Character Recognition from Virtual Scenes and Vehicle License Plates Using Genetic Algorithms and Neural Networks

Stephen Karungaru, Kenji Terada and Minoru Fukumi

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51473

1. Introduction

Character recognition remains one of the vital research areas mainly because its application to human-machine and machine-machine communication. One example application that needs this technology is vehicle number plate recognition. With millions of vehicles on the roads today, human resources alone are insufficient in recognizing, tracking or controlling their movements. Another area in character recognition is in virtual scenes. In such scenes, the characters are written in the air by hand and captured using a cheap USB camera placed in front of a subject. Such characters are termed "Air Characters" in this work.

In this chapter, we present a character recognition method for virtual scenes (air characters) and vehicle number plate recognition using neural networks and evolutionary computation. We combine neural networks learning, image processing and template matching to create a novel character recognition system. To speed up the system and deal with size and orientation issues, we employ a genetic algorithm. Furthermore, to control the size of both the neural network inputs and the template, we also apply a genetic algorithm to guide the search.

Fortunately, many useful technologies in automatic detection and recognition have already been proposed to recognize characters. In vehicle license plate detection and recognition research is widely carried out by many researchers in many countries because of the many applications that benefit from it ranging from traffic control, crime prevention, automatic parking authentication systems, etc. Recognition of air characters will open new areas in human-machine interfaces especially in replacing the TV remote control devices and enabling non-verbal communication. Three steps are necessary in such systems. That is, the size and orientation invariant segmentation of the characters, normalization of other factors like brightness, contrast, illumination, etc. and the recognition of the characters themselves.

In [1] we proposed a robust license plate recognition method which recognized characters using a combination of neural networks, template matching and genetic algorithms. In this work, we improve the system by the introduction of the the bilateral filter for noise reduction,the variable threshold method for better image binalization and linear regression to extract more shape features that creates a better feature vector.

Today there are many OCR devices in use based on a plethora of different algorithms [21]. Examples include a wavelet transform based method for extracting license plates from cluttered images achieving a 92.4% accuracy [2] and a morphology-based method for detecting license plates from cluttered images with a detection accuracy of 98% [3] . Hough transform combined with other preprocessing methods is used by [4, 5]. In [6] an efficient object detection method is proposed. More recently, license plate recognition from low-quality videos using morphological and Adaboost algorithm was proposed by [7]. It uses the haar like features proposed by [8] for face detection. Furthermore, in our earlier work [9], we proposed a license plate detection algorithm using genetic algorithms. All of the popular algorithms sport high accuracy and most high speed, but still many suffer from a fairly simple flaw: mis-recognition that is often very unnatural to the human point of view. That is, mistaking a ″5" for an ″S", or a ″B" for a ″8",etc.

In this work, we extend and largely improve the work in [1] to also recognize the characters using more features and a hybrid system consisting of neural networks and template matching. The genetic algorithm used in [1] has been re-designed to improve its detection accuracy. We extract bifurcation, end and corner points from candidate characters and used a hybrid system made of neural networks, template matching and genetic algorithms to solve such mis-recognition problems. We also test this method using air characters that prove that the method is effective for different types of characters. However, the results are affected by character segmentation whose results are not perfect yet.

The effectiveness of various types of neural networks to solve a variety of problems has recently been shown in [10] for partially connected neural networks (PCNN), [11] for recurrent neural networks (RNN) and [12] for perceptron neural networks. This adds confidence to the use of neural network in learning problems. Character recognition using neural networks to determine a threshold is proposed by [13]. However, since character shearing is not handled exhaustively, the accuracy is not high enough. As stated, although a lot of work has been done in this area, as far as we can tell, our proposed use of genetic algorithms and artificial templates has not been used anywhere else.

2. Database

2.1. License plates

This work uses license plates images of vehicles that were taken near a parking lot with the target vehicle coming towards the camera and then turning towards the right or left[1].

The license plates are divided into several categories based on colors and arrangement of the characters on the plates. In private vehicles, the plates have a black background and white characters, for taxis it is the exact opposite, white backgrounds with black characters and a variety of other kinds based on special regions, etc. Moreover, there are single and double row plates. The characters on the plates are all alphanumeric (All upper-case). However, the alphabets I, O and Z are not used in the license plates.

In the database used in our experiments, there are 6444 images of 46 cars each captured in a different number of frames. Each of the images is 320*240 pixels. In some of the images, there are no vehicles and hence no license plates to detect. Although, there are single and double row license plates, the background color of plates is either black or white. Moreover, the number of characters in the plates may also differ. Therefore, the length of the license plates is also different. A sample of these plates is shown in Fig. 1.

The details of the extraction, from the database, of the license plates locations and the character segmentation methods adapted in this work can be found in [1].

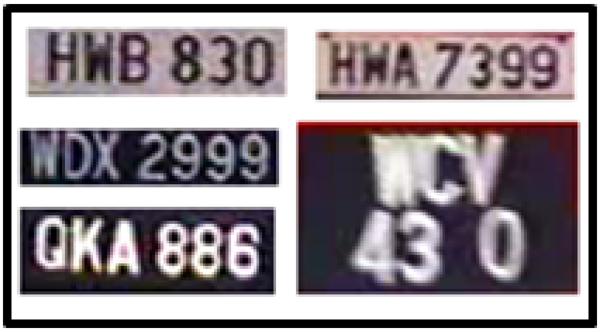

Figure 1. Example of plates in the Database

2.2. Air characters

The database for the air characters is created using images captured in our laboratory. An USB camera is placed on top of a computer display in front of a subject. In this experiment, the subject is asked to use the right hand's pointing finger to write characters in the air in front of the camera. Preprocessing steps required include the detection of the face region to make sure that it does not interfere with the character segmentation because both region are characterised by similar skin color. The tip of the finger is extracted and then tracked to trace the character. The process is as follow:

- An open palm in front of the camera indicates the home position. This is important to disable the unnecessary detections at the beginning.
- Extract the face region to make sure the skin pixels inside it don't interfere with the tracking.
- If only the pointing finger is visible, start tracing the movement.
- To end a character, pause for about 1 second.

The detection of the face can be accomplished using several methods. In this work, we use the method that uses neural networks and genetic algorithms proposed in [20].

The segmentation of the palm region uses colour, based on the HSV colour space and dynamic thresholds. Colour based extraction is chosen because it is fast to process and invariant to rotation. The conversion from the RGB color space to the HSV color space can be performed using the following expressions.

$$H = arcos \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)+(G-B)}} \tag{1}$$

$$S = 1 - 3\frac{min(R,G,B)}{R+B+G} \tag{2}$$

$$V = \frac{1}{3}(R+B+G) \tag{3}$$

An example air character capture scene is shown in Fig. 2.

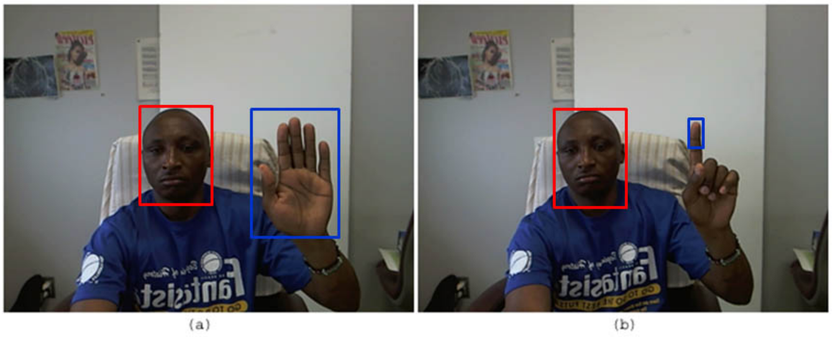

Figure 2. Air character capture scene. (a) Home position (b)Start tracing position

3. Character region segmentation

Candidate character regions should be extracted from target images to improve the search accuracy and speed up the process. Raster scanning images for a character is both slow and inefficient. Therefore, a character segmentation method is vital.

Character segmentation is a process in which the areas that are likely to contain characters (search candidates) are extracted from the image. Thereafter, character recognition methods are applied only inside the segmented candidate regions, hence, speeding up the search. However, character segmentation is not such a trivial problem. Although it is assumed that only one character will be included in each segmented area, this is not usually the case. Any method assuming one character per region will thus fail because such recognition methods (for example, template matching) rely on the number of characters and the size of the extracted region to set their parameters. Therefore character segmentation must be accurate otherwise the results of recognition will be adversely affected.

3.1. Noise reduction

Once the character candidate regions have been determined, noise reduction is carried out before image binalization by smoothing the region using the bilateral filter. Smoothing reduces the noise and average brightness of an image. The bilateral filter is defined by eq.4.

$$f(i,j) = \frac{\sum\limits_{n=-\omega}^{\omega}\sum\limits_{m=-\omega}^{\omega} f(i+m,j+n)\cdot exp(-\frac{m^2+n^2}{2\sigma_1^2})exp(-\frac{(f(i,j)-f(i+m,j+n))^2}{2\sigma_2^2})}{\sum\limits_{n=-\omega}^{\omega}\sum\limits_{m=-\omega}^{\omega} exp(-\frac{m^2+n^2}{2\sigma_1^2})exp(-\frac{(f(i,j)-f(i+m,j+n))^2}{2\sigma_2^2})} \tag{4}$$

where:

σ_1,σ_2 are the geometric spreads chosen based on the desired amount of low-pass filtering required.

However, with the bilateral filter, we must decide the optimal window size during smoothing. The best results were obtained with filter window sizes of 5*5 and 7*7. In addition, the value of parameters σ_1 and σ_2 is set to 5.

3.2. Image binalization

The most basic image binalization technique is the selection of a single threshold value for the whole image. All the grey levels below this value are classified as black, and those above white. However, it is almost impractical to clearly segment an image into objects and background with a single threshold value because of noise and illumination effects.

Therefore, in this work, to binalize the image, the Variable Threshold Method (VTM) is used. It sets a domain in an image and uses the average of the brightness in the domain to set a two level threshold. The processing windows are set in the image beginning at the top left corner of the image. Inside a window, the extraction thresholds are automatically set using average region brightness. However, it is not easy to decide the optimal size of the window for best extraction accuracy and short processing time. Therefore, we experimented using several window sizes between 5 and 20.

To improve the performance of the conventional VTM method, we employ the Discriminant Analysis (DA) method to automatically determine the threshold in the windows. The DA method classifies data into two classes solving for eigenvectors which maximize between-class variance and minimizes the within-class variance. The algorithm proceeds as follows:

For a given image, where ω_i, M_i, σ_i^2 and M_T represents the total number of pixels in class, class average brightness, class variance and overall average brightness respectively, the within-class variance (σ_W^2) is given by:

$$\sigma_W^2 = \frac{\omega_1\sigma_1^2 + \omega_2\sigma_2^2}{\omega_1 + \omega_2} \tag{5}$$

and the between-class variance (σ_B^2) can be calculated using:

$$\sigma_B^2 = \frac{\omega_1\omega_2(M_1 - M_2)^2}{(\omega_1 + \omega_2)^2} \tag{6}$$

The total variance is given by:

$$\sigma_T^2 = \sigma_W^2 + \sigma_B^2 \tag{7}$$

The threshold can be determined by maximizing σ_B^2 in the following equation.

$$\frac{\sigma_B^2}{\sigma_W^2} = \frac{\sigma_B^2}{\sigma_T^2 - \sigma_B^2} \tag{8}$$

After image binalization, labeling is carried out to extract the blobs. We expect one blob per segment. Therefore, we select the largest blob as the candidate and delete the others as noise. The major advantage of noise deletion is the reduction in the number of the blobs. This improves the systems speed by eliminating unnecessary computation. The results of this process are shown in Fig. 3.

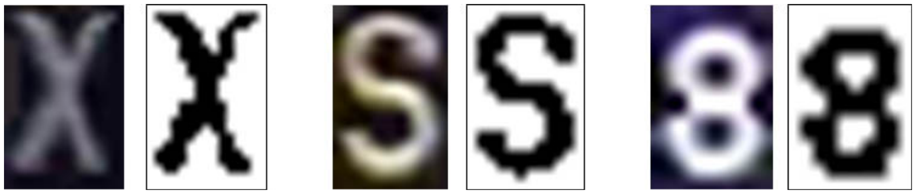

Figure 3. Binalization using the VTM and DA methods.

4. Feature extraction

Two types of features are used to recognize the characters, brightness and shape. Brightness features can be captured from the image data directly. For shape features, after image thinning, linear regression is used to extract straight lines and circular regions common in several characters and numbers.

4.1. Image thinning

A image skeleton is useful because it represents the shape of the object in a relatively small number of pixels [16]. This reduction of information speeds up the other analysis or recognition processes performed after. Thinning is an iterative technique, which extracts the skeleton of an object as a result. In every iteration, the edge pixels having more than one adjacent background pixels are eroded if their removal does not change the topology of the character.

We employ a skeletonization method called the Zhang-Suen Thinning Algorithm [17] because it is fast and easy to process. This skeletonization algorithm is a parallel method that obtains a new value depending only on the previous iteration's value. The algorithm can be implemented using two iterations. In the first iteration, a pixel is deleted (in order) if

- it has at lease two and at most six neighbours
- left, top and bottom neighbours exist
- right, top and bottom neighbours exist

The second iteration phase is similar to the first except the order of the last two processes is reversed. The algorithm terminates if no more pixels can be deleted after the two iterations. Figure 4 shows some example results achieved using the Zhang-Suen Thinning Algorithm.

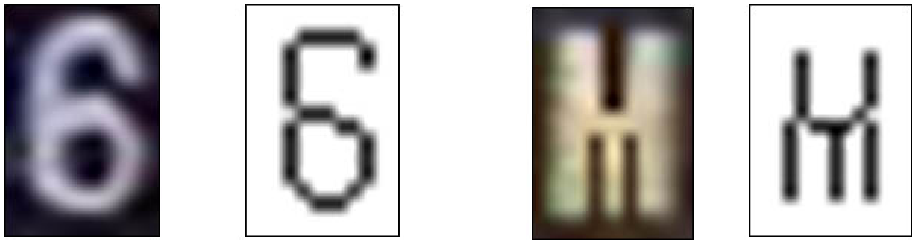

Figure 4. Thinning results using the Zhang-Suen Thinning Algorithm.

However, the results were not perfect for all characters. A closer observation of the ″M" in Fig. 4 shows that some areas still have more than one pixel width. A pruning algorithm is therefore necessary to remove such noise. The results of pruning are shown in Fig.5.

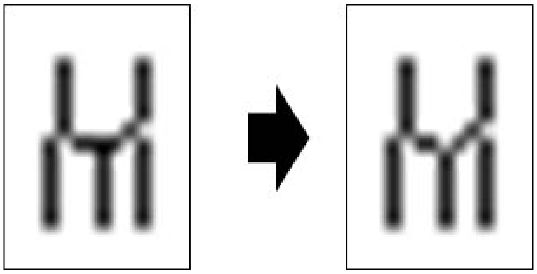

Figure 5. Noise pruning results.

4.2. Bifurcation and end points extraction

To define the shapes of the characters, it is important to extract the bifurcation and end-points from the results of the thinning operation. We define a bifurcation point as one with three neighbors and an endpoint as one with only one neighbor. In this work, we deal with license plate characters which are printed and hand written virtual characters. Table 1 shows the number of bifurcation and end-points for each character extracted manually by visual observation.

Extraction of the bifurcation and end points from virtual characters produces similar results to those shown in table 1 because they are based on our intuition about how characters should look like.

However, printed characters like the ones on license plates produce different rather surprising results. For such characters, the binalization and thinning results are somewhat different. For example Fig.6 shows the thinning results of the character ″V" extracted from a license plate.

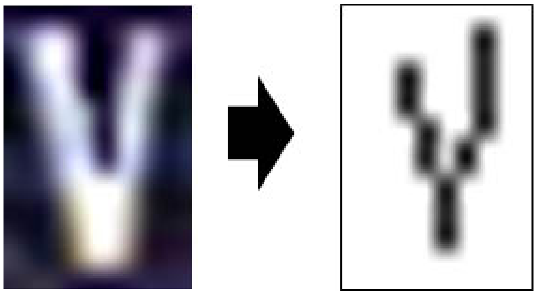

Figure 6. Noise pruning results.

Whereas the manual extraction gives no bifurcation points and 2 end points, the results of automatic extraction are 1 bifurcation point and 3 end points. In fact, ″V" and ″Y" have the same number of bifurcation and end points. Therefore, there are significant differences in the number of bifurcation and end points for the virtual and printed characters. In

Chara-cter	Bifur points	End pts	Char-cter	Bifur pts	End pts	Char-cter	Bifur pts	End pts
0	0	0	C	0	2	O	0	0
1	0	2	D	0	0	P	1	1
2	0	2	E	1	3	Q	1	2(1)
3	0	2	F	1	3	R	1(2)	2
4	1(2)	2	G	1(0)	3(2)	S	0	2
5	0	2	H	2	4	T	1	3
6	1	1	I	0	2	U	0	2
7	0	2	J	1(0)	3(2)	V	0(1)	2(3)
8	1(2)	0	K	1(3)	4	W	0(3)	2(5)
9	1	1	L	0	2	X	1(2)	4
A	2	2	M	0(5)	2(3)	Y	1	3
B	1(2)	0	N	0(2)	2(4)	Z	0	2

Table 1. Manual bifurcation and end-points extraction.

table 1, the values in the brackets show extracted points for the character when the value was different for the virtual and printed extractions. As shown in the table, characters ″4",″8",″B",″G",″J",″K",″M",″N",″Q",″R",″V",″W" and ″Y" are affected.

Figure 7 shows a set of bifurcation and end points extracted from license plate characters.

4.3. Corner points extraction

Bifurcation and end points provide information about the general shape of the characters and can help divide the character in segments for further processing. These segments are assumed to contain either straight lines or curves. However, to help differentiate characters like ″S" and ″5" or ″2" and ″Z" which have two end points each and no bifurcation points, or ″B" and ″8" with two bifurcation points and no end points, it is important to extract the corner points. Corner points help further classify segments with both straight and curved sections.

4.4. Shape features extraction

To extract the shape features, the bifurcation and end points are used. We check for straight lines and circles between a combination of the points.

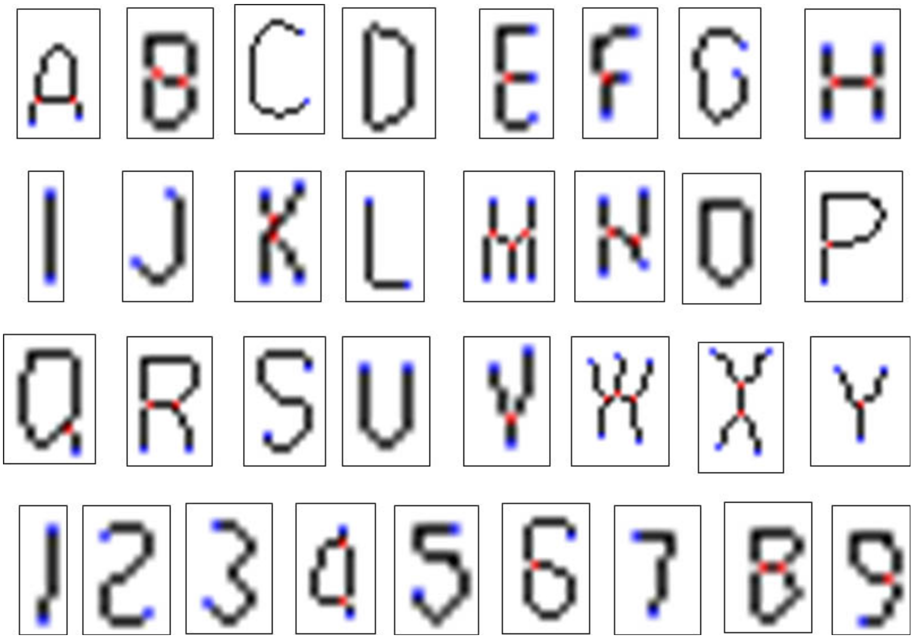
Figure 7. Bifurcation (red) and end (blue) points extracted from license plate characters.
3

4.4.1. Number of segments and length

Based on the number of bifurcation and end points, we can determine the number of segments making up the character. Each segment is created between bifurcation points, end points or a combination of both types of points. Moreover, each segment length in pixels can then be calculated. This length is then normalized by dividing it by the height of the character. The normalization should offer some size invariance during training and testing.

4.4.2. Lines: Linear regression

There is need to determine which of the segments are lines and which are circles. Initially, we should assume that the segment is a line and use a line extractor to find it. Hough transform [18] is a widely used line extractor. However, in this work, we cannot use it because we need to determine how well the extracted line fits the data. Therefore, we use linear regression to fit the line because we can calculate an error that tells us how well the lines fit.

Linear regression [19] is the least squares estimator of a linear regression model with a single explanatory variable. It fits a straight line through the set of points by minimizing the vertical distances between the points of the data set and the fitted line. The fitted line has the slope equal to the correlation between y and x corrected by the ratio of standard deviations of these variables.

After fitting a segment, we set a threshold for the error to decide if the segment is a line or should be passed to the curves(circle) extractor.

4.4.3. Curves

Most characters contain some form of circle like curve. Therefore, we can process for circularity as a character feature. This value is calculated between 0 to 1. As circularity approaches 1, a near perfect circle is extracted. It can be calculated using Eq. 9. Area and circumference are used.

$$\frac{4S\pi}{L^2} \tag{9}$$

Where S is the enclosed area and L is the circumference.

The area and circumference can be easily calculated from the segment information.

5. Character recognition

In this work, we chose Neural Networks (NN) as the main classifiers because of their proven effectiveness to learn multi-dimensional and non-linear data [10–12]. There are 26 alphabets and 10 numerals that must be recognized.

5.1. Neural network

Although neural networks can learn from large non-linear data, the processes requires thousands of training examples and huge computation time. Therefore, to effectively use neural networks in real time, their structure should be simple and the number of classes to be learned should be minimized. In this work, instead of creating one neural network to learn the 36 characters, we use "divide and conquer" method to train highly specialized compact forms.

The subdivisions are based the following features:

- Number of bifurcation points
- Number of end points
- Number of segments
- Number of straight segments
- Straight Line angles
- Number of curved segments
- Circularity (0 to 1 value) of each curved segment
- Length of segment

Table 2 shows a list of all the features extracted from the characters.

Using this information, we can divide the characters into seven neural networks as shown in Table 3.

The neural networks are selected to be 3 layered trained using the back propagation algorithm [14]. The size of the training sample is 15x20pixels. There are also nodes to represent the presence of lines(1), angles(3), circles(1) and their numbers(2). Therefore, the number of units in the input layer is 307.

The system is trained to produce an output of 0.95 for the node representing the character or numeral being learned and 0.05 for all the other output nodes. To further reduce the size of these neural networks, improving the training and test speeds, structural learning with knowledge [15] is used to supplement the error back propagation method.

Chara-cter	Bifur points	End points	Corner points	Straight Lines	Angles	Curves
0	0	0	0	0	0	1
D	0	0	0	2	0	2
0	0	0	0	0	0	1
1	0	2	2	1	0	0
2	0	2	3	3	0	1
3	0	2	2	0	0	1
5	0	2	3	3	0	1
7	0	2	1	2	1	0
C	0	2	0	1	0	2
G	0	2	0	0	0	1
I	0	2	2	1	0	0
J	0	2	0	1	0	1
L	0	2	1	2	0	0
S	0	2	2	4	0	2
U	0	2	0	0	0	1
Z	0	2	0	3	1	0
6	1	1	1	2	0	1
9	1	1	1	2	0	2
E	1	3	2	5	0	0
F	1	3	1	4	0	0
P	1	1	1	3	0	1
Q	1	1	1	2	0	1
T	1	3	0	2	0	0
V	1	3	0	3	2	0
Y	1	3	0	3	2	0
4	2	2	1	4	0	1
8	2	0	0	0	0	2
A	2	2	0	2	2	1
B	2	0	2	2	0	2
H	2	5	0	5	0	0
K	2	4	0	4	2	0
N	2	4	0	5	2	0
R	2	2	1	5	1	1
X	2	5	0	5	4	0
M	3	5	0	7	2	0
W	3	5	0	7	2	0

Table 2. All extracted character features.

6. Genetic algorithm

The neural network training and the character template creation data is extracted using a genetic algorithm for better normalization. The genetic algorithms can extract character

	Features	Characters in Group
1	No Bifurcation or End points	0, D, O
2	Angled lines (3 or less)	7, Z, V, Y, A
3	Angled lines (3 or more)	4, K, N, X, M, W
4	More than 3 straight lines(no angles)	E, F, H, T
5	Curves and Straight Lines	2, 5, J, R, Q, P, B
6	Curves Only	3, C, U, 6, 9, 8, S
7	Straight lines only	1, I, L

Table 3. All extracted character features.

regions invariant to size and orientation. The two characteristics are vital especially in neural network training.

Based on the selected character region size, the largest parameter that must be coded is 20. Therefore, 5 bits are used to code each parameter and the orientation angle. This information is used to determine the length of the genetic algorithm chromosome. The genetic algorithm determines the position, size and shearing angle of each sample. Moreover, the scaling rates and the translation values each require 5 bits. Therefore, the genetic algorithm chromosome length is 36. This genetic algorithm is binary coded to allow for bit manipulation during training.

The GA chromosome is designed as follows;

1. The position of the sample is represented by l_x and l_y.
2. Translation in the x and y directions by t_x and t_y respectively.
3. Rotation angle by ang. The angle changes are from $-20 \leq ang \leq 20$
4. The scaling rate by scl. Scale of 1 is 20 pixels. The maximum allowed expansion is up 32 pixels (This is the maximum number we can represent using binary code).

The character segmentation genetic algorithm's chromosome is shown in Fig.8.

The fitness of the GA is based on the color of the characters. During reproduction, we save the elite solution, and then use the top 40% of the fittest individuals to reproduce 80% of the next population. The remaining 20% of the next population is reproduced by selection of one of the parents from the top 40% group and the other from the remainder of the population. This method improves the search space by ensuring that we not only retain the best individuals for reproduction but also explore the rest of the population for other possible candidates. The matching process is terminated after 50 generation and the elite solution is the character for training the neural network or creation of the template.

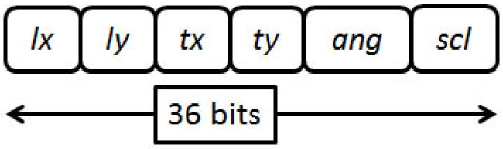

Figure 8. Character segmentation GA's structure.

7. Template matching

Although template matching is computationally expensive especially for large images, it is used in this work as a preprocessing step to help divide the characters into different higher classes. To reduce the computational cost, we have selected a relatively small character region.

The templates used in this work for each of the characters are constructed from the same data used to train the neural network. Therefore the initial size of the template is 15*20 pixels. Each template is the average of 10 images selected at random from the minimum 20 images extracted for neural network training. Note that the height and the width of the template are fixed.

8. Experiments

This work uses license plates images of vehicles that were taken near a parking lot with the target vehicle coming towards the camera and then turning towards the right or left[1] for license plate character recognition and air characters captured in our laboratory as explained in sec. 2.2. We use 3 subjects each writing the characters a total of 4 times.

However, the characters ″I", ″O",″T" and ″Z" are missing in the license plate database.

This work is carried out using a computer with the following specifications. The processor is Intel core 7 CPU, operation at 3.47GHz and an installed memory of 4GB.

8.1. Procedure

The overall system procedure is as follows.

8.1.1. Training

1. Train the neural networks separately, for the licence plates and air characters
2. For the license plate, there are about 10 examples per character
3. For the air characters, use 3 characters from each subject, total 9 examples each.

8.1.2. Testing

1. Extract the character features and use them to decide the neural network to learn.
2. Use template matching to make an initial guess.
3. Run the neural network to confirm the results from step 2.
4. If results of steps 2 and 3 are same, end.
5. Otherwise use the results of the neural network as the final result.

9. Results

There are two experiments in this work. License plate and air character recognition. The experiments are still ongoing but we can report that we achieved better results than those in [1], in license plate recognition. The air characters have so far produced an accuracy of over 94%.

For each character region, the neural network and template matching method were initially tested individually. To use the two methods in a hybrid system, this time we make an initial guess using the template matching method and confirm the results using a neural network.

The results of these computer simulations for character recognition are shown in table 4. These are the average results for all the 30 characters learned. The total number of characters used for testing was 4268.

Table 4 shows the results of character recognition for license plates recognition.

	Percentage Accuracy (%)			
Method	Letters	Numbers	Average	Average time (Msec)
NN only	96.1	97.3	95.2	5.0
TM only	89.3	91.0	90.3	5.0
Hybrid	96.3	98.3	96.8	6.0

Table 4. Character Recognition Results for license plates

Table 5 shows the results of character recognition for air characters.

	Percentage Accuracy (%)			
Method	Letters	Numbers	Average	Average time (Msec)
NN only	92	94	93	12.0
TM only	90	89	90	13.0
Hybrid	93	95	94	15.0

Table 5. Character Recognition Results for air characters

For comparison, the method described in [1] achieved an accuracy of 94% using a neural network and template matching for licence plate detection only. The results of this work show a 3% accuracy improvement because of the different features used.

10. Discussion

Initially, we hoped to use the same neural network to recognize all characters weather printed or virtual. However, although human visual observation make it look like they are similar, the results of the thinning process produced completely different rather surprising results. Therefore, two system are required.

Neural network training depends on the number of training samples available. In this work, the samples used per character vary between a minimum of 10 samples for letters (U, X) to a maximum of 200 for letter W and numerals $(7, 9)$. Generally, neural network require a lot of data to train. This phenomenon is also observed here where the accuracy results of characters with more training samples are better.

The air character data collection needs some improvement to reduce the processing time. Although face detection is useful for position extraction etc, it takes valuable time that could be used to improve the character segmentation process.

11. Conclusion

In this chapter, we presented a character recognition method for virtual scenes (air characters) and vehicle number plate recognition using neural networks, template matching and evolutionary computation. We combine neural networks learning, image processing and template matching to create a novel character recognition system. To speed up the system and deal with, size and orientation issues, we employ a genetic algorithm. In addition, to control the size of both the neural network inputs and the template, we apply a genetic algorithm to guide the search. Average accuracy of about 97% and 94% were achieved for the license plate and virtual characters respectively.

In future we must find ways to combine the different recognition systems to universally recognize all characters and expand the work to include the recognition of lower case characters as well. Computation time especially for air character recognition must also be improved.

Author details

Stephen Karungaru, Kenji Terada and Minoru Fukumi
Department of Information Science and Intelligent Systems, University of Tokushima, Japan

12. References

[1] Stephen Karungaru, Minoru Fukumi, Norio Akamatsu and Takuya Akashi (2009). Detection and Recognition of Vehicle License Plate using Template Matching, Genetic Algorithms and Neural Networks, *Trans. of International Journal of Innovative Computing*, Information and Control, Vol.5, No.7, pp.1975-1985.

[2] Ching-Tang Hsieh, Yu-Shan Juan, Kuo-Ming Hung (2005). Multiple License Plate Detection for Complex Background, *Advanced Information Networking and Applications* , pp.389-392.

[3] Jun-Wei Hsieh, Shih-Hao Yu, Yung-Sheng Chen (2002). Morphology-Based License Plate Detection from Complex Scenes, *Proc. of International Conference on Pattern Recognition* , pp. 176-179.

[4] Yanamura Y., Goto M., Nishiyama D, Soga M, Nakatani H, Saji H (2003). Extraction And Tracking Of The License Plate Using Hough Transform And Voted Block Matching, Proc. of IEEE IV Intelligent Vehicles Symposium , pp.243-6.

[5] Kamat V., Ganesan S (1995). An efficient implementation of the Hough transform for detecting vehicle license plates using DSPĄfS, *Proc. of Real-Time Technology and Applications Symposium*, pp.58-9.

[6] Viola P., Jones M (2001). Rapid Object Detection Using a Boosted Cascade of Simple Features, *Proc. of Computer Vision and Pattern Recognition* , vol.1, pp.511-518.

[7] Chih-Chiang Chen, Jun-Wei Hsieh (2007). License Plate Recognition from Low-Quality Videos. *Proc. of the IAPR Conference on Machine Vision Applications*, pp. 122-125.

[8] P. Viola and M. J. Jones (2004). Robust real-time face detection, *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154.

[9] Stephen Karungaru, Minoru Fukumi and Norio Akamatsu (2005). License Plate Localization Using Template Matching, *Proc. of 9th International Conference on Mechtronics Technology*, Vol.1, No.T4-3, pp.1-5, Kuala Lumpur.

[10] Y. Abe, M. Konishi and J. Imai (2007). Neural network based diagnosis system for looper height controller of hot strip mills, *International Journal Innovative Computing, Information and Control* , vol.3, no.4, pp.919-935.

[11] Fekih, A., H. Xu and F. Chowdhury (2007). Neural networks based system identification techniques for model based fault detection of nonlinear systems, *International Journal Innovative Computing , Information and Control*, vol.3, no.5, pp.1073-1085.

[12] L. Mi and F. Takeda (2007). Analysis on the robustness of the pressure-based individual identification system based on neural networks, *International Journal Innovative Computing, Information and Control*, vol.3, no.1, pp.97-110.

[13] M.Fukumi, Y.Takeuchi H.Fukumoto, Y.Mitsukura, and M.Khalid (2005). Neural Network Based Threshold Determination for Malaysia License Plate Character Recognition, *Proc. of 9th International Conference on Mechatronics Technology*, Vol.1, No.T1-4, pp.1-5, Kuala Lumpur.

[14] Kah-Kay Sung (1996). Learning and example selection for object and pattern recognition, PhD Thesis, MIT AI Lab.

[15] M. Ishikawa (1993), Structure learning with forgetting, *Neural networks journal* , Vol. 9, No. 3, pp 509-521.

[16] Parker, J., R. (1994) ”Practical Computer Vision using C", Wiley Computer Publishing.

[17] Zhang, T. Y. and Suen, Ching Y. (1984) ”A Fast Parallel Algorithms For Thinning Digital Patterns", Communication of the ACM, Vol 27, No. 3, Maret 1984, pp.236-239.

[18] Duda, R. O. and P. E. Hart (1972) ”Use of the Hough Transformation to Detect Lines and Curves in Pictures," Comm. ACM, Vol. 15, pp. 11-5.

[19] Kenney, J. F. and Keeping, E. S. (1962) "Linear Regression and Correlation." Ch. 15 in Mathematics of Statistics, Pt. 1, 3rd ed. Princeton, NJ: Van Nostrand, pp. 252-285

[20] Stephen Karungaru, Minoru Fukumi, Norio Akamatsu (2005), ”Genetic Algorithms Based On-line Size and Rotation Invariant Face Detection," Journal of Signal Processing, Vol.9, No.6, pp.497-503, November 2005.

[21] Eric W. Brown, 2010, ”Character Recognition by Feature Point Extraction", http://www.ccs.neu.edu/home/feneric/charrec.html.