text{where} \quad \mathrm{LBP}^{\,i}(x,y)=\begin{cases}1, & \text{if } I(x,y)>I(x_i,y_i)\\ 0, & \text{otherwise}\end{cases}$$



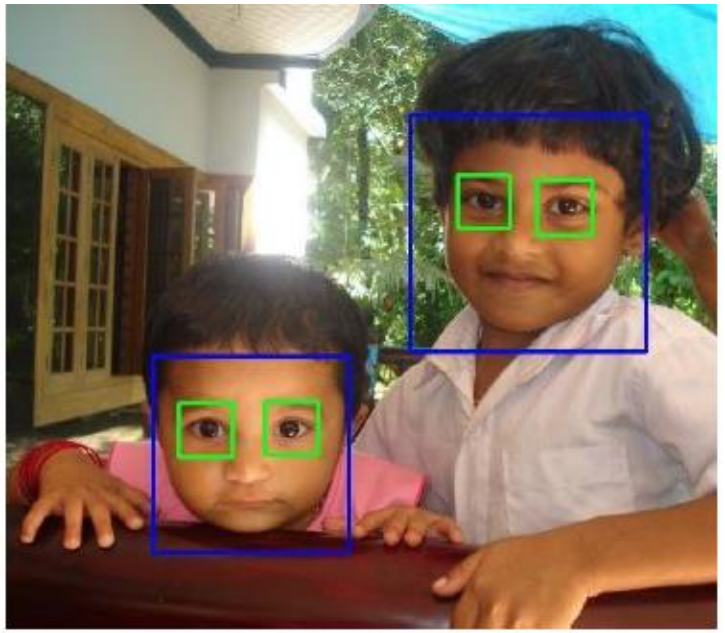


<result>

Class CascadeClassifier (1/2)

- **Apply the function of eye detection**
  : Opencv also provides the function for eye detection

```
#include<stdio.h>
#include<stdlib.h>
#include<iostream>
#include<opencv/cv.h>
#include<opencv/highgui.h>
#include<opencv2/opencv.hpp>
#include<opencv2/objdetect/objdetect.hpp>
#include<opencv2/ml/ml.hpp>
using namespace cv;
using namespace cv::ml;
using namespace std;
```



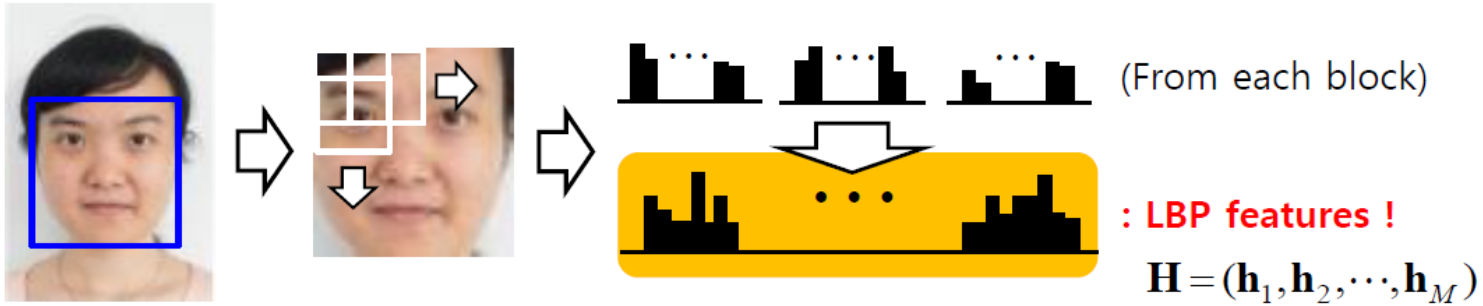
Class CascadeClassifier (2/2)

```
void main(){
        CascadeClassifiercascade;
        cascade.load(“~/OpenCV3.1.0/sources/data/lbpcascades/lbpcascade_frontalface.xml");
        Matimg=imread(“face1.bmp”,1);//imageread(color)
        vector<Rect>faces;
        cascade.detectMultiScale(img,faces,1.1,4,0|CV_HAAR_SCALE_IMAGE,Size(10,10));
        for(y=0;y<faces.size();y++){
                Pointlb(faces[y].x+faces[y].width,faces[y].y+faces[y].height);
                Pointtr(faces[y].x,faces[y].y);
                rectangle(img,lb,tr,Scalar(0,255,0),3,8,0);
                                                        }
        imshow(“Face”,img);
        waitKey(50000);
        }
```

Implementation

- Feature extraction
  - To encode such patterns, histogram is employed
    - Since the range of LBP image is also 0~255,
      conventional histogram-based representation can be adopted !

$$\mathbf{h} = (h_1, h_2, \cdots, h_{256}) \quad \text{where} \quad h_k = \frac{N_k}{\sqrt{\sum_{q=1}^{256} N_q^2 + \delta}}, \quad N_k = \sum_{f_{\text{LBP}}(x,y) \in k} 1$$



Implementation

- Your task
  1. Use class CascadeClassifier to find the face in the images
  2. Convert gray image to LBP images
  3. Construct the 256-bin histogram
  4. Compute Euclidean distance between two LBP features (similarity computation)

$$\text{Similarity}: S(\mathbf{H}^{en}, \mathbf{H}^{ver}) = \frac{1}{256}\sum_{i=1}^{256} \| \mathbf{h}_i^{en} - \mathbf{h}_i^{ver} \|$$

(en : enroll image, ver : verification image)

To determine whether given two images are same or not, we need to set the threshold value !