# Video-Based Handwritten Chinese Character Recognition

Xiaoou Tang, *Senior Member, IEEE*, Feng Lin, and Jianzhuang Liu, *Senior Member, IEEE*

***Abstract*—In this work, we propose a video-based handwritten Chinese character recognition system. By using several error correction techniques, the algorithm works effectively against various shadow and noise problems for video-based stroke-tracing of complex Chinese characters. The stroke temporal information similar to an online OCR system is accurately extracted. Recognition experiments on a large set of handwritten Chinese characters clearly demonstrates the efficacy of the system.**

***Index Terms*—Handwritten Chinese characters, online stroke tracing, video-based OCR.**

## I. Introduction

CHINESE characters are used by nearly one-quarter of the population in the world. However, several decades after the computer was introduced into the Chinese community, the input of Chinese characters into computers is still a difficult problem, primarily because the Chinese language is not alphabetic. Each Chinese character is a beautiful, but complicated square graph composed of different strokes organized in a highly artistic and imaginative manner. Handwritten character recognition is an efficient alternative for Chinese character computer input. However, with over 5000 categories of characters, the handwritten chinese character data are complex, diverse, and deformable, thus are very difficult for automatic recognition.

Handwritten character recognition has been studied for many years. The methods for recognizing handwritten characters are generally divided into two categories on the basis of data input: optical methods (offline) and pen-based methods (online) [1]. The offline systems usually use optically scanned character images as input data [6]–[15]. Because of the difficulty in extracting reliable stroke information and the missing of temporal information, classification accuracy of most offline systems remains too low for practical usage.

The online-based systems, on the other hand, have attained high recognition rates, and many commercial systems are already available [2]–[5]. This is mostly because the stroke temporal information is recorded for each character. However, one has to use an extra writing board as the input equipment. Writing on the slippery surface of the writing board is cumbersome and time consuming. The extra cost of the pen and board, as well as the inconvenience of using them, greatly undermines the popularity of pen-based online systems.

In this work, we develop the first video based handwritten Chinese character recognition system that combines the advantages of both the online and offline approaches. Since writing on paper is the most natural way for handwriting, the system allows users to write on any regular paper just like using the offline system. However, instead of waiting for a whole page to be written and then scanned into a computer, we use a video camera attached on the computer to capture a sequence of the character images while it is being written on the paper. Then using video processing techniques, the temporal information of each stroke is extracted. Thus we can capture the stroke temporal information similar to the online system. This allows the system to achieve high recognition accuracy like an online system, but without requesting users to write awkwardly on the slippery surface of an online writing board. Unlike the online system writing board that has to be purchased separately, most computers nowadays have a web camera attached for net meeting type of application, therefore, no additional special equipment is required for the system.

In addition, the camera provides the computer a convenient way to interact with users through computer vision. Besides recognizing handwriting on paper, the computer vision system developed for the VCR system can also be adapted for recognition of sketches on classroom blackboard, which has great potential for applications in video conferencing and remote video lecturing. Another advantage of the camera-based system is the ability to miniaturize the system [17]. For handheld computer systems, the size of the input systems is a key limiting factor for further reducing the size of the system. Comparing to keyboard and touch screen, a web camera can be much smaller, limited only by the size of the small lens.

The key step of the VCR system is to effectively extract the stroke dynamic information from the video sequence. Once the stroke dynamic information is extracted, the recognition process is exactly the same as the traditional online system. To extract stroke dynamic information of a VCR system, a method for video based English character recognition is proposed in [17]–[19]. The method simulates a traditional online system and tries to track the pen-tip motions. First the pen-tip has to be located in each video frame. To locate the pen-tip, the shape of the pen-tip has to be known. Then a Kalman filter is used to track the pen-tip based on position, velocity and acceleration of the pen. The pen-tip trajectory is then segmented using the velocity of the pen-tip and the curvature of the trajectory. Finally, the pen-up and pen-down segments are recognized by a $k$-nearest neighbors classifier. The methods are quite complicated even for the simple English characters. For the complex Chinese characters with many more stroke segments, the method becomes more difficult to use. In order to deal with the large number of strokes in Chinese characters, we develop a new approach using a pixel tracing scheme. The method does

Manuscript received February 28, 2003; revised November 27, 2003. This work was supported in part by RGC Grant CUHK4378/99E and Grant CUHK4357/02E from the Research Grants Council of the Hong Kong.

The authors are with the Department of Information Engineering, The Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: xtang@ie.cuhk.edu.hk; flin@ie.cuhk.edu.hk; jzliu@ie.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCSVT.2004.839975

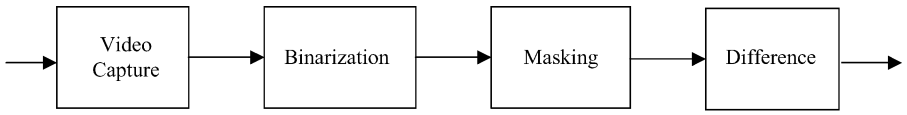

Fig. 1. Diagram for video-based handwritten character pixel sequence extraction.

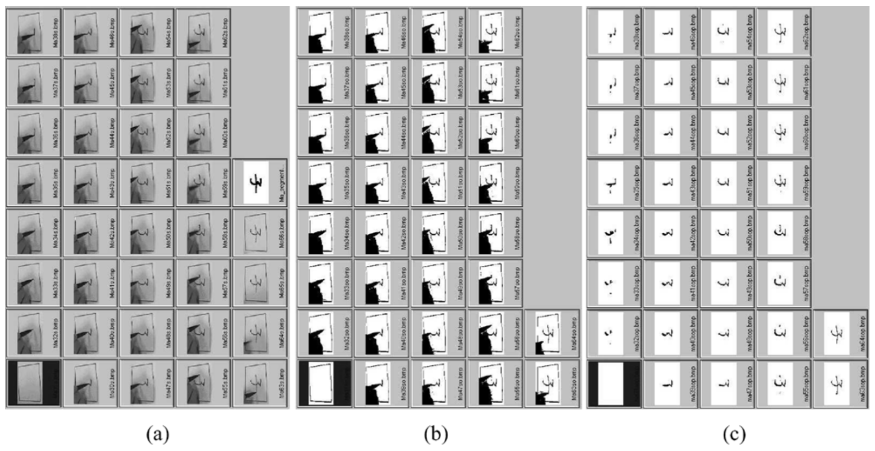

Fig. 2. Video sequence processing example. (a) Original video images. (b) Binary images. (c) Binary images masked by the reference character.

not depend on the shape of the pen-tip. In addition, no filter or classifier is involved in the method.

## II. Simple Pixel Tracing

A simple diagram of the video processing steps is shown in Fig. 1 [16]. As a user writes on a piece of paper, a video camera captures the evolving character continuously. An example sequence of the captured images is shown in Fig. 2(a). These images are then converted into binary images through thresholding. As we can see from the binary images shown in Fig. 2(b), the background noise including the hand, pen and their shadows occupies a significant portion of the binary image. Fortunately, after the character is written and the pen moves away to the next one, the finished character is left alone in a clean image, as shown by the last image in Fig. 2(b). In the third step of the process, using this clean character image as a reference mask, we extract only the character part of each binary image in the video sequence. Fig. 2(c) shows the results of such a masking operation. From the processed image sequence we can clearly see how each stroke is gradually written out. By computing the difference between neighbor images, we can locate both the time (frame number) and location of each written pixel, which give us the similar type of information that is available to traditional online systems.

However, this seemingly straightforward approach does not work most of the time due to the hand and pen shadows in the video sequence. We encounter some difficulties in implementing the last step in the system, i.e., differenciation. As we can see from Fig. 2(c), because of the background noise, many pixels on a stroke appear several times in a video sequence even before they are written. It is difficult to distinguish whether a black pixel in an image is due to a true stroke, a pen shadow, or simply a binarization error. For example, at the beginning of the sequence, instead of the first few pixels on the character, many noise pixels appear because of the pen and hand movement in the image. Simply computing the image difference cannot produce the intended pixel list. Even in the middle of the image sequence, we can also see some noise pixels appear at the tips of the stroke caused by the slight misalignment of images. Such misalignment of images will produce even more noisy pixels after we compute the image differences. In addition, to compute the difference between all neighbor images and later combine all the difference images are not trivial in computation. To derive a stable pixel sequence, we propose the following improved approach.

Instead of computing the image differences, we first conduct a thinning operation on the last finished binary character to reduce the line width to one pixel. Then for each pixel on the thinned character, we look at a $3 \times 3$ neighborhood around the same location in each binary image in the video sequence. If any one value of the nine pixels is nonzero in a image we assign a switch value 1 to that image frame, otherwise a value 0 is assigned. Using a $3 \times 3$ neighborhood can avoid the problem of slight misalignment between images. Tracing through the video sequence from the first image to the last image where the final character is recorded, we obtain a binary switch line shown in Fig. 3 for each pixel on the thinned character. To find the frame number that a pixel is first written, we trace from the end of the binary line in Fig. 3. Since the pixel must be on the finished character, all the binary lines end with switch value one. As we trace back the frames along the binary switch line, sometimes the switch value will turn to zero and remain zero until the first frame at the beginning of the line, as shown in Fig. 3(a). In this case, we can clearly identify the frame (marked by a circle in Fig. 3) at the turning point as the one in which the pixel first appears. We will call this frame the emergence frame (EF).

Sometimes, because of noise, a pixel may disappear from one or a few frames even after it is already written, as shown in Fig. 3(b). To get around such situations, we continue to check

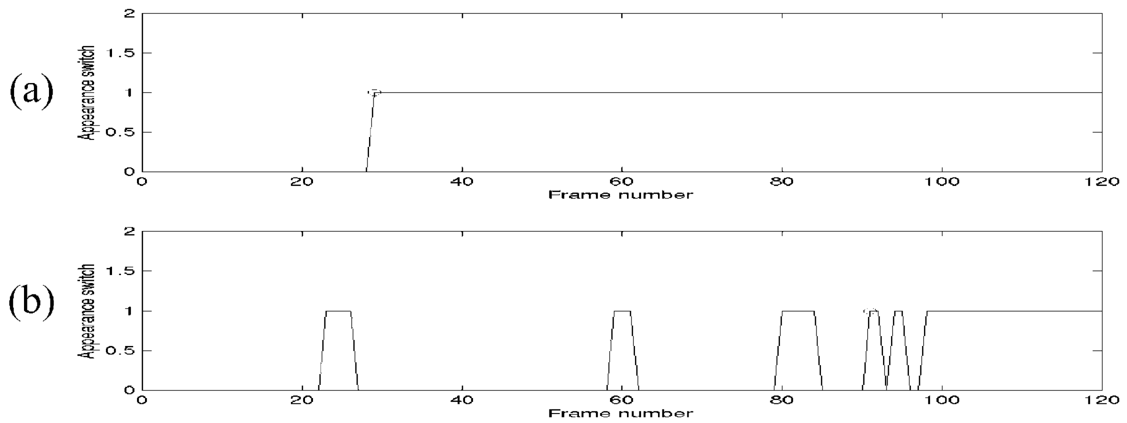

Fig. 3. Sample binary switch lines of two pixels on the character. Symbol "o" marks the frame number that the pixel first appears in the video.

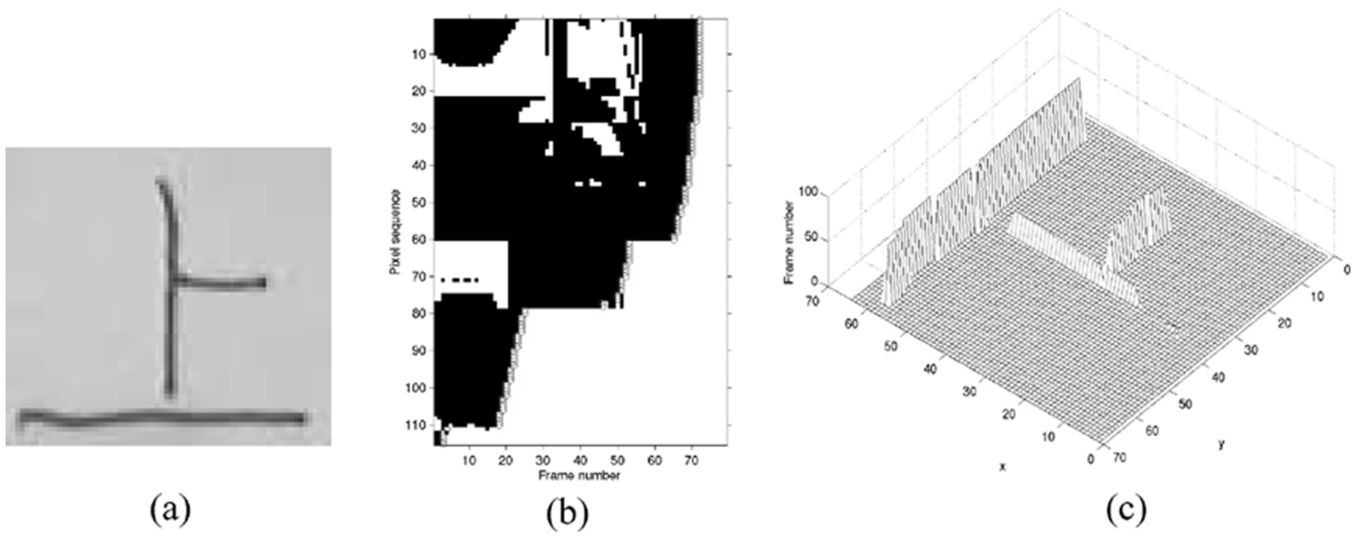

Fig. 4. Character tracing results. (a) Original images. (b) Pixel appearance frame number ("o" marks) trace plot. (c) Appearance frame number plots of character pixels.

the next five frames. If they are all zeros, we confirm the point as the turning point, otherwise the search continues. In Fig. 3(b), we also see that the pixel appeared several times even before it is written. These apparently are due to the hand and pen shadows in the image. We sort the pixels on the character by their EF number and then stack the binary lines of all the pixels into a binary matrix according to the sort order. So the first row of the matrix is the binary switch line of the first pixel on the character, and the last row of the matrix is the binary line of the last pixel on the character. Displaying the binary switch matrix as a black and white image in Fig. 4(b) for the character in (a), we can clearly see the trace line (circles in the image) dividing the one region and zero region. Each monotonically increasing section of the trace line corresponds to one stroke in the character. Each horizontal section of the trace line represents a jump in frame numbers from one pixel to the next, thus corresponds to the pen-up transition period from one stroke to the next. Finally, we show the extracted spatial and temporal information of the character in Fig. 4(c), where the $x$ and $y$ axes represent the location of each pixel, the $z$ axis represents the EF number of the pixel. We can see that the frame number increases as a stroke is written in one direction, and the frame number jumps from one stroke to the next.

## III. Shadow-Resistance Tracing

However, in some cases, pixels may remain under the pen shadow until they are written on the paper, so these pixels appear occupied throughout the video sequence. This problem becomes more severe for more complex Chinese characters. In order to detect the frame within which the pixel is first written we need to address the shadow problem carefully. We propose a more robust stroke tracing algorithm and some effective error-correction techniques for stroke tracing of sophisticated hand written Chinese characters.

To develop a shadow-resistance tracing algorithm, we first extract a one-pixel-wide, fully written character using a thinning algorithm on the final reference binary image of the video sequence. For each pixel on this character, we count the black-pixel number in an $n$-by-$n$ window surrounding the pixel location in all the binary frames. In a binary frame, the number should be zero if the current pixel location is completely empty,

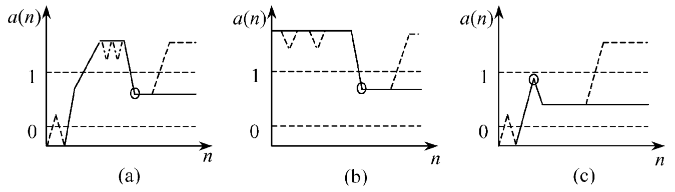

Fig. 5. Three types of pixel occupancy line. The small circle in the figure marks the emergence frame location.

i.e., free of shadows and strokes, and be $n^2$ if the pixel is fully under a shadow, i.e., all pixels in the window are black. Thus, tracing through the whole sequence of video frames, for each pixel on the character, we can obtain a series of black-pixel numbers falling between these two extreme values, which we call the pixel occupancy line (POL). By analyzing the patterns of the POL we can detect the emergence frame within which the pixel first emerges.

There are usually three scenarios under which a pixel is written. First, the pen goes to a new location and starts to write a stroke along a similar direction of the pen shadow. The pixel is empty before the pen shadow is brought upon it as the pen starts to write the stroke. Then the pen tip passes over the pixel. The solid line in Fig. 5(a) illustrates the POL for this scenario, with a few possible variations shown by the dashed lines attached. The figure shows that the POL is brought to the maximum value by the shadow, then falls to a lower value after the pen tip passes the pixel. The second scenario is shown in Fig. 5(b), in which the pixel is under the shadow of the pen and the hand from the start until it is written. So, the POL stays at a high value until the pixel emerges. Finally, in the third scenario, when a stroke is written in a direction relatively perpendicular to the pen shadow, the POL will rise from zero to a local maxima, then falls back to a lower value, as the pen tip enters then leaves the pixel local area. The POL is shown in Fig. 5(c). Unlike the first two scenarios where the pixel to be written is in the shadow of the pen, for the last scenario, the pixel is not in a shadow before it is written.

There are two horizontal dashed lines marked by “0” and “1” in Fig. 5. They represent two boundary thresholds we use to locate the emergence frame. After the pixel is written, there is always a stroke line passing through the $n$-by-$n$ window surrounding the pixel. The minimum number of black pixels in the window should be the window width multiplied by the stroke width. Therefore we define the lower threshold as $T_0 = k_0 w$, where $k_0$ represents the stroke width, and $w$ represents the window width. We define the higher threshold as the upper limit of the POL value when the pen tip is on top of the current pixel. When the pen tip is in the center of the local area, its shadow can only occupy less than half of the window. Also, the stroke is only halfway through the window. So, we define the higher threshold as $T_1 = (T_0 + w^2)/2$.

The emergence frame is detected through the following algorithm.

Let $n$ be the frame number, and $a(n)$ be the POL value.

```
(1) Find the minimum frame number $n_0$, so that
all the POL values after the frame is above the
0-threshold,
    $n_0 = \min\{n_i : a(n) > T_0, \forall n > n_i\}$.
    /* Since the value cannot fall below the
0-threshold after the pixel is written */
(2) Compute the derivative of the POL,
    $\Delta a(n+1) = a(n+1) - a(n)$, $\forall n > n_0$.
(3) For $n = (n_0 + 1) : \max(n)$
 If $a(n) < T_1$ and $\Delta a(n) \times \Delta a(n+1) \le 0$ then /* Local extreme points between $T_0$ and $T_1$ */
  If $\Delta a(n) < 0$ then /* POL descends across $T_1$*/
   $n_0 = n$, break; /* First local minima is the EF for the first two scenarios */
  Else if $\Delta a(n) > 0$ then /* POL rise across $T_0$ */
   $n_0 = n$, break; /* First local maxima is the EF for the third scenario */
  End
 End
 End
```

Fig. 6 shows some EF detection results for a character. The ten corresponding pixels are marked in the reference image. We can clearly see the increasing temporal frame number as the character is written out. Except point 2 on the character, which illustrates the third scenario of writing a stroke, all other points conform to the first scenario of stroke writing. Finally, The complete tracing result of the character is plotted in Fig. 7, where the $z$ axis represents the EF number of the pixel. In this case, we did not try to control the lighting and writing process very carefully as in the case of the example in Fig. 4. Although the majority of the pixels are correctly traced, there are still several types of error exist, as illustrated in the figure. Next, we develop several error correction techniques to improve the tracing result.

## IV. Error Correction Through Static Tracing

There are four types of error as shown in Fig. 7. The first type of error is due to that the EF estimation error is generally in the range of $[-1, +1]$ pixels. The temporal order among contiguous pixels may be distorted, as shown in Fig. 7(a). The second type of error happens at the end points of a stroke. As the pen is entering the starting position of a stroke, its shadow may be identified as the starting points before the actual writing, thus resulting in smaller EF values for the first few pixels, as shown in Fig. 7(b). Similarly, as the pen leaves a stroke, the shadow may cause a delay in the EF estimation. The third type of error is caused by some random video noises, as shown in Fig. 7(c). Fig. 7(d) demonstrates the last type of error at the stroke crossing point, where a previous written stroke may affect the EF estimation of a later stroke.

Fortunately the errors are significantly fewer than correct pixels on the character. To correct the errors, we conduct static tracing of pixel segments in the reference character image. Under ideal conditions, if the EF numbers of the pixels on a stroke segment are plotted against their statically traced spatial order, we should get a straight spatial-temporal line, with a slope gradient corresponding to the inverse of the pen velocity. However, in most cases, the line is broken by various errors, as shown by the examples in Fig. 8. Since the majority of the pixels are correctly traced, we select the

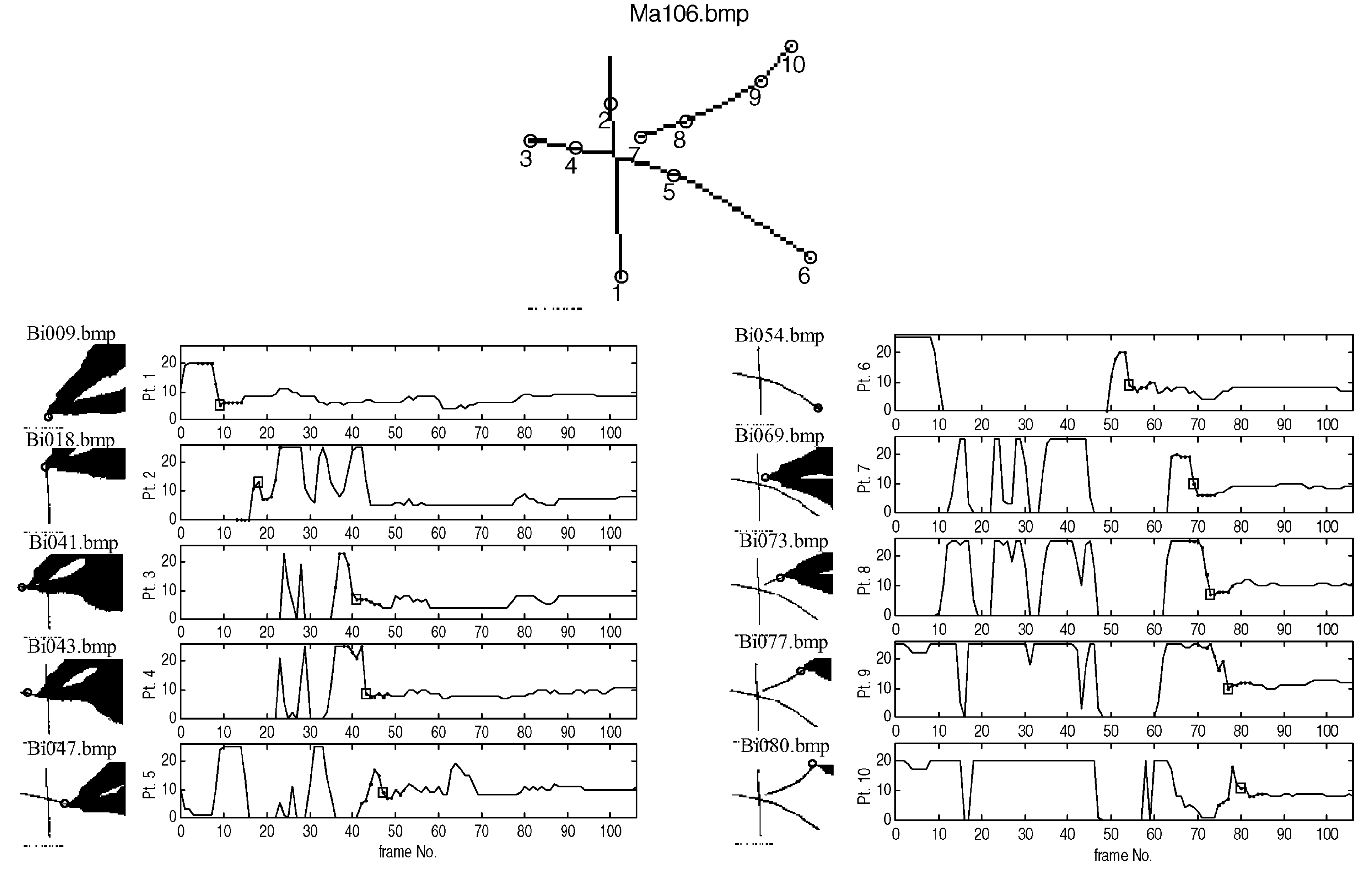

Fig. 6. Emergence frame detection results for the ten pixels marked on the reference character. The detected emergence frame is marked by a small square on each POL.

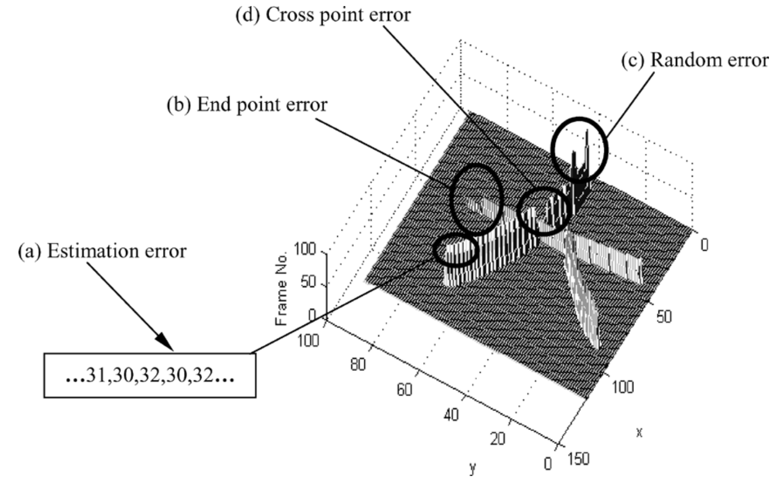

Fig. 7. Emergence frame number plot with the errors.

longest, monotonous, and continuous segment in the plot and conduct a first-order polynomial fitting. Then the fitted line is used as the final temporal information of the stroke pixels. The fitting result is shown in Fig. 8. The two dashed lines in the figure illustrate the boundary of the line-segment that is used for fitting.

The success of this error correction scheme depends on correct static stroke tracing. For effective static stroke tracing, we focus on the two key processing steps, feature point extraction and broken stroke connection [20]–[23], and develop several new algorithms to improve the performance of these two steps.

### A. Feature Point Extraction

For feature point extraction, Rutoviz crossing number is generally used for skeleton images [25], [27], [28]. However, due to problems introduced at the thinning step, Rutoviz crossing number cannot detect all the fork points in a skeleton image. Several new measures were proposed by Liu *et al.* [27] and Abuhaiba *et al.* [24]. However both methods tend to detect more fork points than the actual numbers. In this paper, we propose a simple approach that can remove all bug pixels introduced at the thinning step.

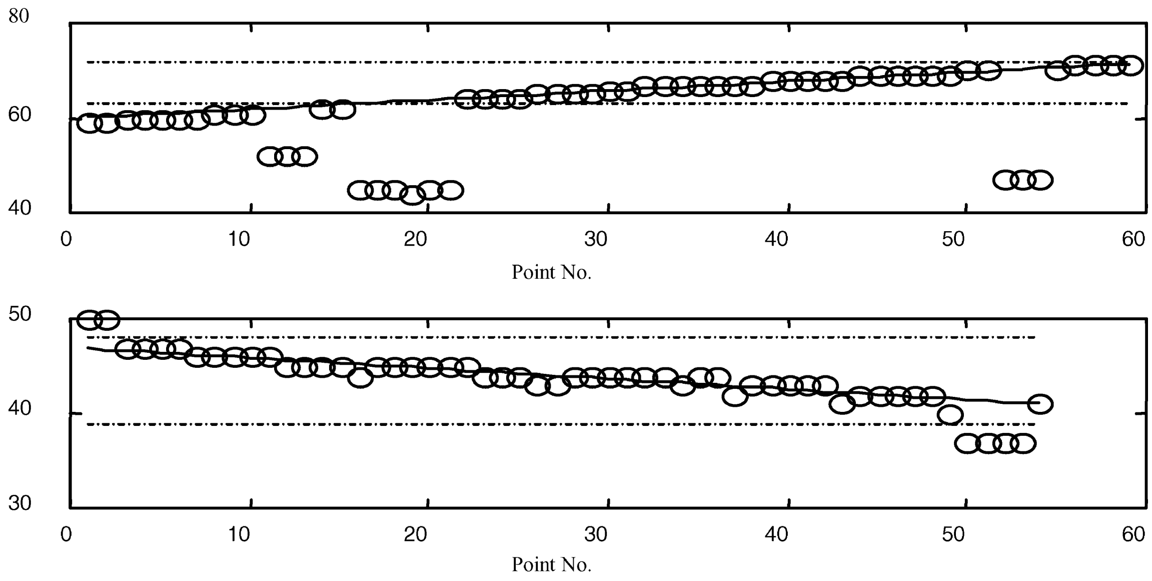

Fig. 8. Spatial-temporal plots of stroke segments with line fitting results. Horizontal axis represents the pixel spatial order; vertical axis represents the pixel temporal order.

Rutoviz's definition of crossing number for a pixel $p$ is

$$N_c(p) = \frac{1}{2} \cdot \sum_{i=1}^{8} |x_{i+1} - x_i| \qquad \begin{array}{|c|c|c|} \hline x_4 & x_3 & x_2 \\ \hline x_5 & p & x_1 \\ \hline x_6 & x_7 & x_8 \\ \hline \end{array} = x_9$$

where $x_i$ ($i = 1, \ldots, 9$) are the adjacent points of pixel $p$ and $x_1 = x_9$. The skeleton pixels in the thinning image can be classified into three classes.

If $N_c = 1$, the analyzed point is an end point.
If $N_c = 2$, the analyzed point is a connective point.
If $N_c > 2$, the analyzed point is a fork point.

This taxonomy is based on the assumption that the width of the skeleton is strictly one pixel. This is not always true for a thinned skeleton. Fig. 9 shows several exceptions, where a skeleton has two-pixel width in the fork region and pixels in the fork region have $N_c = 2$ instead of $N_c > 2$. We find that these exceptional (bug) points can be group into two categories. The first category is shown in Fig. 9(a), where we have a four-pixel block and each pixel in the block has an 8-connected neighbor at the corners of the block. We correct this case by moving two diagonal pixels outward by one step. An example after such a correction is shown in Fig. 9(b), where all pixels in the skeleton satisfy the three-class rules.

The second category can be defined as pixels with more than two 4-connected neighbors. Fig. 9(c)–(f) illustrate some examples, where bug pixels are marked by bold font $\boldsymbol{D}$ and $\boldsymbol{X}$. These pixels are in the fork section but has $N_c = 2$. They each has three or four 4-connected neighbors, so we can detect them easily by locating pixels with more than two 4-connected neighbors.

In order to remove these bug pixels while preserving the skeleton continuity at the fork section, we develop a scan-line ordered deletion scheme. Scanning the skeleton image from left to right for each horizontal line, line by line from top to bottom, we delete the first bug pixel encountered. Then the next bug pixel will be checked again for the number of 4-connected neighbors. If it still has more than two 4-connected neighbors, it will be removed. However, if it no longer has more than two 4-connected neighbors because of the removal of previous bug pixel, it will be kept in the skeleton. This process is continued until all bug pixels are processed. As shown in Fig. 9, all the removed bug pixels are marked as bold $\boldsymbol{D}$, and all the bug pixels that are switched to normal pixels are marked as bold $\boldsymbol{X}$. After the correction, all pixels in the skeleton satisfy the three-class rules. Thus all the feature points of a character, including end points and fork points, can be accurately extracted.

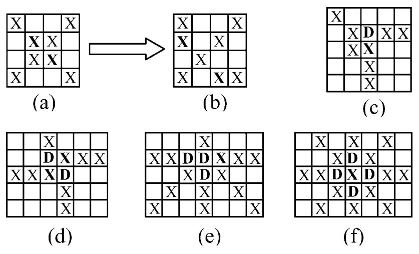

Fig. 9. Example fork point bugs and their corrections.

## B. Stroke Tracing

After the extraction of character feature points ($N_c = 1$ or $> 2$), we can trace the skeletons of the stroke segments from one feature point to the other. This simple tracing procedure is called regular stroke tracing.

Due to the thinning distortion, a fork point may be split into several fork points in a fork region, as shown in Fig. 10(a). To overcome this type of thinning distortion, we adopt the maximum circle criterion [26] to group the fork points belonging to the same fork region.

At this point, we have extracted stroke segments bounded by end points and fork points. In order to obtain the complete strokes, we need to connect stroke segments that are disconnected by the fork points. Intuitively, we can look at the temporal difference between two segments and connect two segments that are close to each other both spatially and temporally. However,

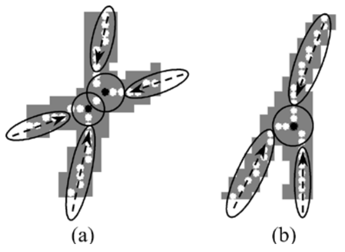

Fig. 10. Thinning distortion and stroke direction estimation.

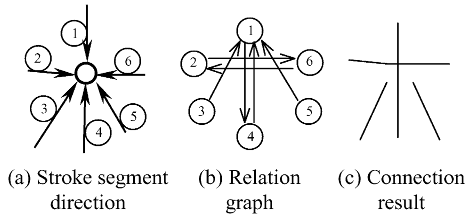

Fig. 11. Stroke segment connection using bidirectional graph.

due to the large number of errors at the crossing points, the temporal estimation of some short segments are usually not stable, thus are not appropriate for error correction. To determine which pair of stroke segments needs to be connected at the fork point, we propose a simple bidirectional graph method.

To connect stroke segments at a fork point, several methods have been developed in recent years. Liao and Huang developed a curve fitting method [26]. All possible pairs of stroke segments connected at the same fork point are fit by the Bernstein-Bezier curve. The connections that produce large enough curvature radius at the fork point are selected. Liu *et al.* propose a method using polygons to approximate skeleton segments [27]. They develop a complicated set of heuristic rules according to the number of stroke segments related to a fork point. Both methods are fairly complicated, thus may not be robust enough for the large variations of hand-written Chinese characters. In this paper, we propose a new approach based on a novel bidirectional graph to connect Chinese character strokes.

First, we need to compute the incoming direction of all stroke segments intersecting with the fork point. Because of thinning distortion, the running direction of skeleton pixels around the fork point can be quite different from the actual orientation of the stroke segment, as shown in Fig. 10(b). To accurately estimate the stroke segment direction, only a section of skeleton pixels right outside the maximum circle of the fork point are used for direction computation, as marked in Fig. 10.

After the stroke segment direction computation, we connect stroke segments at a fork point based on a bidirectional graph. Fig. 11 illustrates an example fork point with six intersecting stroke segments. The incoming directions of the six segments are shown in Fig. 11(a). We represent the directional relationship of the stroke segments by a graph shown in Fig. 11(b), where each node corresponds to a stroke segment. A directional edge is linked from one node to the other if the receiving node segment is closest to $180^\circ$ from the starting node segment, among all the segments. An additional requirement is that the angle between the two segments has to be above a threshold $T_d$ (in our experiment, $T_d = 135^\circ$).

TABLE I
ERROR STATISTICS FOR THE RECOVERY OF STROKE TEMPORARY INFORMATION

| | Error | | | |
|---|---|---|---|---|
| | Direction | Order | Structure | Total |
| Number | 346 | 91 | 53 | 490 |
| Percent (%) | 4.26 | 1.12 | 0.65 | 6.03 |
| Data | 1251 Characters with 8117 Strokes | | | |

For example, the node segment that intersects node-3 at an angle closest to $180^\circ$ is node-1. Therefore, a directional link from node-3 to node-1 is established. However, for node-1, the closest opposite node is node-4, thus a directional link from node-1 to node-4 is set up. Since node-1 is also the closest opposite node for node-4, a directional link from node-4 to node-1 can also be set up. In this way, a complete relational graph for the six segments can be built as shown in Fig. 11(b). There are two bidirectional links 1–4 and 2–6, and two uni-directional links 3–1 and 5–1. Only segment pairs with bidirectional links are connected and the final stroke-connecting result is shown in Fig. 11(c). The method is straightforward and effective.

## V. EXPERIMENTS

To test the VCR system, ten students were invited to write a total of 1251 Chinese characters. Their handwritings were captured by a video camera in real time. There are 8117 strokes in these characters. For such a large data set, we achieve around 94% accuracy for the stroke temporal tracing. The stroke order and direction are both correctly extracted like a regular online system.

Table I summarizes the statistics of the tracing results. There are three types of error for the tracing results.

***Direction Error***: The stroke writing direction is wrongly estimated.

***Order Error***: The stroke writing order is wrongly estimated.

***Structure Error***: The stroke has error skeleton structure (e.g., misconnected with other strokes, or one stroke is broken into several segments) introduced by the stroke static tracing and cannot be corrected by the temporal tracing scheme.

As we can see from the table, most of the errors are direction errors, which mostly happen on short strokes since there are often too few pixel samples for a stable estimation. Fortunately, the short strokes are sometimes considered as dot strokes, so their directions maybe ignored at the recognition stage.

Since the traced strokes have all the properties required by the input of an online character recognition system, a traditional online recognition method can be utilized for the recognition. We adopt the attributed relational graph (ARG) matching method [2], which is a graph based method that can characterize the relational structure of strokes of a Chinese character. An input character can be recognized by comparing its ARG with those of the model characters.

TABLE II
CHARACTER RECOGNITION ACCURACY USING THE ARG METHOD

| Rank | TOP 1 | TOP2 | TOP3 | TOP4 | TOP5 | Number of Errors Caused by Different Steps | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | Tracing | ARG | Total |
| Number | 1106 | 1166 | 1181 | 1193 | 1203 | 30 | 18 | 48 |
| Percent (%) | 88.41 | 93.21 | 94.40 | 95.36 | 96.16 | 2.40 | 1.44 | 3.84 |

Recognition results are shown in Table II. Among the 1251 characters, 1203 are correctly recognized within the top five candidates. Among the errors, 30 are due to the stroke tracing errors and 18 are due to the ARG recognition algorithm. The total recognition accuracy is 88.41% for the first rank and 96.16% for the fifth rank. The results are encouraging compared to the results obtained by Fink *et al.* [19], which is the first and only character recognition experiment on a VCR system. They report around 76% recognition accuracy on a data set containing 201 German city names. Notice we should not make direct comparison between the two experiments since they are for different languages. However, this at least shows that our method works well for the complicated Chinese characters.

## VI. CONCLUSION

We have demonstrated a stroke-tracing algorithm for a handwritten Chinese character recognition system that combines the advantages of both the online and the offline systems. Using a regular video camera, we can capture the temporal information of written characters as they are written on regular papers. Thus the high-accuracy online recognition algorithm can be used for recognition. As a new concept, there are still many issues that remain to be resolved before the system can be employed for practical usage.

## REFERENCES

[1] *Handbook of Character Recognition and Document Image Analysis*, H. Bunke and P. S. P. Wang, Eds., World Scientific, Singapore, 1997.
[2] J. Liu, W. K. Cham, and M. M. Y. Chang, "online Chinese character recognition using attributed relational graph matching," *Inst. Elect. Eng. Proc. Vision Image Signal Processing*, vol. 143, no. 2, pp. 125–131, 1996.
[3] ——, "online Chinese character recognition by incorporating human knowledge," *Int. J. Uncertainty, Fuzziness and Knowledge-Based Syst.*, vol. 5, no. 1, pp. 13–29, 1997.
[4] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 8, pp. 787–808, Aug. 1990.
[5] S. D. Connell and A. K. Jain, "Writer adaptation for online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 3, pp. 329–346, Mar. 2002.
[6] N. Arica and F. T. Yarman-Vural, "An overview of character recognition focused on offline handwriting," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 5, pp. 216–233, May 2001.
[7] S. Madhvanath and V. Govindaraju, "The role of holistic paradigms in handwritten word recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 2, pp. 149–164, Feb. 2001.
[8] J. T. Favata, "Offline general handwritten word recognition using an approximate BEAM matching algorithm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 9, pp. 1009–1021, Sep. 2001.
[9] N. Arica and F. T. Yarman-Vural, "Optical character recognition for cursive handwriting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 6, pp. 801–813, Jun. 2002.
[10] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1438–1454, Nov. 2002.
[11] M. Y. Chen, A. Kundu, and J. Zhou, "offline handwritten word recognition using a hidden Markov model type stochastic network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 481–496, May. 1994.
[12] C. L. Liu, M. Koga, and H. Fujisawa, "Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 11, pp. 1425–1437, Nov. 2002.
[13] Y. Xiong, Q. Huo, and C. Chan, "A discrete contextural stochastic model for the offline recognition of handwritten Chinese characters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 7, pp. 774–782, Jul. 2001.
[14] K. W. Cheung, D. Y. Yeung, and R. T. Chin, "Bidirectional deformable matching with application to handwritten character extraction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 8, pp. 1133–1139, Aug. 2002.
[15] F. Lin and X. Tang, "offline handwritten Chinese character stroke extraction," in *Proc. Int. Conf. Pattern Recognition*, 2002, pp. 249–252.
[16] X. Tang, C. Y. Chan, and J. Liu, "Handwritten Chinese character recognition through a video camera," in *Proc. IEEE Int. Conf. Image Processing*, Vancouver, Canada, Sep. 2000, pp. 692–695.
[17] M. E. Munich and P. Perona, "Visual input for pen-based computers," in *Proc. Int. Conf. Pattern Recognition*, 1996, pp. 33–37.
[18] ——, "Visual signature verification using affine arc-length," in *Proc. Computer Vision and Pattern Recognition*, 1999, pp. 180–186.
[19] G. A. Fink, M. Wienecke, and G. Sagerer, "Video-based online handwriting recognition," in *Proc. Int. Conf. Document Analysis and Recognition*, 2001, pp. 226–230.
[20] Y. Kato and M. Yasuhara, "Recorder of drawing order from scanned images of multi-stroke handwriting," in *Proc. Int. Conf. Document Analysis and Recognition*, 1999, pp. 261–264.
[21] S. Lee and J. C. Pan, "Offline tracing and representation of signatures," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 4, Jul.–Aug. 1992.
[22] S. W. Lee, Y. Y. Tang, and P. S. P. Wang, "Advances in oriental document analysis and recognition techniques," *Int. J. PRAI*, vol. 12, no. 1, Feb. 1998.
[23] J. R. Parker, "Handprinted digit recognition by stroke tracing," in *Proc. 3rd Australian and New Zealand Conf. Intelligent Information Systems*, 1995, pp. 64–69.
[24] I. S. I. Abuhaiba, M. J. J. Holt, and S. Datta, "Processing of binary images of handwritten text documents," *Pattern Recognit.*, vol. 29, no. 7, pp. 1161–1177, 1996.
[25] L. Lam, S. W. Lee, and C. Y. Suen, "Thinning methodologies – A comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 9, pp. 869–885, Sep. 1992.
[26] C. W. Liao and J. S. Huang, "Stroke segmentation by Bernstein-Bezier curve fitting," *Pattern Recognit.*, vol. 23, no. 5, pp. 475–484, 1990.
[27] K. Liu, Y. S. Huang, and C. Y. Suen, "Robust stroke segmentation method for handwritten Chinese character recognition," in *4th Int. Conf. Document Analysis and Recognition*, vol. 1, 1997, pp. 211–215.
[28] L. Wang and T. Pavlidis, "Direct gray-scale extraction of feature for character recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 10, pp. 1053–1067, Oct. 1993.