ON-LINE GESTURE RECOGNITION BY FEATURE ANALYSIS

Joonki Kim

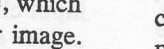
IBM Research Center
Yorktown Heights, NY 10598

Abstract

A feature analysus method was developed for the recognition of hand generated gestures (or markings). Gesture recognition differs from handwriting recognition because gestures are often generated in different proportions, rotations, and sometimes in mirror images. The features are based on direction changes and they are applied successfully to gestural variations. This recognition system is a part of a keyboardless direct manipulation interface to a spreadsheet application.

KEYWORDS: Direct manipulation, gestural variations, gesture recognition, electronic tablet, feature analysis.

Introduction

In recent years, recognition of hand-generated tablet-captured symbols has progressed to symbols other than handwritten characters and include shorthand, editing symbols, and flow chart symbols [1,2,3]. Recognition of such symbols usually depends on shape consistency. The difference between different instances of the same symbol must be less than the difference between different symbols. However, there is a class of symbols in which the same symbol can differ, in some respects, more than different symbols, yet recognizable by humans. An example is a commonly used delete symbol (), which varies in size, orientation, and sometimes in mirror image. Computer recognition of such symbols has been difficult because computers were not taught the features that will correctly classify such symbols.

Problems in gesture recognition

Unlike handwriting, there is no fixed set of gestures defined and used over the population. Proofreader's editing symbols are defined but not widely used. Additionally, gesture generation is not an overlearned skill for most people. Gesture generation is usually a slow conscious process. There seem to be room to define a set of gestures and let users learn and use them. The same gesture is often made differently by the same person at different times, and recognition should be designed for this possibility. Gestures also come in different sizes and orientations. Handwriting, in most cases, is on a writing line (which may be invisible), and have fixed heights. On the other hand, some gestures such as scoping can be small for a small scope like one letter but large for a large scope like a paragraph. The difference measures and features developed for handwriting recognition do not always work for gestures. Thus, a new measure of difference is necessary for reliable recognition of gestures. Figure 1 shows some delete gestures and circular enclosing scope gestures for a spreadsheet application.

We identified the following variations (Figure 2) in gestures.

1. Non-linear scaling
2. Rotation
3. Mirror image
4. Reverse direction at production time

We call them gestural variations.

In many handwriting recognition systems, the same letter, produced differently, is often recognized as more than one shape but later mapped to one symbol. For example, Suen [4] has identified 16 different ways of producing "E". Recognizing all of them as possible variations requires too much time and space. The same approach is possible for gestures - recognize the variations as different shapes, and then map all different shapes to the same gesture. Again, this allows recognition but each gesture may have to be recognized as many shapes and we encounter vocabulary explosion.

We have concluded that it is necessary to develop a new recognition algorithm to handle gestural variations. A new feature, direction change, was found to handle gestural variations. A recognition system that utilizes directions for fixed shape gestures and direction changes for varying gestures is developed.

Application

The gesture recognizer is a part of a spreadsheet application interface. Instead of a keyboard and a separate screen, a specially made transparent tablet on top of a flat screen [5] is used as an input/output device. With a cursor following the pen when the pen is close to the tablet surface and

electronic ink trace when the pen is touching the tablet surface, it is possible to make accurate marks on the screen-tablet combination. The user enters text by writing and enters commands and operands by gestures directly on the tablet. Without a keyboard, both gesture recognition and handwriting recognition are necessary. The fixed feature analysis gesture recognition developed in this paper and an adaptive handwriting recognition [6] are used. There are many other functions required for such keyboardless interface, and gesture recognition is one of them. A more detailed description of the interface can be found at [7].

Vocabulary

Gesture alphabet

Based on a paper and pencil study of gestures [8,9], we selected a few representative shapes as the initial gesture alphabet (Figure 3). This alphabet is a mix of gestures that vary little (fixed shape) and gestures that vary in many ways (varying shape). We simplified the problem by using single stroke gestures only. This is not a major limitation for the small vocabulary set, but future systems should recognize multi-stroke gestures.

Gesture commands

Usually, a gesture by itself does not make much sense. A number of gestures are put together to form a command, just the way alphabet letters are put together to form a word. Usually, a gesture command consists of a command (like move), a source, and a target. The recognizer described in this paper recognizes basic gesture alphabet shapes. Another function[10] that understands the application and screen display information collects the gesture alphabet elements together to form a complete command and send it to the application (spreadsheet in our example) for execution.

Output coordinates

In order to apply the gesture command to the spreadsheet, it is necessary to find the part of the display information marked by the gesture. For example, if the gesture is a caret, its tip coordinates are necessary to identify the part of the display which in turn identifies the location in the spreadsheet where space is to be inserted.

A set of output coordinates (usually tips, begin/end points, and/or surrounding box coordinates) are defined for each gesture. The recognizer, after successful recognition, determines the appropriate set of coordinates.

Preprocessing

Generally, tablet sampling rate is set high enough to capture all important points[11]. Then, this highly sampled data is filtered to a reduced set of points necessary for recognition. Over the years, filtering has evolved to include functions such as smoothing, wild point correction, hook removal, dot reduction and trim[12,13,14,15]. A trim filter works on internal points of a stroke (from pen down to pen up), and selects the points necessary for recognition. Another way of looking at the trim function is piecewise linear approximation of a stroke.

Smoothing filter

Inaccuracy of the tablet, and digitization noise makes smooth lines wiggly. A digital low pass filter[16] was used to remove wigglyness and produce smooth output.

Acceleration filter

We developed a new trim filter for gesture recognition[17]. It relies on pen dynamics, namely velocity and acceleration. The mass of a human hand holding the pen and the force that can be applied to it limits acceleration. Large acceleration is from large velocity change and/or angle changes (Figure 5). Electronic tablets sample pen positions at a fixed time interval, typically 100 samples per second. This provides a convenient means to compute velocity and acceleration. At point i, the magnitude of acceleration ac(i) is:

```
ac(i)=sqrt(sq(x(i+1)-2*x(i)+x(i-1))
          +sq(y(i+1)-2*y(i)+y(i-1))).
```

When ac(i) is greater than a threshold, there is a significant change in speed, and/or direction. Such a point is probably necessary for recognition and it is kept. A similar idea has been used successfully to detect wild points[15].

Direction filter

A second-level filter based on directions is developed for gestures. This filter is necessary because we use only 12 directions and hence introduce quantization noise. After the second-level filter, for example, a caret, which may had 30 points initially will usually be reduced to three points.

Development of a new algorithm

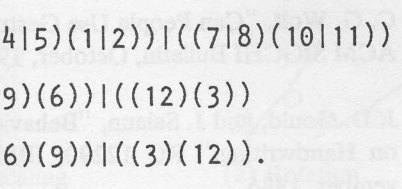
Directions are used to recognize gestures for two main reasons. Recognition of handwritten characters using directions is well established, and we have developed a new algorithm that uses direction changes. We can use the same input characterization for both.

12 directions

Eight directions (N E W S NE SE SW NW) have been widely used to recognize handwritten characters. For example, the letter "L" could be characterized as S-E. But eight was not enough for gestures. We used 12 directions (1 through 12, corresponding to clock directions).

Fixed shape recognition

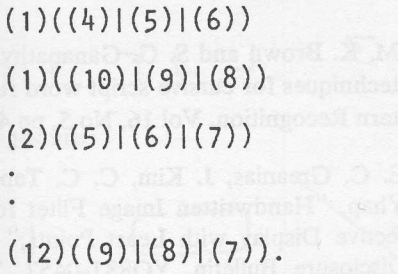
Fixed shape gestures can be recognized conveniently and speedily by direction analysis. Following is a list of example gesture shapes and directions under ideal conditions:



A flow table with 12 inputs for each direction is used as a first stage recognition for fixed shape gestures. This first stage recognition can handle minor rotations and size variations, but not major gestural variations.

Varying shape recognition

An arrowhead can be characterized as:



We can map 24 different shapes to one arrowhead. This is rather clumsy and inefficient. 24 expressions are really a single expression

```
(n)((n+3)|(n+4)|(n+5)|(n-3)|(n-4)|(n-5))
where 1 <= n <= 12.
```

An arrowhead can be characterized as any first direction and second direction differing from the first by 3, 4, 5, -3, -4, or -5.

Non-linear scaling is handled by ignoring vector length. Rotation is handled by ignoring the first direction and concentrating on direction changes. Mirror image is handled by the same expression, but with negative direction change. Reverse direction is handled by the expression written backward.

When the first stage recognition fails, a second stage recognition is invoked. This second stage consists of codes tailored for each symbol and uses direction changes extensively. The delete symbol can be described as 4 or more vectors, total direction change of 12, and beginning and ending tails.

Figure 6 shows a delete symbol as captured by the tablet, after filtering, directions and direction changes. There are 9 points and 8 vectors connecting them. Direction change from the first to seventh vector is -12, and a delete gesture is recognized.

Result

The recognition algorithm as discussed above was implemented using the C programming language. It consists of about 2000 lines of source code, and requires about 100KB of storage. It runs on various computers and in either interactive or batch mode.

An experiment was set up with IBM PC AT, RTPC, and two tablet display combinations. One tablet display combination consists of a transparent tablet over LCD display with electronic ink, and another consists of an opaque tablet and a separate graphic display with electronic ink. A previous study concerned a transparent tablet over flat display[5].

Twenty subjects were presented with spreadsheets containing instructions to manipulate the spreadsheet (Figure 7). Strokes generated by the tablet were collected without recognition and later analyzed. Data gathered from this experiment was used for two studies. One study compares the keyboard against the direct manipulation gestural interface[18]. This study evaluates the gesture recognition algorithm. Not all strokes were gestures, and because of human and tablet errors, not all gesture strokes were usable for this study. Each gesture, as captured by the tablet, was viewed on a CRT and subjective decisions were made on acceptability.

Writings from six writers were used to improve accuracy, and the improved recognition was tested against all twenty writers. Improvements are of two types. First, boundaries for different directions were changed. Initially, each of the 12 directions was 30 degrees wide, but the vertical and horizontal direction ranges were enlarged with positive effect. This is due to customary slanted writing habits. Second, feature extraction was modified to accommodate the variety of styles from six subjects.

The accuracy was initially about 80%. The recognition results after improvement are shown in Figure 8. Recognition rate is (recognized/written), and accuracy rate is ((correctly recognized)/ recognized).

Most unrecognizable gestures have hooks or retraces. Small hooks at the beginning and end of a stroke are removed, but when the size of the hook approaches the size of the gesture itself, and become a retrace, it is very difficult to decide if it is a hook or part of a stroke. Improved hardware and user adaptation may ease this problem.

Mis-recognitions are among similarly shaped gestures. For example, a caret pointing down and slanted too much can be recognized as a general arrowhead. A dot, written as a short line, can be recognized as a line gesture.

Summary

A new recognition system that recognizes gestures has been developed. This is a fixed feature analysis system using directions and direction changes. This system can recognize gestures that were not recognizable previously.

A designer of a recognition system should consider the vocabulary and decide on a set of features that will distinguish all symbols uniquely. The feature set will include many of the features that have been used in the past and may include some new features based on direction changes. Direction changes add to the previously-established set of features and expand the vocabulary of the recognizer.

Acknowledgement

This work is a part of a larger effort to build a keyboardless direct manipulation interface to a spereadsheet application. The author is grateful to the management for providing such environment, and to all the group members for their support. Special thanks are to C. Wolf, M. Sacks and K. Leone for their help in the data collection effort.

References

[1] C. G. Leedham, A. C. Downton, "On-Line Recognition of Shortforms in Pitman's Handwritten Shorthand," Proc. 7th Int. Conf on Pattern Recognition, pp.1058-1060, July, 1984.

[2] T. Sakai, K. Odaka, and T. Toiba, "Several Approaches to Development of On-Line Handwritten Character Input Equipment," Proc. 7th Int. Conf on Pattern Recognition, pp.1052-1057, July, 1984.

[3] H. Murase, T. Wakahara, and M. Umeda, "Online Recognition Algorithm for Hand-Sketched Flowchart by Candidate Lattice Method," Systems, Comuters, Controls, Vol 14, No. 3, pp. 37- 46, 1983.

[4] C. Y. Suen, "Handwriting Generation, Perception and Recognition," Acta Psychologica, 54, pp. 295-312, 1983.

[5] C. C. Tappert, A. S. Fox, J. Kim, S. E. Levy, and L. L. Zimmerman, "Handwriting Recognition on Transparent Tablet over Flat Display," Proceedings of the SID, Vol.28/1, pp.67-74, 1987.

[6] H. Ellozy, Private Communications, 1986.

[7] J. R. Rhyne, and C. G. Wolf, "Gestural Interfaces for Information Processing Applications," RC 12179, IBM Research, September, 1986.

[8] C. G. Wolf, "Can People Use Gesture Commands?" ACM SIGCHI Bulletin, October, 1986.

[9] J. D. Gould, and J. Salaun, "Behavioral Experiments on Handwriting," RC 12290, IBM Research, November, 1986.

[10] J. Rhyne, "Dialogue Management for Gestural Interfaces," RC 12244, IBM Research, October, 1986.

[11] J. Kim, and C. C. Tappert, "Handwriting Recognition Accuracy versus Tablet Resolution and Sampling Rate," Proc. 7th Int. Conf on Pattern Recognition, pp. 917-918, July, 1984.

[12] M. K. Brown and S. G. Ganapathy, "Preprocessing techniques for cursive script word recognition," Pattern Recognition, Vol 16, No 5, pp 447-458, 1983.

[13] E. C. Greanias, J. Kim, C. C. Tappert, and E. F. Yhap, "Handwritten Image Filter for Enabling Effective Display with Least Points," IBM Technical Disclosure Bulletin, YO881-0457, Vol 27, No 8, January, 1985.

[14] J. Kim, and C. C. Tappert, "Tablet Image Filter for Recognition Purposes," IBM Technical Disclosure Bulletin, YO881-0091, Vol 28, No 3, August, 1985.

[15] J. Kim, and C. C. Tappert, "Correction of Wild Points within an Electronic Tablet Data," IBM Technical Disclosure Bulletin, YO884-0269.

[16] H. Ellozy, Private Communications, 1986.

[17] J. Kim, "Acceleration Filter for On-Line Hand Input Recognition," YO887-0342, IBM Technical Disclosure Bulletin, 1987.

[18] C. Wolf, "A Comparative Evaluation of Gestural and Conventional Interfaces," RC 13187, IBM Research, October, 1987.

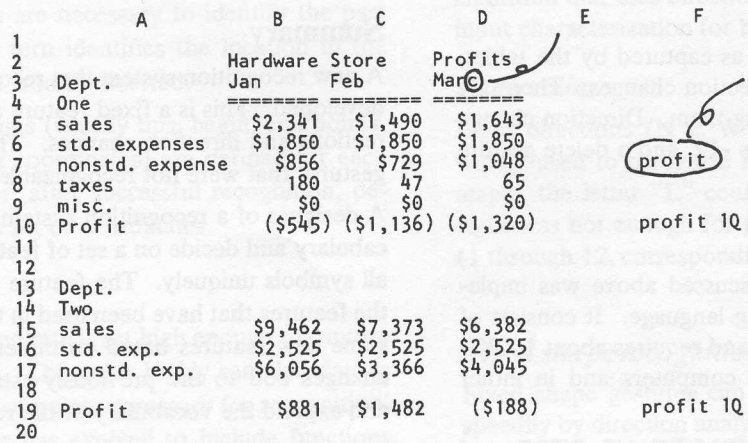


Figure 1. Direct manipulation gestures.

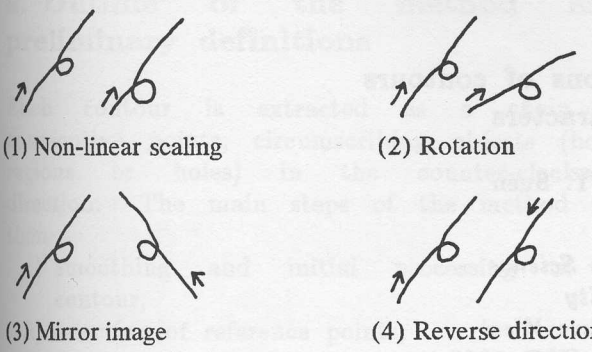


Figure 2. Gestural variations.

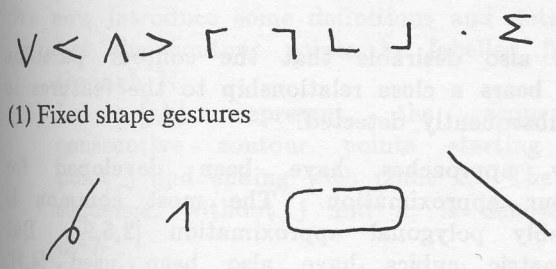


Figure 3. Gesture recognition alphabet

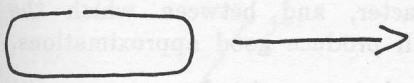


Figure 4. Move command built from scope, line and arrowhead.

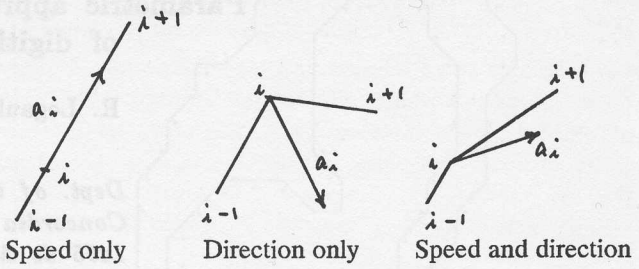
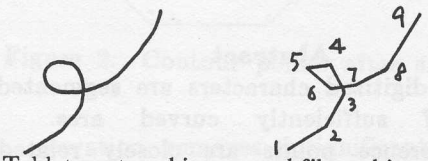


Figure 5. Speed and/or direction changes



points	1	2	3	4	5	6	7	8	9
directions		2	1	11	9	5	3	2	1
direction changes		-1	-2	-2	-4	-2	-1	-1	

|<------ -12 ------->|

(2) Recognition

Figure 6. Recognition of delete gesture.

	transparent	opaque	both
written	804	1032	1836
recognized	795(98.9%)	1015(98.4%)	1810(98.6%)
correctly recod	737(92.7%)	955(94.1%)	1692(93.5%)

Figure 7. Recognition result for 20 subjects

	A	B	C	D	E	F	G
1	m71e		***ERASE A12..G12***				
2							
3	The	Mountain	Supply	and	Valley	Hikers	Discount
4							
5							
6							
7							
8							
9	Item	Cost	Jan	Feb	Mar	Apr	May
10	pack b20	2480	102	89	87	95	115
11	pack b24	2699	67	60	62	89	87
12	pack b26	3050	88	64	64	56	78
13	knife n21	1400	14	12	21	22	27
14	knife n22	3295	14	15	17	19	30
15	knife m4	2650	20	23	34	32	34
16							
17							
18	pack	Totals	257				
19	knife	Totals					
20							

Figure 8. Spreadsheet with instruction and gesture input. Lower right corner gesture is not recognized.