|  |
| --- |
|  |
| Human Face Recognition Attendance System  Face Recognition For Everyone |
| |  |  |  | | --- | --- | --- | |  |  |  | |

**Face Recognition Algorithm**

Over the last ten years or so, face recognition has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding. There are different types of face recognition algorithms, for example:

• Eigenfaces (1991)

• Local Binary Patterns Histograms (LBPH) (1996)

• Fisherfaces (1997)

• Scale Invariant Feature Transform (SIFT) (1999)

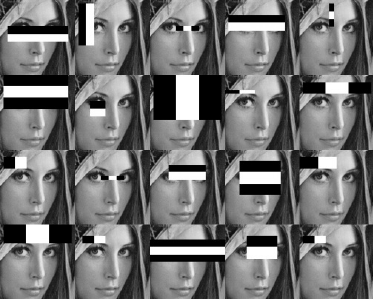
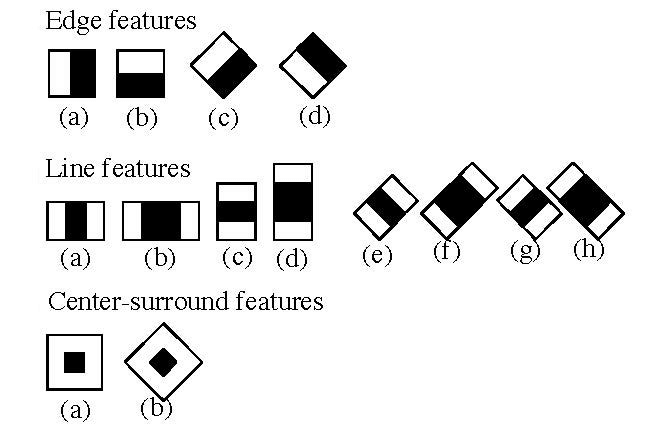
• Speed Up Robust Features (SURF) (2006)

Each method has a different approach to extract the image information and perform the matching with the input image. However, the methods Eigenfaces and Fisherfaces have a similar approach as well as the SIFT and SURF methods. One of the oldest (not the oldest one) and more popular face recognition algorithms: Local Binary Patterns Histograms (LBPH).

**How the Face Detection Works**

Firstly the image is imported by providing the location of the image. Then the picture is transformed from RGB to Grayscale because it is easy to detect faces in the grayscale. After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images done if needed. The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

The next step is to use algorithm. Algorithm used for finding the location of the human faces in a frame or image. All human faces shares some universal properties of the human face like the eyes region is darker than its neighbor pixels and nose region is brighter than eye region.

The haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, center detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.

The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.

**Local Binary Patterns Histograms (LBPH) (1996)**

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector.

**How LBPH algorithm work step by step:**

LBPH algorithm work in 5 steps.

1. **Parameters:** the LBPH uses 4 parameters:

* **Radius:** the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
* **Neighbors:** the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
* **Grid X:** the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
* **Grid Y:** the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

1. **Training the Algorithm:** First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let’s see the LBPH computational steps.
2. **Applying the LBP operation:** The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

The image below shows this procedure:

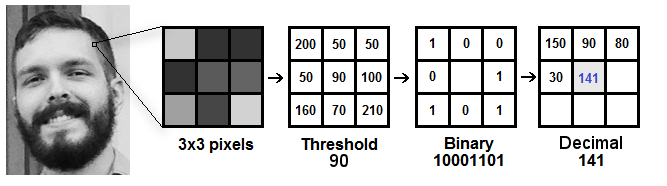
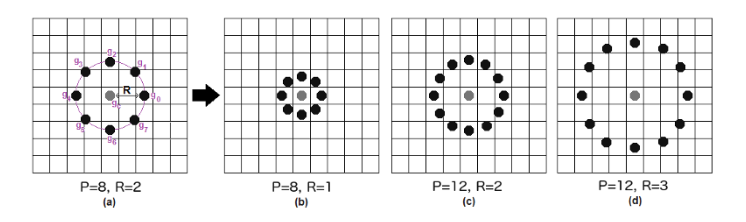


Fig: LBP Operation

Based on the image above, let’s break it into several small steps so we can understand it easily:

* Suppose we have a facial image in grayscale.
* We can get part of this image as a window of 3x3 pixels.
* It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
* Then, we need to take the central value of the matrix to be used as the threshold.
* This value will be used to define the new values from the 8 neighbors.
* For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
* Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
* Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
* At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.



It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

1. **Extracting the Histograms:** Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids,

as can be seen in the following image:

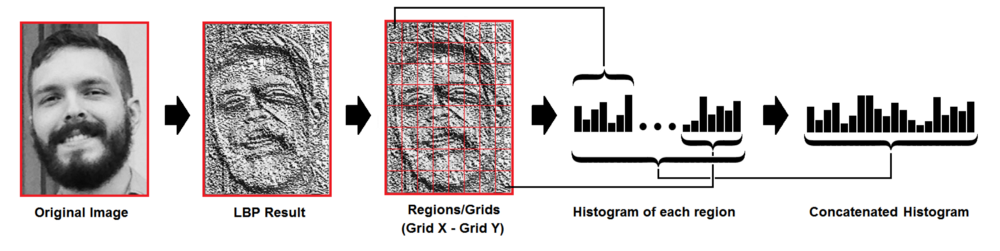


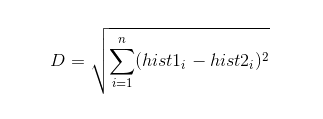
Fig: Extracting The Histogram

Based on the image above, we can extract the histogram of each region as follows:

* As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
* Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

1. **Performing the face recognition:** In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

* So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
* We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: Euclidean distance, chi-square, absolute value, etc. In this example, we can use the **Euclidean distance** (which is quite known) based on the following formula:



* So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a ‘confidence’ measurement. Note: don’t be fooled about the ‘confidence’ name, as lower confidences are better because it means the distance between the two histograms is closer.
* We can then use a threshold and the ‘confidence’ to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

**References**

* Ahonen, Timo, Abdenour Hadid, and Matti Pietikainen. “Face description with local binary patterns: Application to face recognition.” IEEE transactions on pattern analysis and machine intelligence 28.12 (2006): 2037–2041.
* Ojala, Timo, Matti Pietikainen, and Topi Maenpaa. “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns.” IEEE Transactions on pattern analysis and machine intelligence 24.7 (2002): 971–987.
* Ahonen, Timo, Abdenour Hadid, and Matti Pietikäinen. “Face recognition with local binary patterns.” Computer vision-eccv 2004 (2004): 469–481.
* LBPH OpenCV: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec\_tutorial.html#local-binary-patterns-histograms
* Local Binary Patterns: http://www.scholarpedia.org/article/Local\_Binary\_Patterns