Eigenvectors adopted for geometrical shapes recognition

Ghada thanoon talee
Department of Computer Science
College of Computer Science and Mathematics
University of Mosul

Received 15 / 04 / 2010 **Accepted** 05 / 01 / 2011

الخلاصة

الأشكال الهندسية فرع م هم جدا وضروري في مجال تمييز الأنماط لذا نرى العديد من البحوث التي عملت في هذا المجال.

في هذا البحث تم تمييز الأشكال الهندسية الأساسية مثل (الدائرة، المربع، المثلث، المعين،...، الخ) وذلك ببناء قاعدة بيانات لهذه الأشكال بالاعتماد على (المتجه المميز والقيمة المميزة). تم دراسة (القيمة المميزة و المتجه المميز) في هذا البحث كمعامل لتمييز الأشكال والتي أظهرت نسبة تمييز قوية ، تم توظيف شبكة الانتشار العكسي (backprogation) من اجل تسريع عملية التمييز.

Abstract

Geometrical shapes are so important in pattern and image recognition So many research use proposed in this direction .

In this paper study basic geometrical shapes [circle, triangle, rectangle,…,etc] are studied and small database was build for them based on their eigenvalue. Eigenvalues and eigenvectors studied in this paper as shape descriptor parameters, which found it can be strongly used due to the high difference shapes.

backprogation neural network was achieved to a speed path recognition process.

1. Introduction

"A picture is worth one thousand words". This proverb comes from Confucius-a Chinese philosopher about 2500 years ago. Now, the essence

of these words is universally understood. A picture can be magical in its ability to quickly communicate a complex story or a set of ideas that can be recalled by the viewer later in time.

Visual information plays an important role in our society, it will play an increasingly pervasive role in our lives, and there will be a growing need to have these sources processed further. The pictures or images are used in many application areas like computer vision, architectural and engineering design, fashion, journalism, advertising, entertainment, etc. Thus it provides the necessary opportunity for us to use the abundance of images. However, the knowledge will be useless if one can't find it. Face to the substantive and increasing apace images, how to search and to retrieve the images that we are interested in facility is a fatal problem: it brings a necessity for image retrieval systems. As we know, visual features of the images provide a description of their content. Content-based image retrieval (CBIR), emerged as a promising mean for retrieving images and browsing large images databases. CBIR has been a topic of intensive research in recent years. It is the process of retrieving images from a collection based on automatically extracted features from those images.[1]

shape (from Old English esceap, shap, etc., originally meaning created thing) of an object located in some space is the part of that space occupied by the object, as determined by its external boundary – abstracting from other properties such as colour, content, and material composition, as well as from the object's other spatial properties (position and orientation in space; size).

Mathematician and statistician David George Kendall defined shape this way:

Shape is all the geometrical information that remains when location, scale and rotational effects are filtered out from an object.

Simple two-dimensional shapes can be described by basic geometry such as points, line, curves, plane, and so on. (A shape whose points belong all to the same plane is called a plane figure.) Most shapes occurring in the physical world are complex. Some, such as plant structures and coastlines, may be so arbitrary as to defy traditional mathematical description – in which case they may be analysed by differential geometry, or as fractals.[2]

Basically, shape-based image retrieval consists of measuring the similarity between shapes represented by their features. Some simple geometric features can be used to describe shapes. Usually, the simple geometric features can only discriminate shapes with large differences; therefore, they are usually used as filters to eliminate false hits or combined with other shape descriptors to discriminate shapes. They are not suitable to be stand alone shape descriptors. A shape can be described by different aspects.[1]

we introduce eigenregions, which are geometrical features that encompass area, location and shape properties of a region. Eigenregions are obtained by analyzing segmented image regions with Principal Component Analysis (PCA).

Principal component analysis has already successfully been implemented in image classification for many tasks, but usually on the whole image. As opposed to other geometrical region features, eigenregions can be used and result in significant classification improvement even if the image regions are spatially incoherent.

They are also visually significant and computationally efficient. Another key result obtained with eigenregions is that for a large dataset of natural images, the largest variance in region geometry is due to the area and not to shape or position.[3]

2. shape descriptor features

Efficient shape features must present some essential properties such as:

- identifiability: shapes which are found perceptually similar by human have the same features that are different from the others.
- translation, rotation and scale invariance: the location, the rotation and the scaling changing of the shape must not affect the extracted features.
- affine invariance: the affine transform performs a linear mapping from coordinates system to other coordinates system that preserves the "straightness" and "parallelism" of lines. Affine transform can be constructed using sequences of translations, scales, flips, rotations and shears. The extracted features must be as invariant as possible with affine transforms.
- noise resistance: features must be as robust as possible against noise, i.e., they must be the same whichever be the strength of the noise in a given range that affects the Pattern Recognition Techniques, Technology and Applications.
- occultation invariance: when some parts of a shape are occulted by other objects, the feature of the remaining part must not change compared to the original shape.
- statistically independent: two features must be statistically independent. This represents compactness of the representation.
- reliability: as long as one deals with the same pattern, the extracted features must remain the same. In general, shape descriptor is a set of numbers that are produced to represent a given shape feature. A descriptor attempts to quantify the shape in ways that agree with human intuition (or task-specific requirements). Good retrieval accuracy requires a shape descriptor to be able to effectively find perceptually similar shapes from a database. Usually, the descriptors are in the form of a vector. Shape descriptors should meet the following requirements:

- the descriptors should be as complete as possible to represent the content of the information items.
- the descriptors should be represented and stored compactly. The size of a descriptor vector must not be too large.
- the computation of the similarity or the distance between descriptors should be simple; otherwise the execution time would be too long.

Shape feature extraction and representation plays an important role in the following categories of applications:

- shape retrieval: searching for all shapes in a typically large database of shapes that are similar to a query shape. Usually all shapes within a given distance from the query are determined or the first few shapes that have the smallest distance.
- shape recognition and classification: determining whether a given shape matches a model sufficiently, or which of representative class is the most similar.
- shape alignment and registration: transforming or translating one shape so that it best matches another shape, in whole or in part.
- shape approximation and simplification: constructing a shape with fewer elements (points, segments, triangles, etc.), so that it is still similar to the original. Many shape description and similarity measurement techniques have been developed in the past. [1][4][5]

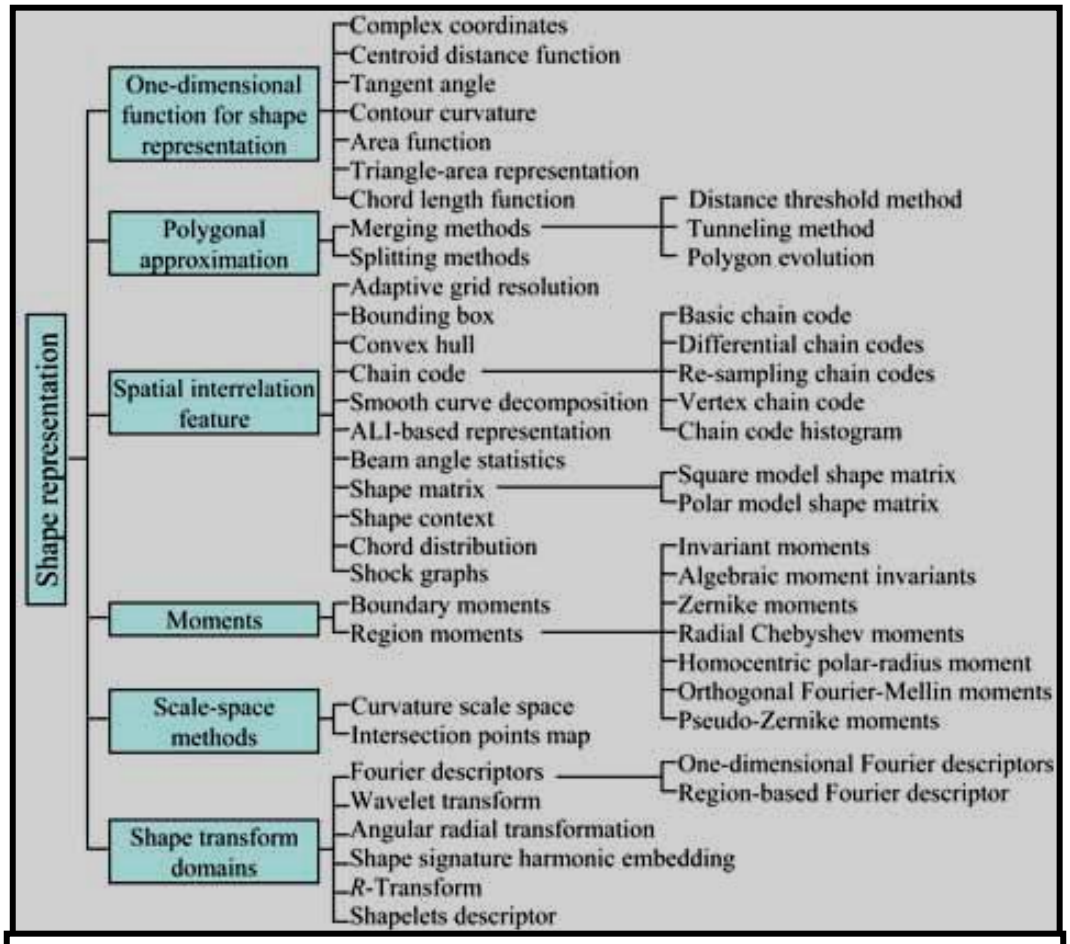


Fig. 1. An overview of shape description techniques[1]

3. Aim of the research :-

In this paper, Eigenvalues were proposed to be used for recognizing, the difference between geometrical shape. A data base was built for most known geometrical shapes, and to be used later. The same parameters which were used in the database will be input to a neural network [backprogation neural network] for recognition.

4. The Eigenvalues :-

In mathematics, eigenvalue, eigenvector, and eigenspace are related concepts in the field of linear algebra. Linear algebra studies linear transformations, which are represented by matrices acting on vectors. Eigenvalues, eigenvectors and eigenspaces are properties of a matrix. They are computed by a method described below, give important information about the matrix, and can be used in matrix factorization. They have applications in areas of applied mathematics as diverse as finance and quantum mechanics.

In general, a matrix acts on a vector by changing both its magnitude and its direction. However, a matrix may act on certain vectors by changing only their magnitude, and leaving their direction unchanged (or, possibly, reversing it). These vectors are the eigenvectors of the matrix. A matrix acts on an eigenvector by multiplying its magnitude by a factor, which is positive if its direction is unchanged and negative if its direction is reversed. This factor is the eigenvalue associated with that eigenvector. An eigenspace is the set of all eigenvectors that have the same eigenvalue. The concepts cannot be formally defined without prerequisites, including an understanding of matrices, vectors.[6]

In linear algebra, there are two kinds of objects: scalars, which are just numbers; and vectors, which can be thought of as arrows, and which have both magnitude and direction (though more precisely a vector is a member of a vector space). In place of the ordinary functions of algebra, the most important functions in linear algebra are called "linear transformations", and a linear transformation is usually given by a "matrix", an array of numbers. Thus instead of writing f(x) we write M(v) where M is a matrix and v is a vector. The rules for using a matrix to transform a vector are given in the article linear algebra.

If the action of a matrix on a (nonzero) vector changes its magnitude but not its direction, then the vector is called an eigenvector of that matrix. A vector which is "flipped" to point in the opposite direction is also considered an eigenvector. Each eigenvector is, in effect, multiplied by a scalar, called the eigenvalue corresponding to that eigenvector. The eigenspace corresponding to one eigenvalue of a given matrix is the set of all eigenvectors of the matrix with that eigenvalue.

Many kinds of mathematical objects can be treated as vectors:

ordered pairs, functions, harmonic modes, quantum states, and frequencies are examples. In these cases, the concept of direction loses its ordinary meaning, and is given an abstract definition. Even so, if this abstract direction is unchanged by a given linear transformation, the prefix "eigen" is used, as in eigenfunction, eigenmode, eigenface, eigenstate, and eigenfrequency.

When a transformation is represented by a square matrix A, the eigenvalue equation can be expressed as shown in eq.(1)

$A\mathbf{x} - \lambda I\mathbf{x} = \mathbf{0}$.....(1)

This can be rearranged to be shown in eq.(2)

$(A - \lambda I)\mathbf{x} = \mathbf{0}$.....(2)

If there exists an inverse to be shown in eq.(3)

$(A - \lambda I)^{-1}$,.....(3)

then both sides can be left multiplied by the inverse to obtain the trivial solution: x = 0. Thus we require there to be no inverse by assuming from linear algebra that the determinant equals zero:

$\det(A - \lambda I) = 0$.

The determinant requirement is called the characteristic equation (less often, secular equation) of A, and the left-hand side is called the characteristic polynomial. When expanded, this gives a polynomial equation for λ. The eigenvector x or its components are not present in the characteristic equation. [3][6]

5. Backpropagation network

Backpropagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task. It was first described by Arthur E. Bryson and Yu-Chi Ho in 1969, but it wasn't until 1986, through the work of David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams, that it gained recognition, and it led to a "renaissance" in the field of artificial neural network research.

It is a supervised learning method, and is an implementation of the Delta rule. It requires a teacher that knows, or can calculate, the desired output for any given input. It is most useful for feed-forward networks (networks that have no feedback, or simply, that have no connections that loop). The term is an abbreviation for "backwards propagation of errors". Backpropagation requires that the activation function used by the artificial neurons (or "nodes") is differentiable. [7]

5.1 Complex Problems

The field of neural networks can be thought of as being related to artificial intelligence, machine learning, parallel processing, statistics, and other fields. The attraction of neural networks is that they are best suited to solving the problems that are the most difficult to solve by traditional computational methods.

Consider an image processing task such as recognizing an everyday object projected against a background of other objects. This is a task that even a small child's brain can solve in a few tenths of a second. But building a conventional serial machine to perform as well is incredibly complex. However, that same child might NOT be capable of calculating 2+2=4, while the serial machine solves it in a few nanoseconds.

A fundamental difference between the image recognition problem and the addition problem is that the former is best solved in a parallel fashion, while simple mathematics is best done serially. Neurobiologists believe that the brain is similar to a massively parallel analog computer, containing about 10^10 simple processors which each require a few milliseconds to respond to input. With neural network technology, we can use parallel processing methods to solve some real-world problems where it is very difficult to define a conventional algorithm.[8][9]

5.2 The Feed-Forward Neural Network Model

If we consider the human brain to be the 'ultimate' neural network, then ideally we would like to build a device which imitates the brain's functions. However, because of limits in our technology, we must settle for a much simpler design. The obvious approach is to design a small electronic device which has a transfer function similar to a biological neuron, and then connect each neuron to many other neurons, using RLC networks to imitate the dendrites, axons, and synapses. This type of electronic model is still rather complex to implement, and we may have difficulty 'teaching' the network to do anything useful. Further constraints are needed to make the design more manageable. First, we change the connectivity between the neurons so that they are in distinct layers, such that each neuron in one layer is connected to every neuron in the next layer. Further, we define that signals flow only in one direction across the network, and we simplify the neuron and synapse design to behave as analog comparators being driven by the other neurons through simple resistors. We now have a feed-forward neural network model that may actually be practical to build and use.

Referring to figures 2 and 3, the network functions as follows: Each neuron receives a signal from the neurons in the previous layer, and each of those signals is multiplied by a separate weight value. The weighted inputs are summed, and passed through a limiting function which scales the output to a fixed range of values. The output of the limiter is then broadcast to all of the neurons in the next layer. So, to use the network to solve a problem, we apply the input values to the inputs of the first layer, allow the signals to propagate through the network, and read the output values.

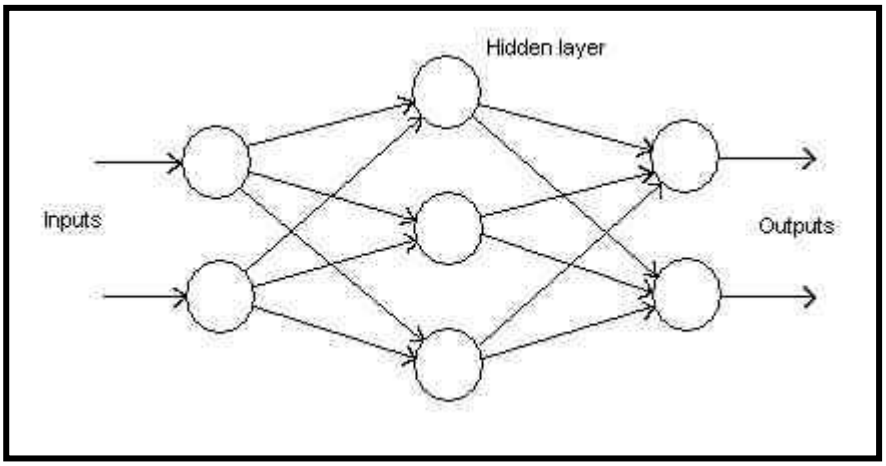


Figure 2: A Generalized Network

A generalized network can be seen in figure(2), Stimulation is applied to the inputs of the first layer, and signals propagate through the middle (hidden) layer(s) to the output layer. Each link between neurons has a unique weighting value.

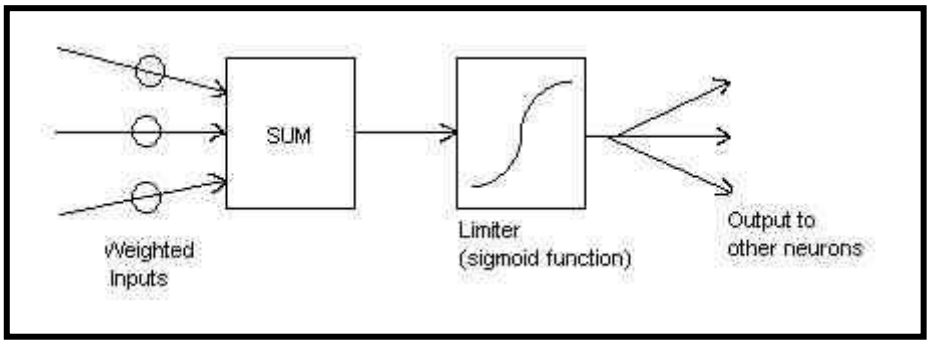


Figure 3: The Structure of a Neuron

The structure of the neuron can be seen in figure(3), Inputs from one or more previous neurons are individually weighted, then summed. The result is non-linearly scaled between 0 and +1, and the output value is passed on to the neurons in the next layer.

Since the real uniqueness or 'intelligence' of the network exists in the values of the weights between neurons, we need a method of adjusting the weights to solve a particular problem. For this type of network, the most common learning algorithm is called Back Propagation (BP). A BP network learns by example, that is, we must provide a learning set that consists of some input examples and the known-correct output for each case. So, we use these input-output examples to show the network what type of behavior is expected, and the BP algorithm allows the network to adapt.

The BP learning process works in small iterative steps: one of the example cases is applied to the network, and the network produces some output based on the current state of it's synaptic weights (initially, the

output will be random). This output is compared to the known-good output, and a mean-squared error signal is calculated. The error value is then propagated backwards through the network, and small changes are made to the weights in each layer. The weight changes are calculated to reduce the error signal for the case in question. The whole process is repeated for each of the example cases, then back to the first case again, and so on. The cycle is repeated until the overall error value drops below some pre-determined threshold. At this point we say that the network has learned the problem "well enough" - the network will never exactly learn the ideal function, but rather it will asymptotically approach the ideal function.

Summary of the backpropagation technique:

1. Present a training sample to the neural network.
2. Compare the network's output to the desired output from that sample. Calculate the error in each output neuron.
3. For each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is the local error.
4. Adjust the weights of each neuron to lower the local error.
5. Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights.
6. Repeat from step 3 on the neurons at the previous level, using each one's "blame" as its error. [8][9][10]

5.3 Research algorithm

Actual algrithm for a 3-layer network (only one hidden layer):

Initialize the weights in the network (often randomly)

Do

For each example e in the training set

O = neural-net-output(network, e) ; forward pass

T = teacher output for e

Calculate error (T - O) at the output units

Compute delta_wh for all weights from hidden layer to output layer ; backward pass

Compute delta_wi for all weights from input layer to hidden layer ;

backward pass continued

Update the weights in the network

Until all examples classified correctly or stopping criterion satisfied

Return the network.[7]

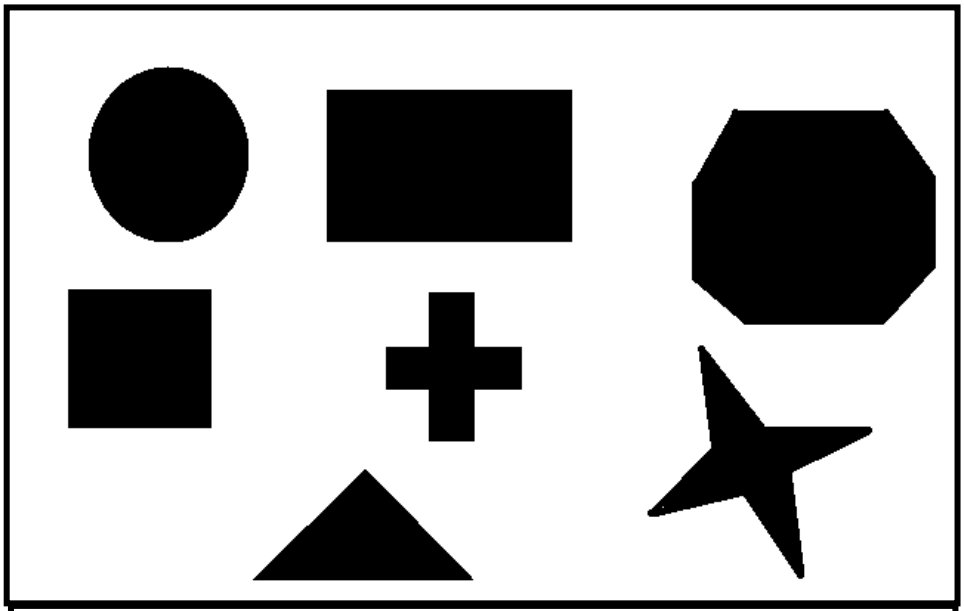
6. PROPOSED PROCEDUR

The proposed algorithm can be summarized by the following steps:-

Step-1 :
Enter the image which contains the shapes that will be recognize (circle, rectangle, triangle, polygon, star,…,etc).
Step-2 :
Scanning process to select the shape which will be entered to the system.
Step -3:
Size re-arrangement: in this task the size of the selected shape will be in different sizes, so to make a standardized size for the whole shapes used, the proposed technique
Setp-4 : eigenvalue evaluation
Compute the eigenvalues for the shape and eigenvectors
Step 5:
Make the eigenvector as inputs to a neural network(backpropgation) and this network will recognize the shape from other shapes

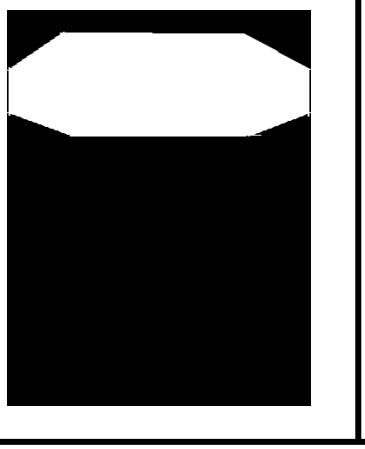
7. **Results discussion** (with sample example)

1. The studied shapes shown in figure(4) adopted to be as input. Shown shapes was ploted using paint software then scanned to be stored.



Figure(4): original image

2. After scanning process will get a shape from original image.
3. A polygon will be selected (as example).
4. After resizing the image we will eliminate all the lines in horizontal and verticals, i.e. the polygon was truncated, the process can be seen very clear in figure(5), and figure(6) so the result image of truncated process is shown in figure(7).



Figure(5): Truncated polygon

Figure(6): Truncated polygon

Figure(7): Result of truncated process

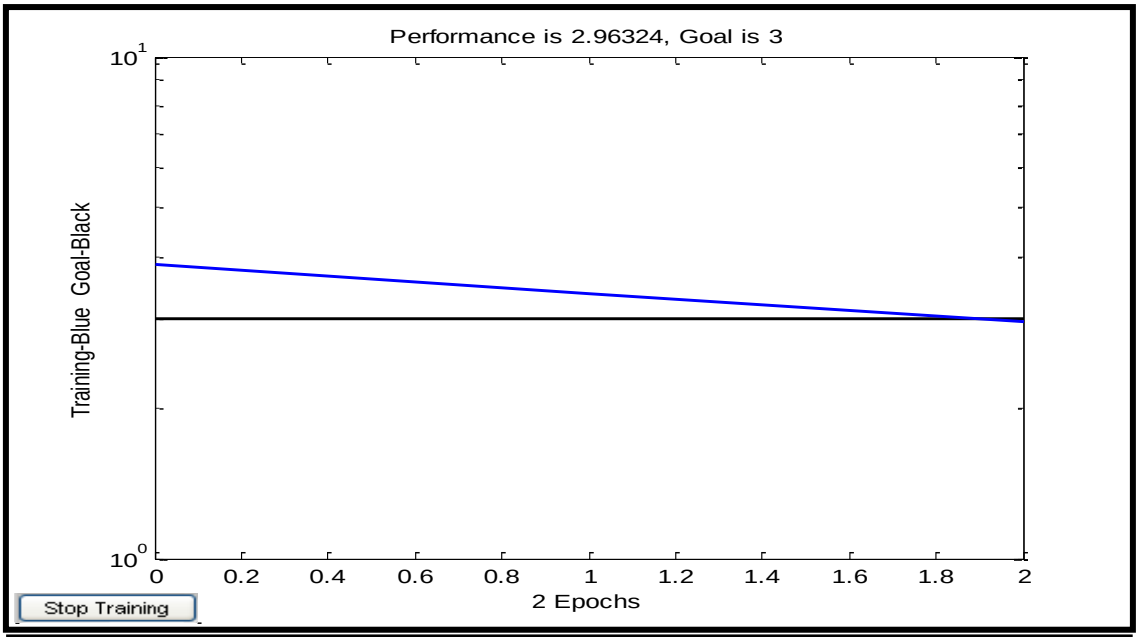
5. Compute eigenvalue for the image whose value explain in table(1), the number of eigenvalue equal to size of the image (Row*Colum).

Table(1): eigenvalue for polygon shape

sequences	Eigenvalue									
	real	Imaj	real	imaj	Real	imaj	real	Imaj	real	Imaj
1	1.0e*+002		1.1423+0		-0.2225 +0		0.0686+0		-0.0199 +0.0147i	
2	-0.0199 – 0.0147i		0.0253+0		0.0086 + 0.0186i		0.0086 - 0.0186i		-0.0064 +0.0163i	
3	-0.0064 – 0.0163i		0.0174+0		-0.0115 +0.0088i		-0.0115 - 0.0088i		-0.0124 +0.0016i	
4	-0.0124 – 0.0016i		0.0075 + 0.0095i		0.0075 - 0.0095i		0.0015 + 0.0117i		0.0015 - 0.0117i	
5	0.0103 + 0.0024i		0.0103 - 0.0024i		-0.0077 +0.0057i		-0.0077 - 0.0057i		0.0003 - 0.0085i	
. . .	0.0067 + 0.0035i		0.0067 - 0.0035i		0.0040 + 0.0062i		-0.0022 +0.0065i		-------	

6. These value used as an input to backprogation neural network and the results describe in the figure (8).

Eigenvectors adopted for geometrical shapes recognition.



Figure(8): explain result of training process

Table(2): results of applied shapes

Shape	Actual value	Target	Excution time	Epoch	Size	Error
Circle	0.906342	1	0.1090	11	1*128	0.0936578
Circle	0.980561	1	0.1250	7	1*64	0.019439
Circle	0.919055	1	0.1250	7	1*32	0.080945
Circle	0.953084	1	0.1400	3	1*50	0.046916
Trangle	1.91219	2	0.1250	1	1*128	0.08781
Trangle	1.92867	2	0.1090	6	1*64	0.07133
Trangle	1.92015	2	0.1250	2	1*32	0.07985
Trangle	1.9529	2	0.0940	2	1*50	0.0471
Polygon	2.96324	3	0.0780	2	1*128	0.03676
Polygon	2.95198	3	0.1090	4	1*64	0.04802
Polygon	2.91479	3	0.0940	4	1*32	0.08521
Polygon	2.94658	3	0.0940	2	1*50	0.05342
Rectangle	3.91088	4	0.0940	3	1*128	0.08912
Rectangle	3.97946	4	0.0780	3	1*64	0.02054
Rectangle	3.9587	4	0.0940	2	1*32	0.0413
Rectangle	3.93627	4	0.0940	2	1*50	0.06373
Cross	4.94287	5	0.1100	3	1*128	0.05713
Cross	4.98591	5	0.0930	1	1*64	0.01409
Cross	4.72595	5	0.1090	2	1*32	0.27405
Cross	4.9719	5	0.1090	1	1*50	0.0281
Star	5.95725	6	0.0907	1	1*128	0.04275
Star	5.99071	6	0.1250	1	1*64	0.00929
Star	5.81463	6	0.7080	3	1*32	0.18537
Star	5.35593	6	0.1100	1	1*50	0.64407

7. Table(2) show the output of the applied shapes on the network in addition to the error (which can be seen very low), also the applied algorithm was stable with different image size.

7. Conclusion:

- using eigenvalues as a feature for geometric shapes in digital image gave high recognition even in image containes different geometrical shapes.
- neural network (backpropagation net) will speed up the process specially when we adopt training types.
- gemoetrical shapes recognition are so important to be recognized by using mathmatical models, using eigenvalues will be new scientifical approch.
- big difference can be seen between different shapes even may they are littelbit closed to each others

Reference;

1) Yang Mingqiang1, 2, Kpalma Kidiyo1 and Ronsin Joseph1**,** **'**A Survey of Shape Feature Extraction Techniques', 1IETR-INSA, UMR-CNRS 6164, 35043Rennes, 2Shandong University, 250100, Jinan, 1France 2China.

2) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., (2009) a non-profit organization, Privacy policy About Wikipedia Disclaimers.

3) Clement Fredembach, Michael Schroder, Sabine'susstrunk, (2003), 'Eigenregions for Image Classification'.

4) www.facebook.com, (2008), 'A Survey of Shape Feature Extraction Techniques'.

5) Hung-Chun ISOMORPHIC NETWORK', Department of Electrical Engineering University of Texas at Arlington, Texas 76019. Yau Michael T. Manry,: 'SHAPE RECOGNITION WITH NEAREST NEIGHBOR.

6) Eigenvalue, eigenvector and eigenspace, From Wikipedia the freeency clopedia (2010). http://en.wikipedia.org/wiki/Eigenvalue,_eigenvector_and_eigenspace.[internet]

7) Grapy, Daniel, (2007), 'principle of artificial neural network', 2nd addition, publish by world scientific publishing, Co.Pte.ltd.

8) Christos Stergiou, Dimitrios Siganos, 'NEURAL NETWORKS', http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html.[internet]

9) Rao, valluru B., (2006), 'c++ neural network and fuzzy logic', MM&tbox, IBTIDG_Box-world wide, Inc.isbn:1558515526

10) 'Backpropagation Neural Network', Copyrights© 2007-2008.