|  |
| --- |
|  |
| Human Face Recognition Attendance System  Face Recognition For Everyone |
| |  |  |  | | --- | --- | --- | | kmh mubin | 6/23/19 | CSE 299 Junior Design | |

**Abstract:**

Authentication is one of the significant issues in the era of information system. Among other things, human face recognition (HFR) is one of known techniques which can be used for user authentication. However, it is difficult to estimate the attendance precisely using each result of face recognition independently because the face detection rate is not sufficiently high.

Continuous observation improves the performance for the estimation of the attendance we constructed the lecture attendance system based on face recognition, and applied the system to classroom. This paper first review the related works in the field of attendance management and face recognition. Then, it introduces our system structure and plan.

Finally, experiments are implemented to provide as evidence to support our plan. The result shows that continuous observation improved the performance for the estimation of the attendance.

**Keywords:** *Human Face Recognition, Attendance system, Raspberry Pi*

**Introduction:**

As we are making a system which can recognize face and match with its own database. It will make the attendance system more authentic. Our primary goal is to help the lecturers, improve and organize the process of track and manage student attendance and absenteeism.

Additionally, we seek to provide a valuable attendance service for both teachers and students. Reduce manual process errors by provide automated and a reliable attendance system uses face recognition technology.

Increase privacy and security which student cannot presenting himself or his friend while they are not. Flexibility, Lectures capability of editing attendance records.

**What Is Face Detection?**

**Face detection** is a type of computer vision technology that is able to identify people’s faces within digital images. This is very easy for humans, but computers need precise instructions. The images might contain many objects that aren’t human faces, like buildings, cars, animals, and so on.

It is distinct from other computer vision technologies that involve human faces, like facial recognition, analysis, and tracking.

**Facial recognition** involves identifying the face in the image as belonging to person X and not person Y. It is often used for biometric purposes, like unlocking your smartphone.

**Facial analysis** tries to understand something about people from their facial features, like determining their age, gender, or the emotion they are displaying.

**Facial tracking** is mostly present in video analysis and tries to follow a face and its features (eyes, nose, and lips) from frame to frame. The most popular applications are various filters available in mobile apps like Snapchat.

**How Do Computers “See” Images?**

The smallest element of an image is called a pixel, or a picture element. It is basically a dot in the picture. An image contains multiple pixels arranged in rows and columns.

You will often see the number of rows and columns expressed as the image resolution. For example, an Ultra HD TV has the resolution of 3840x2160, meaning it is 3840 pixels wide and 2160 pixels high.

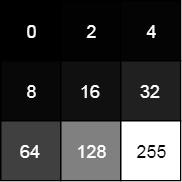
But a computer does not understand pixels as dots of color. It only understands numbers. To convert colors to numbers, the computer uses various color models.

In color images, pixels are often represented in the RGB color model. RGB stands for Red Green Blue. Each pixel is a mix of those three colors. RGB is great at modeling all the colors humans perceive by combining various amounts of red, green, and blue.

Since a computer only understand numbers, every pixel is represented by three numbers, corresponding to the amounts of red, green, and blue present in that pixel.

In grayscale (black and white) images, each pixel is a single number, representing the amount of light, or intensity, it carries. In many applications, the range of intensities is from 0 (black) to 255 (white). Everything between 0 and 255 is various shades of gray.

If each grayscale pixel is a number, an image is nothing more than a matrix (or table) of numbers:



In color images, there are three such matrices representing the red, green, and blue channels.

**What Are Features?**

A feature is a piece of information in an image that is relevant to solving a certain problem. It could be something as simple as a single pixel value, or more complex like edges, corners, and shapes. You can combine multiple simple features into a complex feature.

Applying certain operations to an image produces information that could be considered features as well. Computer vision and image processing have a large collection of useful features and feature extracting operations.

Basically, any inherent or derived property of an image could be used as a feature to solve tasks.

**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).**

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

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 Local Binary Patterns Histograms Work?**

**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.

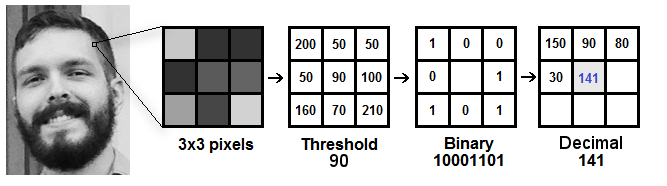
**Grid X:** the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

**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.

**2.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.

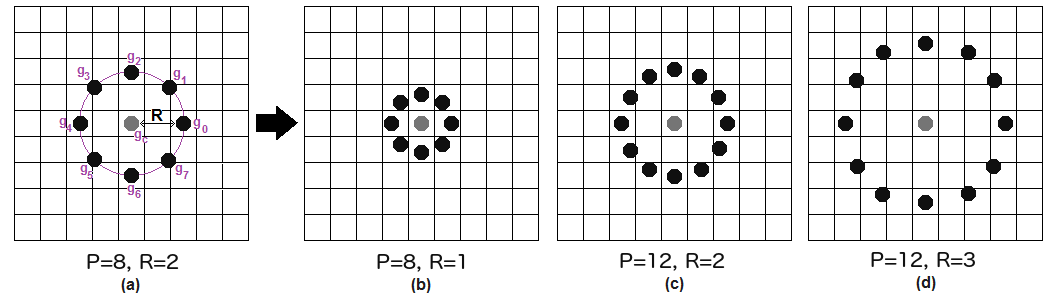
**3.Applying the LBP operation:** The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

The image below shows this procedure:



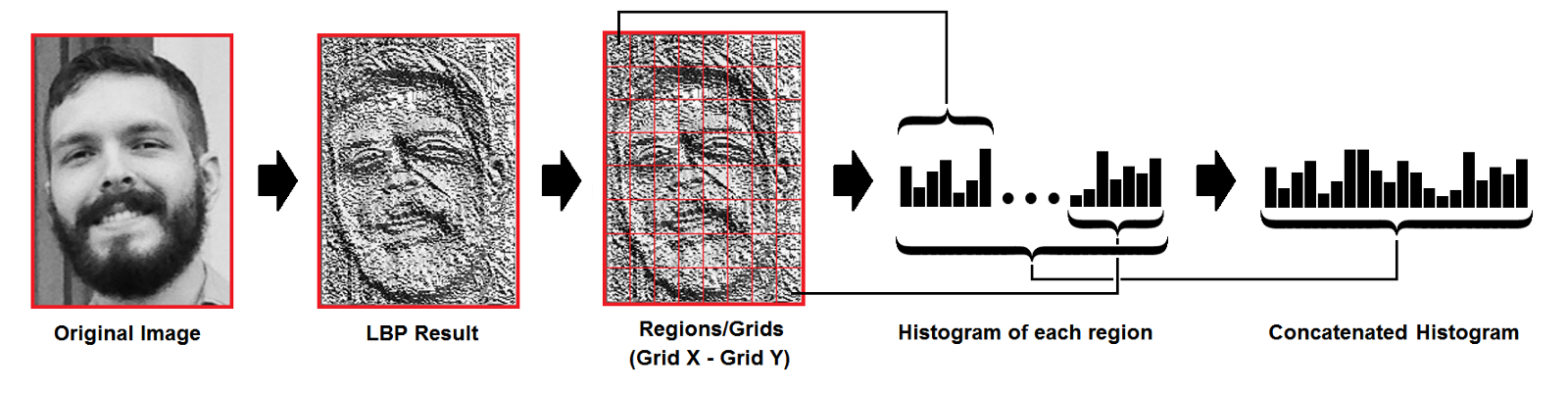
Based on the image above, let’s break it into several small steps so we can understand it easily:

* 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.
* Note: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.



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.

**4.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:



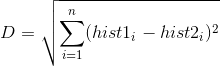
Based on the image above, we can extract the histogram of each region as follows:

* 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.

The LBPH algorithm is pretty much it.

**5.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.