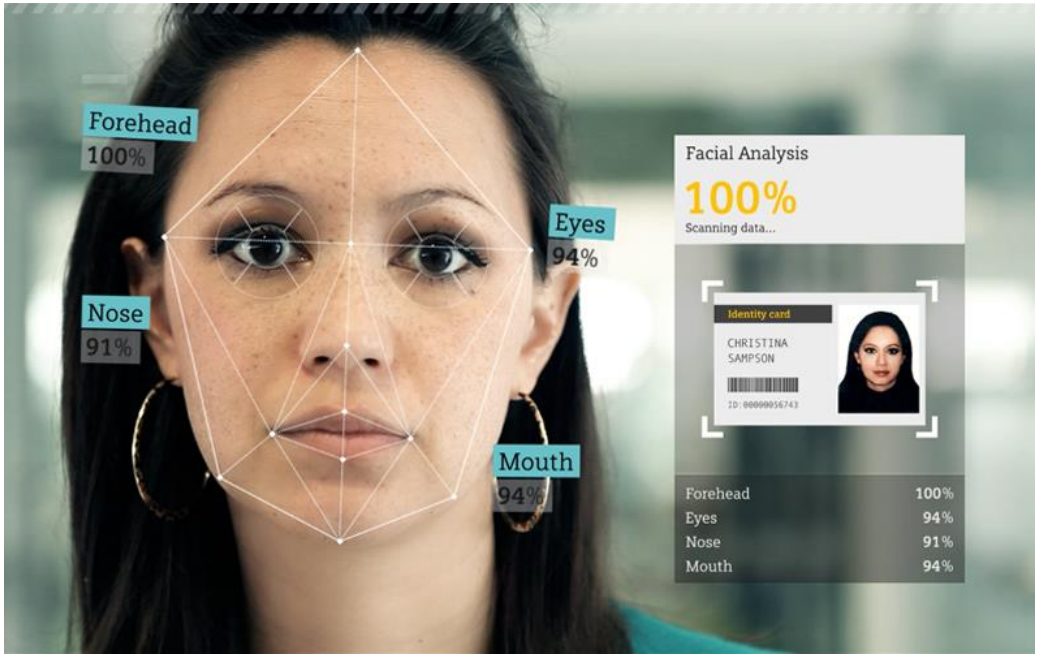
Face Recognition vs. Face Detection

What is difference between recognition and detection?

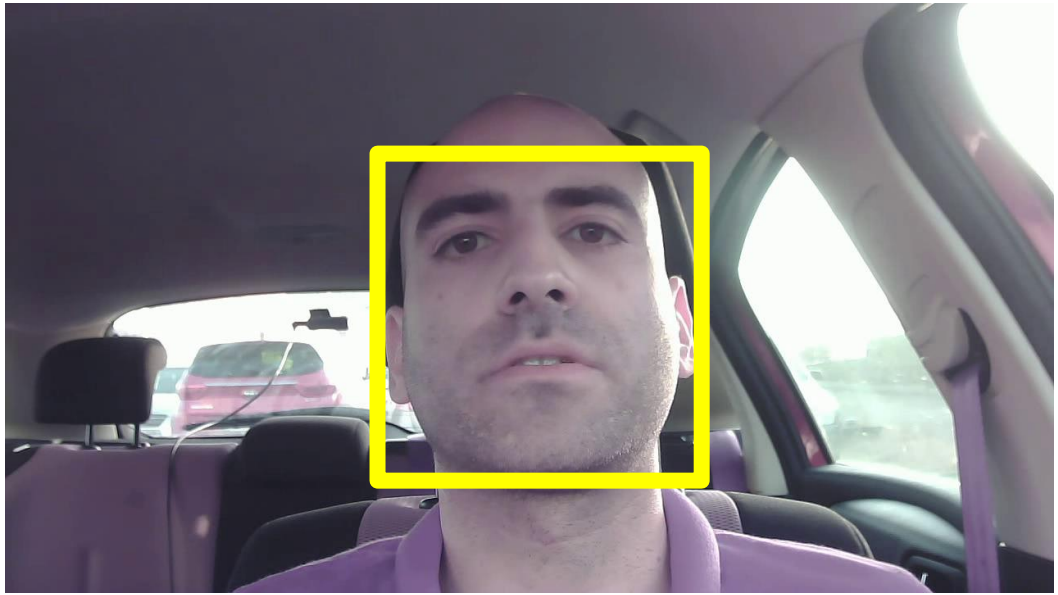


https://recfaces.com/articles/how-accurate-is-facial-recognition

Face Recognition

Face Recognition can be considered as multi step process:

1. Face Detection



Face Recognition

Face Recognition can be considered as multi step process:

2. Face Alignment (for example, with the use of face detector or facial landmarks)

Face Recognition

Face Recognition can be considered as multi step process:

3. Feature Extraction

feature vector

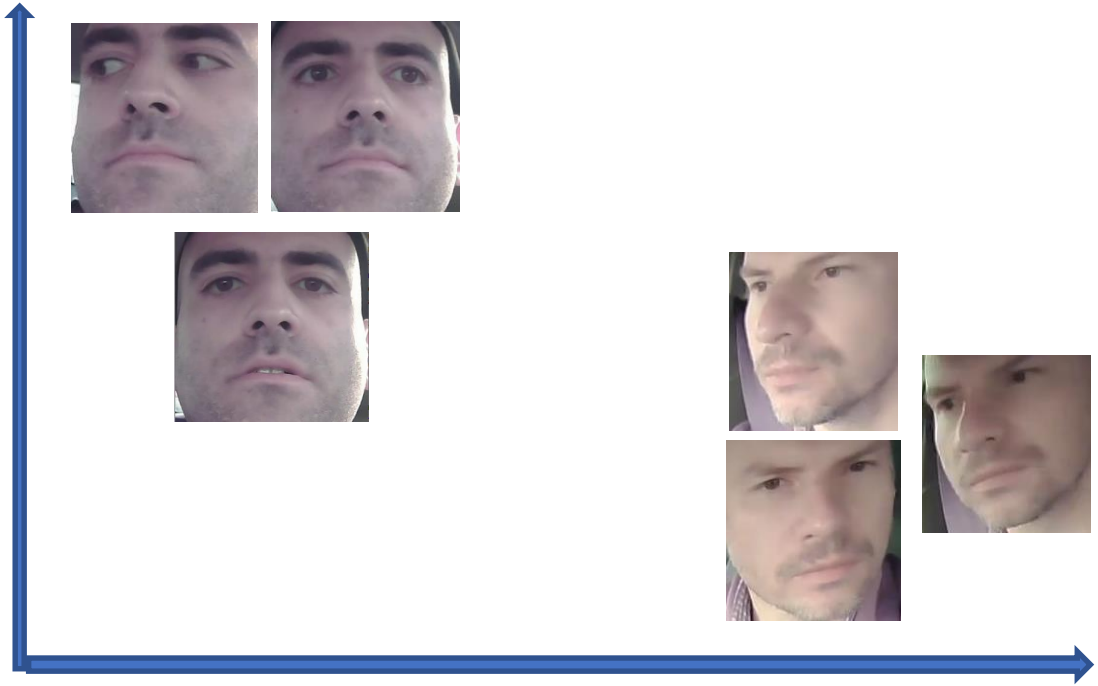
$$\begin{bmatrix} 0{,}912 \\ 0.128 \\ 0.384 \\ 0.865 \\ 0.951 \\ 0.156 \\ 0.684 \\ . \\ . \\ . \\ 0.665 \end{bmatrix}$$

Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Face Recognition

Face Recognition can be considered as multi step process:

4. Recognition phase



Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Face Recognition

Face Recognition can be considered as multi step process:

4. Recognition phase

Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Face Recognition

Face Recognition can be considered as multi step process:

4. Recognition phase

new image/observation

???

Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Face Recognition

Face Recognition can be considered as multi step process:

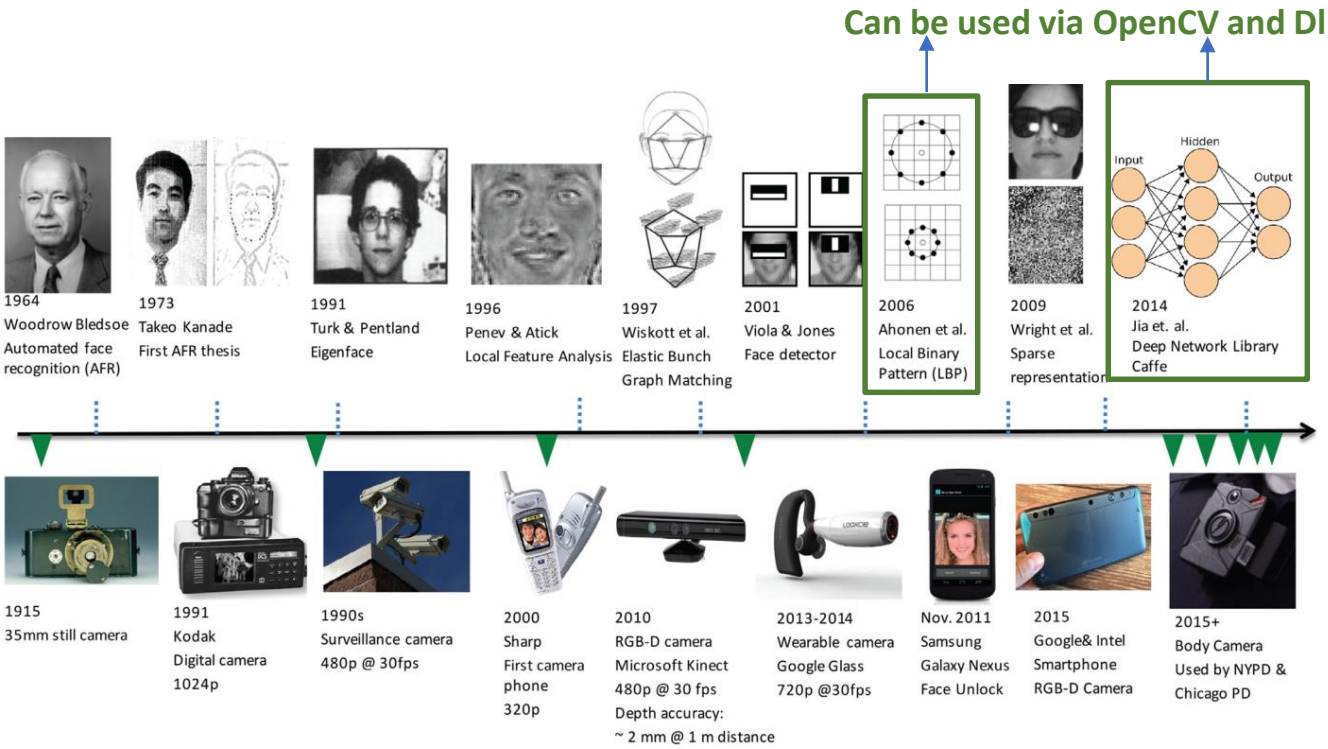
4. Recognition phase

Based on comparison of new feature vectors with existing ones

Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Face Recognition

Face Recognition can be considered as multi step process:



Jain, Anil & Nandakumar, Karthik & Ross, Arun. (2016). 50 Years of Biometric Research: Accomplishments, Challenges, and Opportunities. Pattern Recognition Letters. 79. 10.1016/j.patrec.2015.12.013.

Simple Face Recognition using OpenCV and LBP features

- **LBP?**

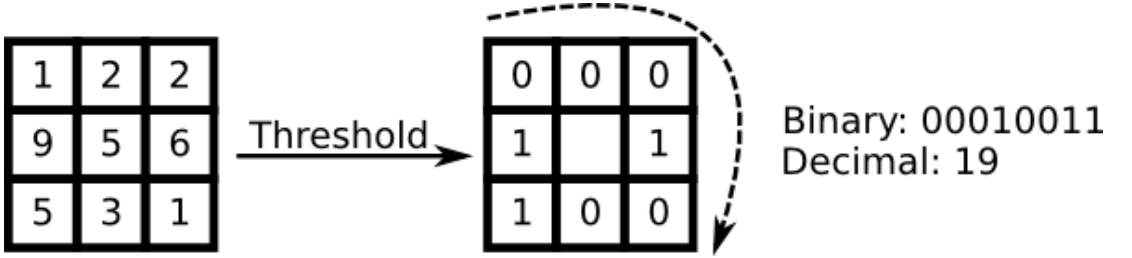
Simple Face Recognition using OpenCV and LBP features

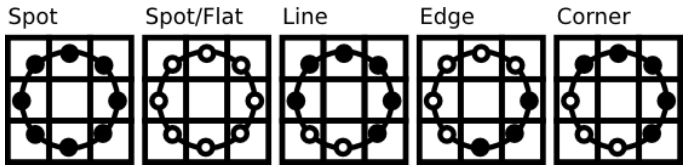
- **LBP?**
- The local binary patterns (LBP) were introduced by Ojala et al. [2, 3] for the texture analysis. The main idea behind LBP is that the local image structures (micro patterns such as lines, edges, spots, and flat areas) can be efficiently encoded by comparing every pixel with its neighboring pixels. In the basic form, every pixel is compared with its neighbors in the 3 × 3 region.

https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

Simple Face Recognition using OpenCV and LBP features

- **LBP?**
- The local binary patterns (LBP) were introduced by Ojala et al. [2, 3] for the texture analysis. The main idea behind LBP is that the local image structures (micro patterns such as lines, edges, spots, and flat areas) can be efficiently encoded by comparing every pixel with its neighboring pixels. In the basic form, every pixel is compared with its neighbors in the 3 × 3 region. The result of comparison is the 8-bit binary number for each pixel; in the 8-bit binary number - the value 1 means that the value of neighbor pixel is greater than the center pixel





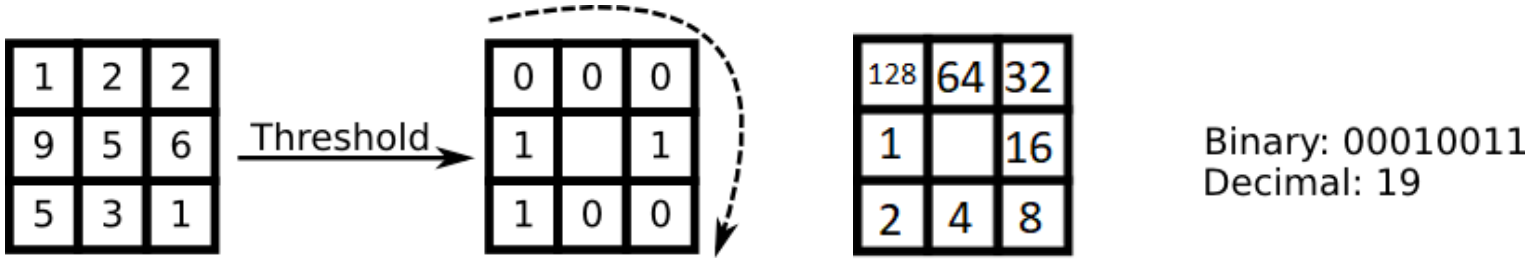
https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

Simple Face Recognition using OpenCV and LBP features

- In the case of 8 neighbors, we can use the following formula:

$$LBP = \sum_{n=0}^{7} s(i_n - i_c)2^n \qquad s(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & otherwise \end{cases}$$

i_n (neighbor pixel value); i_c (center pixel value)



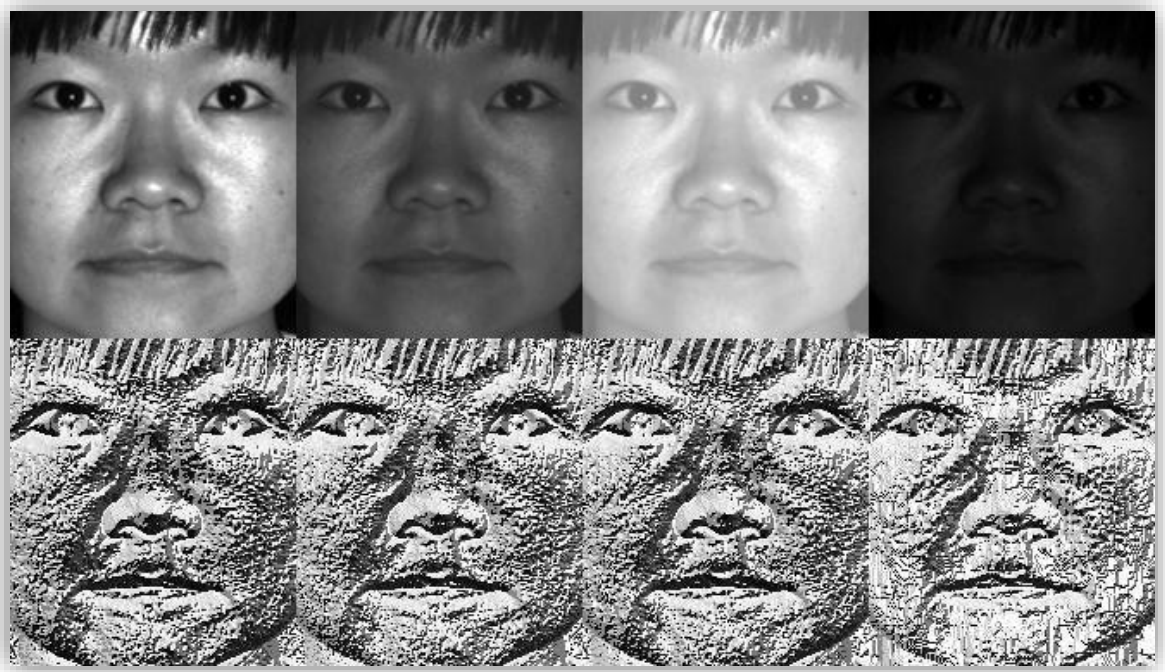
1*1 + 1*2 + 0*4 + 0*8 + 1*16 + 0*32 + 0*64 + 0*128 = **19**

- You can use different direction, but it must be same for all images

https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

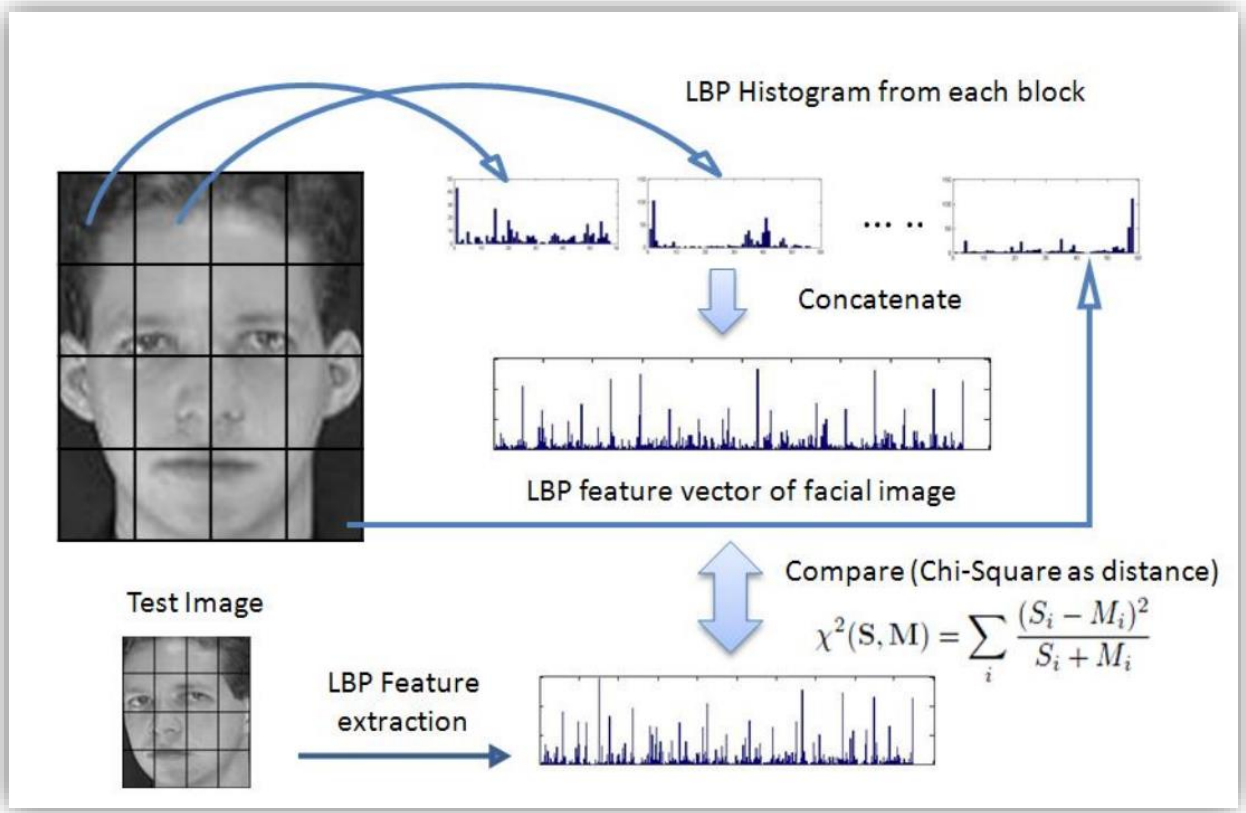
Simple Face Recognition using OpenCV and LBP features

- **LBP?**



https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html#local-binary-patterns-histograms

Face Recognition



Zhao, Xuran & Evans, Nicholas & Dugelay, Jean-Luc. (2011). A co-training approach to automatic face recognition.

Face Recognition

create()

```
static Ptr<LBPHFaceRecognizer> cv::face::LBPHFaceRecognizer::create ( int    radius = 1 ,
                                                                      int    neighbors = 8 ,
                                                                      int    grid_x = 8 ,
                                                                      int    grid_y = 8 ,
                                                                      double threshold = DBL_MAX
                                                                    )
```

static

Python:

cv.face.LBPHFaceRecognizer_create([, radius[, neighbors[, grid_x[, grid_y[, threshold]]]]]) -> retval

Parameters

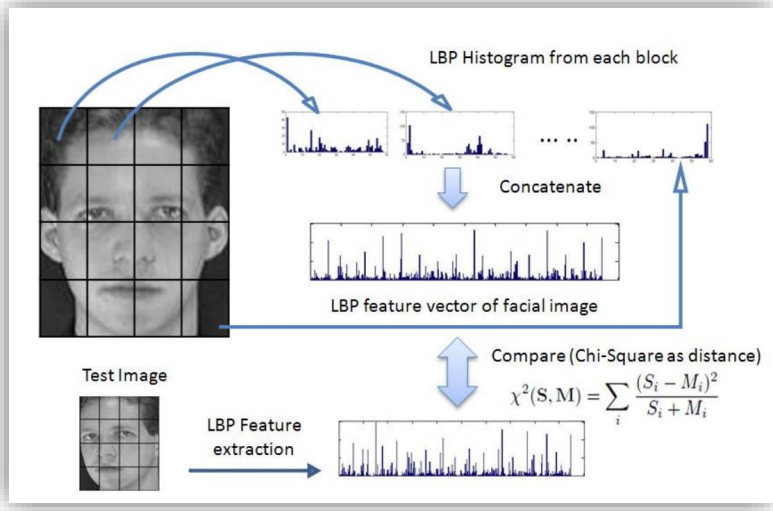
- **radius** The radius used for building the Circular Local Binary Pattern. The greater the radius, the smoother the image but more spatial information you can get.
- **neighbors** The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use `8` sample points. Keep in mind: the more sample points you include, the higher the computational cost.
- **grid_x** The number of cells in the horizontal direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **grid_y** The number of cells in the vertical direction, 8 is a common value used in publications. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector.
- **threshold** The threshold applied in the prediction. If the distance to the nearest neighbor is larger than the threshold, this method returns -1.

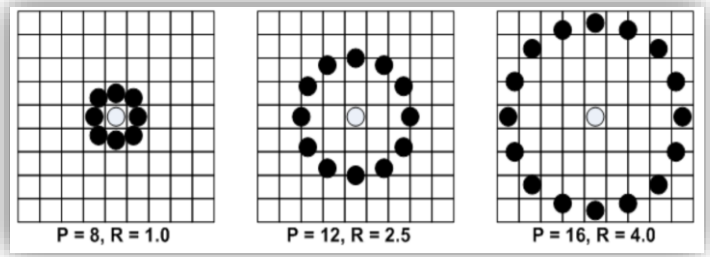
Notes:

- The Circular Local Binary Patterns (used in training and prediction) expect the data given as grayscale images, use cvtColor to convert between the color spaces.
- This model supports updating.

Model internal data:

- radius see **LBPHFaceRecognizer::create**.
- neighbors see **LBPHFaceRecognizer::create**.
- grid_x see LLBPHFaceRecognizer::create.
- grid_y see **LBPHFaceRecognizer::create**.
- threshold see **LBPHFaceRecognizer::create**.
- histograms Local Binary Patterns Histograms calculated from the given training data (empty if none was given).
- labels Labels corresponding to the calculated Local Binary Patterns Histograms.





https://docs.opencv.org/4.x/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html

Face Recognition

predict() [2/3]

```
void cv::face::FaceRecognizer::predict ( InputArray src,
                                         int &      label,
                                         double &   confidence
                                       )            const
```

Python:

```
cv.face.FaceRecognizer.predict(          src            ) -> label, confidence
cv.face.FaceRecognizer.predict_collect( src, collector ) -> None
cv.face.FaceRecognizer.predict_label(    src            ) -> retval
```

Predicts a label and associated confidence (e.g. distance) for a given input image.

Parameters

src	Sample image to get a prediction from.
label	The predicted label for the given image.
confidence	Associated confidence (e.g. distance) for the predicted label.

train()

```
virtual void cv::face::FaceRecognizer::train ( InputArrayOfArrays src,
                                               InputArray         labels
                                             )
```

Python:

```
cv.face.FaceRecognizer.train( src, labels ) -> None
```

Trains a **FaceRecognizer** with given data and associated labels.

Parameters

src	The training images, that means the faces you want to learn. The data has to be given as a vector<**Mat**>.
labels	The labels corresponding to the images have to be given either as a vector<int> or a **Mat** of type CV_32SC1.

write() [1/2]

```
virtual void cv::face::FaceRecognizer::write ( const String & filename ) const
```

Python:

```
cv.face.FaceRecognizer.write( filename ) -> None
```

Saves a **FaceRecognizer** and its model state.

Saves this model to a given filename, either as XML or YAML.

Parameters

filename	The filename to store this **FaceRecognizer** to (either XML/YAML).

read() [1/2]

```
virtual void cv::face::FaceRecognizer::read ( const String & filename )
```

Python:

```
cv.face.FaceRecognizer.read( filename ) -> None
```

Loads a **FaceRecognizer** and its model state.

Loads a persisted model and state from a given XML or YAML file .

https://docs.opencv.org/4.x/dd/d65/classcv_1_1face_1_1FaceRecognizer.html#aa2d2f02faffab1bf01317ae6502fb631