A Virtual Character Recognition System Based on Optical Detection of Red Light and its Embedded Implementation

Kai Ding, [+]Lianwen Jin, Hanyu Yan
College of Electronic Information
South China University of Technology
Guangzhou Guangdong, China
[+]E-Mail lianwen.jin@gmail.com

***Abstract*—This paper proposes a novel "air-writing" character recognition system (ACRS) based on optical detection of red light, with which a user can write a character in the air with a red light-emitting device. The trajectories of the light can be captured and detected by a camera during the writing process and then a character reconstruction algorithm is employed to convert them to a 2-D plan (as a virtual character trajectory) after a serials of preprocessing steps, including background analyzing and noise erasing. Finally, the virtual character trajectory is considered as a kind of inkless character pattern and recognized by a minimum classifier. To expand the application of our system, we implement an ARM7-based embedded "air-writing" character recognition system (E-ACRS). Experiments show that both ACRS and E-ACRS are able to recognize cursive digits and English alphabetical characters with encouraging accuracy.**

I. INTRODUCTION

The computer was created several decades ago, and since then many technological breakthroughs have occurred. But most computers still receive input from traditional low-bandwidth devices such as a keyboards or a mouse[1]. As we know these devices are both inconvenient and unnatural for providing three-dimensional or high degrees-of-freedom inputs, which motivates the research on Human-Computer Interaction (HCI) to develop a machine can understand audio, visual, or touch-based information[2].

Vision-based HCI, proposed to make a machine that can "see" objects like humans[3, 4, 5], is a popular field in HCI, and can be associated with many technologies to expand their application[6, 7, 8, 9]. For instance, vision-based character recognition is a novel application among them, and many novel applications, such as visual-based pen input[5], finger-writing virtual character recognition[8, 9] and so on, are presented in the literature.

In this paper, we propose a novel application based on vision HCI: a virtual "air-writing" character recognition system based on optical detection of red light, which we call an "air-writing" character recognition system (ACRS). The basic idea of ACRS is that users can write characters virtually with a red light-emitting device in the air in a wireless fashion. With this characteristic, our system can be used as a novel tool with interesting applications, such as demonstrations in the conference, classes, and lectures.

The diagram of the proposed ACRS is shown in Figure1. The ACRS process can be described as follows: The trajectories of virtual characters are detected and tracked in real time while a user writes characters in the air with a red light-emitting device, and then a character reconstruction algorithm is employed to convert them to a trajectory on a 2-D plan (virtual character trajectory) after a serials of preprocessing steps, including background analyzing and noise erasing. Finally, the recognition algorithm based on a minimum classifier recognizes the virtual character. To expand the application of ACRS, we implement it on an ARM7-based embedded platform (we call this system E-ACRS), which is much smaller, cheaper, and more convenient.

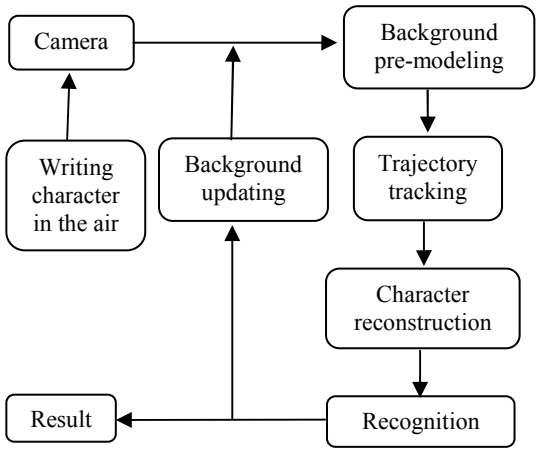


Figure 1. Diagram of the ACRS.

This paper is partially sponsored by fundings from GDSTP (no. 2007B010200048) and GZSTP (no. 07A1208009)

II. SYSTEM AND ALGORITHMS

Figure1 shows an overview of our system, which is constituted by three modules: background analyzing (which includes background pre-modeling and updating), trajectory tracking and character recognition (which consists of character reconstruction, feature extraction, and classification). Details of these modules will be introduced in the following sections.

A. Background analyzing

Background analyzing is an important and challenging module of our system. A typical method used to differentiate pixels in the disparity images that belong to character trajectory from those that belong to the backgrounds is background subtraction. A model of the background is maintained, and each disparity image is compared to that model on a pixel-wise basis. Pixels that are sufficiently different from the accumulated statistical model at each pixel are considered to be part of the foreground[10].

In ACRS, the characteristics of background can be summarized in the following three aspects. First, as shown in Figure2, the image captured by a digital camera with special red light filtering is binary. Therefore, the color of some image patterns, which are generated by the noise, is the same as the color of the character trajectories. Second, as shown in Figure3, the background is unsteady and will be changed during the writing process according to the movement of users' body or hand. Third, the shape and the size of the "white area" (which color is white in our demo) captured vary greatly during writing process (Figure3).

Characteristic 1 means that color information cannot be used to discriminate between the background and foreground, and characteristics 2 and 3 indicate that the background is unsteady and the characteristics of the trajectory point hard to obtain. All of these characteristics state clearly that the typical background subtraction methods are not suitable for our system.

To overcome these problems, some simple but effective techniques are proposed in this section. The basic assumption of our algorithm is that the area of the noise will not be enlarged greatly during the writing process. For the writing process is very short (about 1 or 2 seconds), so this assumption is reasonable and suitable for practical situations.

Under this assumption, the background model of our system is constituted by two separate models: a pre-model and an updating model. The pre-model is obtained before writing the current character and is utilized as the basic background model of our algorithm. The updating model is acquired before writing the next character and the function of this model is to update the pre-model if the area of the background is enlarged before writing next character.

Besides the above techniques, we also implement several other techniques to increase the stability of our system. (1) Enlarge the noise area of the background model to avoid the effect of slight movements of the background and (2) give a warning message to customers if area of noise area is too large. If the trajectories we obtain are much different from real trajectories in such situations, the performance of the ACRS will be greatly degraded.

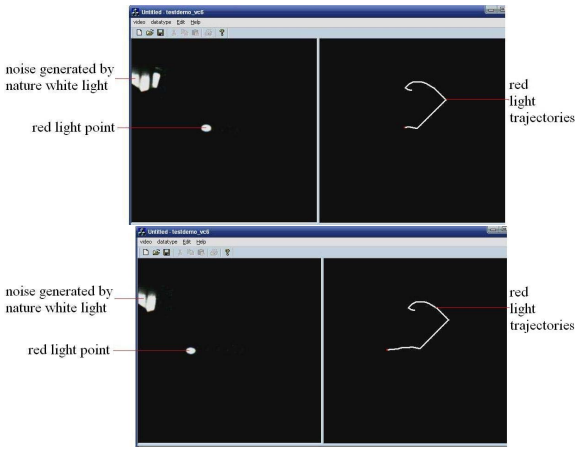


Figure2. Variation of background.

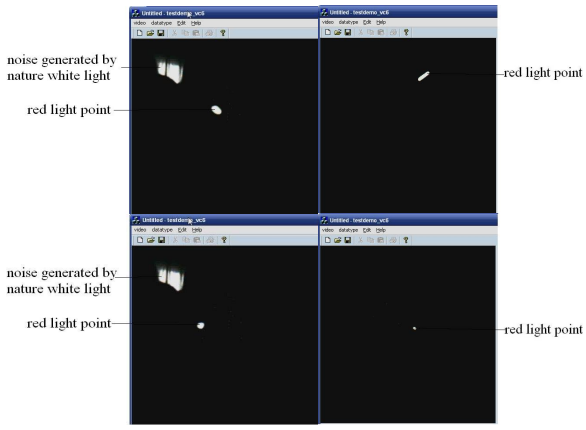


Figure3. Variation of the shape of trajectory point

B. Trajectory tracking

A trajectory point captured by the camera is not just only a point but an area (shown in Figure2 and 3), so we call the area of ***the trajectory points*** the trajectory area and the center of this area as ***the trajectory point***, which is the point we want to obtain in our trajectory tracking algorithm.

Trajectory tracking is the most crucial element of ACRS. Reducing the complexity of the tracking algorithms and judging whether the detected point is the trajectory point or not are the key issues of this module.

1) Algorithm of background elimination

To explain algorithms clearly, we set the value of points that is white in Figure 2 and 3, as 1 (we call them "white points" or "white areas", for the color of them is white in our demo), and the value of the other points as 0.

The algorithm for traditional background subtraction, which is employed just before detecting trajectory points in traditional methods, is described as follows: Suppose B is the background model, and A is the image with the trajectory area and noise. The whole trajectory area could be extracted through the following equation:

$$C[i][j]=\begin{cases}1 & if \quad A[i][j]-B[i][j]>0\\ 0 & other\end{cases} \qquad (1)$$

where $C[i][j]$ represents the pixel that is located in ith row and jth column of image C. And then the trajectory point is searched for in image C.

In this traditional algorithm, both the image *A* and *B* are checked thoroughly and the formulation (1) is employed by all the points in image *A* and *B* however, our interesting area are the "white areas" (the value of points in these areas is 1), which are just a very small part of the image. Therefore, much unnecessary computation is implemented in this algorithm.

To decrease the algorithm complication, we present an improved background elimination algorithm in this section: first the points in image *A* are checked according to a given order: when the "white points" are found, then the formulation (1) is employed to judge whether the point is the trajectory point or not. According to this, only the searched "white points" in image *A* are processed by formulation (1).

2) Algorithm of trajectory tracking

Using prior information properly can promote the performance of algorithms greatly. As we known, the starting point of a character is probably in the upper left portion of the image, and the other points of the character are near their previous points. Accordingly, we set the initial point of our tracking algorithm as point (0, 0) if the point we are detecting is the starting point of the character, otherwise, the previous trajectory point will be set as the initial point of our algorithm. And then a fast and robust trajectory tracking algorithm, which is described below, is implemented to obtain the trajectory point.

To explain the trajectory tracking algorithm clearly, a distance measure is introduced first before describing the details of the algorithm:

$$SCD(M,N)=k \ if \ \ k\le \| M-N \| \le k\sqrt{2} \qquad (2)$$

where *M* and *N* represent vectors with two dimensions and "|| ||" means the norm of a vector. And $SCD(M,N)$ is the distance between *M* and *N* in this distance measure.

The process of the algorithm is described as follows:

a. Coarse location: the purpose of this step is to find the coarse position of the trajectory point. As shown in Figure4, suppose point A is the initial searching point, and then the points with the least values of $SCD(A,P)$ around point *A* will be searched first. Finally, a random point in the trajectory area, just like the point B in Figure4, can be found rapidly and considered as the coarse location of trajectory point.

b. Determine the trajectory point: As shown in Figure4, a horizontal line and a vertical lines are constructed based on point B, and then the cross point C, D, E, F will be obtained easily. Next, the top, bottom, left, and right points of the trajectory area will be rapidly searched through the following search algorithm: set F as the initial point of the search algorithm, and then the points above F will be searched till the top point of the trajectory area is found. The bottom, left and right points can be obtained similarly. Finally, the circumscribed rectangle of the trajectory area is constructed and the trajectory point is set as the center of the rectangle.

c. Judging step: As shown in Figure3, though the shape and the size of the "white area" vary greatly, its size must be in a range, thus it can be determined whether the detected point in step (2) is the real trajectory point or not. For example, if the width or height of the circumscribed rectangle constructed in step (2) is too large, or the ratio between them is unreasonable, our algorithm will refuse to accept this point as the trajectory point. And then the trajectory tracking algorithm will be continued until the trajectory point is obtained, or just use the previous trajectory point if no trajectory point is found in this frame.

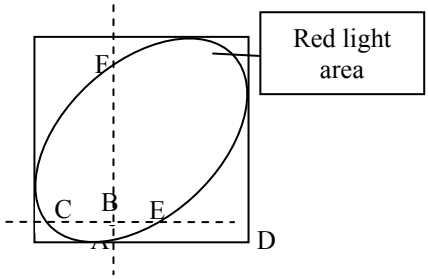


Figure 4 Diagram of detecting algorithm

3) Algorithm of noise erasing

Some of the algorithms indicated above not only can accomplish their purpose, but can also erase noise. For example, the background updating algorithm can eliminate the noise generated by the movement of the background before writing. The trajectory tracking algorithm can eliminate the noise that is located in the unsearched areas, because the variation of background in the unsearched area will have no effect on the trajectory tracking algorithm, which means that the noise in these unsearched area is erased by our algorithm indirectly. For example, as shown in Figure2, though the background is changing during the writing process, our algorithm still can track the trajectories properly.

C. Character recognition

1) Reconstruction of air-writing character

The reconstruction of the virtual character is implemented by linking all detected trajectories together. However, because the trajectories of the character are inkless and cursive, one important issue is how to make them more regular. In this section several algorithms are proposed to solve this problem:

a. Smooth filtering: the purpose of this step is to reduce stroke shape variation, which is produced by the unconscious movement of the user's hand during the writing process. Suppose the array P represents the whole trajectories, and the ith point of the trajectories $P[i]$ is modified by the following moving average smoothing filter:

$$P[i]' = \frac{1}{2t+1}\sum_{k=i-t}^{i+t} P[k] \quad (3)$$

where $t <= i <= N - t$, N is the total number of trajectory points, and t (whose value is 2 in our system) is an empirical parameter.

b. Connecting adjacent points: the digital differential analyzer[11] (DDA) algorithm is utilized in this step to connect to consecutive points.

c. Re-sampling: Re-sampling is for the purpose of reducing the distance variation between two adjacent trajectory points, and eliminating the effect of writing speed variation. The sequence of the trajectory points in each character is re-sampled by a sequence of equidistance points (a distance of 3 unit lengths is used in our system).

d. Connecting adjacent points: the DDA algorithm is employed to link all the re-sampled points to construct a 2-D character image, and some reconstructed samples are shown in Figure5.

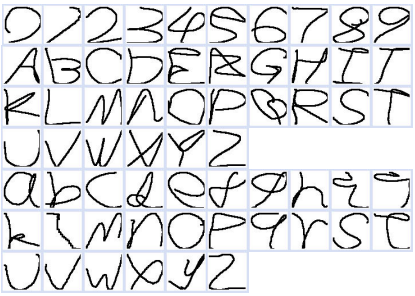


Figure5. Reconstructed character samples

2) Feature extraction and classification

Because all strokes of reconstructed characters are connected (because it is difficult to distinguish the start and the end point of a stroke when the character is written in the air with the light emitting device) and our algorithms need to be implemented in an ARM7 hardware platform, the feature and classifier algorithms should be effective and involve less computation. The Gradient feature[12] (an effective feature for handwritten character recognition[13, 14]) and minimum distance classifier (the most simple classifier in pattern recognition) are used in our ACRS and E-ACRS.

III. E-ACRS

For PC platforms, the advantage of the ARM7 based platform is lower cost, smaller size, and broader application et al. Therefore, we implemented our ACRS into an ARM7 platform with and eye to expanding the application of ACRS

The ARM7 we adopt is the WinBond platform[15] with CPU frequency of 81 MHZ and 8 M RAM memory. The characteristic of this platform is represented in three aspects: (1) Support of TV or VGA display through utilizing GUP and LCD/TV driver on the WinBond CPU. (2) A friendly GUI easily implemented through the advanced 2D/2.5D effect accelerator. (3) Support of Video-In, CMOS driver, and DMA channel that can reduce the cost of the CPU greatly.

The diagrams of our ARM7-based ACRS hardware and system architecture are shown in Figure6 and 7.

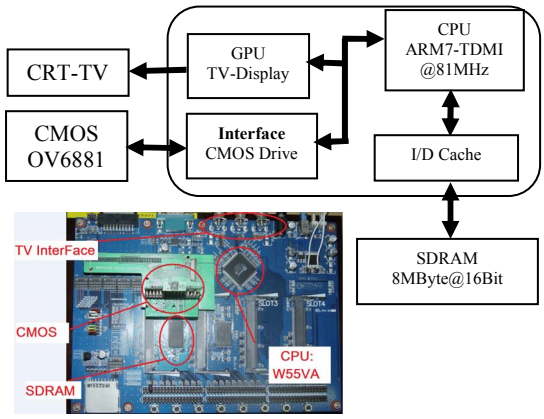


Figure 6. Diagram of hardware structure

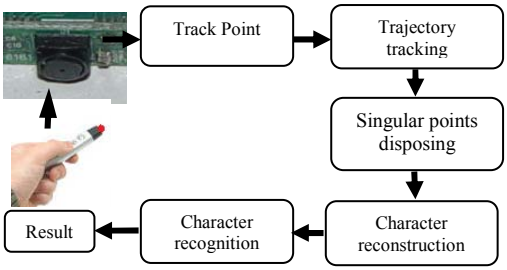


Figure 7. System architecture of E-ACRS

Compared with ACRS, the characteristics of E-ACRS are different in two aspects: First, the background analyzing process is omitted in E-ACRS; second, the process of singular point disposition is carried out before character reconstruction. The details and the reason of these changes will be presented below.

To make our system much more convenient for practical use, we utilize the light emitting device to realize an additional function—"mouse function", which means that the function of the light-emitting device is similar to a mouse. Considering this, we call the light-emitting device an "air mouse".

A. Trajectory capture

For the limited resources of the ARM7-based embedded platform, we can't implement the algorithms just as on a PC platform and many algorithms need to be optimized and simplified. Two aspects of the algorithm optimization of this module are indicated below.

The original resolution of camera is 400×400, which would cause heavy computation in the tracking and searching algorithms. Therefore, we zoom out the original resolution to 128×128 linearly, which decreases the amount of processing data and cost time without degrading the performance of E-ACRS seriously.

Since heavy computation mostly occurs in the background analyzing algorithm, this process is omitted in our E-ACRS to

simplify the algorithm complexity. Alternatively, the red light filter and red light LED are replaced by an infrared filter and an infrared LED respectively. Because infrared sources in normal scenes do not have enough energy to activate the filter, therefore the noise, which is generated by natural infrared sources, can be erased by the infrared filter. Therefore, the background analyzing module is not needed and can be left out in our E-ACRS.

B. Trajectory tracking

Based on experimental observations, we found that the performance of our system is not sensitive to the exact location of the trajectory points; especially the smoothing algorithm is implemented in the character reconstruction module. Therefore, we can set the coarse position of trajectory area as the exact position of the final trajectory point, just like point B in Figure4.

C. Singular point disposal

Singular points are often generated by system noise, such as impulse current, in the E-ACRS. In the trajectory tracking module, these points can be viewed as the trajectory point (as the point *B* shown in Figure8), which affects the performance of our system greatly.

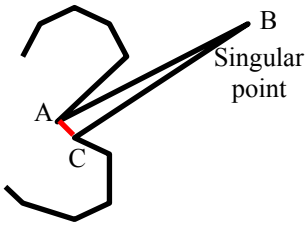
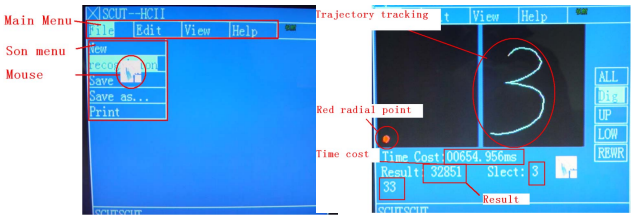


Figure 8. Algorithm for judging singular points

In this section, some strategies are presented to identify and eliminate the singular points: suppose points *A*, *B*, *C* are three consecutive points, as shown in Figure8. The point B will be considered as a singular point if the following two conditions are satisfied: (1) $\angle ABC$ (the value of angle between line *AB* and line *BC*) is less than a limiting value; and (2) the distances between points *A* and *B*, and points *B* and *C*, are both much larger than the distance between points *A* and *C*.

D. Air mouse

In this section, a particular implementation—air mouse, is utilized in our E-ACRS to make our system much more convenient in practical applications.



(a) Demonstration of air mouse (b) UI of E-ACRS

Figure9. "Air mouse" & UI of E-ACRS.

The basic idea of the air mouse is that users can use the air mouse to implement some complex operations just like the mouse for a PC. It means that the light emitting device not only can be used as a character input device during writing state but also can be used as a "mouse" to implement other operations during a non-writing state (as shown in Figure9). Besides, a wireless transceiver module is designed to send some given values to the main processing platform to implement some additional functions, such as recognition candidate selection and menu selection. With no doubt, our air mouse also can be considered as a wireless mouse and utilized in many other situations.

IV. EXPERIMENTS

To evaluate the performance of our systems, a series of experiments were conducted for the purpose of the recognition of isolated digits and uppercase and lowercase English characters. A subset of the handwriting online character database, released by China 863 Project, is used as training data. The dataset contains 60 writers' samples. We collected the testing database using ACRS and E-ACRS. It contains 10 sets of digits and uppercase and lowercase English characters written by different people.

The original dimension of the gradient feature is 128, and linear discrimination analysis (LDA) is implemented to reduce the feature dimension to 40. The result for the training and testing databases are shown in Table 1 and 2 respectively.

TABLE I. EXPERIMENT RESULT ON TRAINING DATASET

Database	Recognition results
digit	94.67%
upper case	95.13%
lower case	88.97%

TABLE II. EXPERIMENT RESULT ON TESTING DATASET

Platform	"Top-N" recognition results	Database		
		digit	uppercase	lowercase
PC	*N*=1	89.00%	70.96%	77.31%
	N=3	97.00%	94.62%	95.38%
	N=5	98.00%	98.85%	97.69%
ARM7	*N*=1	80.00%	80.76%	65.38%
	N=3	90.00%	90.38%	84.61%
	N=5	100.00%	96.15%	94.23%

From Tables 1 and 2, we can see that the recognition accuracy is not good enough for testing dataset. This may be because our testing data (which are captured by ACRS and E-ACRS) are in a cursive, one-stroke style, which is quite different from the training data (collected by normal digital pen or tablet). As shown in Figure5 and 10, the samples we collected are cursive or slanted; some of them are hard to identify even by eye. However, even we use un-correlated training and testing datasets, the accuracy of "Top-3" and "Top-5" recognition results are encouraging in both ACRS and E-ACRS. Besides, the average processing time (containing the time of reconstruction, feature extraction, and classification) is 631ms/character for E-ACRS.

Figure10. Cursive samples of 2, 8, N, Q, d, g, i, j

V. CONCLUSION

In this paper, we describe a novel "air-writing" character recognition system (ACRS) based on optical detection of red light and its embedded implementation (E-ACRS). With them, users can input characters virtually in the air just using a simple light-emitting device, without any additional device such as keyboard, touch screen, or digital pen. Our systems provide a novel interesting character inputting modality.

Our preliminary experiments show that the performance of our ACRS and E-ACRS is encouraging, although they still need to be improved to meet practical requirement. Further investigations, such as collecting large samples by our ACRS as training datasets; solving the problem of one-stroke style cursive character recognition; and automatic slant detection and correction, can help make our system more robust and practical.

REFERENCES

[1] Maybury, M.T. (ed). Intelligent Multimedia Interfaces. MIT Press, 1993

[2] Lam, K.M. and Yang, H. Locating and extracting the eye in human face images. *Pattern Recognition,* Vol.29, No. 5, 1996, 771-779.

[3] V. Pavlovic, R. Sharma et al., Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review, *IEEE Transactions on PAMI,* 1997, Vol. 19, No. 7, pp. 677-695.

[4] Kenji Oka, Yoichi Sato, Hideki Koike, Real-time Tracking and Gesture Recognition, *IEEE Trans. Computer Graphics and Application*, December 2002, pp. 64-71.

[5] M.E. Munich,; P. Perona, Visual input for pen-based computers, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 24 no. 3, 2002. pp. 313 -328.

[6] C. Hardenberg, Francois Bérard, Bare-Hand Human Computer Interaction, *Workshop on perceptive User Interface,* Florida, USA, Nov, 15-16, 2001, pp. 1-8.

[7] F. Quek, et al., Finger mouse: A freehand pointing interface, *Proc. of International Workshop on Automatic Face and Gesture Recognition*, Zurich, Switzerland, 1995, pp. 372-377.

[8] Lianwen Jin, Duanduan Yang et al, A Novel Vision based Finger-writing Character Recognition System, *Proc. of ICPR*, 2006, Vol. 1, pp. 1104-1107.

[9] Duanduan Yang, Lianwen Jin et al, An Effective Robust Fingertip Detection Method for Finger Writing Character Recognition System, *Proc. of the Fourth International Conference on Machine Learning and Cybernetics*, Guangzhou, 18-21 August 2005, pp. 4991-4996.

[10] K. Toyama, J. Krumm, B. Brumitt and B. Meyers, "Wallflower: Principles and Practice of Background Maintenance," In Proc. of the International Conference on Computer Vision, 1999.

[11] Siyuan Chen, Qunzhan Zhang, Computer Graphics, 1 edition, press of metallurgy industry, Beijing, 2003.

[12] Chenlin. Liu, K. Nakashima, et al, Handwritten digit recognition: investigation of normalization and feature extraction techniques, *Pattern Recognition*, 37(2), pp. 265-279, 2004.

[13] Chenglin Liu, Masashi Koga, Hiromichi Fujisawa, Gabor Feature Extraction for Character Recognition: Comparison with Gradient Feature, *In Proc. 8th ICDAR*,2005, vol.1, pp. 121-125.

[14] Kai Ding, Zhibin Liu, Lianwen Jin et al., A Comparative Study of Gabor Feature and Gradient Feature for Handwritten Chinese Character Recognition, *Proc. of ICWAPR,* 2007, pp. 1182-1186.

[15] http://www.winbond.com.tw/china/chs/.