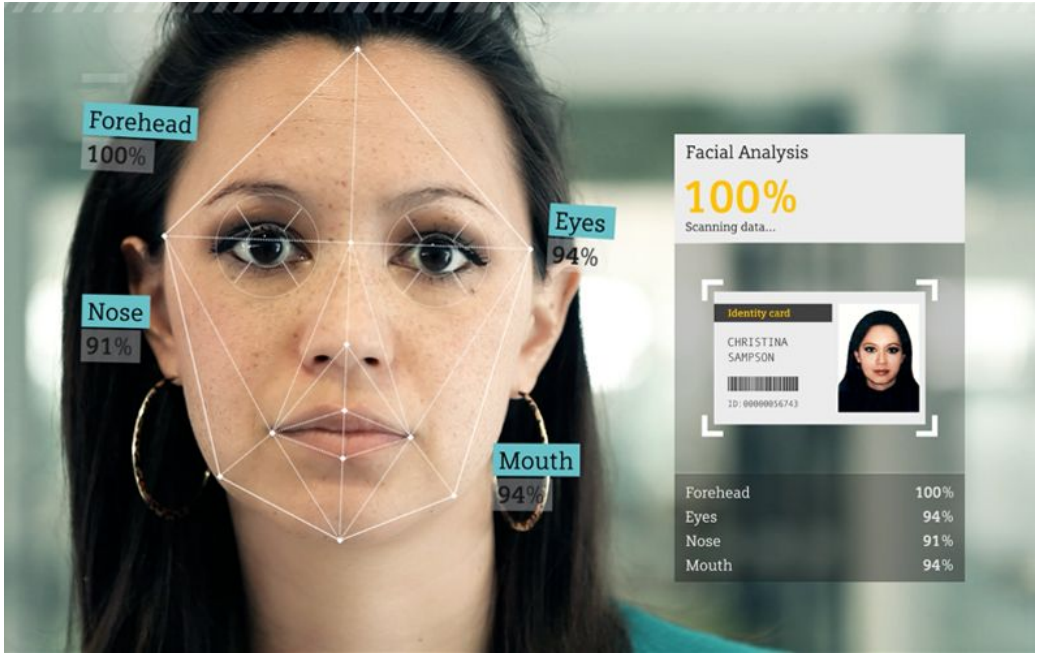
Face/Object Recognition

Face Recognition vs. Face Detection

What is difference between recognition and detection?

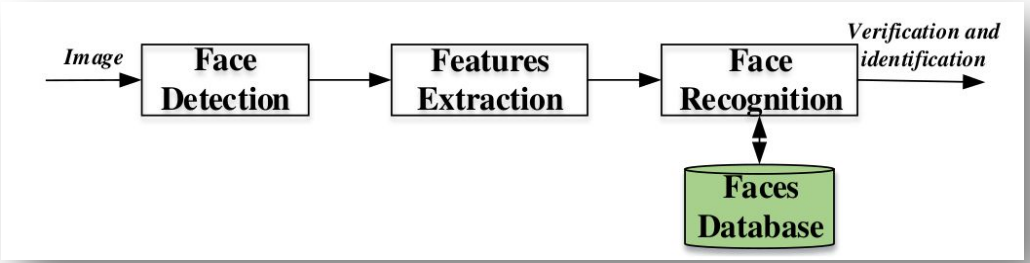


https://recfaces.com/articles/how-accurate-is-facial-recognition

Face Recognition

Face Recognition can be considered as multi step process:

1. Face detection
2. Feature extraction
3. Face recognition



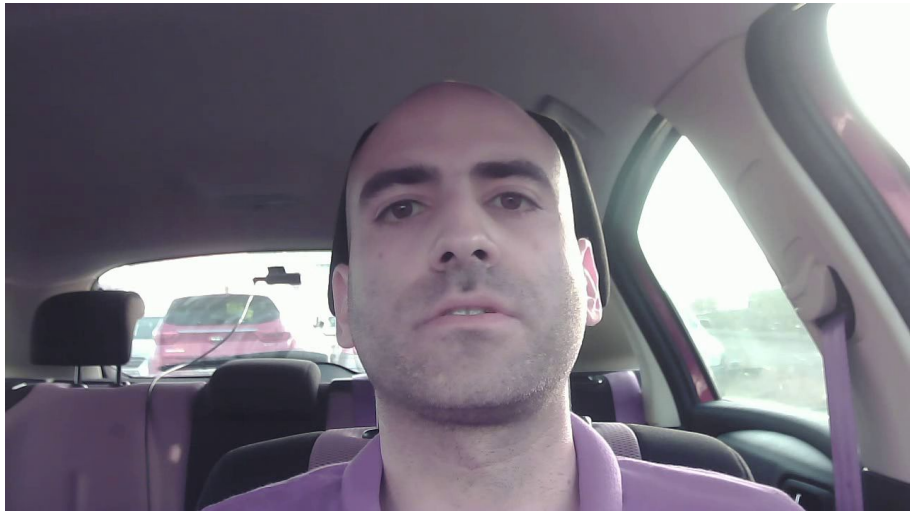
Three basic steps are used to develop a robust face recognition system: (1) face detection, (2) feature extraction, and (3) face recognition (shown in Figure 1) [3,23]. The face detection step is used to detect and locate the human face image obtained by the system. The feature extraction step is employed to extract the feature vectors for any human face located in the first step. Finally, the face recognition step includes the features extracted from the human face in order to compare it with all template face databases to decide the human face identity.

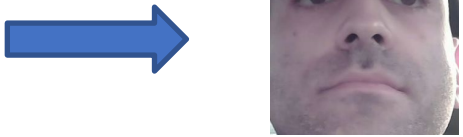
Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

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:

1. Face detection

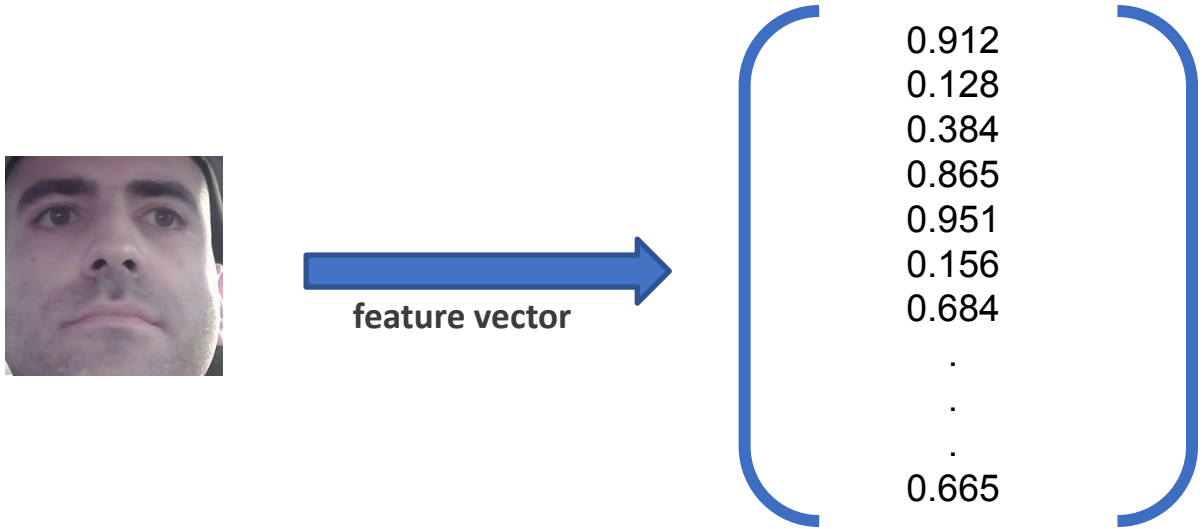
Face Detection: The face recognition system begins first with the localization of the human faces in a particular image. The purpose of this step is to determine if the input image contains human faces or not. The variations of illumination and facial expression can prevent proper face detection. In order to facilitate the design of a further face recognition system and make it more robust, pre-processing steps are performed. Many techniques are used to detect and locate the human face image, for example, Viola–Jones detector [24,25], histogram of oriented gradient (HOG) [13,26], and principal component analysis (PCA) [27,28]. Also, the face detection step can be used for video and image classification, object detection [29], region-of-interest detection [30], and so on.

Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

Face Recognition

Face Recognition can be considered as multi step process:

2. Feature extraction



Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

Face Recognition

Face Recognition can be considered as multi step process:

2. Feature extraction

Feature Extraction: The main function of this step is to extract the features of the face images detected in the detection step. This step represents a face with a set of features vector called a "signature" that describes the prominent features of the face image such as mouth, nose, and eyes with their geometry distribution [31,32]. Each face is characterized by its structure, size, and shape, which allow it to be identified. Several techniques involve extracting the shape of the mouth, eyes, or nose to identify the face using the size and distance [3]. HOG [33], Eigenface [34], independent component analysis (ICA), linear discriminant analysis (LDA) [27,35], scale-invariant feature transform (SIFT) [23], gabor filter, local phase quantization (LPQ) [36], Haar wavelets, Fourier transforms [31], and local binary pattern (LBP) [3,10] techniques are widely used to extract the face features.

Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

Face Recognition

Face Recognition can be considered as multi step process:

3. Recognition phase

e.g. comparison of new feature vectors with existing ones

Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

Face Recognition

Face Recognition can be considered as multi step process:

3. Recognition phase

Face Recognition: This step considers the features extracted from the background during the feature extraction step and compares it with known faces stored in a specific database. There are two general applications of face recognition, one is called identification and another one is called verification. During the identification step, a test face is compared with a set of faces aiming to find the most likely match. During the identification step, a test face is compared with a known face in the database in order to make the acceptance or rejection decision [7,19]. Correlation filters (CFs) [18,37,38], convolutional neural network (CNN) [39], and also k-nearest neighbor (K-NN) [40] are known to effectively address this task.

Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. *Sensors* **2020**, *20*, 342. https://doi.org/10.3390/s20020342

Face Recognition Methods

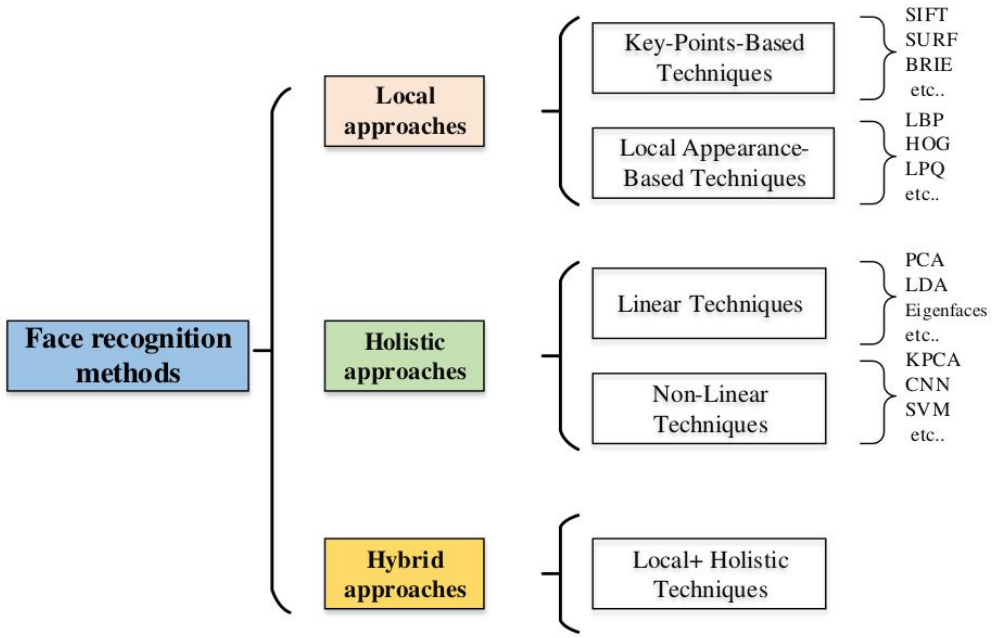


Figure 2. Face recognition methods. SIFT, scale-invariant feature transform; SURF, scale-invariant feature transform; BRIEF, binary robust independent elementary features; LBP, local binary pattern; HOG, histogram of oriented gradients; LPQ, local phase quantization; PCA, principal component analysis; LDA, linear discriminant analysis; KPCA, kernel PCA; CNN, convolutional neural network; SVM, support vector machine.

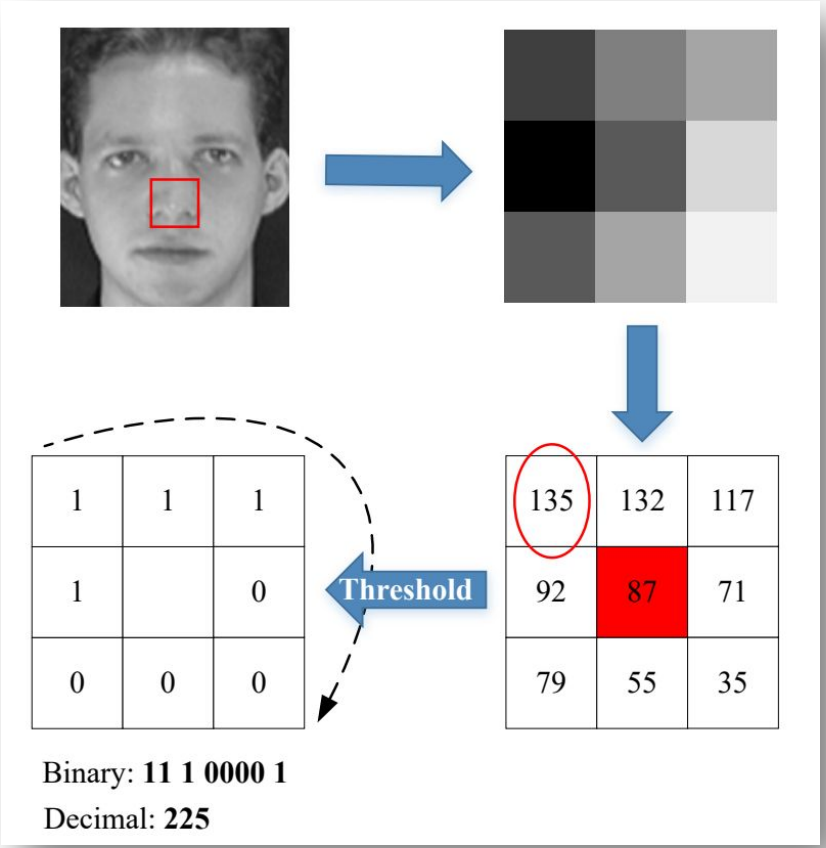
Many of these methods can be used with the use of existing frameworks, e.g. OpenCV, Dlib, PyTorch, scikit-image.

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.
Kortli, Y.; Jridi, M.; Al Falou, A.; Atri, M. Face Recognition Systems: A Survey. Sensors 2020, 20, 342. https://doi.org/10.3390/s20020342

Face Recognition - LBP

LBP features

- The local binary patterns (LBP) were introduced by Ojala et al. 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.

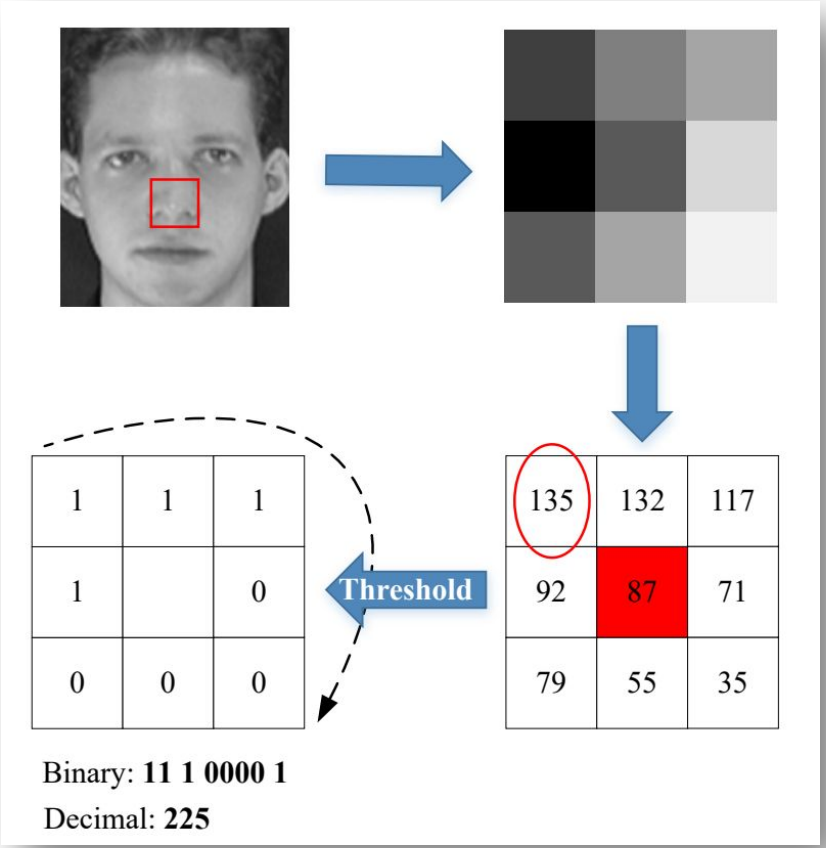


Kamencay, Patrik et al. “A new method for face recognition using convolutional neural network.” *Advances in Electrical and Electronic Engineering* 15 (2017): 663-672.

Face Recognition - LBP

LBP features

- The result of **comparison** is the 8-bit binary number; in the 8-bit binary number - If the neighbor's intensity is greater than or equal to the central pixel, assign a binary value of 1; otherwise, assign 0.
- This binary number is then converted to a decimal value, called the LBP code (you can use different direction, but it must be same for all images).



Kamencay, Patrik et al. “A new method for face recognition using convolutional neural network.” *Advances in Electrical and Electronic Engineering* 15 (2017): 663-672.

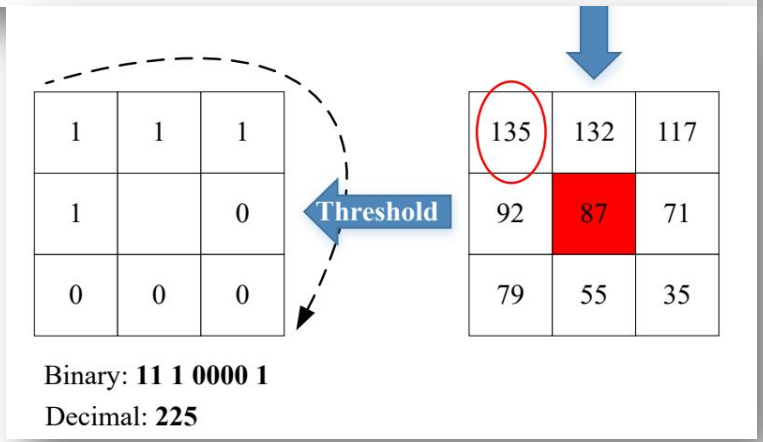
Face Recognition - LBP

A more formal description of the LBP operator can be given as:

$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$



https://docs.opencv.org/4.7.0/da/d60/tutorial_face_main.html

Kamencay, Patrik et al. "A new method for face recognition using convolutional neural network." *Advances in Electrical and Electronic Engineering* 15 (2017): 663-672.

Face Recognition - LBP

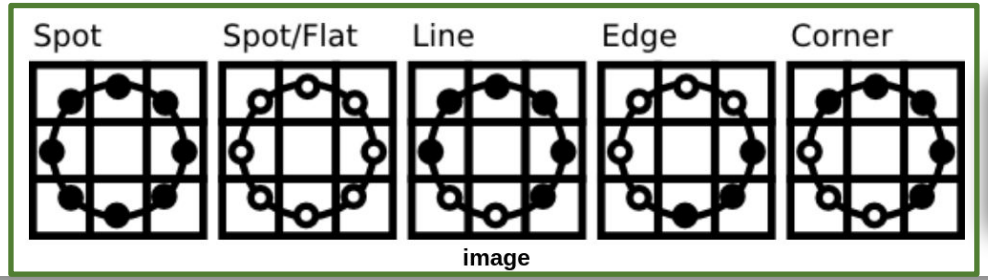
A more formal description of the LBP operator can be given as:

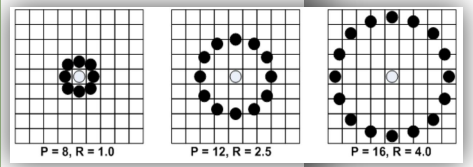
$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

This description enables you to capture very fine grained details in images. In fact the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [3] . The idea is to align an abritrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:





https://docs.opencv.org/4.7.0/da/d60/tutorial_face_main.html

Kamencay, Patrik et al. "A new method for face recognition using convolutional neural network." *Advances in Electrical and Electronic Engineering* 15 (2017): 663-672.

Face Recognition - LBP

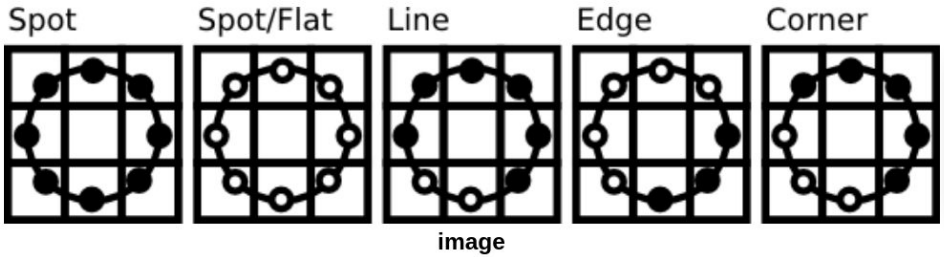
A more formal description of the LBP operator can be given as:

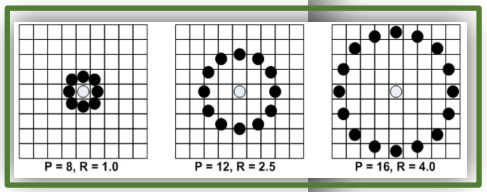
$$LBP(x_c, y_c) = \sum_{p=0}^{P-1} 2^p s(i_p - i_c)$$

, with (x_c, y_c) as central pixel with intensity i_c; and i_n being the intensity of the the neighbor pixel. s is the sign function defined as:

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases}$$

This description enables you to capture very fine grained details in images. In fact the authors were able to compete with state of the art results for texture classification. Soon after the operator was published it was noted, that a fixed neighborhood fails to encode details differing in scale. So the operator was extended to use a variable neighborhood in [3] . The idea is to align an abritrary number of neighbors on a circle with a variable radius, which enables to capture the following neighborhoods:



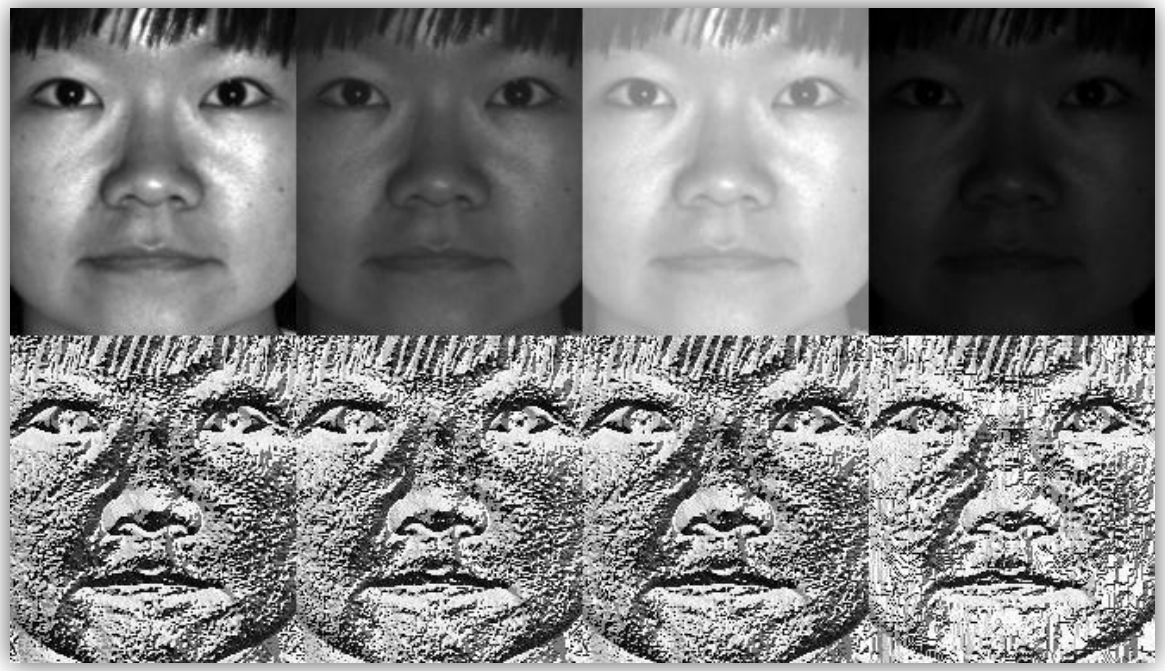


https://docs.opencv.org/4.7.0/da/d60/tutorial_face_main.html

Kamencay, Patrik et al. "A new method for face recognition using convolutional neural network." *Advances in Electrical and Electronic Engineering* 15 (2017): 663-672.

Face Recognition - LBP

LBP features – visualization



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

Face Recognition - LBP

Using LBP for face recognition:

- The transformed LBP image is divided into $k \times k$ regions to save the spatial information about the object.
- An LBP histogram is calculated for each region.
- Region histograms are stacked sequentially into a single face feature histogram
- We can use various methods to compare the histograms (calculate the distance between histograms), e.g. Euclidean distance, chi-square, absolute value.

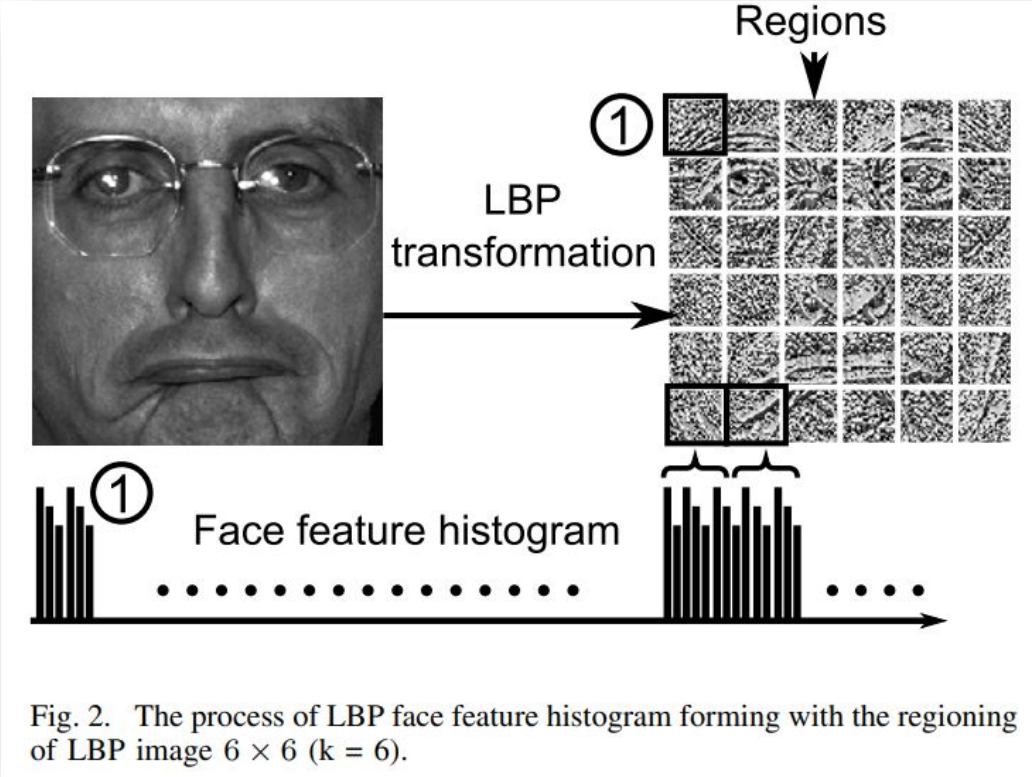


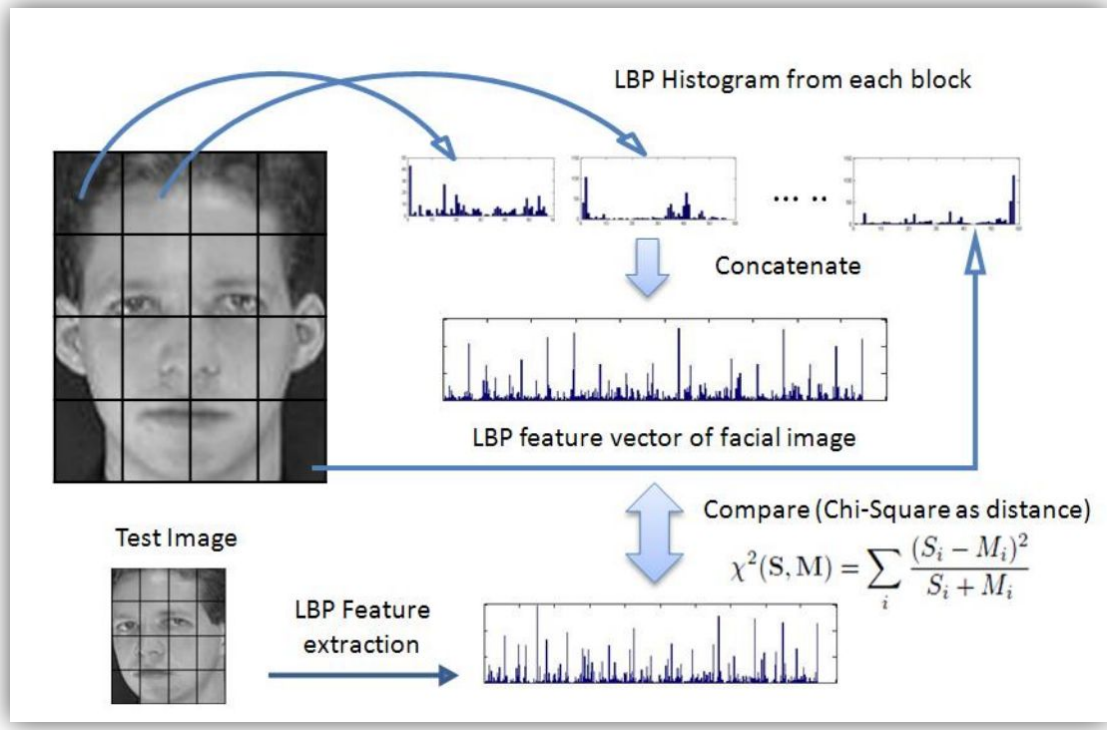
Fig. 2. The process of LBP face feature histogram forming with the regioning of LBP image 6 × 6 (k = 6).

Nikisins, Olegs & Greitans, Modris. (2012). A mini-batch discriminative feature weighting algorithm for LBP - Based face recognition. 170-175. 10.1109/IST.2012.6295521.

Face Recognition - LBP

Using LBP for face recognition:

- The transformed LBP image is divided into $k \times k$ regions to save the spatial information about the object.
- An LBP histogram is calculated for each region.
- Region histograms are stacked sequentially into a single face feature histogram
- We can use various methods to compare the histograms (calculate the distance between histograms), e.g. Euclidean distance, chi-square, absolute value.



Nikisins, Olegs & Greitans, Modris. (2012). A mini-batch discriminative feature weighting algorithm for LBP - Based face recognition. 170-175. 10.1109/IST.2012.6295521.
Zhao, Xuran & Evans, Nicholas & Dugelay, Jean-Luc. (2011). A co-training approach to automatic face recognition.

OpenCV + LBP

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

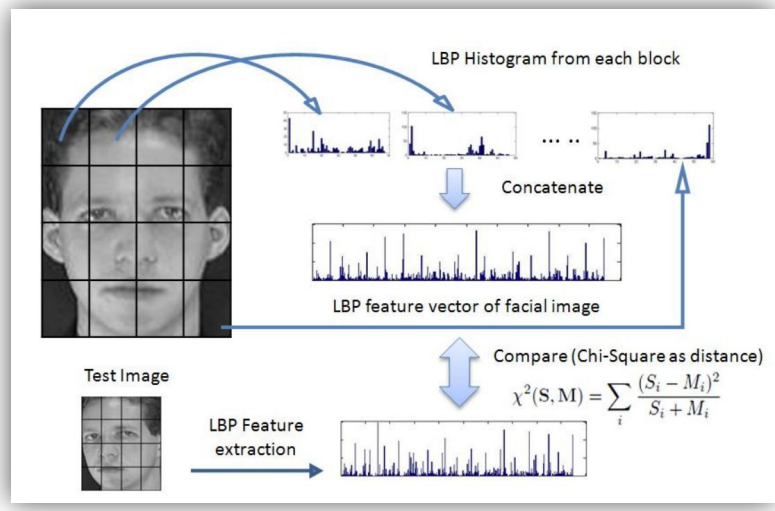
radius	The radius used for building the Circular Local Binary Pattern. The greater the radius, the smoother the image but more spatial information you can get.
neighbors	The number of sample points to build a Circular Local Binary Pattern from. An appropriate value is to use `8` sample points. Keep in 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.

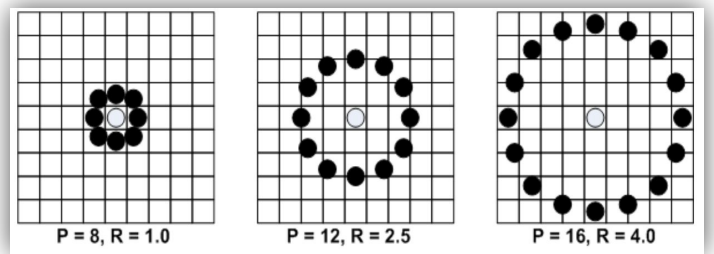
Notes:

- 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

OpenCV + LBP

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

labels = [0, 0, 0, 1, 1, 1]
np.array(labels)

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 .

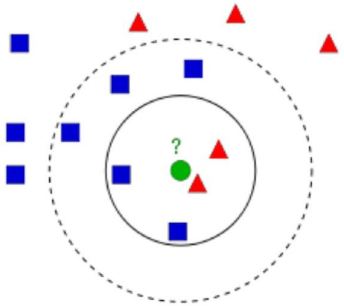
https://docs.opencv.org/4.x/dd/d65/classcv_1_1face_1_1FaceRecognizer.html#aa2d2f02faffab1bf01317ae6502fb631

OpenCV + LBP

In OpenCV, the final face recognition step is performed using a nearest neighbour classifier (k-NN).

Theory

kNN is one of the simplest classification algorithms available for supervised learning. The idea is to search for the closest match(es) of the test data in the feature space. We will look into it with the below image.



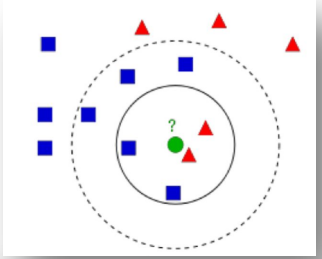
image

In the image, there are two families: Blue Squares and Red Triangles. We refer to each family as a **Class**. Their houses are shown in their town map which we call the **Feature Space**. You can consider a feature space as a space where all data are projected. For example, consider a 2D coordinate space. Each datum has two features, a x coordinate and a y coordinate. You can represent this datum in your 2D coordinate space, right? Now imagine that there are three features, you will need 3D space. Now consider N features: you need N-dimensional space, right? This N-dimensional space is its feature space. In our image, you can consider it as a 2D case with two features.

https://docs.opencv.org/4.x/dd/d65/classcv_1_1face_1_1FaceRecognizer.html#aa2d2f02faffab1bf01317ae6502fb631
https://docs.opencv.org/4.9.0/d5/d26/tutorial_py_knn_understanding.html

OpenCV + LBP

In OpenCV, the final face recognition step is performed using a nearest neighbour classifier (k-NN).



Now consider what happens if a new member comes into the town and creates a new home, which is shown as the green circle. He should be added to one of these Blue or Red families (or *classes*). We call that process, **Classification**. How exactly should this new member be classified? Since we are dealing with kNN, let us apply the algorithm.

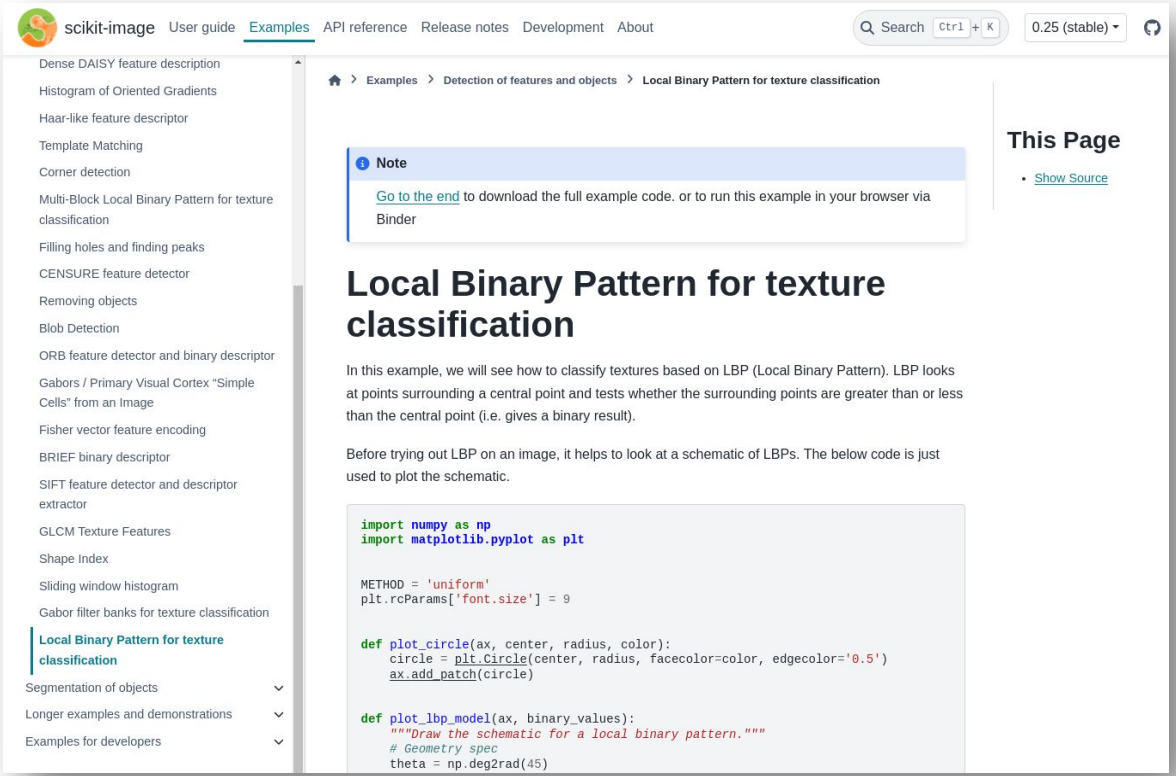
One simple method is to check who is his nearest neighbour. From the image, it is clear that it is a member of the Red Triangle family. So he is classified as a Red Triangle. This method is called simply **Nearest Neighbour** classification, because classification depends only on the *nearest neighbour*.

But there is a problem with this approach! Red Triangle may be the nearest neighbour, but what if there are also a lot of Blue Squares nearby? Then Blue Squares have more strength in that locality than Red Triangles, so just checking the nearest one is not sufficient. Instead we may want to check some **k** nearest families. Then whichever family is the majority amongst them, the new guy should belong to that family. In our image, let's take k=3, i.e. consider the 3 nearest neighbours. The new member has two Red neighbours and one Blue neighbour (there are two Blues equidistant, but since k=3, we can take only one of them), so again he should be added to Red family. But what if we take k=7? Then he has 5 Blue neighbours and 2 Red neighbours and should be added to the Blue family. The result will vary with the selected value of k. Note that if k is not an odd number, we can get a tie, as would happen in the above case with k=4. We would see that our new member has 2 Red and 2 Blue neighbours as his four nearest neighbours and we would need to choose a method for breaking the tie to perform classification. So to reiterate, this method is called **k-Nearest Neighbour** since classification depends on the *k nearest neighbours*.

https://docs.opencv.org/4.x/dd/d65/classcv_1_1face_1_1FaceRecognizer.html#aa2d2f02faffab1bf01317ae6502fb631
https://docs.opencv.org/4.9.0/d5/d26/tutorial_py_knn_understanding.html

scikit-image + LBP

Alternatively (to OpenCV), we can use **scikit-image** to extract LBP features.



https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html

scikit-image + LBP

skimage.feature.local_binary_pattern(*image, P, R, method='default'*)

Compute the local binary patterns (LBP) of an image. [source]

LBP is a visual descriptor often used in texture classification.

Parameters:

image : *(M, N) array*
2D grayscale image.

P : *int*
Number of circularly symmetric neighbor set points (quantization of the angular space).

R : *float*
Radius of circle (spatial resolution of the operator).

method : *str {'default', 'ror', 'uniform', 'nri_uniform', 'var'}, optional*
Method to determine the pattern:

`default`
Original local binary pattern which is grayscale invariant but not rotation invariant.

`ror`
Extension of default pattern which is grayscale invariant and rotation invariant.

`uniform`
Uniform pattern which is grayscale invariant and rotation invariant, offering finer quantization of the angular space. For details, see [1].

`nri_uniform`
Variant of uniform pattern which is grayscale invariant but not rotation invariant. For details, see [2] and [3].

`var`
Variance of local image texture (related to contrast) which is rotation invariant but not grayscale invariant.

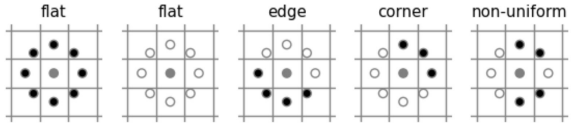
Returns:

output : *(M, N) array*
LBP image.

```
image = data.brick()
lbp = local_binary_pattern(image, n_points, radius, METHOD)


def hist(ax, lbp):
    n_bins = int(lbp.max() + 1)
    return ax.hist(
        lbp.ravel(), density=True, bins=n_bins, range=(0, n_bins), facecolor='0.5'
    )


# plot histograms of LBP of textures
fig, (ax_img, ax_hist) = plt.subplots(nrows=2, ncols=3, figsize=(9, 6))
plt.gray()
```



The figure above shows example results with black (or white) representing pixels that are less (or more) intense than the central pixel. When surrounding pixels are all black or all white, then that image region is flat (i.e. featureless). Groups of continuous black or white pixels are considered “uniform” patterns that can be interpreted as corners or edges. If pixels switch back-and-forth between black and white pixels, the pattern is considered “non-uniform”.

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html

scikit-image + LBP

skimage.feature.local_binary_pattern(*image, P, R, method='default'*)

Compute the local binary patterns (LBP) of an image. [source]

LBP is a visual descriptor often used in texture classification.

Parameters:

image : *(M, N) array*
2D grayscale image.

P : *int*
Number of circularly symmetric neighbor set points (quantization of the angular space).

R : *float*
Radius of circle (spatial resolution of the operator).

method : *str {'default', 'ror', 'uniform', 'nri_uniform', 'var'}, optional*
Method to determine the pattern:

`default`
Original local binary pattern which is grayscale invariant but not rotation invariant.

`ror`
Extension of default pattern which is grayscale invariant and rotation invariant.

`uniform`
Uniform pattern which is grayscale invariant and rotation invariant, offering finer quantization of the angular space. For details, see [1].

`nri_uniform`
Variant of uniform pattern which is grayscale invariant but not rotation invariant. For details, see [2] and [3].

`var`
Variance of local image texture (related to contrast) which is rotation invariant but not grayscale invariant.

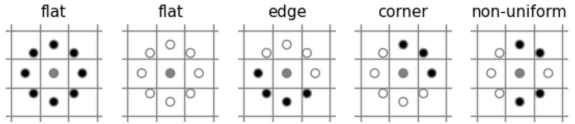
Returns:

output : *(M, N) array*
LBP image.

```
image = data.brick()
lbp = local_binary_pattern(image, n_points, radius, METHOD)


def hist(ax, lbp):
    n_bins = int(lbp.max() + 1)
    return ax.hist(
        lbp.ravel(), density=True, bins=n_bins, range=(0, n_bins), facecolor='0.5'
    )


# plot histograms of LBP of textures
fig, (ax_img, ax_hist) = plt.subplots(nrows=2, ncols=3, figsize=(9, 6))
plt.gray()
```



The figure above shows example results with black (or white) representing pixels that are less (or more) intense than the central pixel. When surrounding pixels are all black or all white, then that image region is flat (i.e. featureless). Groups of continuous black or white pixels are considered “uniform” patterns that can be interpreted as corners or edges. If pixels switch back-and-forth between black and white pixels, the pattern is considered “non-uniform”.

https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_local_binary_pattern.html

Where to go from here?

Wang, Mei, and Weihong Deng. "Deep face recognition: A survey." *Neurocomputing* 429 (2021): 215-244.

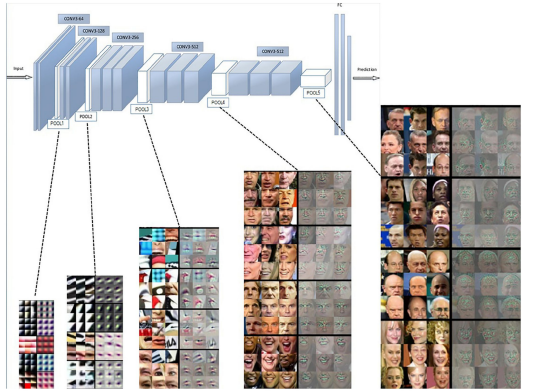


Fig. 2. The hierarchical architecture that stitches together pixels into invariant face representation. Deep model consists of multiple layers of simulated neurons that convolute and pool input, during which the receptive-field size of simulated neurons are continually enlarged to integrate the low-level primary elements into multifarious facial attributes, finally feeding the data forward to one or more fully connected layer at the top of the network. The output is a compressed feature vector that represent the face. Such deep representation is widely considered as the SOTA technique for face recognition.

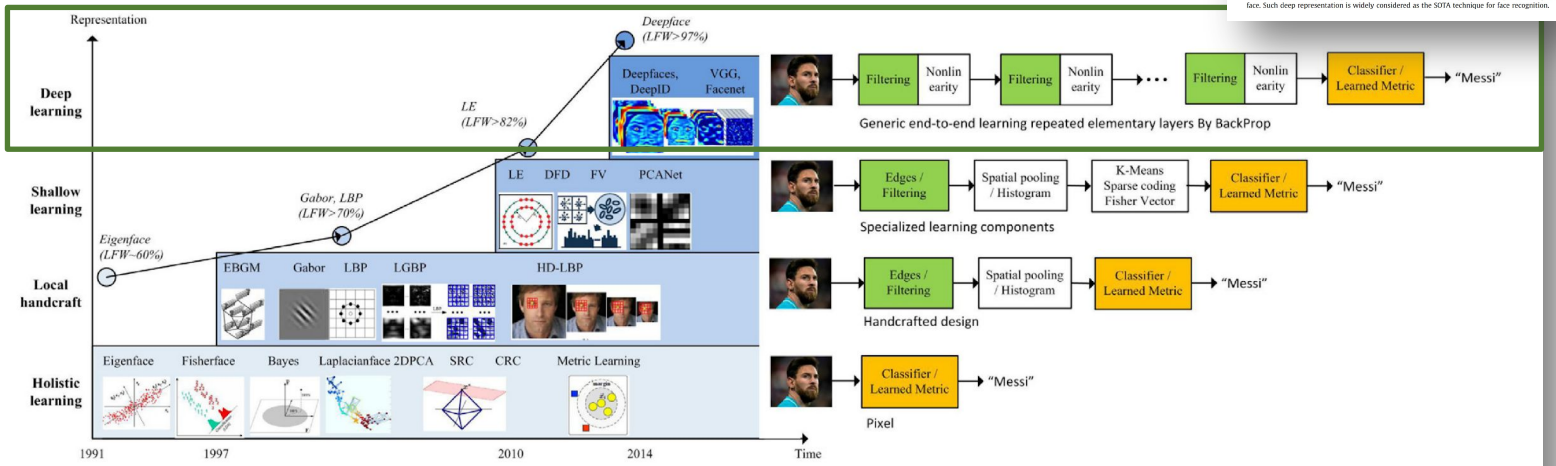


Fig. 1. Milestones of face representation for recognition. The holistic approaches dominated the face recognition community in the 1990s. In the early 2000s, handcrafted local descriptors became popular, and the local feature learning approaches were introduced in the late 2000s. In 2014, DeepFace [20] and DeepID [21] achieved a breakthrough on state-of-the-art (SOTA) performance, and research focus has shifted to deep-learning-based approaches. As the representation pipeline becomes deeper and deeper, the LFW (Labeled Face in-the-Wild) performance steadily improves from around 60% to above 90%, while deep learning boosts the performance to 99.80% in just three years.