Mansoura Engineering Journal

Volume 19 | Issue 4 Article 7

12-1-2021

Object Matching by Image Contours Using Neural Networks.

A. El-Shami
Department of Mathematics., University of Suez Canal., Egypt.

H. Niemann
Department of Computer Science., University of Erlangen., Germany.

A. Tolba
Department of Electrical Engineering., University of Suez Canal., Egypt.

Recommended Citation

El-Shami, A.; Niemann, H.; and Tolba, A. (2021) "Object Matching by Image Contours Using Neural Networks.," *Mansoura Engineering Journal*: Vol. 19 : Iss. 4 , Article 7.
Available at: https://doi.org/10.21608/bfemu.2021.164258

Mansoura Engineering Journal (MEJ) Vol. 19, No. 4, December, 1994.

Object Matching by Image Contours using Neural Networks

مضاهاة الأشكال لحدود الصور باستخدام الشبكات العصبية

A. El-Shami,[1] H. Niemann,[2] H.Hawidi[1] and A. Tolba[3]

1 Department of Mathematics, Uni. of Suez Canal, Egypt,
2 Department of Computer Sci., Uni. of Erlangen, Germany,
3 Department of Electrical Engineering, Uni. of Suez Canal, Egypt.

ملخص البحث:

يتناول البحث بناء نظام لمضاهاة الصور ثلاثية وثنائية الأبعاد بناءا على الصور ثنائية الأبعاد. أولا تحول الصورة الرمادية إلى صورة ثنائية بطريقة ديناميكية . يتم التعرف على الشكل بواسطة الخطوط والمنحنيات التى يتم تحديدها بواسطة تحويل هوف (Hough Transform) وكود السلسلة (Chain-Code) والتى تمثل المدخلات إلى مصنف الشبكة العصبية التى يتم تدريبها للتعرف على الصور بواسطة طريقة الإنتشار الخلفى. لحل مشكلة اختلاف عدد السمات المستخلصة للأشكال المختلفة وما يترتب عليها من مشكلة إختيار عدد النيورونات فى طبقة الإدخال فى المصنف العصبى يتم تكرار الخطوط والمنحنيات فى نفس المواضع لزيادة عددها إلى حد معين. أختبرت الطريقة المقدمة على عدد ٣٦٠ شكل حقيقى حيث قامت الشبكة العصبية بالتعرف على جميع الأشكال التى تعرضت لها سابقا رغم عدم الإكتمال أو الإزاحة أو الدوران.

Abstract This paper deals with the implementation of 3-d object matching system. The shape of the object is identified from the image lines and curves using Hough transform, chain code and backpropagation neural networks. This is achieved by first dynamically thresholding the grey level image, then segmenting the image into its linear components with both Hough transform and chain coding. A backpropagation framework is used for classifying the image into one of possible surfaces based on the extracted vertices and line segments. To fix the number of input layer neurons, the image features are normalized. The approach is tried on a variety of real objects and appears to hold great promise.

Index Terms - 3-D Object recognition, shape matching, chain code, Hough transform, surface classification, neural networks.

1. Introduction

Research and development in computer vision has increased dramatically over the last thirty years. Application areas that have been extensively studied include character recognition, medical diagnosis, traget detection and remote sensing. Recently , machine vision for automating the manufacturing process have received considerable attention with the growing interest in robotics. Although some commercial vision systems for robotics and industrial automation do exist, their capabilities are still very primitive [1].

Object recognition approaches in computer vision can conceptually be classified into two categories. The first, or traditional approach, involves the use of statistical and structural techniques. Over the years this approach, by itself, has proven to be inadequate in handling some of the more difficult real world problems where noise and improper illumination exist, and the problem domain has not been constrained to well-defined geometric objects, The second approach attempts to overcome these problems in much the same way a human does, through the use of contextual information, experience, or expert knowledge. The advantages derived from the second approach, however, typically come at the expense of speed [2] .

The inference of the shape of a 3-D object from its image (or images) has been the concern of many researchers for the past two decades. Many researchers have considered inferring shape by considering special contours on the surface [1-3] or by considering distribution of the contours of constant image intensity as a function of 3-D surface shape [4]. Shape parameters can also be calculated from regular patterns on the surface.

2. System Definition

2.1. System Hardware

The image processing system used in this research was implemented on HP-workstation in a UNIX environment. The system incorporates several sets of high-level processing stages. Figure 1 displays the general processing steps performed on each object in order to accurately identify it. As indicated in the figure, matching was performed after various stages of processing.

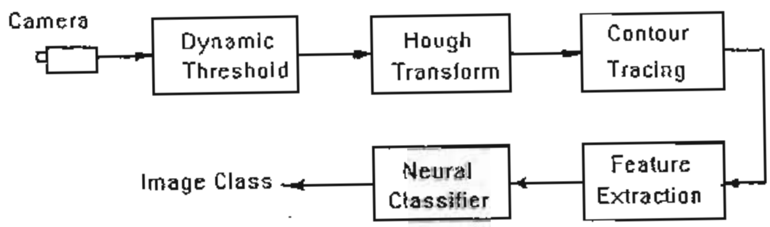

Figure 1. Basic processing flow

2.2. Object models, features and matching [1]:

An object recognition system can be broken down into a training phase and a classification phase. The three major components of the system are feature extraction, object modelling, and matching. The use of models for image understanding has been studied extensively. However, most of the models that have been investigated are relatively simple and do not provide adequate descriptions for recognizing complex scenes. Models based on geometric properties of an object's visible surface or silhouette are commonly used because they describe objects in terms of their constituent shape features.

The problem of selecting the geometric features that are components of the model is integrally related to the problem of model definition. Image features such as edge, corner, line, curve and boundary curvature define individual feature components of an image. These features and their spatial relations are then combined to generate object descriptions.

Given a set of models that describe all aspects of all objects to be recognized, the process of model-based recognition consists of matching features extracted from a given input image with those of the models. The general problem of matching may be regarded as finding a set of features in the given image that approximately matches one model's features.

Matching techniques using 2-D global, local, or relational features, provide a way to recognize and locate parts on the basis of a few key features. Matching using features becomes a model-driven process in which model features control the matching process. In this paper we present a matching process which is invariant to translation and rotation, and is not sensitive to noise and image distortion. Our matching is a syntactic matching based on global features . Recognition is based on template matching between the model edge template and the edge image in the generalized Hough transform space.

For the matching process we need to use features that are invariant to scaling, translation and rotation. One of such features are the line segments forming the object shape which are obtained as described in Section 3. The verticies of line segments are inputted to the neural network after normalization for either training or classification. These features have a compression property that is desirable for managable training of the NN.

2.3. Dynamic Image Thresholding:

The grey-level image is dynamically first thresholded to generate a binary image [5]. This process helps in reducing image noise. Since the real images could not be guaranteed to be homogeneously illuminated it was necessary to implement a dynamic thresholding scheme. The suitable threshold limit is locally determined from the neighbouring pixels. The mean value of the intensities of a window is computed. The intensity of each grey level pixel is then scaled to the 0 to 1 interval.

3. Image Segmentation:

The shape of an object can be described either in terms of its boundary or in terms of the region it occupies. Shape representation based on boundary information requires image edge detection and following. Region-based shape representation requires image segmentation in several homogenious regions. Thus, edge detection and region segmentation are dual approaches in image analysis.

3.1. Line and curve detection using Hough Transform:

A binary image must be further processed to produce more useful information that can be used in the detection of simple shapes (e.g. straight lines, curves) or arbitrary shaped objects. In this paper, we shall describe a well known method for the detection of parametric curves in general and for the detection of straight lines in particular. Let us suppose that we search for straight lines on a binary image. The simplest approach is to find all possible lines determined by pairs of pixels and to check if subsets of the binary image pixels belong to any of these lines. The Hough transform uses a parametric description of simple geometrical shapes (curves) in order to reduce the computational complexity of their search in a binary image. The parametric description of a line is given by a linear equation $y=ax+b$. Each line is represented by a single point in the parameter space (a,b). For every pixel that possess value 1 at the binary edge detector output, the corresponding line equation is formed. The appropriate parameter matrix elements $P(a,b)$ are computed. Each parameter matrix element has the number of binary edge detector output pixels that satisfy the linear equation of a line. If this number is above a certain threshold a line is declared [3].

3.2. Contour Tracing Using Chain Code:

After detecting image edges and lines a contour tracer is the primary source of information for the recognizer to work with. The process of tracing the contour segments the image and makes it possible to locate several distinct objects in a single scene. Short gaps are closed and short lines are eliminated. Vertices are then extracted. The chain code technquie is used for contour tracing [4].

3.3. Image normalization:

For the recognition system to be invariant against translation, zoom and rotation, the object image is rotated around its major axis within a fixed window size [6].

3.4 Region closing and elimination of short lines:

Open regions are closed and short line segments are eliminated to eliminate the effects of noise and nonuniform illumination.

3.5 Normalization of segmentation results:

We are given a set of coordinates of both start and end points of every line segment. Unfortunately, the number of coordinates is different from one image to the other. To hold the number of features fixed for all images and to keep the number of input neurons constant, line segments are repeated over and again in the same position such that the number of vertices is increased to be suitable for the input layer of the neural network and becomes independent of the object shape. We implemented a special algorithm for solving this problem. The algorithm concept depends on redrawing the line segments for the image which have a specific number of lines many times in the same position. The final image is the same as the original one but the number of vertices became suitable for use in the input layer of the neural network. Figures 2 a,b show the result of applying this algorithm on an original image of 512x512 pixels size (Fig. 2-a).

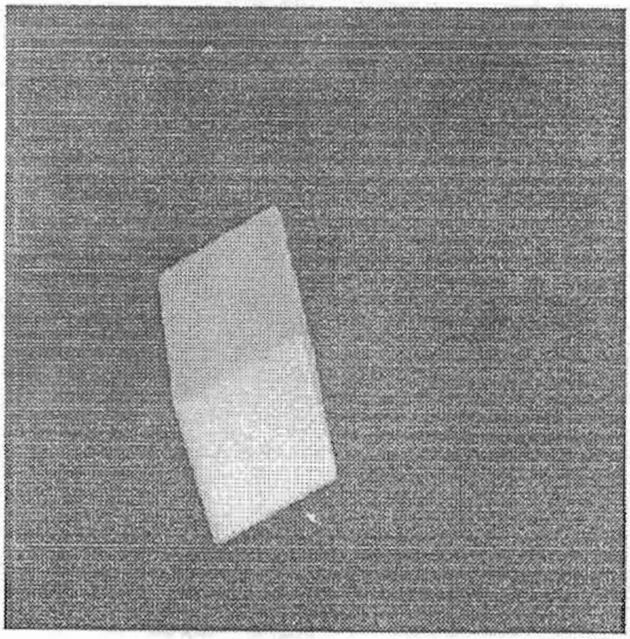

Figure 2: (a) Original Image of size 512x512 pixels

Figure 2-b shows the segmented image after the application of the normalization process. The algorithm of the normalization process is as follows:

Normalization algorithm:

```
While not end of image-base do
      read object
      get one line segment
      store the number of segment  indices
compute the maximum number of lines in segmented object (max)
While not end of image-base segmentation do
      read segmented object
      While number of line segments of the object  < max do
            redraw the same line segment
      store new segmented object to use in Neural Network Classifier
```

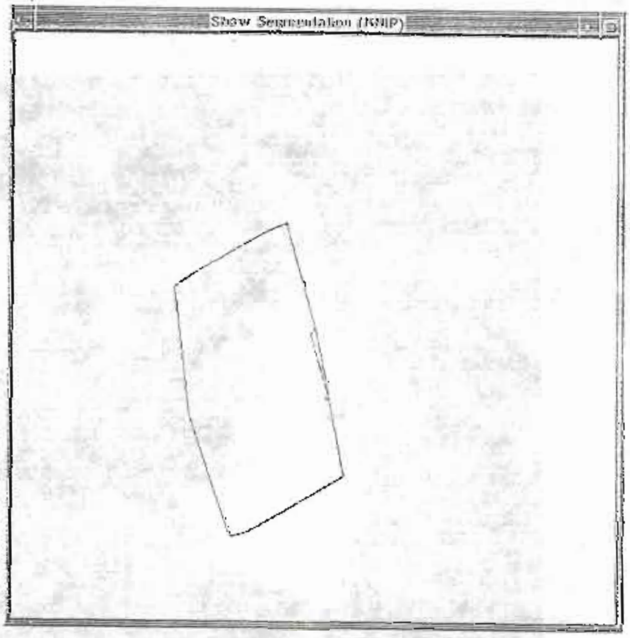

Figure 2: (b) Result of normalized segmentation

Tables 1-a, 1-b show the results of application of the normalized segmentation on the object shown in Figure 2.

x1	y1	x2	y2
265	310	251	261
183	438	197	434
197	434	280	384
267	312	265	310
230	165	209	174
155	206	135	220
209	174	155	206
135	220	148	332
148	332	183	438
251	261	264	317
267	312	255	255
267	317	264	317
267	312	255	255
267	317	267	312
264	317	265	310
255	255	230	165
280	384	267	317

(a)

x1	y1	x2	y2
265	310	251	261
183	438	197	434
197	434	280	384
267	312	265	310
230	165	209	174
155	206	135	220
209	174	155	206
135	220	148	332
148	332	183	438
251	261	264	317
267	312	255	255
267	317	264	317
267	312	255	255
267	317	267	312
264	317	265	310
255	255	230	165
280	384	267	317
..	..	..	..
..	..	..	..
265	310	251	261
183	438	197	434
197	434	280	384
267	312	265	310
230	165	209	174
155	206	135	220
209	174	155	206
135	220	148	332
148	332	183	438
251	261	264	317
267	312	255	255
267	317	264	317
267	312	255	255
267	317	267	312
264	317	265	310
255	255	230	165
280	384	267	317

(b)

4. Neural Network Models for Pattern Matching:

Pattern recognition requires a number of distinct steps. Noise removal, edge enhancement, segmentation, feature extraction and classification. Neural network algorithms for most of these steps have been suggested. Neural networks are at present the unique device, as far as we know, capable of tackling complex visual image recognition [7] . We can say that, the neural network is sufficient and a powerful tool for classifying a set of images in a specific time.

Image understanding requires the segmentation of an image into individual objects. These objects are then subject to feature extraction and classification. A neural network serves in this case as a classifier.

Neural networks could be applied for recognition of a whole image frame in applications such image matching (for example face identification) in image bases. The photograph of a person is first acquired, and input to the trained neural network for classification. This problem could be solved by training a neural network to learn the grey levels of the given image instead of using a large number of neurons in the input layer. In this case the input layer includes only 256 neurons. The hidden layer includes 16 neurons. The output layer includes a number of neurons equal to the number of image classes. A three layer backpropagation network performs the learning and classification tasks together. Figure 3 shows the response of the neural network by learning the grey levels of the input image for different numbers of iterations. The network gets some difficulties in learning the marginal grey levels due to the asymptotic nature of the sigmoidal function.

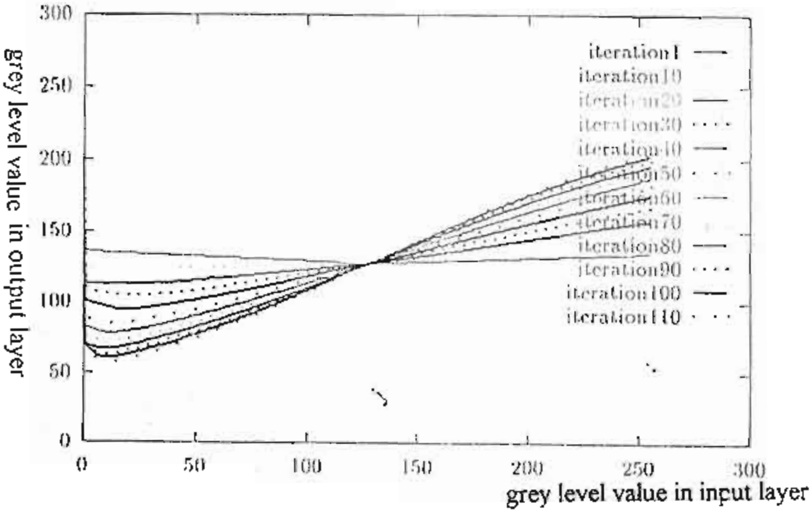

Fig. 3 Learning the image grey levels for different numbers of iterations

Figure 4-c Learned image for iterations 100-110

The resulting error values for different numbers of iterations are shown in Fig.5 .

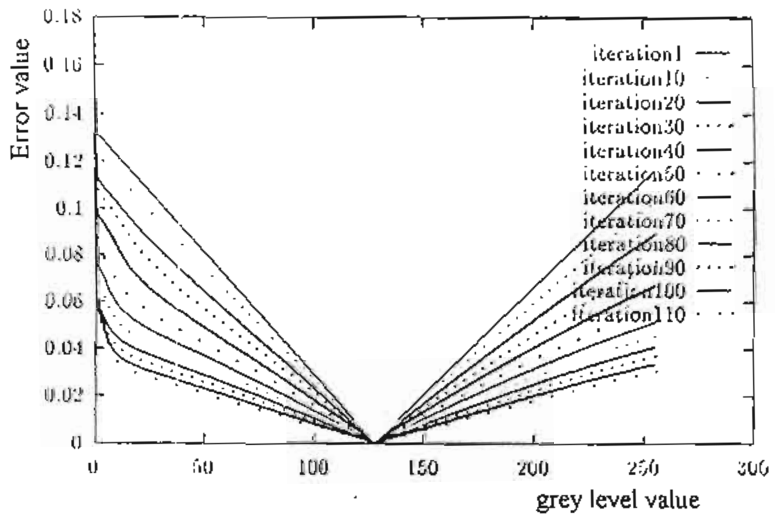

Fig. 5 Error values for iteration range from 1 to 110

The effect of the learning rate on the output grey levels at a fixed number of iterations (256 in this case) is shown in figure 6.

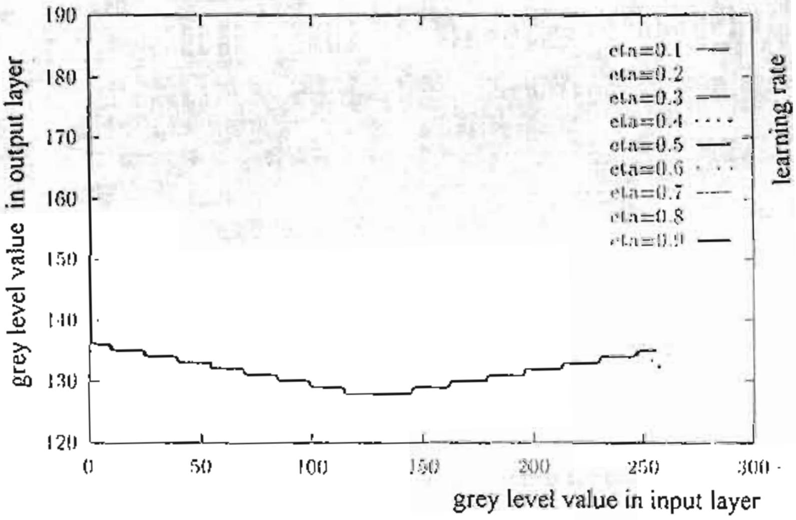

Fig. 6 Effect of learning rate on grey levels

Figure 4 shows an original image of size 512x512 pixels and the learned images after specific numbers of iterations.

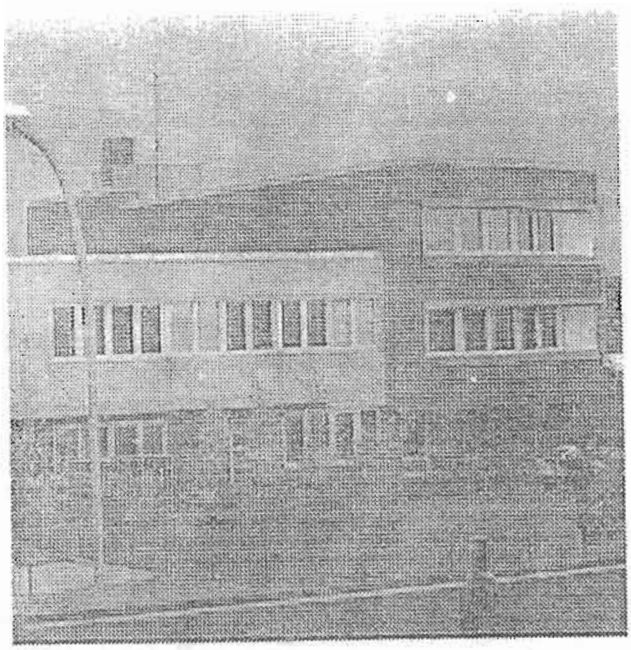

Figure 4-a Original image of size 512 x 512 pixels

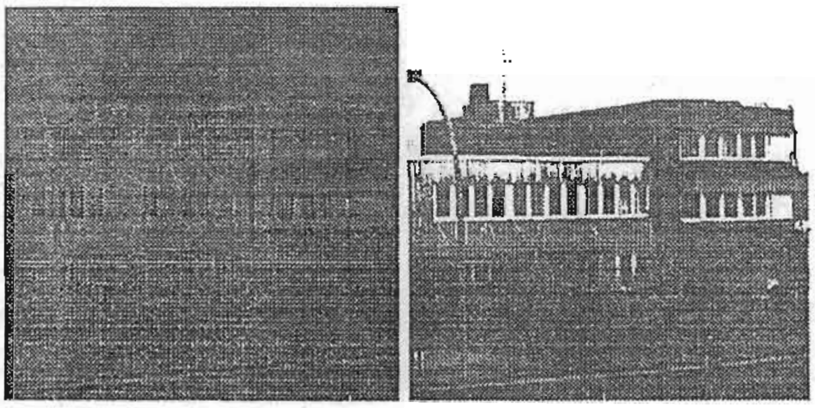

Fig. 4-b Learned image after 1-10 iterations

Table 2 shows the convergence of the neural network by learning the grey levels and its relation to the number of training iterations.

Input	*Output at different iterations*										
Grey Value	1	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
101	128	101	101	101	101	101	101	101	101	101	101
102	128	102	102	102	102	102	102	102	102	102	102
103	128	103	103	103	103	103	103	103	103	103	103
104	128	104	104	104	104	104	104	104	104	104	104
105	128	105	105	105	105	105	105	105	105	105	105
106	128	106	106	106	106	106	106	106	106	106	106
107	128	107	107	107	107	107	107	107	107	107	107
108	128	108	108	108	108	108	108	108	108	108	108
109	128	109	109	109	109	109	109	109	109	109	109
110	128	110	110	110	110	110	110	110	110	110	110
111	128	111	111	111	111	111	111	111	111	111	111
112	128	112	112	112	112	112	112	112	112	112	112
113	128	113	113	113	113	113	113	113	113	113	113
114	128	114	114	114	114	114	114	114	114	114	114
115	128	115	115	115	115	115	115	115	115	115	115
116	128	116	116	116	116	116	116	116	116	116	116
117	128	117	117	117	117	117	117	117	117	117	117
118	128	118	118	118	118	118	118	118	118	118	118
119	128	119	119	119	119	119	119	119	119	119	119
120	128	120	120	120	120	120	120	120	120	120	120
121	128	121	121	121	121	121	121	121	121	121	121
122	128	122	122	122	122	122	122	122	122	122	122
123	128	123	123	123	123	123	123	123	123	123	123
124	128	124	124	124	124	124	124	124	124	124	124
125	128	125	125	125	125	125	125	125	125	125	125
126	128	126	126	126	126	126	126	126	126	126	126
127	128	127	127	127	127	127	127	127	127	127	127
128	128	128	128	128	128	128	128	128	128	128	128
129	128	128	129	129	129	129	129	129	129	129	129
130	128	129	130	130	130	130	130	130	130	130	130
131	128	130	130	131	131	131	131	131	131	131	131
132	128	131	131	131	131	131	131	131	131	131	131
133	128	132	132	132	132	132	132	132	132	132	132
134	128	133	133	133	133	133	133	133	133	133	133
135	128	134	134	134	134	134	134	134	134	134	134
136	128	135	135	135	135	135	135	135	135	135	135
137	128	136	136	136	136	136	136	136	136	136	136
138	128	137	137	137	137	137	137	137	137	137	137
139	128	138	138	138	138	138	138	138	138	138	138
140	128	139	139	139	139	139	139	139	139	139	139
141	128	140	140	140	140	140	140	140	140	140	140
142	128	141	141	141	141	141	141	141	141	141	141
143	128	142	142	142	142	142	142	142	142	142	142
144	128	143	143	143	143	143	143	143	143	143	143
145	128	144	144	144	144	144	144	144	144	144	144
146	128	145	145	145	145	145	145	145	145	145	145
147	128	146	146	146	146	146	146	146	146	146	146
148	128	147	147	147	147	147	147	147	147	147	147
149	128	148	148	148	148	148	148	148	148	148	148
150	128	149	149	149	149	149	149	149	149	149	149

5. Neural Network Classifier:

Neural net architectures can be used to construct many different types of classifiers. The multilayer perceptron architecture [7] is currently the most widely applied NN to learn complicated mappings. It attempts to map an input pattern to a desired output pattern using a set of connecting weights and nonlinear mapping functions. In this paper, we implemented a three layer perceptron. It includes N input neurons, H hidden neurons and K output neurons. There is a bias unit connected to both the hidden and output layers.

6. Experiments:

The mathcing system was trained with 360 training objects belonging to 12 different classes each including 30 different objects. The output of the neural classifier is given below in table 3 for different numbers of learning iterations . The features extracted from the segmentation process of each object (represented by 160 lines) are the starting and end coordinates of each line. A neural network with 640 input neurons (160x4), 15 hidden layer units, and 12 output neurons is trained with 360 objects with the backpropagation technique. The network trained the objects in 4.5 minutes. After training of the network a test set including both objects from the training set and new objects is introduced to the network. Some objects are incomplete. Most of the tested objects are correctly recognized. Most of the noisy images are classified with minimum false rate. Table 4 shows the results for classification of 300 objects belonging to twelve different classes. The response of the output neurons to the different classes ranges from zero to one. The network have some problems to classify not learned images. The rotation and translation of the object does not affect the recognition results.

7. Conclusions

Neural net architecture's form a flexible framework that can be used to construct efficient image recognition systems. A three layer back propagation neural network is designed for recognition of 2-d and 3-d objects. The performance of the system is evaluated on the hand of a large test set including 360 objects of different types, sizes and positions. The recognition results presented show that high speed object recognition can be performed with a neural network based system.

8. References

[1] Roland Chin, and Charles Dyer
"Model-based recognition in Robot-Vision", in Computer Surveys, Vol.18, No. 1, March 1986.

[2] Bir Bhanu et. al
"Recognition of 3-D objects in range images using a butterfly micro processor", Pattern Recognition, Vol 22, No. 1, pp. 49-64, 1989

[3] H. Niemann
"Pattern Analysis and Understanding", Springer, Heidelberg, Germany, 1990.

[4] D. Paulus
"Object Oriented Image Segmentation" IEEE Proc. of the 4th Int. Conf. on Image Processing and its Applications pp.482-485, Maastrich, Holland, 1992.

[5] A. Tolba, M. Elshahat , et al.
"A machine vision system for measurement of biological shapes", in the proceedings of :ISPRS Symposium, "Close-Range Photogrammetry Meets Machine Vision", ETH Zurich-Switzerland, September 3-7, 1990.

[6] M. Hamid, A. Tolba, and H. Elhindy
"A General Technique For Recognition of Planar Shapes", in Mansoura Engineering Journal (MEJ), Vol.18, No. 3, September, 1993.

[7] Ritter. H ; Martinetz. T and Schulten. K
"Neural Computation and Self-Organizing Maps", Addison Wesley Publishing Company, 1992.

Table 3 Case learning after different numbers of learning iterations

Image	*Output with Different Classes*											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	*0.885*	0.002	0.045	0.005	0.000	0.000	0.046	0.054	0.001	0.001	0.051	0.017
2	0.000	*0.977*	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
3	0.049	0.001	*0.946*	0.019	0.000	0.000	0.015	0.014	0.007	0.000	0.004	0.009
4	0.000	0.006	0.027	*0.967*	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
5	0.000	0.010	0.001	0.002	*0.905*	0.048	0.049	0.000	0.013	0.001	0.048	0.002
6	0.000	0.008	0.000	0.002	0.050	*0.942*	0.012	0.000	0.008	0.028	0.002	0.012
7	0.045	0.006	0.009	0.001	0.053	0.010	*0.934*	0.011	0.008	0.000	0.000	0.005
8	0.046	0.007	0.010	0.001	0.000	0.000	0.010	*0.934*	0.011	0.033	0.004	0.007
9	0.000	0.013	0.010	0.020	0.025	0.004	0.005	0.019	*0.967*	0.017	0.000	0.000
10	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.035	0.012	*0.948*	0.037	0.015
11	0.052	0.003	0.003	0.015	0.053	0.002	0.000	0.001	0.002	0.017	*0.921*	0.005
12	0.032	0.002	0.014	0.005	0.000	0.027	0.010	0.006	0.002	0.025	0.004	*0.970*

Case learning with iteration 3000 at error= 0.051

Table 3 (continuation) : Case learning after different numbers of learning iterations

Image	*Output with Different Classes*											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	0.003	0.003	0.028	0.020	0.041	0.015	0.005	0.002	0.004	0.004	0.012	0.001
2	0.000	0.088	0.009	0.017	0.045	0.014	0.005	0.001	0.004	0.004	0.011	0.000
3	0.002	0.002	0.047	0.019	0.041	0.014	0.006	0.001	0.005	0.005	0.009	0.001
4	0.001	0.004	0.032	0.042	0.050	0.018	0.009	0.001	0.006	0.004	0.009	0.001
5	0.001	0.006	0.011	0.018	0.024	0.008	0.003	0.001	0.002	0.003	0.008	0.000
6	0.001	0.007	0.009	0.017	0.024	0.010	0.003	0.001	0.002	0.003	0.010	0.000
7	0.002	0.005	0.015	0.018	0.031	0.011	0.004	0.002	0.003	0.004	0.011	0.000
8	0.002	0.005	0.019	0.011	0.036	0.011	0.003	0.001	0.004	0.003	0.012	0.001
9	0.000	0.008	0.017	0.022	0.033	0.009	0.006	0.001	0.004	0.004	0.013	0.000
10	0.000	0.006	0.011	0.010	0.028	0.010	0.002	0.001	0.003	0.002	0.011	0.000
11	0.001	0.003	0.021	0.019	0.032	0.010	0.005	0.001	0.003	0.003	0.009	0.001
12	0.014	0.009	0.080	0.039	0.097	0.067	0.016	0.012	0.022	0.021	0.053	0.005

Case learning with iteration 3000 at error= 0.073

Image	*Output with Different Classes*											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
2	0.000	*0.977*	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
3	0.049	0.001	*0.946*	0.019	0.000	0.000	0.015	0.014	0.007	0.000	0.004	0.009
4	0.000	0.006	0.027	*0.968*	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
5	0.000	0.010	0.001	0.002	*0.906*	0.048	0.049	0.000	0.013	0.001	0.047	0.002
6	0.000	0.008	0.000	0.002	0.049	*0.934*	0.012	0.000	0.008	0.027	0.002	0.011
7	0.045	0.006	0.009	0.001	0.053	0.010	*0.934*	0.011	0.008	0.000	0.000	0.005
8	0.046	0.007	0.010	0.001	0.000	0.000	0.010	*0.935*	0.011	0.033	0.003	0.007
9	0.000	0.013	0.010	0.020	0.025	0.004	0.005	0.019	*0.967*	0.017	0.000	0.000
10	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	*0.948*	0.036	0.015
11	0.052	0.003	0.003	0.015	0.053	0.002	0.000	0.001	0.002	0.017	*0.921*	0.005
12	0.032	0.002	0.014	0.005	0.000	0.027	0.010	0.006	0.002	0.025	0.004	*0.970*

Case learning with iteration 3300 at error= 0.050

Table 4: Output of the neural network classifier for examples of the test objects

Image	*Output with Different Classes*											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
2	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
3	0.049	0.001	0.946	0.019	0.000	0.000	0.015	0.014	0.007	0.000	0.004	0.009
4	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
5	0.000	0.010	0.001	0.002	0.906	0.048	0.049	0.000	0.013	0.001	0.047	0.002
6	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
7	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
8	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
9	0.000	0.013	0.010	0.020	0.025	0.004	0.005	[illegible]	0.967	0.017	0.000	0.000
10	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
11	0.052	0.003	0.003	0.015	0.053	0.002	0.000	0.001	0.002	0.017	0.921	0.005
12	0.032	0.002	0.014	0.005	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
13	0.886	0.007	0.001	0.715	0.000	0.001	0.028	0.021	0.002	0.001	0.002	0.010
14	0.792	0.007	0.001	0.715	0.000	0.001	0.028	0.021	0.002	0.001	0.002	0.010
15	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
16	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
17	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
18	0.000	0.003	0.787	0.002	0.000	0.000	0.003	0.778	0.074	0.014	0.000	0.004
19	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
20	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
21	0.883	0.002	0.044	0.005	0.000	0.000	0.045	0.053	0.001	0.001	0.049	[illegible]
22	0.052	0.003	0.003	0.015	0.053	0.002	0.000	0.001	0.002	0.017	0.921	[illegible]
23	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	[illegible]
24	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
25	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
26	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
27	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
28	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
29	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
30	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
31	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	[illegible]
32	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	[illegible]	[illegible]
33	0.886	0.002	0.045	0.005	0.000	0.000	0.045	[illegible]	0.001	0.001	0.050	[illegible]
34	0.005	0.005	0.003	0.000	0.000	0.005	[illegible]	[illegible]	0.007	[illegible]	[illegible]	[illegible]
35	0.000	0.008	0.000	0.002	0.049	0.943	[illegible]	[illegible]	0.008	[illegible]	[illegible]	[illegible]
36	0.046	0.007	0.010	0.001	0.000	0.000	[illegible]	[illegible]	0.011	[illegible]	0.003	[illegible]
37	0.886	0.002	0.045	0.005	0.000	[illegible]	[illegible]	[illegible]	0.001	[illegible]	0.050	0.016
38	0.000	0.033	0.023	0.143	0.000	0.000	[illegible]	0.027	0.010	[illegible]	0.004	0.001
39	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
40	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
41	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
42	0.000	0.033	0.023	0.143	0.000	0.000	0.000	0.027	0.010	0.053	0.004	0.001
43	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
44	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
45	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
46	0.000	0.008	0.119	0.012	0.000	0.000	0.000	0.063	0.071	0.141	0.032	0.001
47	0.000	0.008	0.000	0.002	0.049	0.943	0.012	0.000	0.008	0.027	0.002	0.011
48	0.046	0.007	0.010	0.001	0.000	0.000	0.010	0.935	0.011	0.033	0.003	0.007
49	0.886	0.002	0.045	0.005	0.000	0.000	0.045	0.054	0.001	0.001	0.050	0.016
50	0.000	0.184	0.157	0.052	0.000	0.003	0.000	0.111	0.004	0.193	0.000	0.009

Table 4 (continuation): Output of the neural network classifier for examples of the test objects

Image	*Output with Different Classes*											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
251	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
252	0.000	0.010	0.000	0.01	0.000	0.026	0.000	0.03	0.011	0.948	0.036	0.015
253	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
254	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
255	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
256	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
257	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
258	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
259	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
260	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
261	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
262	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
263	0.015	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
264	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
265	[illegible]	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
266	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
267	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
268	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
269	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
270	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
271	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
272	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
273	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
274	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
275	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
276	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
277	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
278	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
279	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
280	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
281	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
282	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
283	0.045	0.006	0.009	0.001	0.053	0.010	0.000	0.034	0.011	0.948	0.036	0.015
284	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
285	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
286	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
287	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
288	0.000	0.010	0.000	0.01	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
289	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
290	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
291	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
292	0.00	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
293	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
294	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
295	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
296	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015
297	0.000	0.977	0.000	0.009	0.028	0.008	0.006	0.017	0.012	0.016	0.001	0.000
298	0.000	0.006	0.027	0.968	0.001	0.001	0.000	0.000	0.023	0.003	0.028	0.002
299	0.045	0.006	0.009	0.001	0.053	0.010	0.934	0.011	0.008	0.000	0.000	0.005
300	0.000	0.010	0.000	0.001	0.000	0.026	0.000	0.034	0.011	0.948	0.036	0.015