Vision-Based Single-Stroke Character Recognition for Wearable Computing

Ömer Faruk Özer, *Bilkent University*
Oğuz Özün, C. Öncel Tüzel, and Volkan Atalay, *Middle East Technical University*
A. Enis Çetin, *Sabanci University and Bilkent University*

People want increasingly flexible and mobile ways to communicate using computers. Wearable computing offers advantages but making data entry easy remains a challenge. The authors discuss a new approach for data entry using a head-mounted digital camera to record characters drawn by hand gestures or by a pointer.

Particularly when compared to traditional tools such as a keyboard or mouse, wearable computing data entry tools offer increased mobility and flexibility. Such tools include touch screens, hand gesture and facial expression recognition, speech recognition, and key systems.

However, making data entry easy poses a challenge. New approaches (see the sidebar, "Useful URLs") such as one-handed chording keyboards help us understand the problems and complexities. Using the character recognition systems developed in document analysis, computer vision-based man–machine communication systems are possible.[1,2] For example, personal digital assistants let users write rather than type on a small keyboard, thanks to the success of unistroke, isolated character recognition systems.[3,4] In most of the new data entry approaches, the rate of data entry is lower than that of the traditional keyboard- or mouse-based entry. On the other hand, fast data entry systems require a learning phase most people would rather avoid.

In this article, we describe a new approach for recognizing characters drawn by hand gestures or by a pointer on a user's forearm captured by a digital camera. We draw each character as a single, isolated stroke using a Graffiti-like alphabet. Our algorithm enables effective and quick character recognition. The resulting character recognition system has potential for application in mobile communication and computing devices such as phones, laptop computers, handheld computers, and personal data assistants.

The recognition system and our algorithm

Consider this scenario: A user draws unistroke, isolated characters with a laser pointer or a stylus on their forearm or a table. A camera on their forehead records the drawn characters and captures each character in sequence. The image sequence starts when the user turns the pointer on and ends when they turn

Useful URLs

- **The septambic keyer**, http://wearcam.org/septambic
- **The Twiddler**, www.handykey.com
- **The EyeTap**, http://eyetap.org
- **The Pendragon project**, www.cc.gatech.edu/fce/pendragon
- **Multimodal conversational interaction**, http://vislab.cs.wright.edu
- **Visual gesture research**, www.ifp.uiuc.edu/~jy-lin/gesture/gesture.htm
- **User system ergonomics research**, www.almaden.ibm.com/cs/user.html

1094-7167/01/$10.00 © 2001 IEEE

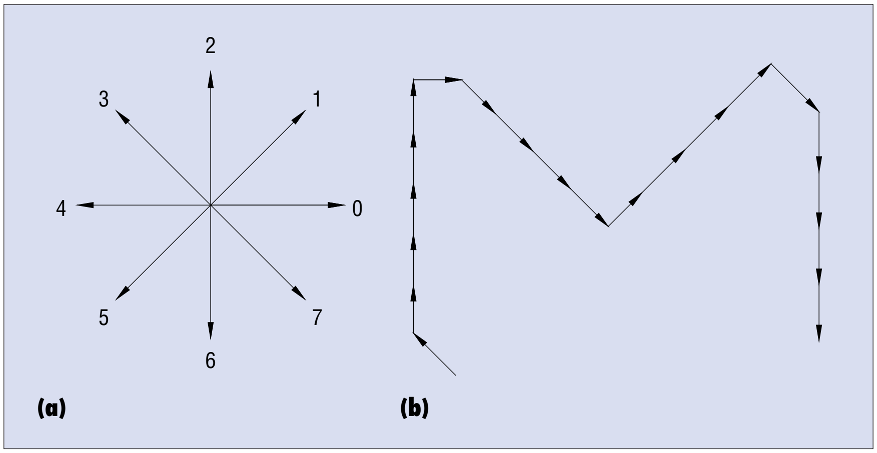

Figure 1. (a) Chain code values for the angles; (b) a sample chain-coded representation of the character *M* = 32222207777111176666.

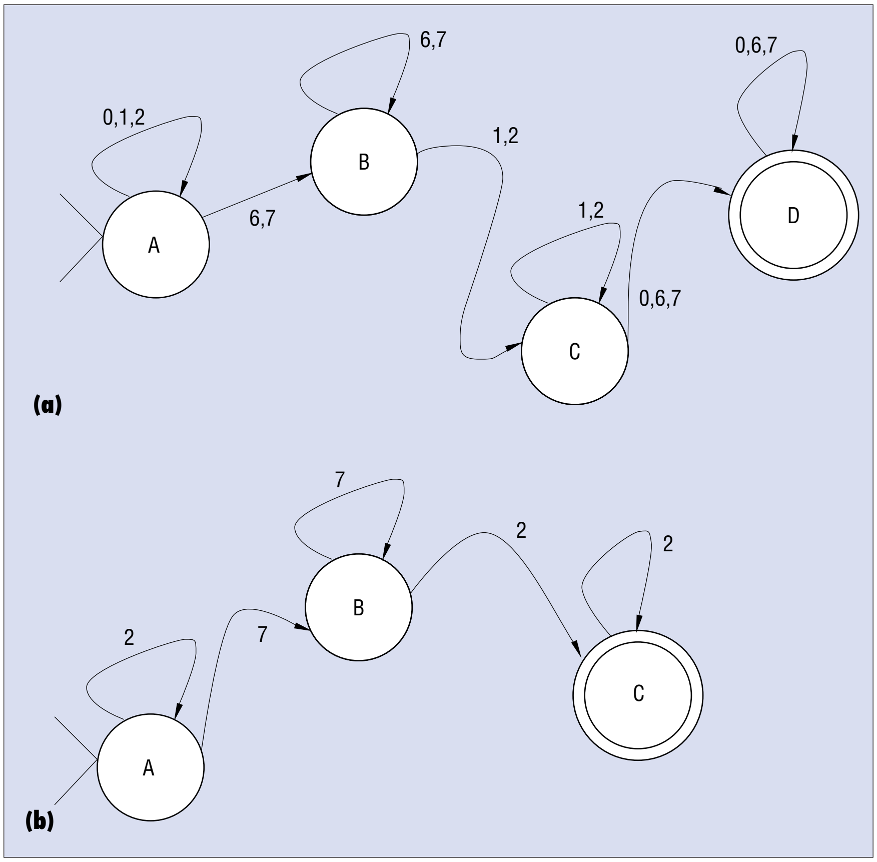

Figure 2. Finite state machines for the characters (a) *M* and (b) *N*.

it off. Thus, discontinuous pointer movements separate each character.

In our approach, a *chain code* describes the unistroke characters drawn. A chain code is a sequence of numbers between 0 and 7 obtained from the quantized angle of the laser point's beam in an equally timed manner. We extract chain code from the beam's relative motion between consecutive images of the video. The chain code is the input for the recognition system.

The recognition system consists of finite state machines corresponding to individual characters. The FSMs generating the minimum error identify the recognized character. However, certain characters such as *Q* and *G* might be confused in a feature set comprising only the chain code. Therefore, the system also considers the beginning and end strokes. The weighted sum of the error from a finite state machine and the beginning and end point error determines the final error for a character in the recognition process.

Our algorithm for character recognition consists of four steps, described in the following paragraphs.

Step 1, extraction of chain code. The system

- finds the position of the red mark the laser pointer produces in each frame,
- generates a chain code according to the angle between two consecutive mark positions, and
- determines beginning and end point coordinates together with the coordinate of a rectangle enclosing the character.

Step 2, analysis using finite state machines. The system

- applies the chain code as input to each state machine,
- determines state changes (additionally, the system increases an error counter by one if a change is not possible according to the current FSM),
- eliminates the corresponding character if a chain code does not terminate in the final state, and
- adds up errors in each state to find the final error for each character.

Step 3, accounting for errors due to beginning and end points. The system

- normalizes beginning and end points of a stroke with respect to the enclosing rectangle,
- determines if the width or the height is larger than a given threshold (if so, it isn't considered a feature), and then
- calculates an error value from the comparison of the normalized beginning and end points of the input character and the candidate character stroke.

Step 4, determining characters. The system

- weights and adds state machine error and position error, and

- recognizes the character with the minimum error.

Figures 1 through 3 illustrate our algorithm. About 20 consecutive images are merged to obtain the *M* image shown in Figure 1b and 3d; the corresponding chain code representation is 32222207777111176666. The FSM for the character *M* is shown in Figure 2a. Consider the laser beam traces of four characters shown in Figure 3.

When the chain code is applied as an input to this machine, the first element, 3, generates an error and the error counter is set to 1. The second element of the chain, 2, is a correct value at the FSM's starting state so the error counter remains at 1 after processing the input 2. The FSM remains in the first state with the other 2s and also with the subsequent 0, as 0, 1, and 2 are the inputs of the machine's first state for *M*. Input 7 makes the FSM go to the next state, and the subsequent three 7s let the machine remain there. Whenever the input becomes 1, the FSM moves to the third state. The machine stays in this state until the single 7 input, and this makes FSM go to the final state. The rest of the input data, 6, makes the machine stay in the final state, and when the input is finished, the FSM terminates. For this input sequence, 1 is the machine's error for character *M*. In practice, this sample chain code determines all other characters using FSMs. However, the other FSMs generate either greater or infinite error values. You can easily see this on the character *N*'s FSM (see Figure 2b). If *M*'s chain code string is an input to this machine, it will never reach the final state and the error will be set to infinity.

Both the time and space complexity of the recognition algorithm are $O(n)$, where n is the number of elements in the chain code. The FSM recognition algorithm is robust as long as the user does not move his arm or the camera while writing a letter. Small changes due to hand trembling while writing can be corrected automatically by look-ahead tokens to improve the recognition rate. The look-ahead tokens act as a smoothing filter on the chain code. Instead of using deterministic FSMs, characters can also be modeled by hidden Markov models (stochastic FSMs) to further increase the system's robustness, but this also increases computational cost.

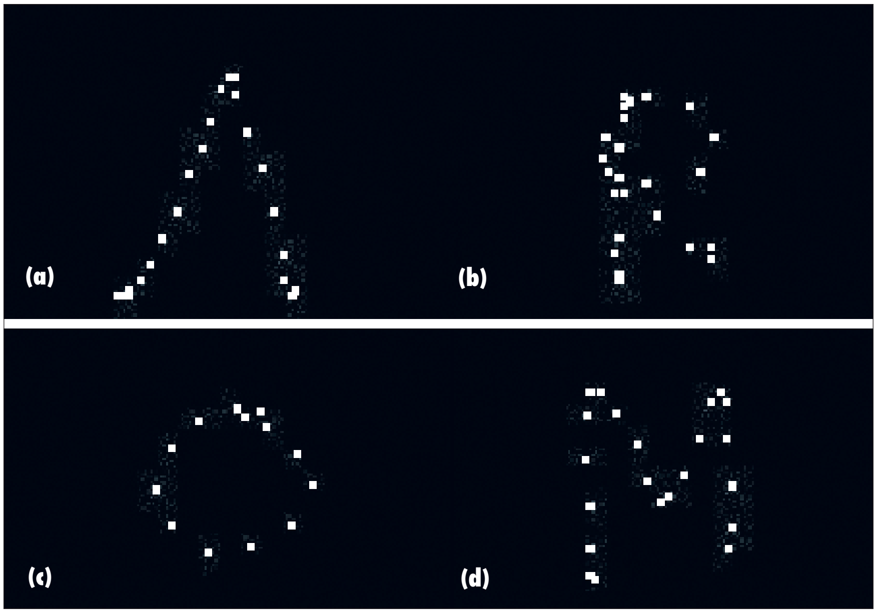

Figure 3. Laser beam traces generated by image sequences corresponding to (a) lambda (*A* in Graffiti), (b) *R*, (c) *O*, and (d) *M*.

Video processing

To extract chain code from the video, marker positions for the images corresponding to a character are processed. If the marker is in the initial frame, you can track it in the consecutive images. In our experiments, we used a red laser pointer to write the characters. Then, we decomposed the images into red, green, and blue components. *Thresholding*—a simple image-processing operation—followed by a connected component analysis identifies the red mark. If you use hand gestures, you might need a skin filter. We can similarly extract and trace other pointers (for example, a pen tip).

A laser pointer is the most robust text entry device in changing lighting and background conditions. As discussed earlier, in an image sequence corresponding to a word, discontinuous pointer movements separate characters. For a laser pointer, at the end of each character the user turns off the light. This marks the end of each character. For each character, we segment the video based on the jumps of the laser pointer's red mark. While the user is writing a character, the transition of the pointer positions in consecutive images should be smooth because the user writes only unistroke characters. The subsequent character will start at a relatively different position because the characters are written one at a time. Therefore, using a laser pointer naturally creates a deliberate discontinuity between two characters.

Two problems mainly arise during image capture and processing: distortion due to perspective projection and marker occlusion. Character distortion occurs when the user draws the hand gestures in a nonorthographic manner. Perspective distortion up to about 45 degrees of difference defined by the laser pointer (or regular pointer) between the camera and the forearm's tangent plane does not affect character recognition. The system fails after 45 degrees because the chain code used in character representation has a quantization level of 45 degrees (the unit circle is represented by eight directions). You can overcome this problem by either increasing the quantization levels and modifying the FSM models accordingly or by using Steve Mann's projective geometry methods[5–7] to provide an efficient solution with the help of feedback from a viewfinder. We don't consider occlusion in this system, because we assume the camera captures the images in front of the marker.

Experimental results

We used a red laser pointer, black background fabric, and a Web camera (an ordinary Philips PC Camera with a Tekram VideoCap C210 capturing card) in our experiment. The Web camera produces 160 × 120 pixel color images at 13.3 frames per second. We used an Intel Celeron 600 processor with 64 Mbytes of memory for all processing.

We have not yet implemented our system on a wearable computer; however, we believe our experimental setup and algorithm illustrate the results we would find with wearable computing applications. The processor we used performs similarly to the processors mentioned in current wearable computers. Furthermore, the Web camera used in our experiments has very similar characteristics with the head-mounted cameras used in wearable computers or the EyeTap (http://eyetap.org).

In our experiments, the user draws a Graffiti-like character using the red pointer on dark background material. In other unistroke recognition systems, you can achieve very high recognition rates.[4] In our system, in spite of perspective distortion, you can attain a recognition rate of 97 percent at a writing speed of about 10 words per minute. We also noted that the recognition process is writer-independent and writers required little training. We used the Graffiti-like alphabet because it resembles the Latin alphabet, and most people can use it without extra effort. Users can also define other single-stroke characters to use as bookmarks or pointers to databases, for example. Although it might be easy to learn other text entry systems, some people are reluctant to take the time to learn unconventional text entry systems. Computationally efficient, low power consuming algorithms exist for the recognition of unistroke characters. We can implement these algorithms in real time with very high recognition accuracy. After a user studies the Graffiti-like alphabet for a few minutes, about 86-percent accuracy is possible. After some practice, accuracy improves to about 97 percent. Almost 100-percent accuracy seems possible.[8]

To estimate the above recognition rate, we used at least 50 samples for each character and a total of 1,354 characters. The system requires an average of 18 image frames per character. Typically, a user draws these in less than 1.5 seconds. This means a data entry rate of more than 40 characters per minute on average. Users can improve writing speed if they spend time learning better ways to write certain characters. For example, the characters *I* and *T* can be drawn and recognized with almost 100-percent accuracy using only three to four frames. In contrast, the character *B* needs at least 50 frames (or more than 3.35 seconds) for reasonable recognition rate accuracy. Perspective distortion also plays a minor role in the system because everything is two-dimensional. In our experiments, we observed that degradation in recognition is, at most, 10 percent around a 45-degree difference between the writing plane and the camera.

We also conducted several tests under different lighting conditions. In daylight, the background's pixel value is about 50 whereas the pixel value of the laser pointer's beam is about 240. In incandescent light, the background's pixel value is about 180 whereas the beam's pixel value is about 250. In fluorescent light, the background's pixel value is about 100 whereas the beam's pixel value is about 240. In all cases, we can easily identify the laser pointer's beam against the dark background because enough contrast exists, especially if the user also wears a dark, solid color. If the user writes characters with her finger, we expect a slightly lower recognition rate. Writing with a finger is much more convenient than writing with a laser pointer; however, detecting the laser pointer's beam is simpler for image analysis.

Our current system's overall writing speed is below the 20-wpm composition rate reported for Graffiti on a PDA.[8] This is because a wearable camera's frame rate is much smaller than a PDA touchscreen's sampling rate. However, a PDA requires much slower writing movements when compared to our approach. Our recognition algorithm is also more complex and robust than the simple recognition algorithms used in PDAs.

Our system's writing speed is also lower than the 35- to 40-wpm transcription speeds of the septambic keyer and the Twiddler. However, regardless of the keyboard, composition writing speed is below 20-wpm for most people. We believe that in a wearable computing environment the composition speed rather than the transcription speed is important. Furthermore, we can achieve the 20 wpm writing speed with very high accuracy in our system (or in today's wearable computing technology) if we use an optimized unistroke alphabet[4] instead of a Graffiti-like alphabet. In such a case, the user would have to learn an alphabet consisting of even more simple strokes.

While our approach hasn't been implemented in wearable computing yet, several interesting applications are possible. For example, our current system is well suited for taking notes while watching a presentation if the camera has a viewfinder.[9–11] The viewfinder provides a feedback loop so the user can review and correct any errors in pointer-written characters as they occur.

We are working on generalizing the system to recognize continuous writing with a finger or stylus. We are also studying an alternative way to recognize characters using a wearable keyboard image and a laser light. You enter characters by shining light onto the character's location on the keyboard image. A finger or stylus can be used to mask the key locations to enter text. If you use an optimized keyboard image (such as the Pendragon Project's Cirrin or IBM's Metropolis), text entry speed can exceed the ordinary keyboard. ■

Acknowledgments

The authors thank the *IEEE Intelligent Systems* guest editor, Steve Mann of the University of Toronto, and the anonymous reviewers whose comments and corrections significantly improved this article.

References

1. O.N. Gerek et al., "Subband Domain Coding of Binary Textual Images for Document Archiving," *IEEE Trans. Image Processing*, vol. 8, no. 10, Oct. 1999, pp. 1438–1446.
2. E. Oztop et al., "Repulsive Attractive Network for Baseline Extraction on Document Images," *Signal Processing*, vol. 75, no. 1, Jan. 1999, pp. 1–10.
3. D. Goldberg and C. Richardson, "Touch-Typing with a Stylus," *Proc. '93 Conf. Human Factors in Computing Systems*, ACM Press, New York, 1993, pp. 80–87.
4. I.S. MacKenzie and S. Zhang, "The Immediate Usability of Graffiti," *Proc. Graphics Interface '97*, Morgan Kaufmann, San Francisco, 1997, pp. 129–137.
5. S. Mann, "Further Developments on Head-Cam: Joint Estimation of Camera Rotation + Gain Group of Transformations for Wearable Bi-Foveated Cameras," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 4, IEEE Press, Piscataway, N.J., 1997, pp. 2909–2912.
6. S. Mann and R.W. Picard, "Video Orbits of the Projective Group: A Simple Approach to Featureless Estimation of Parameters," *IEEE Trans. Image Processing*, vol. 6, no. 9, Sept. 1997, pp. 1281–1295.
7. S. Mann, "Humanistic Computing: WearComp as a New Framework and Application for Intelligent Signal Processing," *Proc. IEEE*, vol. 86, no.11, Nov. 1998, pp. 2123–2151.
8. S. Zhai, M. Hunter, and B.A. Smith, "The Metropolis Keyboard: An Exploration of Quantitative Techniques for Graphical Keyboard

Design," *Proc. 13th ACM Symp. User Interface Software and Technology (UIST 2000)*, ACM Press, New York, 2000, pp. 119–128.

9. S. Mann, "WearCam (The Wearable Camera): Personal Imaging Systems for Long-Term Use in Wearable Tetherless Computer-Mediated Reality and Personal Photo/ Videographic Memory Prosthesis," *Proc. 2nd Int'l Symp. Wearable Computers*, IEEE CS Press, Los Alamitos, Calif., 1998, pp. 124–131.

10. S. Mann, "Smart Clothing: the Wearable Computer and WearCam," *Personal Technologies*, vol. 1, no.1, Mar. 1997.

11. S. Mann, "Telepointer: Hands-Free Completely Self-Contained Wearable Visual Augmented Reality without Headwear and without Any Infrastructural Reliance," *Proc. 4th Int'l Symp. Wearable Computers*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 177–178.

For further information on this or any other computing topic, please visit our Digital Library at http://computer.org/publications/dlib.

The Authors

Ömer Faruk Özer received a BSc in electrical and electronics engineering from Bilkent University. Contact him at the Department of Electrical Engineering, Bilkent University, Ankara TR-06533, Turkey; oozer@ug.bcc.bilkent.edu.tr.

Oğuz Özün is a research assistant at Middle East Technical University. His research interests include computer graphics and computer vision, particularly three-dimensional model reconstruction from a sequence of images. He received a BSc in computer engineering from Middle East Technical University. Contact him at the Dept. of Computer Eng., Middle East Technical Univ., Ankara TR-06531, Turkey; oguz@ceng.metu.edu.tr.

C. Öncel Tüzel works for Meteksan Sistem at Bilknet University. His research interests include computer graphics and computer vision, particularly image-based rendering. He received a BSc in computer engineering from Middle East Technical University. Contact him at Meteksan Sistem ve Bilgisayar Teknolojileri, Bilkent University, Beytepe Koyu No:5, Bilkent, Ankara TR-06533, Turkey; oncel.tuzel@meteksan.com.tr.

Volkan Atalay is a faculty member at Middle East Technical University. Previously, he was a visiting scholar at the New Jersey Institute of Technology. His research interests include computer vision and document analysis. He received a BSc and an MSc in electrical engineering from Middle East Technical University and a PhD in computer science from the Université René Descartes-Paris V, Paris, France. He is a member of the IEEE Computer Society and the Turkish Pattern Recognition and Image Analysis Society. Contact him at the Dept. of Computer Eng., Middle East Technical Univ., Ankara TR-06531, Turkey; volkan@ceng. metu.edu.tr.

A. Enis Çetin is a faculty member of Sabanci University in Istanbul and is also currently on leave from Bilkent University. Previously, he was an assistant professor of electrical engineering at the University of Toronto, Canada, and a visiting professor at the University of Minnesota. He received a BSc in electrical engineering from Middle East Technical University and an MSE, and a PhD from the Moore School of Electrical Engineering, University of Pennsylvania. He is an associate editor of *IEEE Transactions on Image Processing* and a member of the DSP technical committee of the IEEE Circuits and Systems Society. Contact him at the Faculty of Eng. and Natural Sciences, Sabanci Univ., Orhanli TR-81474 Tuzla/Istanbul, Turkey; cetin@sabanciuniv.edu.