# Human Identification via Neural Network

By: Parviz Eshaghi
Tehran- Iran

## Abstract

*This paper presents a novel approach of iris verification based on Learning Vector Quantization Neural Network. The features used in this approach are based on the differences between the lines, rakes, and vessels of each iris considered as being non identical with any other one in the world. And for extracting these features, equipments like edge detection and discrete cosine transform (DCT) are used. The recognition obtained is 98% in small size database given via Learning Vector Quantization Neural Network.*

**Key words**: Canny edge detection, discrete cosine transform, Learning Vector Quantization

## 1. Introduction

There are variable ways of human verification through out the world, as it is of great importance for all organizations, and different centers. Nowadays, the most important ways of human verification are recognition via DNA, face, fingerprint, signature, speech, and iris.

Among all, one of the recent, reliable, and technological methods is iris recognition which is practiced by some organizations today, and its wide usage in the future is of no doubt. Iris is a non identical organism made of colorful muscles including robots with shaped lines. These lines are the main causes of making every one's iris non identical. Even the irises of a pair of eyes of one person are completely different from one another. Even in the case of identical twins irises are completely different. Each iris is specialized by very narrow lines, rakes, and vessels in different people.

The precision of identification via iris is increased by using more and more details. It has been proven that iris patterns are never changed nearly from the time the child is one year old through out all his life.

Over the past few years there has been considerable interest in the development of neural network based pattern recognition systems, because of their ability to classify data. The kind of neural network practiced by the researcher is the Learning Vector Quantization which is a competitive network functional in the field of classification of the patterns.

The iris images prepared as the database is in the form of PNG (portable network graphics) pattern, meanwhile they must be preprocessed through which the boundary of the iris is recognized and their features are extracted. For doing so, edge detection is done by the usage of Canny approach. For more diverse and feature extraction of iris images DCT transform is practiced.

## 2. Feature Extraction

For increasing the precision of our verification of iris system we should extract the features so that they contain the main items of the images for comparison and identification. The extracted features should be in a way that cause the least of flaw in the output of the system and in the ideal condition the output flaw of the system should be zero. The useful features which should be extracted are obtained through edge detection in the first step and the in next step we use DCT transform.

### 2.1 Edge Detection

The first step locates the iris outer boundary, i.e. border between the iris and the sclera. This is done by performing edge detection on the gray scale iris image. In this work, the edges of the irises are detected using the "Canny method" which finds edges by finding local maxima of the gradient. The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds, to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. This method is robust to additive noise, and able to detect "true" weak edges. Figures 1 and 2 are the original and edge images, respectively.

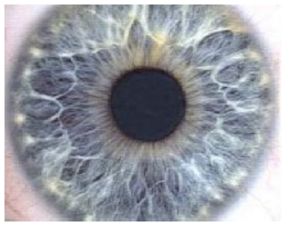


Figure 1: image of a sample iris

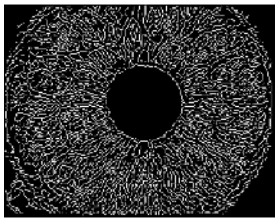


Figure 2: edges of a sample iris

Although certain literature has considered the detection of ideal step edges, the edges obtained from natural images are usually not at all ideal step edges. Instead they are normally affected by one or several of these effects: focal blur caused by a finite depth-of-field and finite point spread function, penumbral blur caused by shadows created by light sources of non-zero radius, shading at a smooth object edge, and local peculiarities or inter reflections in the vicinity of object edges.

Despite the following model does not capture the full variability of real-life edges, the error function (erf) has been used by a number of researchers as the simplest extension of the ideal step edge model for modeling the effects of edge blur in practical applications. Thus, a one-dimensional image (f) which has exactly one edge placed at x = 0 may be modeled as:

$$f(x) = \frac{I_r - I_l}{2}\left(\operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + 1\right) + I_l. \qquad (1)$$

At the left side of the edge, the intensity is $I_l = \lim_{x\to-\infty} f(x)$, and right of the edge it is $I_r = \lim_{x\to\infty} f(x)$. The scale parameter σ is called the blur scale of the edge.

### 2.1.1 Canny Method

The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work. He was very successful in achieving his goal and his ideas and methods can be found in his paper, "A Computational Approach to Edge Detection". In his paper, he followed a list of criteria to improve current methods of edge detection. The first and most obvious is low error rate. It is important that edges existing in images should not be missed and that there be NO responses to non-edges. The second criterion is that the edge points be well localized. In other words, the distance between the edge pixels as found by the detector and the actual edge is to be at a minimum. A third criterion is to have only one response to a single edge. This was implemented because the first 2 were not substantial enough to completely eliminate the possibility of multiple responses to an edge.

The Canny operator works in a multi-stage process. First of all the image is smoothed by Gaussian convolution. Then a simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothed image to highlight regions of the image with important spatial derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as non-maximal suppression. The tracking process exhibits hysteresis controlled by two thresholds: T1 and T2, with T1 > T2. Tracking can only begin at a point on a ridge higher than T1. Tracking then continues in both directions out from that point until the height of the ridge falls below T2. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (Roberts, Prewitt, Sobel for example) returns a value for the first derivative in the horizontal direction (Gy) and the vertical direction (Gx). From this the edge gradient and direction can be determined:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \qquad (2)$$

$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right) \qquad (3)$$

The edge direction angle (theta) is rounded to one of four angles representing vertical, horizontal and the two diagonals (0, 45, 90 and 135 degrees for example).

## 2.2 Discrete Cosine Transform

Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transform (DFT), a DCT operates on a function at a finite number of discrete data points. The obvious distinction between a DCT and a DFT is that the former uses only cosine functions, while the latter uses both cosines and sinusoids (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DCT implies different boundary conditions than the DFT or other related transforms.

The Fourier-related transforms that operate on a function over a finite domain, such as the DFT or DCT or a Fourier series, can be thought of as implicitly defining an extension of that function outside the domain. That is, once you write a function f(x) as a sum of sinusoids, you can evaluate that sum at any x, even for x where the original f(x) was not specified. The DFT, like the Fourier series, implies a periodic extension of the original function. A DCT, like a cosine transform, implies an even extension of the original function.

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio and images (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications:

for compression, it turns out that cosine functions are much more efficient (as explained below, fewer are needed to approximate a typical signal), whereas for differential equations the cosines express a particular choice of boundary conditions.

In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and output data are shifted by half a sample. There are eight standard DCT variants, of which four are common.

The most common variant of discrete cosine transform is the type-II DCT, which is often called simply "the DCT"; its inverse, the type-III DCT, is correspondingly often called simply "the inverse DCT" or "the IDCT". Two related transforms are the discrete sine transform (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transform (MDCT), which is based on a DCT of overlapping data.

The DCT, and in particular the DCT-II, is often used in signal and image processing, especially for lossy data compression, because it has a strong "energy compaction" property. Most of the signal information tends to be concentrated in a few low-frequency components of the DCT.

DCT-II

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, \dots, N-1. \tag{4}$$

This transform is exactly equivalent (up to an overall scale factor of 2) to a DFT of 4N real inputs of even symmetry where the even-indexed elements are zero. That is, it is half of the DFT of the 4N inputs yn, where y2n = 0, y2n + 1 = xn for $0 \leq n < N$, and y4N − n = yn for 0 < n < 2N.

The DCT-II implies the boundary conditions: xn is even around n=-1/2 and even around n=N-1/2; Xk is even around k=0 and odd around k=N.

## 3. Neural Network

In this work one Neural Network structure is used, which is Learning Vector Quantization Neural Network. A brief overview of this network is given below.

### 3.1 Learning Vector Quantization

Learning Vector Quantization (LVQ) is a supervised version of vector quantization, similar to Self organizing Maps (SOM) based on work of Linde et al, Gray and Kohonen. It can be applied to pattern recognition, multi-class classification and data compression tasks, e.g. speech recognition, image processing or customer classification. As supervised method, LVQ uses known target output classifications for each input pattern of the form.

LVQ algorithms do not approximate density functions of class samples like Vector Quantization or Probabilistic Neural Networks do, but directly define class boundaries based on prototypes, a nearest-neighbor rule and a winner-takes-it-all paradigm. The main idea is to cover the input space of samples with 'codebook vectors' (CVs), each representing a region labeled with a class. A CV can be seen as a prototype of a class member, localized in the centre of a class or decision region in the input space. A class can be represented by an arbitrarily number of CVs, but one CV represents one class only.

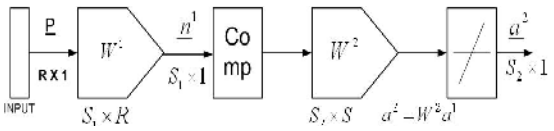
In terms of neural networks a LVQ is a feed forward net with one hidden layer of neurons, fully connected with the input layer. A CV can be seen as a hidden neuron ('Kohonen neuron') or a weight vector of the weights between all input neurons and the regarded Kohonen neuron respectively.

Learning means modifying the weights in accordance with adapting rules and, therefore, changing the position of a CV in the input space. Since class boundaries are built piecewise-linearly as segments of the mid-planes between CVs of neighboring classes, the class boundaries are adjusted during the learning process. The tessellation induced by the set of CVs is optimal if all data within one cell indeed belong to the same class. Classification after learning is based on a presented sample's vicinity to the CVs: the classifier assigns the same class label to all samples that fall into the same tessellation – the label of the cell's prototype (the CV nearest to the sample).

The core of the heuristics is based on a distance function – usually the Euclidean distance is used – for comparison between an input vector and the class representatives. The distance expresses the degree of similarity between presented input vector and CVs. Small distance corresponds with a high degree of similarity and a higher probability for the presented vector to be a member of the class represented by the nearest CV. Therefore, the definition of class boundaries by LVQ is strongly dependent on the distance function, the start positions of CVs, their adjustment rules and the pre-selection of distinctive input features.

Briefly explaining, this network has two layers: a layer of input neurons, and a layer of output neurons. The network is given by prototypes W=(w(i),...,w(n)). It changes the weights of the network in order to classify the data correctly. For each data point, the prototype (neuron) that is closest to it is determined (called the winner neuron). The weights of the connections to this neuron are then adapted, i.e. made closer if it correctly classifies the data point or made less similar if it incorrectly classifies it.

### 3.1.1 Learning Algorithm



(5)

Learning Vector Quantization (LVQ) structure

The number of neurons in the first layer (s1) should be equal at least to the number of neurons in the second layer, i.e, $S_1 \geq S_2$

Generally, the neurons in the first layer are more than the second layer. The behavior of the LVQ Neural Network is expressed by the equation below:

$$n_i^1 = -\| W_i^1 - \underline{P}^T \| \quad (6)$$

*The pure input of the 1st neuron in the first layer*

And also it is written in Vector form as below:

$$n^1 = \begin{bmatrix} \| W_1^1 - \underline{P}^T \| \\ \\ \| W_{S1}^1 - \underline{P}^T \| \end{bmatrix} \quad (7)$$

The output vector of the first layer is:

$$\underline{a}^1 = comp(n^1) \quad (8)$$

Therefore, the vector which has the nearest weight to the input vector is equal to 1, and the rest of the neurons have the zero output. The function of the second layer is to compose the subclasses of one class and to create just one class. $W^2$ matrix in each column has the element 1 and the other elements are zero.

$W_{ji}^2 = 1$ The subclass of i belongs to j class

Kohonen learning rule is used to organize the parameters of the LVQ NN layer in the form below:

$$W_{i*}^1(k+1) = W_{i*}^1(k) + a(\underline{P}^T(k+1) - W_{i*}^1(k)), \quad (9)$$

If: $a_{j*}^2 = t_{j*}(k+1) = 1$ (10)

## 4. Simulation Results

Practically we have done this work for a prepared database for 10 people in which we gave a class for the iris image of the left and right eye of every one and in the long run we obtained 10 classes. It means that in the second layer (s2) of the LVQ NN the number of the neurons is 10, while we put in the first layer (s1) 30 neurons. For each one of these 10 persons we took one image of the left eye iris and another from the right eye iris, and implemented these 20 taken images to the input of neural network after feature extraction by Canny edge detection approach and DCT transform. After learning network for these 20 input images, and testing by the other images from other left and right eyes irises other than the very 20 images we had, finally the true recognition results of our test came to an average of 98%. In this test we also used different noised images.

## 5. Conclusion

In this paper, a novel technique is proposed for iris verification. The classification is performed using LVQ Neural Network. The neural network based approach is found to be a promising one for iris recognition.

## References

[1] Canny, J., *A Computational Approach to Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

[2] Frigo, Matteo, Steven G. Johnson: FFTW, http://www.fftw.org/. A free (GPL) C library that can compute fast DCTs (types I-IV) in one or more dimensions, of arbitrary size.

[3] Frigo, Matteo, Steven G. Johnson, "The Design and Implementation of FFTW3," Proceedings of the IEEE 93 (2), 216–231 2005.

[4] "Bibliography on the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ)", Neural Networks Research Centre, Helsinki University of Technology, 2002.