

machines. Whether the computer system is oriented toward the design of automobiles, bridges, mathematical models, or computer programs, more and more is being demanded of the man-machine communication channel to match the computer's growing ability to quickly answer questions or obey directives.

REFERENCES

- [1] M. I. Bernstein, "Computer recognition of on-line, hand-written characters," The RAND Corporation, Santa Monica, Calif., RM-3753-ARPA, October 1964.
- [2] R. M. Brown, "On-line computer recognition of hand-printed characters," *IEEE Trans. on Electronic Computers*, vol. EC-13, pp. 750-752, December 1964.
- [3] W. Teitelman, "Real-time recognition of hand-drawn characters," *Proc. 1964 Fall AFIPS Joint Computer Conf.*, vol. 26, pp. 559-575.
- [4] I. E. Sutherland, "Sketchpad: a man-machine graphical communication system," *Proc. 1963 Spring AFIPS Joint Computer Conf.*, vol. 23, pp. 329-346.
- [5] M. R. Davis and T. O. Ellis, "The RAND tablet: a man-machine graphical communication device," *Proc. 1964 Fall AFIPS Joint Computer Conf.*, vol. 26.
- [6] G. F. Groner, "Real-time recognition of handprinted text," The RAND Corporation, Santa Monica, Calif., RM-5016-ARPA, August 1966.

The Computer Graphics User/Machine Interface

GARY D. HORNBuckle, MEMBER, IEEE

Abstract—In many instances, computer graphics can provide a powerful, rapid man-machine interface by proper application of simple pattern recognition techniques. The techniques briefly discussed are those used to classify real-time sequences of x, y coordinates such as occur with several graphical input devices, including the RAND tablet.

Several examples show how graphics can improve a class of editors and debuggers generally operated with keyboard/printer consoles.

INTRODUCTION

A PROBLEM currently receiving considerable attention is how best to use the speed and accuracy of digital computers to aid man in the slow, often trial-and-error design process. Time-sharing has solved the economics problem of keeping the machine busy when it would otherwise be idle while the on-line user is thinking. Terminals are now available which allow close man-machine interaction, especially consoles which provide rapid pictorial output and, with the appropriate software, are capable of providing fast, semi-natural, straightforward input. Of primary concern in this paper is the graphical communication, in particular, the application of pattern recognition techniques to aid the on-line user.

There are, of course, an unlimited number of different muscular contortions men perform to pass information. The opposite situation has generally existed with respect to the number of variations in man-computer information exchange. Indeed, punching little holes in paper has

become a way of life for most computer users. To minimize confusion and cost of misplaced holes, some on-line systems have been developed which do little more than provide semi-rapid computer feedback to instruct which holes should be moved. The apparent extension of this "general" information media to some computer display systems has resulted in consoles with as many as 600 different size, color, and shaped buttons [1]. Only recently have computer systems designers become interested not only in what information the user must transmit, but how [2], [3].

Although different man-computer interfaces will be debated and tried for years to come, observing what man does normally during his creative efforts can provide a starting point for the interface designer. In particular, a mathematician does not manipulate equations at a typewriter, nor does a circuit designer prefer a keypunch. Both rely heavily on paper, pencils, and, not least, erasers. A properly designed graphic system can provide an information transfer media similar and in some ways superior [4] to paper and pencil, and computers make particularly good erasures. Several experimental systems [3], [5] have shown that a single hand-held pen-like device, a RAND tablet [7] stylus, for example, being monitored by the computer can provide an information transfer media well matched to the man and quite sufficiently matched to the machine. The critical ingredient is the pattern recognition program interpreting the user's various hand motions.

The amount of effort and lack of results in the field of pattern recognition as evidenced by the number of publications need not be discouraging since most workers have attempted to solve the most general case—all possible fonts of all possible symbols. Much simpler is the task

Manuscript received July 12, 1966; revised October 3, 1966. The work in this paper has been supported in part by the Advanced Research Projects Agency, under Contract SD-185.

The author is at the University of California, Berkeley, Calif.

of specifying a small, well-formed set of symbols allowing only enough variation for the conscientious user to be comfortable. The recognition in this case is to be considered a tool to aid communication. Once the symbol set (interface) has been specified by the system designer, a programmer next determines the distinguishing characteristics, or features, of the symbols and writes a program which searches for the features from the x , y coordinates representing the hand motion. Finally the program decides which symbol was written and displays the results.

As an obvious example, consider the problem of entering electrical circuit diagrams to a computer. The normal graphical procedure is to present the user with several choices of components, each with several orientations. One must, in some order, pick the component, its physical orientation, what it connects to, and its value. But the obvious thing that comes to mind is to simply draw the circuit, much as one would on paper. This requires computer recognition of 20 or so input symbols (or strokes whose combinations make up symbols) such as the 10 digits, \sim for resistors, $|$ and $($ in proper relationship for capacitor, \sim for inductor, \nearrow intersecting $|$ at proper places for transistors, etc. Several very simple recognition schemes have been reported that can accomplish this [6], [7].

The recognition problem associated with such "natural" 2-dimensional real-time graphical input breaks down into two parts: stroke classification and stroke context.

STROKE CLASSIFICATION

The graphical input considered here is in the form of sequences of pen strokes where a stroke is a sequence of x , y coordinates which represent the path of the user's hand as he draws. A stroke terminates when the hand lifts.

The real-time coordinates are normally sampled at a high fixed rate, but the user's hand accelerates. The resulting redundant coordinates are removed since the acceleration information is not often used. Additional coordinate preprocessing is usually necessary to smooth the raw track to remove low-order bit fluctuations due to noise. From the smoothed, thinned track typically containing from 20 to 100 coordinates are derived the features necessary for stroke recognition.

The absolute positional information is used in the stroke context recognition—relative values determine stroke features. Features are generally sought which provide the greatest degree of separation of strokes into distinct classes but which are easily and rapidly determined. For real-time recognition, it is undesirable to provide a class of rejected (unknown) strokes because a guess is at least as good as a reject since the user must always repeat a rejected stroke.

If very many different stroke classes exist, it is unlikely that a set of feature tests can be found to which all stroke coordinates should be subjected. A scheme in

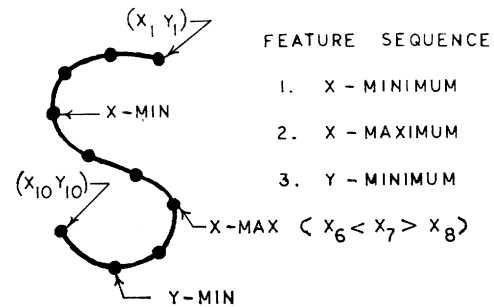


Fig. 1. Relative max-min features. This simplified drawing does not show the usually necessary hysteresis test which eliminates features being generated by very small coordinate fluctuations. That is, a feature is accepted if, and only if, $X_{i-1} > X_i + \Delta < X_{i+1}$ for min in X , or $X_{i-1} < X_i - \Delta > X_{i+1}$ for max, where Δ is the hysteresis parameter.

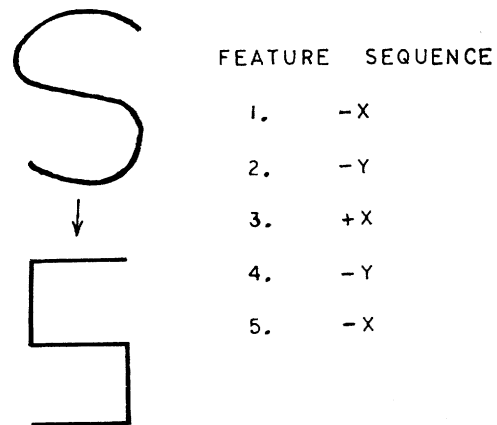


Fig. 2. Features from four axial direction quantization. In this case, hysteresis zones about 16° wide prevent unwanted features from being generated when the pen is moving near ($\pm 8^\circ$) a 45° direction.

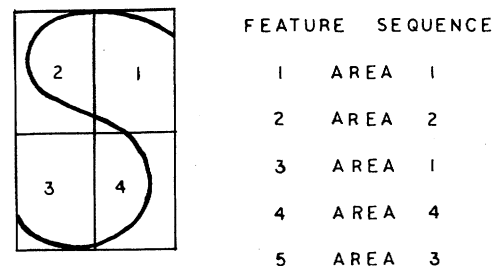


Fig. 3. Subregion sequence features. Hysteresis is provided by considering the lines separating areas having finite width, Δ , across which the pen must travel to change areas. It should be stressed that the areas as shown here should not be displayed or otherwise apparent to the user.

which a few general tests whose results are used to select further, more particular, tests has usually proved better.

Several general schemes exist, each guaranteed not to handle specific important cases without special provisions. Strokes containing many curves are nicely classified by the sequence of sign changes of the first derivative, i.e., relative maxima-minima in x and y (Fig. 1). A similar scheme quantizes the stroke into the 4 axial directions, discarding all distance information (Fig. 2). Another divides the area covered by the stroke into

regions, and extracts the sequence of regions passed through (Fig. 3). These schemes are very simply implemented, require very little computing or memory, are very powerful for simple problems (such as the circuit diagram input example), and are easily expanded or modified because of their generality.

However, superior results are often obtained if the set of features is specifically tailored to the problem. For example, if only 0 and 1 were to be distinguished, the only feature necessary would be the distance between the initial and terminal coordinates. But if later a 2 were added to the stroke set, a completely new feature test would have to be written into the program.



Aside from the few general considerations mentioned above, finding an optimal set of features for a specific problem is still an art requiring considerable experimentation. Fortunately, a high (95 percent or better) degree of recognition is not necessary; the user immediately detects errors due to the output feedback and can take corrective action. Also, most users adapt readily to a given scheme, and protest changes more than the initial learning phase.

STROKE CONTEXT RECOGNITION

Classifying recognized strokes into symbols presents no special difficulty although the context is usually a function of space and time. Vertical and horizontal strokes which cross may be T or $+$; if they don't cross, they may in the future become part of a T or $+$. One can adopt the simple policy that once it is obvious that a new symbol is started, or that a given time span has elapsed in which no input has occurred, previous strokes make up completed symbols which can then be recognized and displayed.

For example, in recognizing the electrical symbols



the resistor  and inductor  are immediately recognizable since they are one-stroke symbols and are not a part of another symbol. The remaining symbols all contain a straight line, which, if drawn first, cannot be recognized until another stroke is added. The second stroke being a $($ would result in capacitor $||$. A second stroke of ∇ would indicate a diode $\nabla|$ or transistor $\nabla|K$; the decision would remain until the third stroke. If the third stroke were not added within a specified lapsed time, then a diode would be displayed.

EQUIPMENT NECESSARY

A graphic console capable of simple pattern recognition input need only have a fairly accurate, moderately fast, coordinate input such as is provided by the RAND tablet or the SRI Mouse [9] and a fairly accurate display for output. These devices need not be tracked in the light-pen sense; computer intervention is required to read X , Y coordinates at less than 1 kHz rate. Feedback from the

device coordinates provides visual tracking. To pick out single characters in a line of 80 or so, the output (feedback) must correspond within ± 0.2 percent of the theoretical (reproducibility), a figure met by all current reputable manufacturers.

Special hardware can be provided to perform the smoothing operation. The Project GENIE display system [8] samples a hardware computed average of 16 RAND tablet-produced coordinates every 5 ms. The computer then simply deletes redundant coordinates to provide the thinned track.

APPLICATIONS—TEXT EDITING

The applications of graphics to input of geometrical constructs is not discussed here. However, graphics can be fruitfully applied to traditionally nongraphical single dimensional computer I/O such as text editing and program debugging.

The class of text editors considered here are computer programs which allow user manipulation of arbitrary strings of alphanumeric symbols. Typical operations allow insertion, deletion, and moving of substrings. Most current on-line editors are operated through keyboard-printer terminals. However, simple transliteration from character printer output to scope output achieves little advantage. Indeed, one might just as well get a faster printer.

However, a graphical text editor can provide advantages not possible with the fastest printer. First is lack of necessity for retyping to verify a change. The scope version should reflect the exact current state of affairs at all times. Second, much less familiarization is necessary because the scope editor should be much simpler. Third, the control and limited text input in many cases may be faster. However, until voice recognition is a reality, typewriter input remains the fastest general text input device for most users.

One need not be very sophisticated in certain instances, and still provide truly graphical input. A typical scope editor has been reported [10] which required manipulation of 27 light buttons displayed on the CRT. For example, for erasing, one picked a left-delimiter, a right-delimiter (each requiring two picks—button and symbol) and then an erase button. A quite different scope editor, VISE [11], was recently completed which performed the same functions by simply drawing a horizontal line through the characters one wished to delete.

The "pattern recognition" in VISE involved distinguishing 3 strokes, a dot which placed a pointer for keyboard insertion, a horizontal line for deletion, and \backslash or $/$ (nonhorizontal, i.e., $\Delta y > \text{constant}$) to indicate extent followed by $.$ to move the text. The only light buttons were for input-output, a pseudo-button for moving (scrolling) text, and the borders reserved for negating partially completed operations. VISE, written for a PDP-5, required less than 2000 machine instructions.¹

¹ The Digital Equipment Corporation PDP-5 is a small general-purpose computer with 4000 words of 12-bit, 6- μ s memory.

GRAPHICAL DEBUGGING

Unfortunately, the majority of the computer industry has largely ignored the problem of program debugging. The attitude has prevailed that given complete documentation for the languages available, the programmer has no excuse for making mistakes. In practice, however, manuals are usually unclear, incomplete, and contain errors, necessitating trial-and-error programmer experimentation. In addition, most nontrivial programs tend to be sufficiently large that logical errors in the program or algorithm itself are a virtual certainty.

Time-sharing, in providing a low-cost means for programmer experimentation, has been the most significant recent contribution to the debugging problem. The on-line environment has provided further impetus for the exploitation of debugging techniques, notable among which is the class of DDT-like machine-language debuggers.

Although recognized by all initiates as an outstanding aid and by all steady users as a virtual necessity, DDT is nevertheless completely non-mnemonic and difficult to learn. The Project GENIE DDT² [12], for example, is operated through a standard (types 33 and 35) teletype console and contains over 60 basic, distinct control operations, each requiring one or two teletype key depressions. This communication sterility achieves maximum input speed. Practically no messages are available—one must have a manual readily available. These shortcomings of DDT can be eliminated by graphic techniques.

To see how a graphical DDT might appear to the user, consider the following operations, common to all machine-language debuggers.

- 1) Display contents of a selected machine register in one of several formats.
- 2) Change contents of any machine register.
- 3) Execute a selected portion of the program.

DDT has devoted 21 commands to selecting register typeout formats and modes. A graphical DDT could display all the different formats for a selected set of cells at once. The selection of a cell and the changing of its contents could be achieved by writing the new value or location directly over the top of the old, thereby causing the old value to be replaced. If an expression in a certain scope field were desired in another field, the operation could be to place the pen down in the former, move it to the latter, and lift. The second field would immediately be displayed identical to the first. A field could be scrolled (i.e. obtain its predecessor or successor) by simply moving the pen vertically up or down within the field.

The symbol set to be recognized for such a graphical DDT would be the symbols used in forming cell location labels and cell value contents, several editing marks such as \wedge to place a pointer for insertion by hand (or keyboard)

and \equiv to erase, and \int or \downarrow to scroll a field with adequate separation of the fields for the move operation.

CONCLUSIONS

Although users can and have been trained to perform quite complex control tasks, simple applications of pattern recognition applied to the user's motions can greatly improve the man-machine interface. This paper has attempted to show how this can be accomplished with a particular class of input devices, those which generate x , y coordinates. Although machine cost in this case may be greater than a button/knob-oriented interface, the number of user control operations to perform an elementary task may be greatly reduced. The action of drawing a symbol directly in place provides implicitly the context (positional) information which would otherwise require several additional button presses. Contrary to much current practice, the burden should be placed on the machine, not the man. In many real-time situations the cost of relatively infrequent machine misinterpretation is small.

REFERENCES

- [1] R. N. Seitz, "AMTRAN, an on-line keyboard computer system for scientific and engineering use," NASA Technical Memorandum, NASA TM X-53243, July 1966.
- [2] J. C. Shaw, "JOSS: A designer's view of an experimental on-line computing system," *Proc. 1964 Fall AFIPS Joint Computer Conf.*, vol. 26, pp. 455-464.
- [3] T. O. Ellis and W. L. Sibley, "On the development of equitable graphic I/O," The RAND Corporation, p. 3415, July 1966.
- [4] I. E. Sutherland, "Computer inputs and outputs," *Scientific American*, vol. 215, pp. 86-96, September 1966.
- [5] A graphical machine language debugging and operating system for an IBM 7040 was in operation at the RAND Corporation from January 1964 until August 1966. The sole input was through a RAND Tablet and consisted of the 10 digits, \cdot , $+$, and $-$. Users were generally ecstatic about the transparency of the man-machine interface. The work was never documented in a distributable form, although a description of the character recognition algorithm can be obtained from the author: G. D. Hornbuckle "G. CHAR, a program which recognizes handwritten digits," January 1, 1964. The experiences gained with the 7040 graphic system are currently being exploited in the RAND project GRAIL op. cit [3].
- [6] G. F. Groner, "Real-time recognition of handprinted text," The RAND Corporation, RM-5016-ARPA, August 1966. Also [3] op. cit. gives further references concerning recognition of handprinted symbols.
- [7] T. O. Ellis, "The RAND tablet device and its communication capabilities," presented at the 1965 University of Michigan Engineering Summer Conference on Computer Graphics.
- [8] G. D. Hornbuckle, "GO, GENIE graphical input/output system," Doc. No. 30.80.10, Project GENIE, Dept. of Elec. Engrg., University of California, Berkeley, Calif., July 1, 1966.
- [9] The SRI "mouse" is a hand-held input device which, when rolled on a table surface, gives x , y coordinate information. Details of the "mouse" can be obtained from D. T. Engelbart of Stanford Research Inst., Menlo Park, Calif.
- [10] G. E. Roudabush et al., "The left hand of scholarship: computer experiments with recorded text as a communication medium," *Proc. 1965 Fall AFIPS Joint Computer Conf.*, pt. 1, vol. 27, pp. 399-411.
- [11] G. H. Heitzman and T. J. Cieslak, "VISE, a visual interactive system for editing," Doc. No. 40.10.140, Project GENIE, Dept. of Elec. Engrg., University of California, Berkeley, Calif., July 26, 1966.
- [12] L. P. Deutsch and B. W. Lampson, "DDT, time-sharing debugging system reference manual," Doc. No. 30.40.10, Project GENIE, Dept. of Elec. Engrg., University of California, Berkeley, Calif., March 4, 1966.

² All references to DDT from this point on concern Project GENIE DDT.