TR-93-002

Bar-Code Recognition System Using Backpropagation Neural Networks

Shu-Jen Liu, Hong-Yuan Liao,
Hsiao-Rong Tyan, and Jun-Wei Hsieh

Bar–Code Recognition System Using Backpropagation Neural Networks

Shu–Jen Liu†, Hong–Yuan Liao†, Hsiao–Rong Tyan♣, and Jun–Wei Hsieh‡

†Institute of Information Science
Academia Sinica, Nan–Kang
Taipei, Taiwan
TEL: 886–2–788–3799
FAX: 886–2–782–4814
E–mail : liao@iis.sinica.edu.tw

♣Department of Computer Science and Information Engineering
Chung–Yuan Christian University

‡Institute of Computer Science and Electronic Engineering
National Central University

Abstract

In this paper, a bar code recognition system using neural networks is proposed. It is well known that in many stores the laser bar code reader is adopted at check–out counters. However, there is a major constraint when this tool is used. That is, unlike traditional camera–based picturing, the distance between the laser reader (sensor) and the target object is close to zero when the reader is applied. This may result in inconvenience in store automation because human operator has to take care of either the sensor or the objects (or both). For the purpose of store automation, human operator has to be removed from the process, i.e., a robot with visual capability requires to play an important role in such system. In this paper, we propose a camera–based bar code recognition system using backpropagation neural networks. The ultimate goal of this approach is to use camera instead of laser reader such that store automation can be achieved. There are a number of steps involved in the proposed system. The first step the system has to perform is to locate the position and orientation of the bar code in the acquired image. Secondly, the proposed system has to segment the bar code. Finally, we use a trained backpropagation neural network to perform bar code recognition task. Experiments have been conducted to corroborate the proposed method.

1. Introduction

With the advent of store automation, the existence of standard bar code for information exchange is indispensable. In many point of sales, the laser bar code reader is used in convention. However, there is a major constraint on the laser bar code reader. That is, unlike traditional camera–based picturing, the distance between the laser reader and the object is normally close to zero when this tool is applied. This may result in inconvenience because human operator must take care of either the sensor or the object. For the purpose of store automation, human operator has to be removed from the process.

Due to the effects of lens distortion and noises, the acquired bar code image may deviate from the ideal one more or less. However, if the deviation is not significant, the recognition system should still be able to deal with it. Having the capability of recovering correct result from partial evidence, the Back–Propagation Neural Network (BPNN) [1–4] is thus selected as a tool to perform the recognition task. The proposed system starts with performing some preprocessing procedures to reduce the size of the raw image into a reasonable one. Then, segmentation process is performed so that the bar code is located. There are a number of segmentation algorithms available in the literature [8–12]. Under the requirement that the operation speed should be as fast as possible, we devise a heuristic segmentation technique to locate the bar code. After the position and orientation (pose) of the bar code is located, we traverse along the perpendicular direction of the bar code stripes. Those active pixels that coincide with the direction of bar code stripes are counted column by column. When the number of active pixels in a column is greater than a threshold, then a '1' is reported, otherwise '0'. A sequence of binary numbers is therefore generated. The binary sequence is then fed into a trained BPNN for final identification. If the output of the BPNN is less than a preset threshold, the candidate is rejected, otherwise the correct bar code is reported. Figure 1 shows the block diagram of the proposed system.

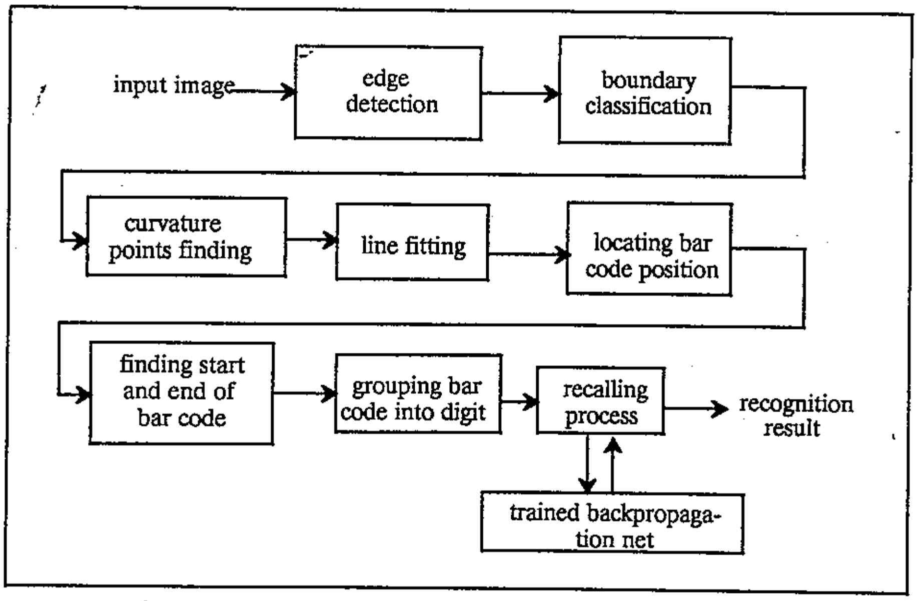


Figure 1. Block diagram of the proposed system

The rest of the paper is organized as follows. Section 2 introduces the encoding rules of the bar code. Section 3 describes the procedures of the proposed system in detail. In Section 4, some experimental results are reported. Finally, conclusion and discussion are made in Section 5.

2. Encoding Rules of Bar Code

In modern society, bar code is extensively used on most of the commodity. The bar code for commodity is just like the identification number for a person. Conventional bar code is composed of two parts: one is the group of stripes, the other is the numbers under the stripes. There are two different types of bar code depending on where it is marked. The first type is recognized as "source marking". The bar code of this type is marked on the product during manufacturing process. The second type is recognized as "in–store marking". Products of this kind are only sold in some particular stores. In this paper we take European Article Number (EAN–13) standard as an example, because it is extensively used by most of the commercial products. Figure 2 shows a sample bar code with EAN–13 standard.

There are totally 13 digits in an EAN–13 code. The center pattern and the guard pattern consist of auxiliary characters. In this standard, each digit is composed of 7 modules. The center pattern and the guard pattern contain 5 modules and 3 modules, respectively. Figure 3 shows the combinations of the center pattern and the guard pattern. The first three digits of an EAN–13 code are country prefix. They are used to identify the nationality of the organization issuing the bar codes. Their four successive digits represent the manufacturer number. Then, the five successive digits represent the specific product item number. The last digit of an EAN–13 code is a check digit. There is another commonly used standard called EAN–8 that contains only 8 digits in length. The latter is used only when the total surface area of a product is less than 120 cm^2. Figure 4 shows examples of the EAN–13 and EAN–8 code.

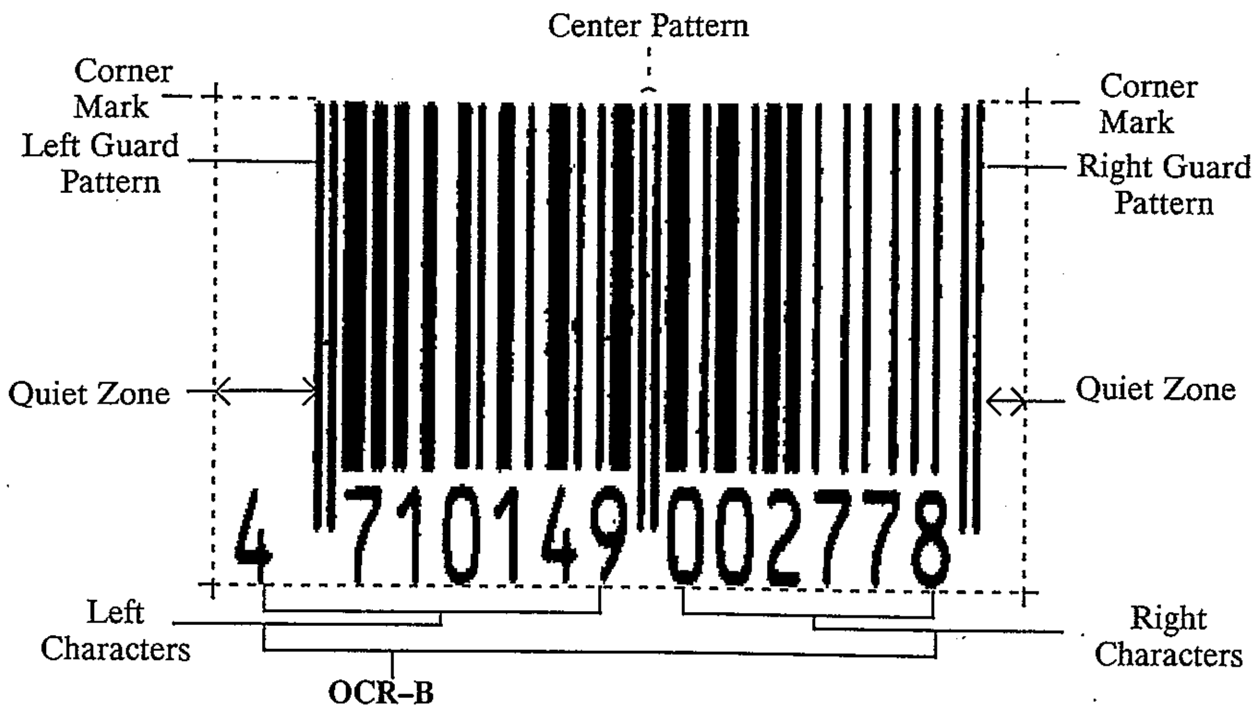


Figure 2. The illustration of EAN–13 standard

Left guard pattern	3 modules	The logic is 101	█▯█
Center pattern	5 modules	The logic is 01010	▯█▯█▯
Right guard pattern	3 modules	The logic is 101	█▯█

Figure 3. Guard pattern and center pattern in EAN–13 standard

For illustrative purpose, we use EAN–13 as an example. In this standard, a code consists of 7 modules. Different combinations of 0/1 will represent different numbers under this format. There are totally three encoding rules for the so-called A, B, C types, respectively.

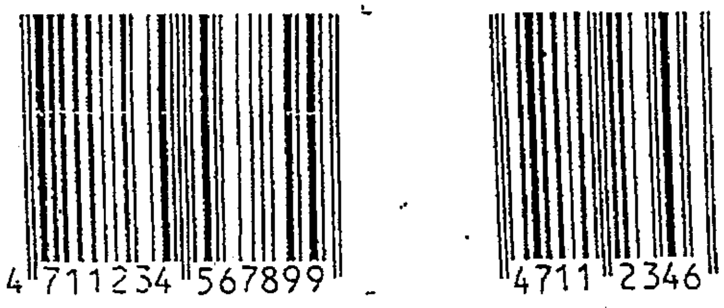


Figure 4. The standard and short version of the EAN code

Figure 5 shows the three encoding formats for the integer number 2. Table 1 illustrates the

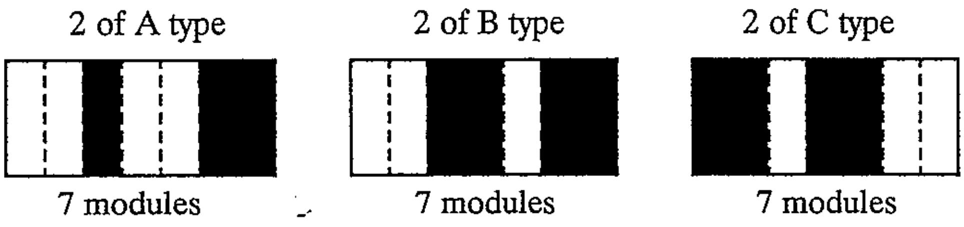


Figure 5. Number "2" encoded by three different rules.

encoding rules for the A, B, C types, respectively. Based on these encoding rules, we can decode a bar code into its corresponding digits easily.

In the EAN–13 standard, the selection of encoding rule thoroughly depends on the first digit of the country prefix. Table 2 illustrates the encoding rules of the six digits which are successive to the first digit. For example, when the rule is applied to a bar code started by the country prefix "4", then its twelve successive digits are A, B, A, A, B, B, C, C, C, C, C, C (Figure 6). The complete set of encoding rules is shown in Figure 7.

number	A type	B type	C type
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Table 1. The list of oriented–modules on logic of 0–9 for A, B, C types

3. The Bar Code Recognition System

3.1 Locating the Position of Bar Code

When an image is acquired, the first thing we have to do is to locate the position of the bar code in the image. There are several steps involved in this procedure. We first perform edge detection. Then, the four neighbors of each pixel are searched. The main purpose of this step is to locate the contour of each bar code stripe and remove some noises.

After the boundary of each stripe is located, we then detect the corner points using curvature characteristics. Pairs of corner points including those points between them are assembled to fit a line segment. The least–squares error method is applied here to guarantee the best line fitting result. After the line fitting process, the image becomes a set of straight line segments. Because of the intricate background, the set of line segments can be in any directions in the image. The orientation from 0° to 180° is the range of the angles between these line

The first digit	The encoding rules applied to the 6 subsequent digits					
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Table 2. Encoding rules applied to the 6 successive digits of the first

segments to the x–axis. In each degree, we compute the number of line segments in that direction. The direction that contains the maximum number of line segments will be selected as the orientation of the bar code.

3.2 Grouping the Bar Code

Now we describe the method for deriving the bounding rectangle of bar code. Since the texture of bar code tends to a specific direction, the height of the bounding rectangle can be easily calculated. As to the width of the bounding rectangle, we can calculate it as follows. Since the bar code stripes of the same orientation have been clustered, the positions of the first line and the last line (from left to right scan) in the current orientation can be confirmed. Based on these two lines, the width of the bounding rectangle can be determined.

bar code	4	7	1	1	2	3	4	5	6	7	8	9	9
types of encoding		A	B	A	A	B	B	C	C	C	C	C	C

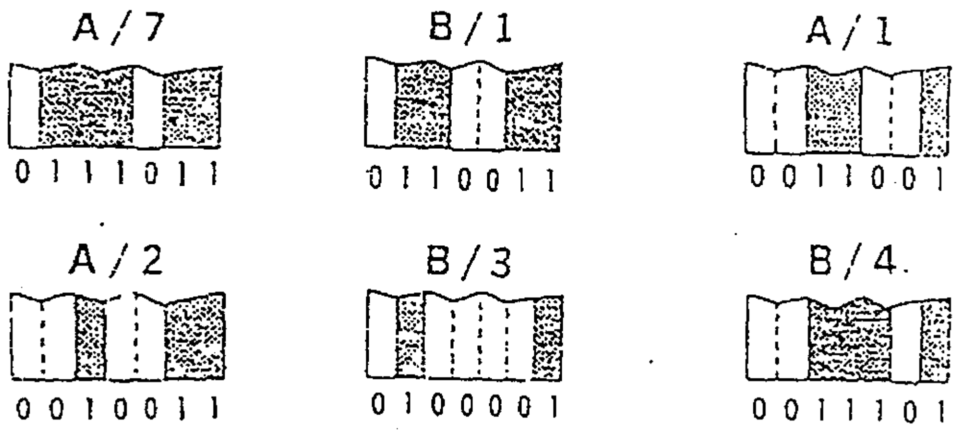
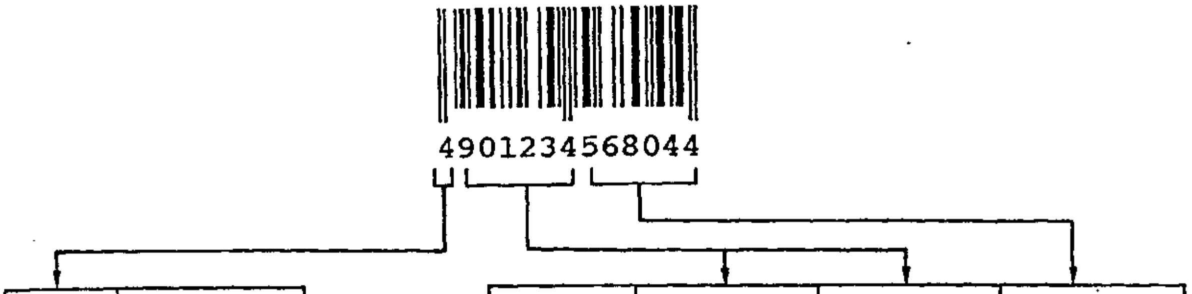


Figure 6. An example of EAN–13 encoding rule

In this phase we do not attempt to determine the exact start and end of the bar code. If the background is not too complex, the bounding area should be equal to the size of bar code. However, the background is usually not so simple. Therefore, further segmentation on the bounding rectangle is required. Under the circumstances, we have to calculate the maximum value of lines that have the same orientation. If the background is monotonic and happens to own the same orientation as the bar code stripes, the segmented area will not be correct. Another potential problem is that part of the existing texture of other object contains the same orientation as the bar code stripes and is very close to these stripes. The area of the bounding rectangle in this case will be larger than the original size. The first problem is easy to solve. We can construct a loop to examine the statistic of orientation from 0° to 180°. The orientations with the values larger than a threshold will be selected. Based on this candidate set we can obtain different areas for testing. This procedure will stop whenever



0	AAAAAA
1	AABABB
2	AABBAB
3	AABBBA
4	ABAABB
5	ABBAAB
6	ABBBAA
7	ABABAB
8	ABABBA
9	ABBABA

	A	B	C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1001110
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Figure 7. Encoding rule of EAN–13 standard.

a satisfactory result is found. For the second problem, it is difficult to deal with. This is due to the area after segmentation is larger than normal size and it may result in incorrect bar code number or even cannot be recognized.

3.3 Recognition Using BPNN

In this section we will introduce how to use the BPNN as a tool for bar code recognition. We introduce the architecture and learning procedure in BPNN firstly. Then, the recalling process for bar code recognition will be detailed in the second subsection.

3.3.1 Learning Procedure in BPNN

This subsection will focus on the BPNN training phase. Firstly, we introduce the architecture of BPNN and describe its learning law. This include how to calculate the errors, and how to propagate these errors for weight adjustment. We will also explain the effect of input patterns to the errors and the way the system outputs approximate the desired outputs. Here, we adopt a three–layered backpropagation net [13] as a tool. The training process as well as the recalling process are both based on this model. Figure 8 shows the architecture of a three–layered BPNN. In this model, each unit in a layer receives its input by summing up the

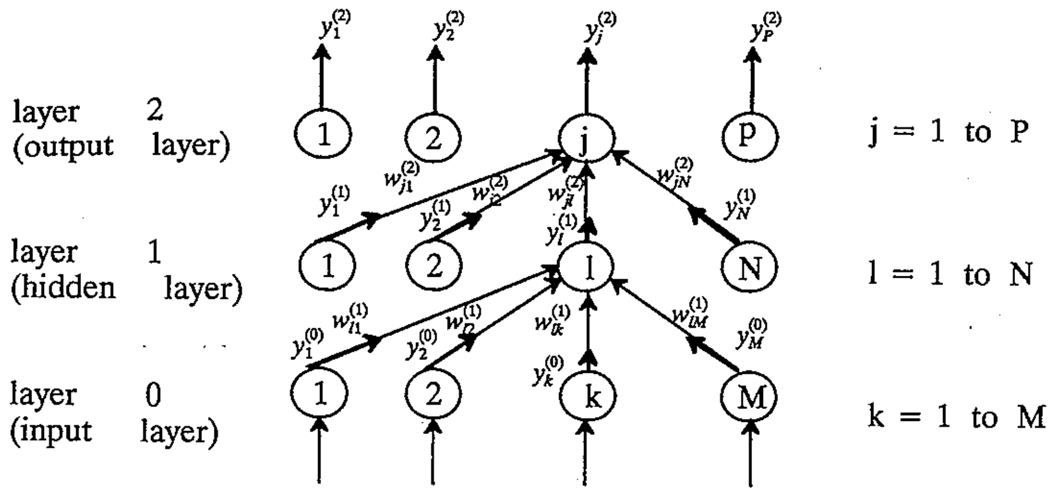


Figure 8. The architecture of BPNN

weighted output signals from all units of its previous layer. Assume the first layer (input layer), the second layer (hidden layer), and the output layer contain M, N, and P neuron units, respectively. Let d_i be the desired outputs. Some basic notations are defined as follows:

$y_i^{(0)}, i = 1,\ldots,M$	(Outputs of layer 0)
$y_i^{(1)}, i = 1,\ldots,N$	(Outputs of layer 1)
$y_i^{(2)}, i = 1,\ldots,P$	(Outputs of layer 2)
$net_i^{(1)}, i = 1,\ldots,N$	(Inputs to layer 1)
$net_i^{(2)}, i = 1,\ldots,P$	(Inputs to layer 2)
$w_{lk}^{(1)}, k = 1,\ldots,M, l = 1,\ldots,N$	(Weights between layer 0 and layer 1)
$w_{jl}^{(2)}, l = 1,\ldots,N, j = 1,\ldots,P$	(Weights between layer 1 and layer 2)

$w_{lk}^{(1)}, k = 1,\ldots,M, l = 1,\ldots,N$ (Weights between layer 0 and layer 1)

$w_{jl}^{(2)}, l = 1,\ldots,N, j = 1,\ldots,P$ (Weights between layer 1 and layer 2)

In different layers, all neurons are of the same type. The neurons in layer 0 fan out the copies of the inputs obtained from the outside world. The fanned out copies multiply the weights between layer 0 and layer 1 eventually becoming the inputs to the hidden layer. The inputs to the hidden layer can be represented by a compact form as follows:

$$net_j^{(1)} = \sum_{k=1}^{M} w_{jk}^{(1)} y_k^{(0)}, 1 \le j \le N \tag{1}$$

By the same token, the output $y_k^{(1)}$ multiplies the weights between layer 1 and layer 2 will finally become the inputs to layer 2,

$$net_i^{(2)} = \sum_{k=1}^{N} w_{ik}^{(2)} y_k^{(1)}, 1 \le 1 \le P \tag{2}$$

In this architecture, neurons only play the role as transfer units.

For the training process, we adopt the generalized delta rule [14] to update $W^{(1)}$ and $W^{(2)}$ such that the accumulated errors E can be reduced to satisfy a preset value as soon as possible. The accumulated error E is defined as follows,

$$E = \frac{1}{2}\sum_{i=1}^{P}\left(d_i - y_i^{(2)}\right)^2 \tag{3}$$

Each time when we get the errors, we adjust the weights such that the errors decreased in the next run.

In order to explore how the weights affect the errors, partial derivative of E is taken with respect to the weight $w_{jl}^{(2)}$ firstly. By chain rule, we have

$$\frac{\partial E}{\partial w_{jl}^{(2)}} = \frac{\partial E}{\partial net_j^{(2)}} \frac{\partial net_j^{(2)}}{\partial w_{jl}^{(2)}}$$

$$= \frac{\partial E}{\partial y_j^{(2)}} \frac{\partial y_j^{(2)}}{\partial net_j^{(2)}} \frac{\partial net_j^{(2)}}{\partial w_{jl}^{(2)}}$$

$$= -\left(d_j - y_j^{(2)}\right) f^{(2)'}(net_j^{(2)}) y_l^{(1)}$$

$$= -\hat{\delta}_j^{(2)} f^{(2)'}(net_j^{(2)}) y_l^{(1)} \tag{4}$$

where $\hat{\delta}_j^{(2)} = d_j - y_j^{(2)}$.

Since we know how $w_{jl}^{(2)}$ (weights between the hidden and output layers) affect the error, we also examine how the weights between the hidden and input layers affect the error. Again, partial derivative of E is taken with respect to $w_{lk}^{(1)}$ and we have the following derivation.

$$\frac{\partial E}{\partial w_{lk}^{(1)}} = \frac{\partial E}{\partial net_l^{(1)}} \frac{\partial net_l^{(1)}}{\partial w_{lk}^{(1)}}$$

$$= \frac{\partial E}{\partial y_l^{(1)}} \frac{\partial y_l^{(1)}}{\partial net_l^{(1)}} \frac{\partial net_l^{(1)}}{\partial w_{lk}^{(1)}}$$

$$= \sum_{j=1}^{P} \left[\frac{\partial E}{\partial net_j^{(2)}} \frac{\partial net_j^{(2)}}{\partial y_l^{(1)}} \right] \frac{\partial y_l^{(1)}}{\partial net_l^{(1)}} \frac{\partial net_l^{(1)}}{\partial w_{lk}^{(1)}} \tag{5}$$

Equation (5) can be further simplified by the following derivations,

$$\frac{\partial E}{\partial net_j^{(2)}} = \frac{\partial E}{\partial y_j^{(2)}} \frac{\partial y_j^{(2)}}{\partial net_j^{(2)}} = -\hat{\delta}_j^{(2)} f^{(2)'}(net_j^{(2)}) \tag{6}$$

$$\frac{\partial net_j^{(2)}}{\partial y_l^{(1)}} = \frac{\partial}{\partial y_l^{(1)}} \sum_{n=1}^{N} w_{jn}^{(2)} y_n^{(1)} = w_{jl}^{(2)} \tag{7}$$

$$\frac{\partial y_l^{(1)}}{\partial net_l^{(1)}} = f^{(1)'}(net_l^{(1)}) \tag{8}$$

$$\frac{\partial net_l^{(1)}}{\partial w_{lk}^{(1)}} = \frac{\partial}{\partial w_{lk}^{(1)}} \sum_{j=1}^{M} w_{lj}^{(1)} y_j^{(0)} = y_k^{(0)} \tag{9}$$

Substitute the results of equations (6), (7), (8), and (9) into equation (5), a more compact form is derived as follows,

$$\frac{\partial E}{\partial w_{lk}^{(1)}} = -\left[\sum_{j=1}^{P} w_{jl}^{(2)} \hat{\delta}_j^{(2)} f^{(2)'}(net_j^{(2)})\right] f^{(1)'}(net_l^{(1)}) y_k^{(0)} \tag{10}$$

Based on the results of equations (5) and (10), $W^{(1)}$ and $W^{(2)}$ can be updated systematically. When the training phase is terminated, the system is able to perform recalling process.

3.3.2 Recognition phase

Since the training of BPNN is completed and the bounding rectangle of bar code is obtained, the next step is to perform the recognition task. Although the area of bar code may not be so exact on the actual beginning and ending lines, it will still be processed in this phase. First of all, we perform histogram analysis in the bounding rectangle. By traversing along the perpendicular direction of the bar code stripes, we calculate the number of active pixels column by column (column is parallel to stripes). If the number of active pixels in a column is larger than a preset threshold, then that column is assigned value 1, otherwise 0. That is, the content within the bounding rectangle is converted into a one dimensional bit array. When we count the number of active pixels of each column, the maximum number can be obtained. Examining these values column by column, we take half of the maximum value as a threshold.

According to the encoding rule of the EAN–13 standard, the start and end character both consist of 3 modules–"101". By measuring the distance between the start and end characters, we can determine the length of the bar code. This process is started by checking the bit array and look for the transitions of 0 –> 1 and 1 –> 0. Then, we group 3 transitions as a unit and normalize its length. The content of each unit is fed into a trained backpropagation neural network and the unknown bar code will be recognized by recalling the trained weight matrix in the neural network. Through this process, it is obvious that scale of the image would not affect the final result. That is, the system is able to deal with any bar code with different sizes.

As to the larger bounding rectangle problem, we examine the length between the start and end characters. If there are several pairs of start and end characters, we calculate each of their length. Then, the bit array between each pair is fed into the trained BPNN for correctness check. This process continues until a valid pair is found. If a valid bar code number cannot be found, we simply reject the bar code.

The real scale of the BPNN for experiment is described as follows.. The designed BPNN includes 56 neurons in its input layer, 48 neurons in its hidden layer, and 40 neurons in its output layer. After converting the content of 3 transitions into a 56–bit length bit array, we use it as input to the BPNN. The BPNN is pre–trained by different types of encoding rules. In our experiment, we use A, B, C, and reverse A types as standards to perform training task. Fig. 9 shows the reverse A type encoding rule.

After the training phase has been completed, the weight matrix is fixed and the system is ready for recalling process. At this time we input the grouped combinations and simply wait for the result.

4. Experimental Result

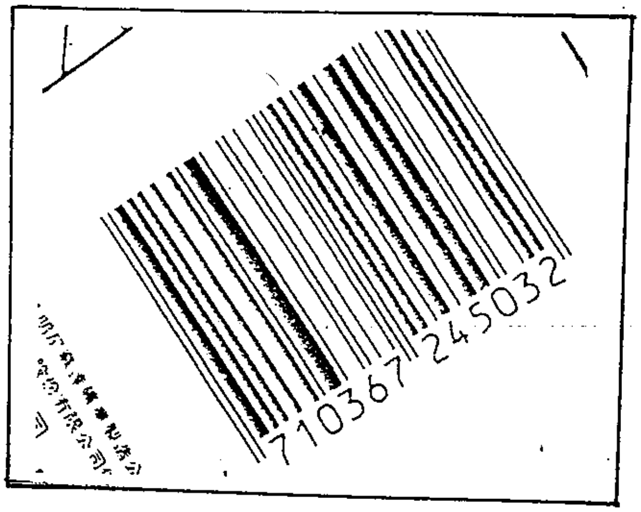
In the experiments we perform bar code recognition on several real images using SPARC 2 workstation. These bar codes are obtained from a variety of commercial products in our daily life. From Figures 10 – 16, part (a) show the original images. These images contain different orientations and backgrounds. Some of them are pure and monotonous, while some are complicated and damaged by paint. After performing preprocessing and line fitting, the intermediate results of these samples are shown in part (b) of Figures 10 – 16. The highlight lines in these figures are the lines after fitting. Next, we try to locate the position and

Digit	The reverse A type
0	1011000
1	1001100
2	1100100
3	1011110
4	1100010
5	1000110
6	1111010
7	1101110
8	1110110
9	0001011

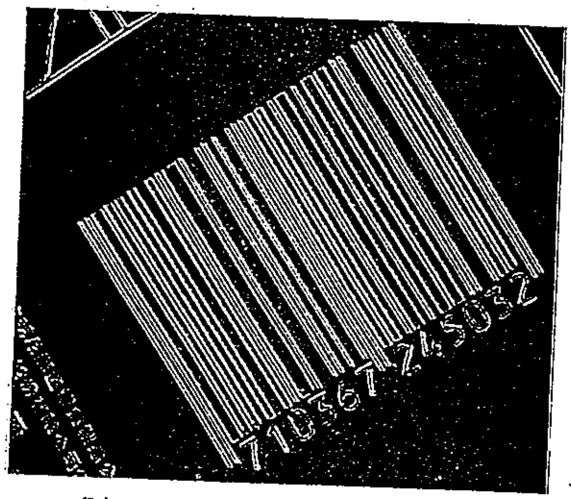
Figure 9. The reverse A type encoding rule

orientation of the bar code based on its texture. The located bounding rectangles are shown in part (c) of Figures 10 – 16. It is obvious that the bounding rectangle encloses the bar code stripes correctly when the background is simple. However, it is possible to have larger bounding rectangle when the background is intricate. After the position of the bar code is located, the grouping process collects the normalized bar code into digits and feed them into a trained BPNN. From Table 3 to Table 9, the correct recalled results of bar codes in part (a) of Figures 10 – 16 are shown, respectively.

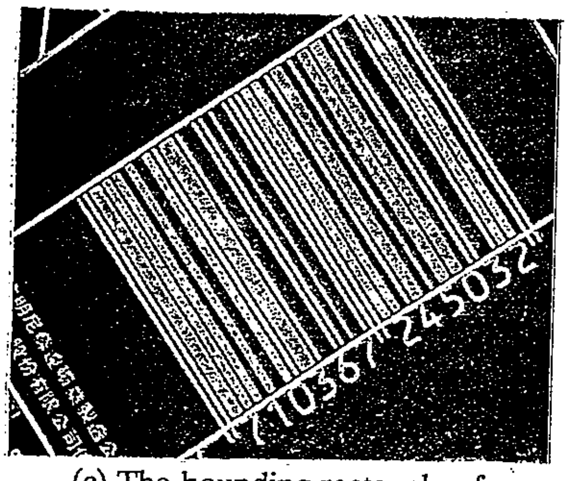
Due to scale and complexity differences in the bar code images, the execution time is case dependent. For the preprocessing part, it takes about 18 ~ 35 seconds to process an image of size under 50K. However, in the recalling process the average execution is about 0.3 second.



(a) Original image of bar code 1



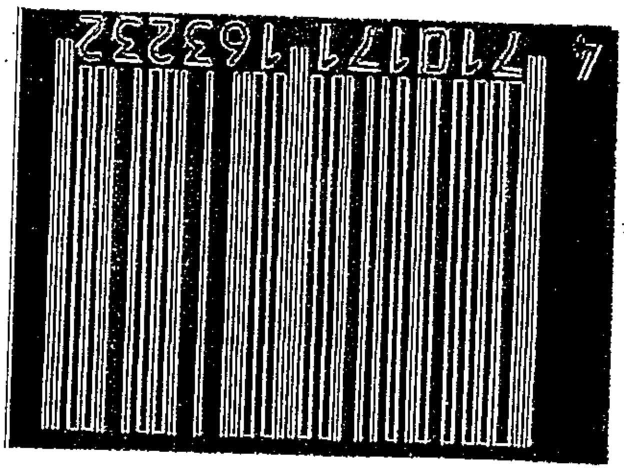
(b) Line fitting result of (a)



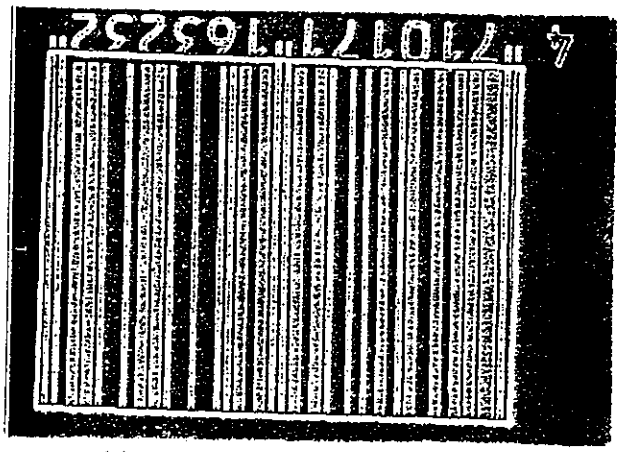
(c) The bounding rectangle of
(a) after segmentation

Figure 10. The results after line fitting and segmentation of bar code 1

(a) Original image of bar code 2

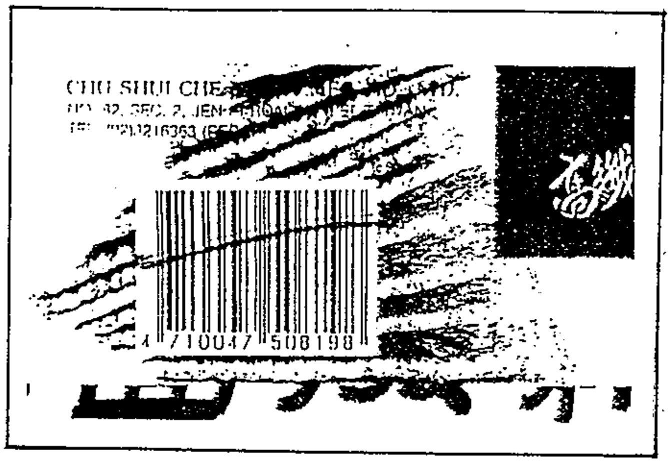


(b) Line fitting result of (a)

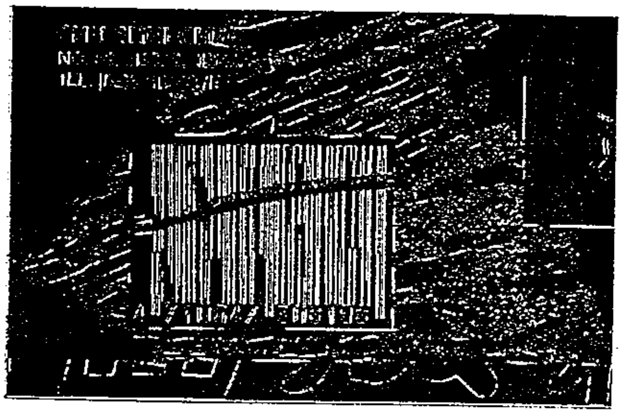


(c) The bounding rectangle of
(a) after segmentation

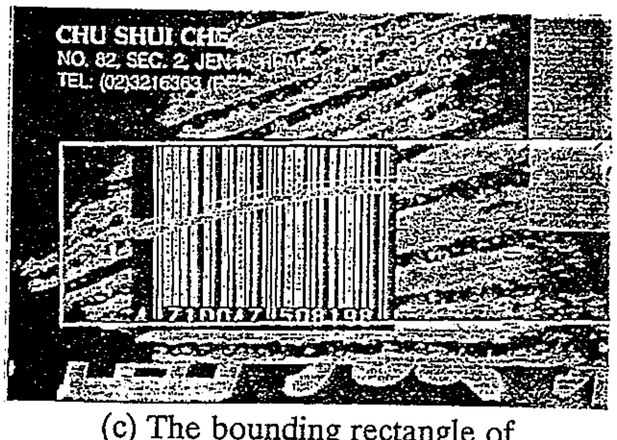
Figure 11. The results after line fitting and segmentation of bar code 2



(a) Original image of bar code 3

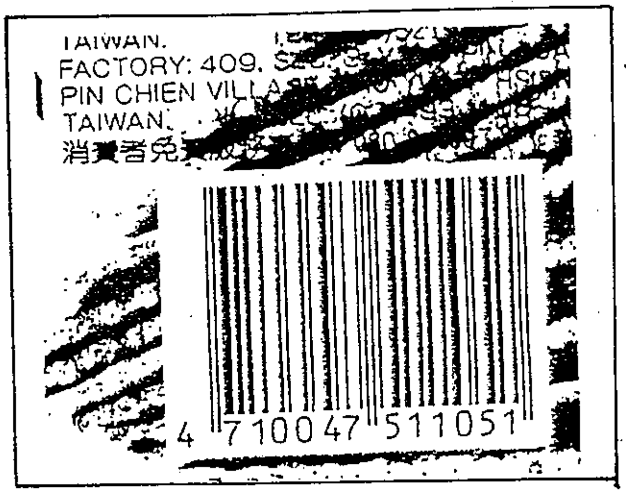


(b) Line fitting result of (a)

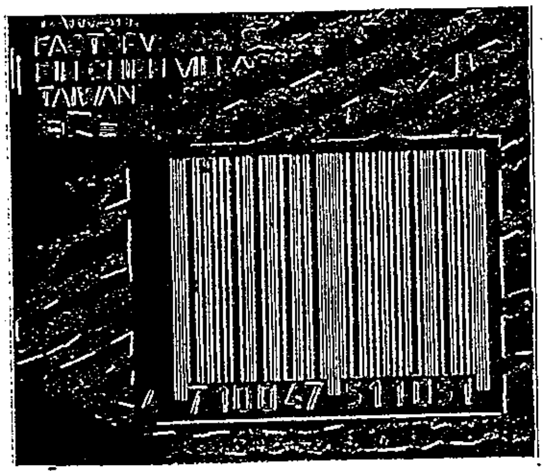


(c) The bounding rectangle of
(a) after segmentation

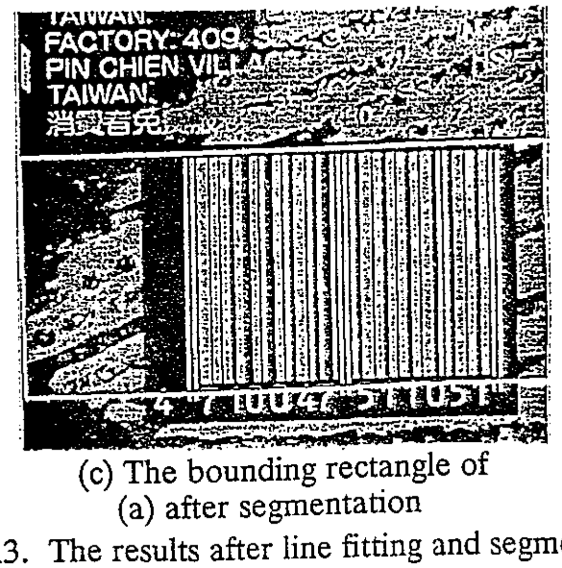
Figure 12. The results after line fitting and segmentation of bar code 3



(a) Original image of bar code 4

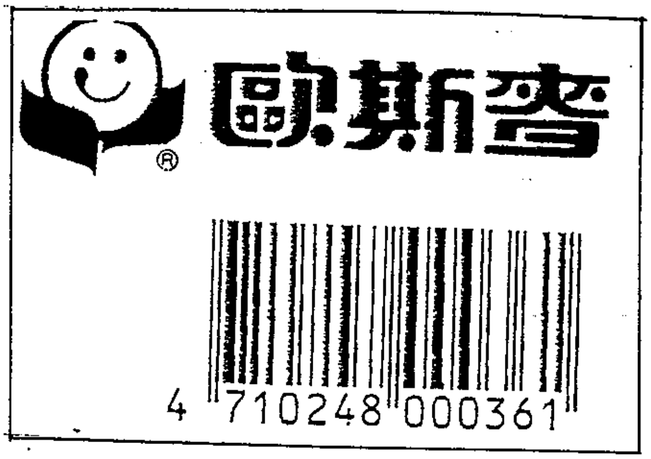


(b) Line fitting result of (a)

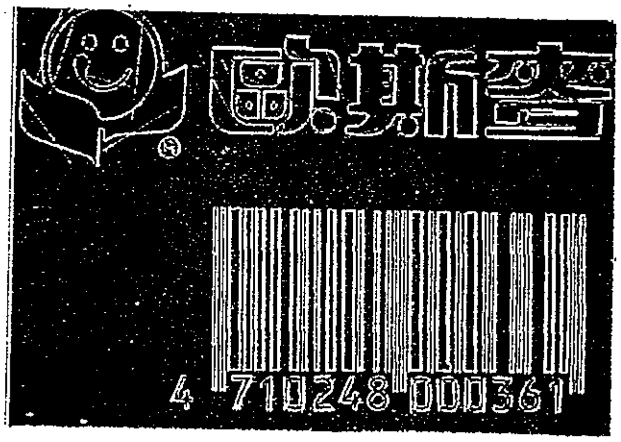


(c) The bounding rectangle of
(a) after segmentation

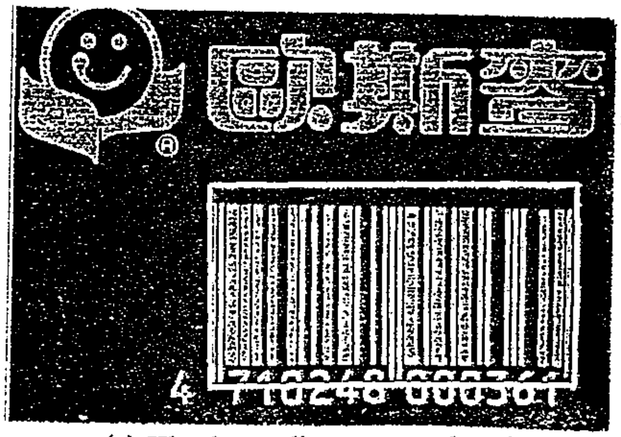
Figure 13. The results after line fitting and segmentation of bar code 4



(a) Original image of bar code 5

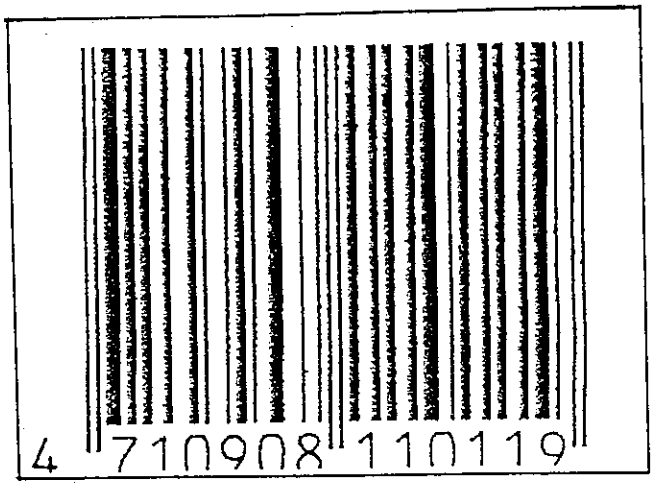


(b) Line fitting result of (a)

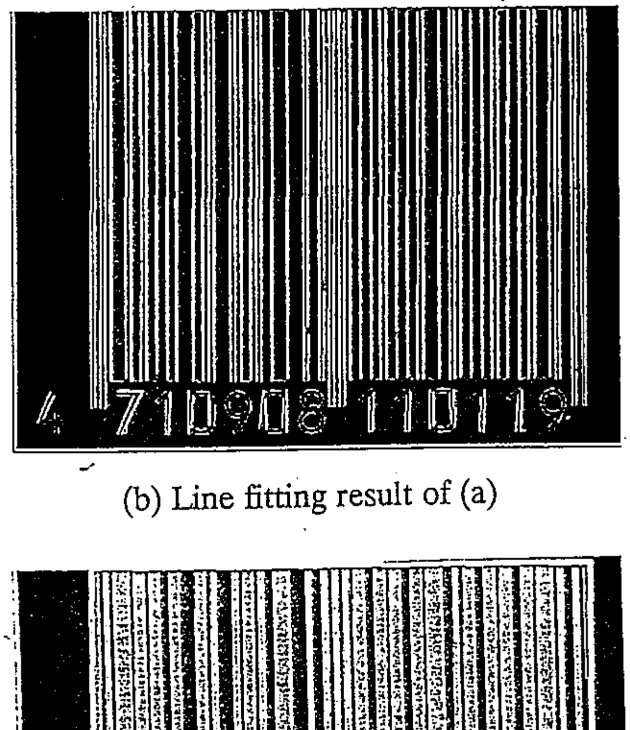


(c) The bounding rectangle of
(a) after segmentation

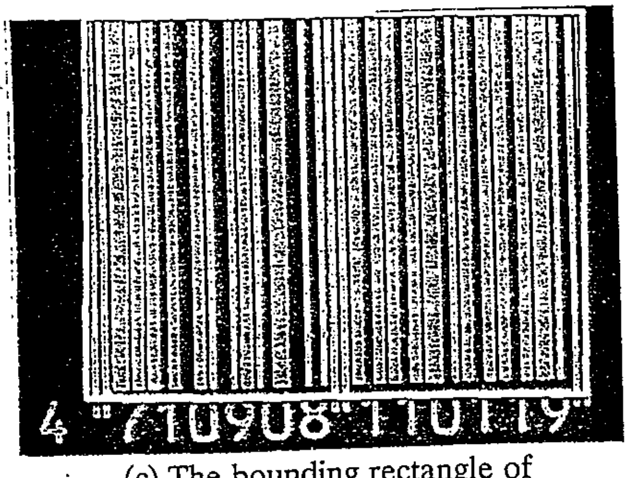
Figure 14. The results after line fitting and segmentation
of bar code 5



(a) Original image of bar code 6

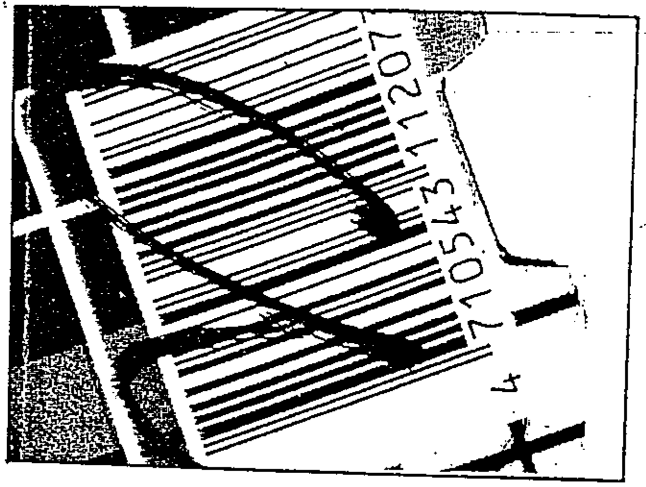


(b) Line fitting result of (a)

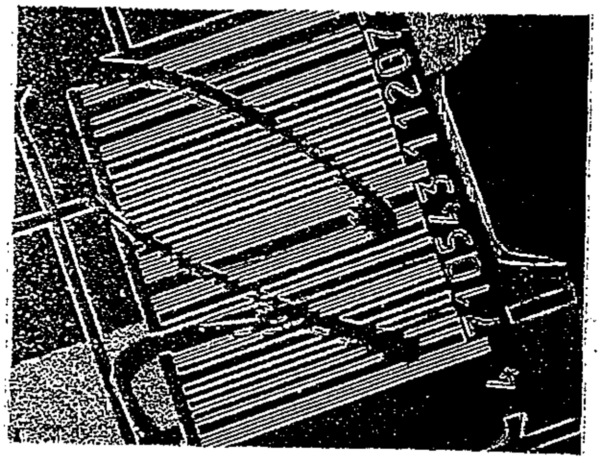


(c) The bounding rectangle of
(a) after segmentation

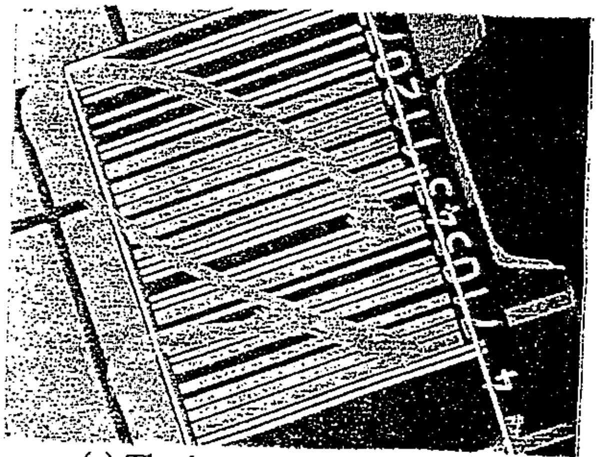
Figure 15. The results after line fitting and segmentation of bar code 6



(a) Original image of bar code 7



(b) Line fitting result of (a)



(c) The bounding rectangle of
(a) after segmentation

Figure 16. The results after line fitting and segmentation of bar code 7

order	input	digit	result
1	00000001111111111111111111111111000000011111111111111111	7	0.889119
2	00000000111111111111111000000000000000011111111111111111	1	0.975827
3	00000000000000000000000001111111111111111000000011111111	0	0.969128
4	00000000011111111111111111111111111111111000000000111111	3	0.975551
5	00000000000000000000000000000000001111110000000000111111	6	0.649445
6	00000000000000000011111100000000000000000000000000111111	7	0.986205
7	11111111111111110000000111111111111111100000000000000000	2	0.887893
8	11111100000000011111111111111111111111100000000000000000	4	0.981922
9	11111100000000000000000111111111111111111111111000000000	5	0.829875
10	11111111111111111111111000000000000000000111111000000000	0	0.968288
11	11111100000000000000000000000000000000000111111000000000	2	0.984319
12	11111111111111100000000111111111111111111000000000000000	3	0.902215

Table 3. The input data code after grouping and normalization and the result after recalling process of Fig.10 (a)

order	input	digit	result
1	00000011111111111111111111111111000000011111111111111111	7	0.748442
2	00000001111111111111111100000000000000011111111111111111	1	0.954561
3	00000000000000000000000011111111111111111000000011111111	0	0.953487
4 /	00000000000000000111111111111111000000000000000001111111	1	0.984115
5	00000000000000011111111000000000000000000000000001111111	7	0.971351
6	00000001111111111111111100000000000000011111111111111111	1	0.954561
7	11111111111111111000000000000000111111111111111110000000	1	0.896941
8	11111110000000011111111000000000000000000000000000000000	6	0.935989
9	11111111000000000000000000000000000000001111111100000000	3	0.989588
10	11111111111111111000000011111111111111111100000000000000	2	0.913110
11	11111111000000000000000000000000000000011111111110000000	3	0.852262
12	11111111111111111000000111111111111111111100000000000000	2	0.888881

Table 4. The input data code after grouping and normalization and the result after recalling process of Fig.11 (a)

order	input	digit	result
1	00000001111111111111111111111111100000001111111111111111	7	0.889119
2	00000001111111111111111100000000000001111111111111111111	1	0.765757
3	00000000000000000000000111111111111111111100000111111111	0	0.765777
4	00000000000000000000000111111111111111110000000011111111	0	0.982378
5	00000000000000001111111111111111111111110000000011111111	4	0.989974
6	00000000000000111111111000000000000000000000000001111111	7	0.874645
7	11111111000000000000000011111111111111111111111100000000	5	0.980606
8	11111111111111111111111000000000000000001111111100000000	0	0.972374
9	11111111000000000000000011111111100000000000000000000000	8	0.974937
10	11111111111111110000000000000000011111111111111111100000	1	0.785016
11	11111111111111111111111110000000111111111000000000000000	9	0.963501
12	11111110000000000000000111111111000000000000000000000000	8	0.934681

Table 5. The input data code after grouping and normalization and the result after recalling process of Fig.12 (a)

order	input	digit	result
1	00000001111111111111111111111111100000111111111111111111	7	0.251669
2	00000001111111111111111000000000000000111111111111111111	1	0.915056
3	00000000000000000000001111111111111111110000000111111111	0	0.830220
4	00000000000000000000000111111111111111110000000111111111	0	0.944430
5	00000000000000111111111111111111111111110000000111111111	4	0.962110
6	00000000000000111111111110000000000000000000000111111111	7	0.722378
7	11111111000000000000000111111111111111111111111110000000	5	0.909748
8	11111111111111111000000000000000011111111111111110000000	1	0.787203
9	11111111111111111000000000000000111111111111111110000000	1	0.896941
10	11111111111111111111111000000000000000000111111111000000	0	0.972374
11	11111111000000000000000111111111111111111111111110000000	5	0.909748
12	11111111111111110000000000000000011111111111111110000000	1	0.844794

Table 6. The input data code after grouping and normalization and the result after recalling process of Fig.13 (a)

order	input	digit	result
1	000001111111111111111111111111111100000011111111111111111	7	0.874357
2	00000001111111111111111110000000000000011111111111111111	1	0.778243
3	00000000000000000000000001111111111111111000000001111111	0	0.916252
4	00000000000000000011111100000000000000011111111111111111	2	0.958840
5	000000000000000111111111111111111111111111000000001111111	4	0.815128
6	00000000000000000000000001111111000000000000000001111111	8	0.871937
7	11111111111111111111111110000000000000000111111110000000	0	0.959800
8	11111111111111111111111110000000000000000111111110000000	0	0.959800
9	11111111111111111111111110000000000000000111111110000000	0	0.959800
10	11111111000000000000000000000000000000000111111110000000	3	0.975122
11	11111111100000011111111100000000000000000000000000000000	6	0.952874
12	11111111111111111000000000000000011111111111111111100000	1	0.879887

Table 7. The input data code after grouping and normalization and the result after recalling process of Fig.14 (a)

order	input	digit	result
1	00000000000000001111111110000000011111111111111111111111	7	0.961151
2	00000001111111111111111000000000000000001111111111111111	1	0.972687
3	00000001111111111111111000000000000000001111111111111111	0	0.972687
4	00000001111111100000000000000001111111111111111111111111	9	0.794892
5	00000001111111111111111110000000000000001111111111111111	0	0.914309
6	00000001111111111111111000000000000000111111111111111111	8	0.915056
7	11111100000000000000000011111110000000000000000000000000	1	0.992650
8	11111111111111111111111100000000000000001111111100000000	1	0.825428
9	11111111111111110000000001111111100000000000000000000000	0	0.969836
10	11111111000000001111111111111111100000000000000000000000	1	0.965772
11	11111111111111110000000000000000111111111111111111100000	1	0.828231
12	11111111111111111100000111111111111111111111111110000000	9	0.249744

Table 8. The input data code after grouping and normalization and the result after recalling process of Fig.15 (a)

5. Conclusion and Discussion

In this paper, we have proposed a bar code recognition system using backpropagation style neural networks. Since the system adopts camera as its sensor, there are some difficul-

order	input	digit	result
1	000000111111111111111111111110000001111111111111111111	7	0.188477
2	000000111111111111111111100000000000000111111111111111111	1	0.741531
3	000000000000000000000001111111111111111110000001111111111	0	0.508362
4	000000111111111111111111100000000000000000000000111111111	5	0.727534
5	000000000000000111111111111111111111111111000000111111111	4	0.979855
6	000000111111111100000000000000000000000000000001111111111	3	0.771845
7	111111111111111111100000000000000111111111111111111000000	1	0.516930
8	111111111111111111100000000000000111111111111111111000000	1	0.516930
9	111111111111111111000000111111111111111111110000000000000	2	0.941698
10	111111111111111111111111100000000000000000111111111000000	0	0.718023
11	111111110000000000000000000000001111111100000000000000000	7	0.983769
12	111111111000000000000000000000000000000001111111100000000	3	0.974279

Table 9. The input data code after grouping and normalization and the result after recalling process of Fig.16 (a)

ties in capturing clean bar code images. However, the texture of the bar code will make up this deficiency by introducing some heuristics to resolve the aforementioned problem.

In the proposed system, the role of the backpropagation net is a powerful tool for bar code recognition. In fact, a trained backpropagation net is able to retrieve correct counterpart of the input (unknown) data from the database even when the input data only contain partially distinguishable information. That is, a trained backpropagation net is capable of dealing with inexact matching problems. As to bar code segmentation process, if the background is not too complicated and there is no texture which is similar to bar code, then the derived bounding rectangle will be right on the bar code. However, if the aforementioned problems happen and hence result in a larger bounding rectangle, then we may acquire several ambiguous candidate start–and–end code pairs. Under the circumstances we have to try the recalling process repeatedly until a correct number emerges. Finally, one of our ultimate goals is to develop the capability of recognizing bar codes on cans. Since the bar code on a can is bended, one has to estimate the degree of distortion and recover it before the segmentation process is performed. The bended bar code recovering process involves a number of interesting issues like: lens analysis, curve fitting and parameter estimation, etc. These issues will be addressed in the future work.

References

[1] T.P. Vopl, J.K.Mangis, A.K.Rigler, W.T.Zink, and D.L.Alkon, "Accelerating the Convergence of the Back Propagation Method," *Biol. Cybern.* 59, 257–263, 1988.

[2] Yoshio Hirose, Koichi Yamashita, and Shimpei Hijiya, "Back–Propagation Algorithm Which Varies the Number of Hidden Units, " *Neural Networks*, Vol. 4, pp. 61–66, 1991.

[3] L.G. Allred and G.E. Kelly, "Supervised Learning Techniques for Backpropagation Networks," Software Support Division (MAS) Ogden Air Logistics Center Hill Air Force Base, Utah.

[4] R. A. Jacobs, "Increased Rates of Convergence Through Learning Rate Adaptation," *Neural Networks*, Vol. 1, pp.295–307, 1988.

[5] Y.J.E. Feng, "An Automated System for Document Recognition," Master Thesis, Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois, August, 1988.

[6] W.C. Lin and C.K. Tsao, "Document Classification Using Associative Memories," *Journal of Neural Network Computing,* Vol. 1, No. 4, pp. 33–41, Spring 1990.

[7] W.C. Lin and C.C. Hsu, "Location of Specific Items in a Document Without Using Character Recognition Techniques," *Eng. Appi. of AI*, Vol. 1, pp. 194–202, September 1988.

[8] R.M. Haralick and L.G. Shapiro, "Image segmentation techniques," *Computer Vision, Graphics, and Image Processing*, Vol. 29, pp. 100–132, 1985.

[9] M. Unser and M. Eden, "Multiresolution Feature Extraction and Selection for Texture Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 11, no. 7, pp. 717–728. July 1989.

[10] S.U. Lee, S.Y. Chung, and R.H. Park, "A Comparative Performance Study of Several Global Thresholding Techniques for Segmentation," *Computer Vision, Graphics, and Image Processing*, Vol. 52, pp. 171–190, 1990.

[11] S.Baronti, A. Casini, F. Lotti, L. Favaro, and V. Roberto, "Variable Pyramid Structures for Image Segmentation", *Computer Vision, Graphics, and Image Processing*, Vol. 49, pp. 346–356, 1990.

[12] P.C. Chen and T. Pavlidis, "Image Segmentation as an Estimation Problem," *Comput. Graph., Image Processing*, Vol. 12, pp.153–172, 1980.

[13] D.E. Rumelhart and J.L. McClelland, Parallel Distributed Process : Explorations in the Microstructure of Cognition, I & II, MIT press, Cambridge MA, 1986.

[14] Robert Hecht–Nielsen, Neurocomputing, Addison–Wesley Publishing Company, Inc., 1990.