7

Video based Handwritten Characters Recognition

Chen-Chiung Hsieh
Tatung University
Taiwan

1. Introduction

Video as an input device[1] is becoming a viable method for information products because of its lower cost along with increasing power of computer processor. Its supporting human-computer communication is evident. (Mann, 1996) used video-cam as the input device of a wearable computer system has reported that video becomes a "visual memory prosthetic", a perception enhancer, allowing one to remember faces better, and see things that normally would never have been seen. Some researches in this area are of assistive technology oriented. (Betke et al., 2002) have developed a visual tracking system that interprets visible-light video to provide computer access for people, in particular, children with severe disabilities. Their system "Camera Mouse" was named because it tracked a body feature like the tip of the nose with a video camera and used the detected motion of the feature to directly control the mouse pointer. However, the spelling board was used to control what was typed by selecting the letters with the Camera Mouse. It took more than 20 seconds to input a character.

(Cantzler & Hoile, 2003) proposed a novel form of a pointing device to control the mouse pointer by moving the camera freely in space in a similar fashion to the way a laser pointer controls a laser dot. They locate the computer display in the camera image by finding structured or large regions. The chosen method exploits the fact that there is much structured content in screen. This is sometimes too restrictive for diversity of window applications. Another vision-based pointer device, Gray level VisualGlove, intended for wearable computers, was presented by (Iannizzotto et al., 2001). While previous methods are mainly based on color for image segmentation and human skin detection, their system only relies on gray level gradient to be of low computation cost. Template matching with the highest degree of confidence, the lowest Hausdorff distance, is chosen as a possible detected object. However, it could not deal with scaled or rotated finger motion.

[1] http://en.wikipedia.org/wiki/Input_device.

2. Related Works

(Zhou et al., 2007) presented the Visual Mouse (VM) system for interaction with display via hand gestures. The method includes detecting bare hands using the fast SIFT algorithm to save long training time of the Adaboost algorithm, tracking hands based on the CAM-Shift algorithm, recognizing hand gestures in cluttered background via Principle Components Analysis (PCA) without extracting clear-cut hand contour, and defining simple and robustly interpretable vocabularies of hand gestures, which are subsequently used to control a computer mouse. However, bare hands are restricted to move on table and can't be extracted when overlapped with complex background.

The extracted and tracked pointing device can be used to do dynamic gesture recognition (Min &Yoo, 1997; Davis & Shan, 1994). A number of researchers have explored the use of hand gestures as a mean of computer input, using a variety of technologies. (Lin & Tang, 2003) proposed a video based handwritten Chinese character recognition system that combines the advantages of both the online and offline approaches (Munich & Perona, 2003). 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 off-line system. The temporal information of each stroke is extracted based on the comparison and tracking of ink pixels (pen-down motions) (Tang et al., 2000, Tang & Lin, 2002). However, their approaches struggle to accurately extract the dynamic information for every single ink pixel, which is often mixed with the pen, hand, and their shadows.

(Chen et al., 2006) presents the design and implementation of a fingertip writing interface which can recognize the moving trajectory of the user's fingertip into alphabets and numerals. The processes are divided into tracking and recognition. For the fingertip tracking process, it deploys background subtraction, skin-color modelling, finger extraction, fingertip positioning, and Kalman filter prediction. To recognize the fingertip trajectories, four types of features are defined for recognition with Hidden Markov Models. However, finger tips are easily confused with facial color and the tracking results are not so reliable.

In this chapter, we are focused on video based pointing method which could be utilized as a choice for information input. Given people's experiences with currently available technology devices, our goal has been to develop a non-contact, comfortable, reliable, and inexpensive communication device that is easily adaptable to serve different applications. Motion, color, contour, and curvature are all combined to reliably extract the fingertips. Kalman filter (Mahmoudi & Parviz, 2006) is also adopted to speed up the tracking process. Section three gives an overview of the proposed system. Section four describes the detection of the moving forefinger. In Section five, the tip of forefinger is extracted and the corresponding positions in each frame are accumulated as the written strokes. Directional strokes, semi-circle strokes, and loop strokes are defined for 1-D character representation. Here, pen-up and pen-down are not easy to be differentiated with only one camera. In Section six, strokes connection rules are induced for pen-up strokes detection. All possible 1-D extracted strokes sequences are matched with the pre-built 1-D character models by the Dynamic Time Warp matching algorithm for character recognition. Experiments were conducted to verify the effectiveness of the proposed system in Section seven. Finally, conclusions and discussions are given in the last section.

3. System Overview

A webcam attached on top of the screen is deployed to capture the user gesture in front of it. The system can detect the fingertip without any marks and track it continuously. The fingertip serves as the pointing device usually is the highest point of the moving hand. Figure 1 show the system architecture which could be divided into two modules.

- Fingertip tracking module: it extracts the forefinger tip by frame differencing and skin color. Then, contour around that point is used to calculate the curvature for verification.
  - Motion Detection: two consecutive images are compared to get the changed areas. A motion mask is formed if the difference is greater than a given threshold. If no motion mask is produced, the previous image is skipped and continues with the next image.
  - Finger Tip Extraction: by overlapping the skin color image with the motion mask, the moving hand and finger would be the largest region after connected component labeling. The fingertip can be defined as the top point of that region.
  - Finger Tip Tracking: If the fingertip is found, the position is recorded and continues to the next frame. If user stops writing, the fingertip would not be found and the system goes to the character recognition module. The system performance could be greatly improved by utilizing Kalman filter to predict the fingertip.

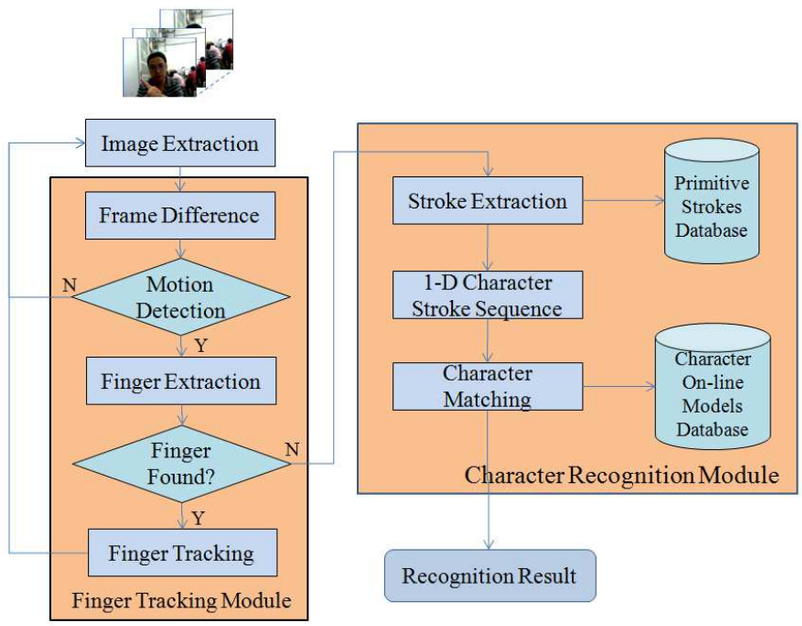


Fig. 1. Functional block diagram of the proposed video-based character recognition system.

- Character Recognition module: it consists of two stages, stroke extraction and character matching. Time warp matching algorithm which could tolerate symbol insertion, deletion, and replacement is used in both of these two stages.
  - Stroke Extraction: from the written character trail, time warp matching algorithm is used to extract all possible strokes defined by the 1-D line segment sequences. In

(Hsieh & Lee, 1992), Hsieh and Lee defined several stroke types for Chinese characters. Here, we defined 3 kinds of stroke types for alphanumeric characters.

- Written Character Representation: all combinations of the extracted strokes are represented as 1-D strings by the temporal information for character matching.
- Character Recognition: time warp matching algorithm is used again to do 1-D strings matching between generated strokes sequence and the stored character on-line model bases. The written trail is recognized as the character model with the minimum distance.

4. Fingertip Detection and Tracking

The tip of forefinger is adopted as the pen tip for writing characters. The forefinger tip has the following properties: skin color, motion region, and of high curvature. Therefore, we design a sequence of functions adaptive skin color detection, temporal frame difference, and connected components labelling to extract it. Adaptive skin color model is developed to produce the skin regions. If the skin regions are of motion regions, those regions would most likely be hand regions. Among the extracted regions, the forefinger tip is usually the highest point and it could be verified by the property of high curvature.

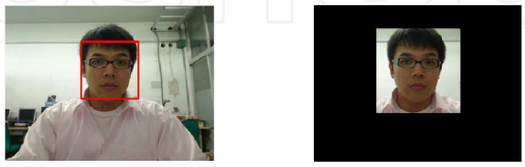
4.1 Adaptive Skin Color Detection

By exploiting information from individual face to create the skin color model for each person will improve the robustness of skin detection because of the reduced amount of color variations within a person's face and hands. In this subsection, we utilize the enhanced adaptive skin color detection method (Hsieh et al., 2010) which is based on individual's face skin color. Firstly, face detection (Viola & Jones, 2001) is applied to find each person's face skin region. By assuming face skin gray levels could be modeled by a Gaussian distribution, non-skin pixels like hair and eyes are of darker gray levels could be excluded by the symmetric property of Gaussian distribution. Remaining pixels most likely to be skin color are transformed into normalized *RGB* color space, say *(r, g, b)*, for building adaptive skin color model. The built skin color model is then used to detect the other skin color pixels such as hands belonging to that person. This method is independent of illumination, ethnic group, shading, and so on. Face detection, face skin region extraction, skin color modeling, ROI setting, and skin color pixels classification are described in detail in the following subsections.

4.1.1 Face Detection

(Viola & Jones, 2001) designed a sequence of classifiers based on a set of haar-like features and combined these increasingly more complex classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object like face regions. (Lienhart & Maydt, 2002) then introduced a novel set of rotated haar-like features which significantly enrich the basic set of simple haar-like features and can also be calculated very efficiently. The characteristic of this method is by use of the black-white haar-like patterns to detect eyes that is independent of the skin colors of people. However, it would produce false alarms along with true positives.

Here, false alarm would be filtered out if the number of skin pixels within the detected face region is less than a given threshold. In order to reduce the computational complexity, the size and location of face are used to set the region of interest (ROI) for face detection in the next frame. It is assumed that user would not move around dramatically in the environment and the location of face in the next frame would be confined in the ROI centered at the previous location of face. If no face is detected, then it is necessary to search face in the whole image. An example is given in Fig. 2. The detected face is enclosed by a rectangle as shown in Fig. 2(a) and the ROI as shown in Fig. 2(b) is set 1.5×1.5 times the size of that rectangle.

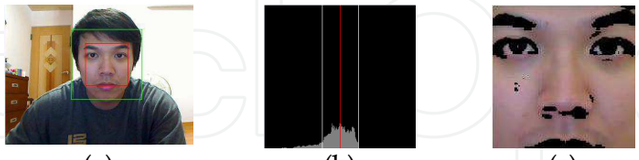


(a) (b)

Fig. 2 (a) Detected face is enclosed by a red rectangle. (b) The ROI set for face detection in the next frame.

4.1.2 Facial Skin Color Sampling

After extraction of face region, we could sample the representative skin color pixels for that person. By observing the histogram of extracted face region, darker pixels like eyes or hair locate at the extreme left. Assume the skin color pixels are of Gaussian distribution in Y (luminance), the shape should be symmetric around the center peak. Therefore, the peak location of the histogram is the mean value. The longer part of the left hand side of the mean could be trimmed by mirroring the right hand side to avoid selecting those darker pixels.
Fig. 3(a) gives an example of detected face and Fig. 3(b) shows the corresponding histogram of the inner region of face. The left hand side is apparently longer than the right hand side. We could trim the darker pixels by the symmetric property of Gaussian distribution. Fig. 3(c) shows that the darker pixels are successfully removed from the inner face region and this demonstrates the mechanism of proposed sampling method.



(a) (b) (c)

Fig. 3 (a) Detected face where the inner region is used for sampling. (b) Luminance histogram of the inner region in (a). (c) Resulting inner region after removal of non-skin color pixels. Note that the non-skin color is indicated in black.

The choice to remove darker pixels in the frontal view inner region is apparent. As shown in (Soriano et al., 2000), the skin scope as defined in Eq. (6) would include white pixels in general. White pixels such as highlights due to illumination reflections happened would be kept in the skin region. Similarly, golden or whitish hair looks like skin color may also be considered as skin color.

$$r = \frac{R}{R+G+B} \tag{1}$$

$$g = \frac{G}{R+G+B} \tag{2}$$

$$Q_+ = -1.3767r^2 + 1.0743r + 0.1452 \tag{3}$$

$$Q_- = -0.776r^2 + 0.5601r + 0.1766 \tag{4}$$

$$W = (r-0.33)^2 + (g-0.33)^2 \tag{5}$$

$$skin = \begin{cases} Q_+ > g > Q_- \\ W > 0.0004 \\ 0.6 > r > 0.2 \end{cases} \tag{6}$$

4.1.3 Adaptive Skin Color Model

R color and normalized *RGB* colors *(r, g)* are used to set up the adaptive skin color model for *(r, g, R)* is less sensitive to changes in light source and suitable for real world applications. Fig. 4 shows the *(r, g, R)* histograms of the skin color pixels in Fig. 3(c). They are symmetric and converge tightly. Therefore, we could model the skin color of that person by Gaussian distributions $GM_i(\mu_i, \sigma_i)$, where μ_i is the mean and σ_i is the standard deviation, i=*r, g,* and *R*. μ_i and σ_i are calculated as in Eq. (7) and (8).

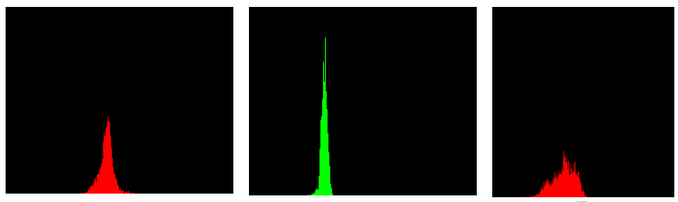


Fig. 4 The skin color distributions of Fig. 3(c) in *(r, g, R)* which could be modeled by three narrow Gaussian distributions.

$$\mu_i = \frac{1}{n}\sum_{(x,y)\in face} I_i(x,y), \quad i = r,g,R \tag{7}$$

$$\sigma_i = \sqrt{\frac{1}{n}\sum_{(x,y)\in face}(I_i(x,y)-\mu_i)^2}, \quad i = r,g,R \tag{8}$$

where n is the number of pixels in the trimmed face region.

4.1.4 ROI setting and skin color detection

ROI setting is designed to speed up the processing time. It can be designed for situations that there are more than one person in the image. Each person has his/her own personal skin color model for hand gesture segmentation. Thus, the accuracy of hand gesture recognition could be increased for occurrence of multiple persons at the same time.

To detect skin color pixels such as the dynamic hand gestures based on the above developed adaptive skin color model, each person has his own ROI as setup in Eq. (9). In most cases,

the hand regions fall within the rectangle as shown in Fig. 5. The actual size of ROI can be adjusted according to application requirements. However, in order to objectively compare the proposed skin color detection method with previously mentioned methods, the ROI is set as the whole image in our experiments.

$$ROI_m(x, y, w, h) = (face_m.x - 5 \times face_m.r, face_m.y - 5 \times face_m.r, 10 \times face_m.r, 10 \times face_m.r), \quad (9)$$

where m represents the index of detected face, (x, y) is the coordinate of the top left corner point of the m-th ROI and (w, h) is its width and height. $face_m$ is a data structure with data members x, y, and r, where $(face_m.x, face_m.y)$ is the center of circle of detected face with radius r.

Fig. 5. ROI setting for hand gesture detection.

With the built personalized skin color model $GM_i(\mu_i, \sigma_i)$, $i=r, g,$ and R, we can now set the upper and low limits for skin color detection as in Eq. (10) and (11), respectively. Note that there is a scale factor for controlling the color range. It is set as two to cover 95%[2] of the samples in our experiments. For each pixel in the set ROI, it is classified as skin color if it satisfies Eq. (12). Fig. 6 shows the mean and the up and low limits for each color component r, g, and R.

$$UpBound_i = \mu_i + 2 \times \sigma_i, \quad i = r, g, R \quad (10)$$

$$LowBound_i = \mu_i - 2 \times \sigma_i, \quad i = r, g, R \quad (11)$$

$$skin = \begin{cases} UpBound_r > r > LowBound_r \\ UpBound_g > g > LowBound_g \\ UpBound_R > R > LowBound_R \end{cases} \quad (12)$$

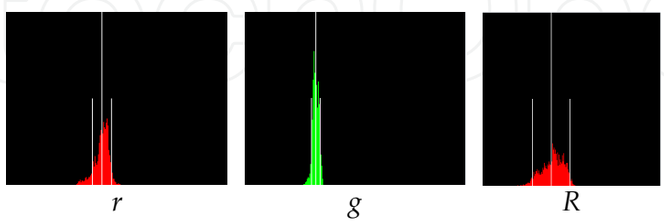


Fig. 6. The scope for each color component (r, g, R). The center line indicates the mean value and the left/right shorter line is the low/up limit.

[2] http://www.itl.nist.gov/div898/handbook/pmc/section5/pmc51.htm

4.2 Fingertip Extraction

By the proposed adaptive skin color model, we could obtain the hand regions. Frame difference is then performed by subtracting the previous frame from current frame. Fig. 7(a) and (b) are two consecutive frames and Fig. 7(c) is the resulting difference frame with threshold set as 20 according to our experience. The motion of hand is apparent in contrast to the other regions in this case. If we restrict the detection of skin pixels within the motion masks, false regions like face or other stationary skin regions can be eliminated for they have no obvious corresponding motion masks.

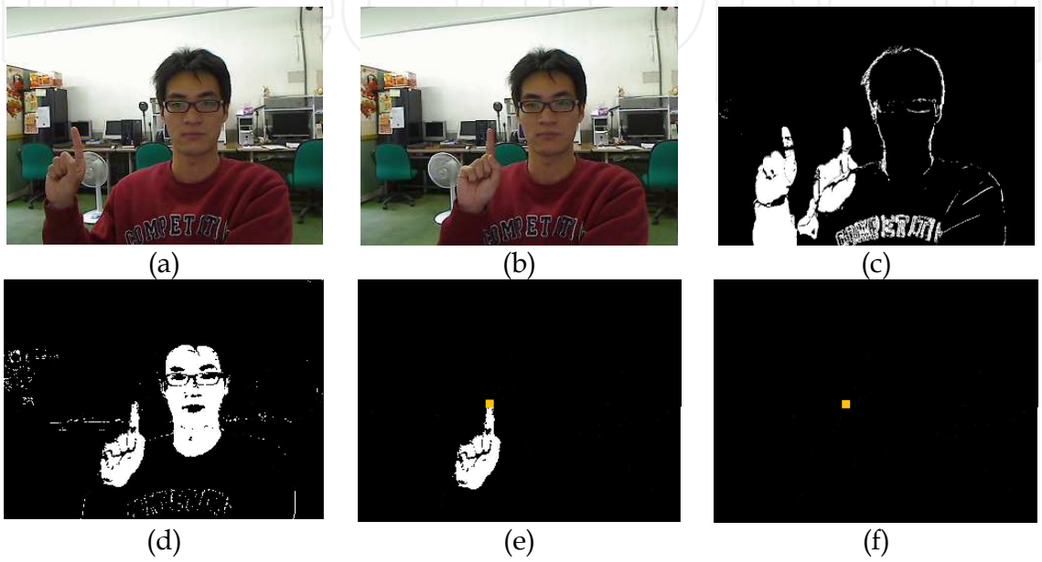


Fig. 7. Finger extraction via adaptive skin color detection and frame differencing. (a) The previous image. (b) The current image. (c) The difference frame with threshold set as *20*. (d) The current skin color image. (e) The extracted current hand region by overlapping motion mask (c) with skin color image (d). Note that the noises were removed. (f) The highest point was extracted as the forefinger tip.

The regions that are both moving regions and skin colors would most likely be hand region. We could define the moving skin color image $H_i(x, y)$ by overlapping motion mask $M_i(x, y)$ with skin color image $S_i(x, y)$ as in Eq. (1).

$$H_i(x, y) = M_i(x, y) \cap S_i(x, y) \tag{13}$$

Through observations and experiments, the fingertip usually is the highest point of moving skin color regions. By connected component labeling, small size regions less than a given threshold are considered as noises and removed. In turn, the moving hand is usually the largest region with size greater than a given threshold, say about 100 pixels. Fig. 7(d) shows the skin color image of Fig. 7(b), and Fig. 7(e) shows the resulting hand region by overlapping Fig. 7(c) and 7(d). The highest end point as shown in Fig. 7(f) was the extracted forefinger tip. It can be further verified by its high curvature.

Kalman filter is adopted to predict the next position of fingertip in the target frame. Because our fingertip is moving in a smooth style, the tracking is based on a dynamic linear system in the time domain. Fig. 8 gives an example for fingertip tracking. The predicted fingertip position is marked by a small yellow triangle and the previous position is marked by a small red circle as shown in Fig. 8(b). Then, the blue rectangle defined as half the size of the image is used for fingertip searching.

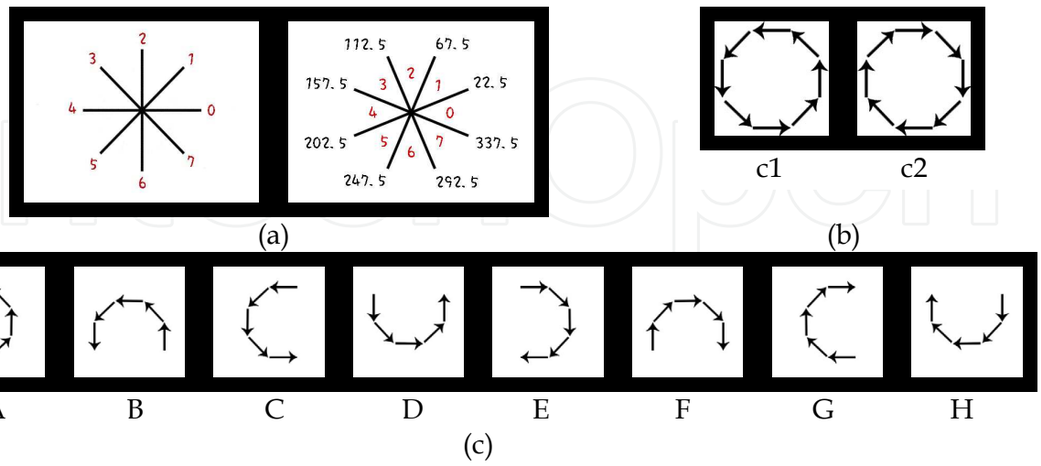
(a) (b)

Fig. 8. Fingertip prediction by Kalman filter. (a) Current fingertip is marked by a blue dot. (b) Previous, current, and predicted fingertip positions are marked by ×, ○, and △, respectively. Note that the blue rectangle represents the search range

From experimental results, the average throughput of fingertip extraction is 13.184 frames per second with Kalman filtering which is faster than the processing speed 10.216 frames per second without Kalman filter.

5. Stroke Extraction

The top point of forefinger extracted by the method discussed in the previous section is deployed as pen-tip to write characters. Each tip position of the forefinger in the image sequence is recorded at the speed of 10-15 frames per second. By connecting consecutive positions along the time series, we can get the written trails.



(a) (b)

(c)

Fig. 9. The defined stroke types for alphanumerical characters. (a) Primitive directional stroke types and their tolerance. (b). Loop stroke types. (d) Semi-circle stroke types.

5.1 Stroke Types

We defined eight primitive directional strokes types, two loop stroke types, and eight semi-circle stroke types as shown in Fig. 9 for alphanumerical character representation. The primitive stroke types are of straight line segments as shown in Fig. 9(a). To tolerate the writing variety among different users, we allow at most 45 degree variation range. Fig. 9(b) and (c) show the loop stroke types and semi-circle stroke types, respectively, by the consisting short straight line segments. Note that the shapes of A~D are written counter-clockwise and E~F are written clockwise. Similarly, the same shape of the two loop stroke types are written in opposite directions.

5.2 Pen-up Stroke Detection

Due to the lost depth information, we induced some rules to prevent extracting too many possible strokes. In this chapter, alphanumeric characters were analyzed to detect the pen-up strokes which move the pen-tip from the end point of current stroke to the starting point of next stroke. For example, the line segment sequence '→', '↓', '← ', and '→ ' can be analyzed to be a compound stroke '⊃' concatenated with a pen-up stroke '→'. For a written 1-D stroke sequence, there may be several ways to decompose that character. Each corresponds to a possible character. Thus, to reduce the number of possible combinations, we could extract only the possible strokes after removal of pen-up strokes.

According to the writing habit, the successive stroke denoted 2 in Fig. 10 is considered as a pen-up stroke which is impossible to be a written stroke. In case one as shown in Fig. 10(a), it shows that we would not write the second stroke in the opposite direction of the first one. Note that the first stroke may not be horizontal. In case two as shown in Fig. 10(b), it reveals that the second stroke would never go north or northwest given a horizontal line segment from right to left. In case three as shown in Fig. 10(c), it demonstrates that the second stroke would not be written from bottom to top given the first stroke written from left to right. For these cases, the second stroke can be extracted and recognized as a pen-up stroke. Furthermore, the pen-up stroke is in fact the directional relationship between the first and the third pen-down strokes.

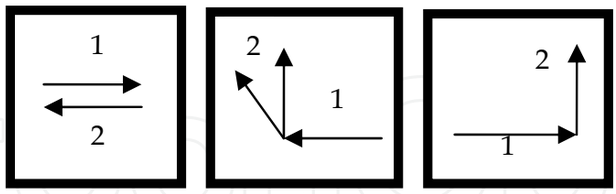


Fig. 10. Three cases for pen-up stroke detection. The second stroke is pen-up stroke given the first pen-down stroke.

5.3 Stroke Extraction by Dynamic Time Warp Matching Algorithm

By connecting tracked fingertip positions in consecutive images, we can obtain the trail which is the written character. The written trail can be parsed to extract the defined stroke types as shown in Fig. 9. Initially, the input trail could be segmented into several connected pen-down strokes by the rules for pen-up strokes detection. Parse each segmented input trail by the finite automata for each defined stroke type, the pen-down strokes could be recognized. Here, the Dynamic Time Warp (DTW) matching algorithm (Weste et al., 1983) is

adopted as the finite automata for pen-down stroke recognition. In fact, DTW is an algorithm for measuring the similarity between two 1-D sequences which may vary in time or speed. The algorithm for pen-down stroke extraction is described as follows.

Algorithm: Stroke extraction
Input: 1-D straight line segment sequence.
Output: Possible strokes for each stroke type.
1. Pen-up stroke detection and segmentation of written trail.
2. **For** each segmented input trail x_i
   $k=0$;
   Repeat
      For each stroke type y_j
         DTW initialization;
         Calculate all the partial sum
            $S_{m,,n} = min\{ S_{m-1,n-1}+R(x_{i,m} ,y_{j,n}), S_{m-1,n},+D(x_{i,m}), S_{m,n-1}+I(y_{j,n})\}$
            between x_i and y_j by dynamic programming;
         If y_j matched with x_i **then**
            A stroke y_j' of type y_j in x_i is obtained;
            Record the starting/end point of y_j' in x_i;
      $k=k+1$;
      $x_i=x_i-$ {the first k line segments};
   Until $x_i = \Phi$;

The distance functions are defined as in Eq. (14)~(16).

$$\mathrm{R}(x_{i,m}, y_{j,n}) = \begin{cases} 0 & ,if\ x_{i,m} = y_{j,n} \\ \left| dir(x_{i,m}) - dir(y_{j,n}) \right| \bmod 6, & otherwise \end{cases} \tag{14}$$

$$\mathrm{D}(x_{i,m}, y_{j,n}) = 1 \tag{15}$$

$$\mathrm{I}(x_{i,m}, y_{j,n}) = 1 \tag{16}$$

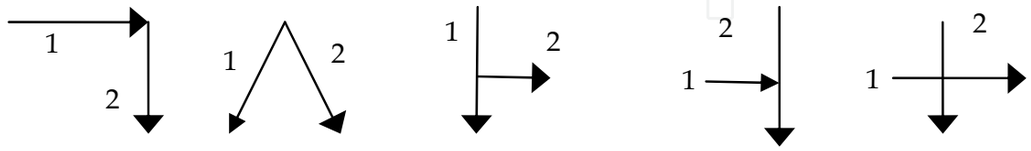
where *dir* is the direction code as shown in Fig. 9(a).

6. Character Recognition

6.1 On-line Model for Character Representation

For the purpose of character recognition, each character is first represented by the on-line model (Hsieh & Lee, 1992) which is actually a 1-D stroke sequence according to the temporal information. Not only stroke types but also stroke relationships are used for character description. For example, the stroke sequence of '4' starts from '/', '—', and ends at '□'. If the stroke relationships between two consecutive strokes are not utilized, then the character '7' could also have the same stroke sequence as '4'. With the relationships introduced into the on-line model, we can not only describe characters but also recognize them. The relationships between any two consecutive strokes can be divided into six classes as shown in Fig. 11.

- Case a: The tail of the first stroke connects to the head of the next stroke as shown in Fig. 11(a).
- Case b: Two consecutive strokes connect at their head positions as shown in Fig. 11(b).
- Case c: The head of the second stroke touches the middle part of the first stroke as shown in Fig. 11(c).
- Case d: The tail of the first stroke touches the side of the second stroke as shown in Fig. 11(d).
- Case e: Two strokes intersect with each other as shown in Fig. 11(e).
- Others: This kind of relationship is used when none of the above cases could be applied. It is simply represented by the direction code as shown in Fig. 9(a).



(a) R_a: tail-head (b) R_b: head-head (c) R_c: side-head (d) R_d: tail-side (e) R_e: intersection
Fig. 11. Stroke relationships between two consecutive strokes.

By the defined stroke types and stroke relationships, on-line models for characters recognition could be built manually. Some examples are given in Fig. 12. The on-line models for the same character may be created differently for the different writing sequences. Usually, we would like to choose the most discriminative one. On the other hand, the on-line model could be used for character recognition if there are no more than two characters share the same model.

A: ↙R_b↘R_3→ B:↓R_b⮌R_a⮌ H:↓ R_c→R_d↓ W:↺R_a↺

Fig. 12. On-line models for alphanumeric characters 'A', 'B', 'H', and 'W'.

6.2 Character Recognition

After strokes extraction, the input character can be represented as many 1-D sequences of extracted pen-down strokes interleaved by pen-up stroke relationships. In Section 5.2, the detected pen-up strokes could be further used to aid the classification of the stroke relationship defined in Fig. 11. These 1-D sequences are then matched with each on-line model in the character set. In this chapter, character recognition is conducted again by the Dynamic Time Warp matching algorithm which could tolerate stroke insertion, replacement, and deletion.

In general, DTW is a method to find an optimal match between two given sequences. It determines the measurement of their similarity in the time domain. We could apply DTW to do matching directly on the written trail which was represented by a sequence of straight line segments. Assume the input sequence is $w(1:m)$ and the reference sequence is $t(1:n)$. The time complexity is $O(mn)$ for DTW at line segment level. However, the detected pen-up strokes could be used to segment the input sequence into $w_1(1:m_1)$, $w_2(1:m_2)$, …, and $w_s(1:m_s)$, where $m>=m_1+ m_2+\ldots m_s$ and s is the number of pen-down strokes. Note that the reference sequence t could also be defined as $t_1(1:n_1)$, $t_2(1:n_2)$, …, and $t_s(1:n_r)$, where $n>=n_1+ n_2+\ldots n_r$

and r is the number of pen-down strokes. Take the fault tolerance of pen-up strokes into consideration, the time complexity would be $O(\sum_{j=1}^{r}\sum_{i=1}^{s} m_i * n_j)$.

To be more robust and save computation time for character recognition, we apply DTW at stroke level instead of straight line segment level. The algorithm is stated as follows.

Algorithm: Character recognition
Input: All possible strokes for each stroke type are stored in linked lists.
Output: The character on-line model with the highest similarity.
1. *max=0;*
2. **For** each character on-line model y_i
    Build the multi-stage graph according to the strokes in y_i;
    For each combination x_j of extracted strokes according to y_i;
        Check validity of x_j by the pen-up strokes in y_i;
        Discard invalid x_j;
        Do DTW matching on pen-down strokes in x_j and y_i ;
        Calculate all the partial sum
            $S_{m,,n} = min\{ S_{m-1,n-1}+R(x_{i,m} ,y_{j,n}), S_{m-1,n},+D(x_{i,m}), S_{m,n-1}+I(y_{j,n})\}$
            between x_i and y_j by dynamic programming;
        If the final score is greater than *max*
            Update *max* and set y_i is the recognized character;

By extracting all the possible strokes from the 1-D straight line segment sequence, the computational cost would become $O(m'_1 \times m'_2 \times \ldots \times m'_s)$, where m'_i ($<m_i$) is the number of strokes of type n'_i, $i=1, 2, \ldots, s$. In our case, $O(m'_1 \times m'_2 \times \ldots \times m'_s \times s^2)$ would become $O(C \times m'_1 \times m'_2 \times \ldots \times m'_s)$ with small s. For a typical example, if $m=n=10$, $(m_1, m_2, m_3)= (n_1, n_2, n_3) =(3, 3, 4)$, and $(m'_1, m'_2, m'_3)=(2, 2, 1)$, we would have $mn=100$, $\sum_{j=1}^{r}\sum_{i=1}^{s} m_i \times n_j =100$, and $m'_1 \times m'_2 \times m'_s \times s^2 = 36$. The time complexity of stroke level DTW would be the best.

7. Experimental Results

The video camera deployed is Logitech QuickCam®Pro for Notebooks with technical specification including auto focus, 2M pixel sensors, 24-bit true colors, and capture speed up to 30 frames per second. No restriction was made on the used camera but the image resolution should be set to 320×240 at least. Note that this is the minimum requirement. The system software is implemented in C language and runs on a Pentium 4 PC.

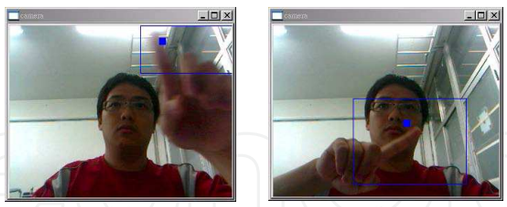
7.1 Fingertip Tracking Results

Accuracy is more important than execution speed in this system. If the extraction of fingertip is not precise, the application will not work effectively. Fig. 13 shows the different writing factors including camera distance, hand rotation, and view angle are all taken into considerations, and the experimental results proved that our system is robust. In addition, if

the moving speed is stable, the fingertip could still be extracted in spite of overlapping with face.

A video sequence consisting of 847 frames is used for testing and the fingertip extraction accuracy achieves 98.58%. Among the fault situations, the first case was due to abrupt changes of luminance. The bright light affected the skin color seriously as shown in Fig. 14(a). In the second case, the moving hand as shown in Fig. 14(b) passed the face. It would be very challenging to extract fingers when fingers were overlapped with face. Both finger and face were of similar skin color and moving regions.

Fig. 13. The extracted fingertips from variety of conditions in a real video sequence.



(a) (b)

Fig. 14. The fault scenarios. (a) The bright light affected the skin color of fingertip. (b) The face affected the fingertip verification.

7.2 Alphanumeric Character Recognition Results

To prove the robustness of proposed video based character recognition system, thirty-six characters including English letters from A to Z and numbers from zero to nine were tested. In order to facilitate the user to write with the proposed system, the input method from Palm Graffiti[3] was adopted. It is released from Palm and in widespread use on PDA. Fig. 15

[3] http://en.wikipedia.org/wiki/Graffiti_(Palm_OS)

gives all the alphanumerical characters. Note that some characters are written different from traditional writing habit.

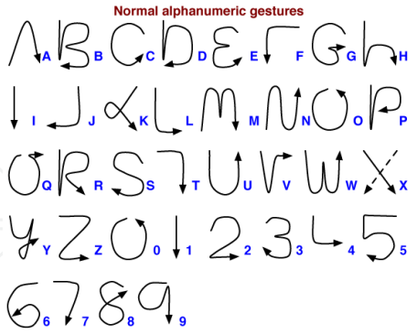
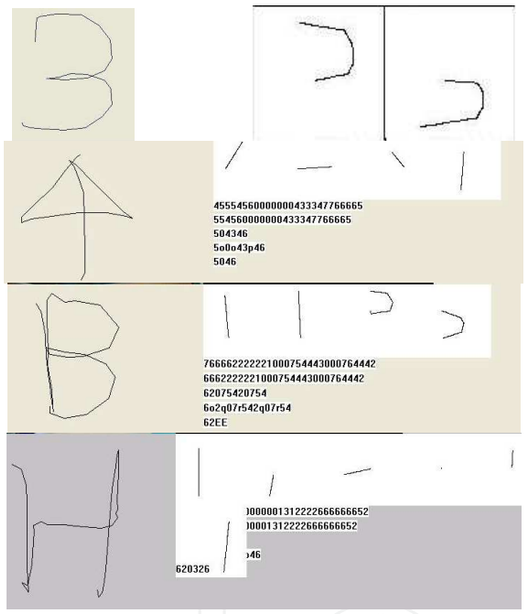


Fig. 15. Palm Graffiti designed for easy writing and easy character recognition.



(a) (b)

Fig. 16. Screenshots of the two stage DTW matching on some input characters. The intermediate process demonstrates the system capability of stroke extraction and character recognition. (a) Written characters. (b) Extracted strokes and the correct on-line model.

Fig. 16(a) shows several characters that were written by fingertip. After pen-up strokes detection, the written 1-D straight line sequence could be segmented for stroke extraction. It was conducted by matching segmented line sequence with each defined stroke types. All possible strokes could be extracted and then arranged as a multi-stage graph according to the character on-line model. Finally, valid combinations of the extracted strokes are matched with the on-line model. Here, the matching was done by two stage DTW matching algorithm. The first one was for stroke extraction and the second one was for character matching. Fig. 16(b) shows the extracted strokes and the recognition results. For example,

character 'B' was recognized as the on-line model "62EE"=pen-down '  ' + pen-up '' + pen-down '  ' + pen-down '  '.

Five persons were invited to write 36 alphanumeric characters five times and we have 900 character test videos. The recognition rate is 82.77% because we can not differentiate '0', '1', and '7' from 'O', 'I', and 'T', respectively, for they have the same written strokes sequences. If these ambiguous characters were counted as correct, the system performance could reach 91.11%. Table 1 shows the time spent for character writing, stroke extraction, and character recognition. The finger writing speed is 3.4543 seconds per character on average, and the character recognition speed is 0.05437second/char including strokes extraction 0.02125 second/char and character recognition 0.03312 second/char. The recognition result comes out immediately after the character was written completely.

	Character writing	Strokes extraction	Character recognition
People 1	2.7965	0.01768	0.02168
People 2	3.9722	0.02387	0.04399
People 3	3.5166	0.02115	0.03465
People 4	3.8168	0.02391	0.03966
People 5	3.1694	0.01963	0.02566
Average	3.4543	0.02125	0.03312

Table 1. The processing speed for character writing, stroke extraction, and character recognition.

Table 2 summaries the accuracies for fingertip extraction and character recognition among different methodologies. The time spent for character recognition was also indicated. Although DTW could achieve recognition rate up to 94%, it spent too much time doing DTW matching. On the other hand, our method based on stroke matching could achieve much higher speed with little sacrifice on the accuracy rate. To improve the recognition rate, we can create more than one on-line models tailed for the mis-recognized characters.

	Fingertip Extraction	Character Recognition	
	Accuracy	Accuracy	Time
HMM2 (Chen et al., 2006)	97.74%	88.1%*	1.08s
DTW (Chen et al., 2006)	97.74%	94%	21.14s
Ours	98.58%	91.11%	0.033s

*with one training sample per character.

Table 2. Comparisons among different methodologies.

8. Conclusions

In this chapter, we firstly designed an adaptive skin color model for fingertip extraction and its application to handwritten character recognition. A cheap webcam is used to serve as the input device and the forefinger tip is extracted by integrating skin-color, motion, curvature, and its relatively higher position. The fingertip position is recorded and tracked for each frame. Thus, a video based pointing method was developed as a choice for information input or computer system operation. Compared with currently available technologies, our system has been a non-contact, comfortable, reliable, and inexpensive communication device that could be easily adaptable to serve special needs like character input. Furthermore, this input device was also integrated with the MS-Windows just like mouse. Here, the fingertip acting as pen tip can be used to write characters. We defined some stroke types including primitive directional strokes, semi-circle strokes, and compound loop strokes for alphanumeric characters. Dynamic time warp matching algorithm was used to extract possible strokes. On-line model was also utilized to formulate the character recognition as a multi-stage searching problem. At each stage, we have extracted stroke candidates for that stroke as defined in the on-line model. The optimal path represents the stroke sequence which complies with the on-line model most. It was found again by the DTW matching algorithm. Among all the found stroke sequences, the one with the highest similarity was picked and the written stroke sequence was recognized as the corresponding on-line model. The system we implemented can be inexpensive, convenient, and intuitive. Experimental results proved that the proposed system is feasible for applications.

Up to now, users of IT products such as computer, notebook, and PDA, all need a convenient character input tool. The video-based handwritten character recognition system provides such an input method without the burden caused by holding tiny stylus pen, writing on a size quite limited touchpad, or inputting through an on-screen virtual keyboard. For future works, the character on-line model base can be extended to cover Chinese characters. For robustness improvement, the fingertip overlapped with moving skin color regions like faces would be difficult to extract. It is a challenge to achieve better recognition accuracy without increasing too much stroke extraction time.

9. References

Betke, M.; Gips, J. & Fleming, P. (2002). The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities, *IEEE Trans. on Neural Systems and Rehabilitation Engineering*, Vol. 10, March 2002, pp. 1-10.

Cantzler, H. & Hoile, C. (2003). A novel form of a pointing device, *Vision, Video, and Graphics*, 2003, pp. 1-6.

Chen, Z. W.; Lin, Y. C. & Chiang, C. C. (2006). Design and implementation of a vision-based fingertip writing interface, *Proceeding of the 18th International Conference on Pattern Recognition (ICPR'06)*, pp. 104-107, 2006.

Davis, J. & Shan, M. (1994). Visual gesture recognition, *Proceedings of IEE on Vision, Image and Signal Processing*, Vol. 141, pp. 101-106, April 1994.

Hsieh, C. C. & Lee, H. J. (1992). Off-line recognition of handwritten Chinese characters by on-line model-guided matching, *Pattern Recognition*, Vol. 25, No. 11, pp. 1337-1352, 1992.

Hsieh, C. C.; Liou, D. H. & Jiang, M. K. (2010). Fast enhanced face-based adaptive skin color model, to be published in the Proceeding of the 2010 International Conference on Image Processing and Pattern Recognition in Industrial Engineering (IPPRIE 2010), Xi'an, China, Aug., 2010.

Iannizzotto, G.; Villari, M. & Vita, L. (2001). Hand tracking for human-computer interaction with graylevel visualglove: turning back to the simple way, *Proceedings of the 2001 workshop on Perceptive user interfaces*, pp 1-7, Orlando, Florida, 2001.

Lin, F. & Tang, X. (2003). Dynamic stroke information analysis for video-based handwritten Chinese character recognition, *Proceedings of the 9th IEEE International Conference on Computer Vision*, pp. 695-700, 2003.

Lienhart, R. & Maydt, J. (2002). An extended set of haar-like features for rapid object detection, *Proceedings of 2002 International Conference on Image Processing*, vol.1, pp. 900-903, 2002.

Mahmoudi, F & Parviz, M. (2006). Visual hand tracking algorithms, *Proc. of the Geometric Modeling and Imaging-New Trends*, London, UK, pp. 228-232, 2006.

Mann, S. (1996). Wearable tetherless computer mediated reality: WearCam as a wearable face recognizer, and other applications for the disabled, *AAAI Fall Symposium on Developing Assistive Technology for People with Disabilities*, pp. 9-11, November 1996.

Min, B. W. & Yoo, H. S. (1997). Hand gesture recognition using hidden Markov modes systems, *Proceedings of the IEEE Int. Conf. on Computational Cybernetics and Simulation*, Vol. 5, pp. 4232-4235, 1997.

Munich, M. E. & Perona, P. (2003). Visual identification by signature tracking. *IEEE Trans. PAMI*, Vol. 25, No. 2, pp. 200-217, Feb. 2003.

Soriano, M.; Huovinen, S.; Martinkauppi, B. & Laaksonen, M. (2000). Using the skin locus to cope with changing illumination conditions in color-based face tracking, *Proc. IEEE Nordic Signal Processing Symposium (NORSIG 2000)*, Kolmarden, Sweden, June 13-15, pp. 383-386, 2000.

Tang, X.; Yung, C. & Liu, J. (2000). Handwritten Chinese character recognition through a video camera, *Proceedings of Int'l Conf. Image Processing*, pp. 692-695, 2000.

Tang, X. & Lin, F. (2002). Video-based handwritten character recognition, *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, pp. 3748-3751, 2002.

Viola, P. & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2001,* vol. 1, pp. 511-518, 2001.

Weste, N.; Burr, D. J. & Ackland, B. D. (1983). Dynamic time warp pattern matching using an integrated multiprocessing array, *IEEE Trnas. on Computers*, Vol. C-32. No. 8, pp. 731-744, August 1983.

Zhou, H.; Xie, L. & Fang, X. (2007). Visual mouse: SIFT detection and PCA recognition, *Proceedings of the 2007 International Conference on Computational Intelligence and Security Workshops*, pp. 263-266, 2007.

Character Recognition
Edited by Minoru Mori

ISBN 978-953-307-105-3
Hard cover, 188 pages
Publisher Sciyo
Published online 17, August, 2010
Published in print edition August, 2010

Character recognition is one of the pattern recognition technologies that are most widely used in practical applications. This book presents recent advances that are relevant to character recognition, from technical topics such as image processing, feature extraction or classification, to new applications including human-computer interfaces. The goal of this book is to provide a reference source for academic research and for professionals working in the character recognition field.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chen-Chiung Hsieh (2010). Video Based Handwritten Characters Recognition, Character Recognition, Minoru Mori (Ed.), ISBN: 978-953-307-105-3, InTech, Available from: http://www.intechopen.com/books/character-recognition/video-based-handwritten-characters-recognition

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen