63-1-2

(10)

REAL-TIME, TIME-SHARED COMPUTER PROJECT

Sixth and Seventh Quarterly Progress Reports

October 8, 1962

Computation Center
Massachusetts Institute of Technology
Cambridge 39, Mass.

Contract Number
Nonr-1841(69)
DSR #8644

Submitted by: [illegible]
Philip M. Morse, Principal Investigator

[illegible]
Herbert M. Teager, Project Head

Summary

Work over the past two quarters has continued in all phases of this project, ranging from devices through software, in the areas of Graphical inputs, Graphical outputs, Problem- and Procedure-oriented languages, and the scheduling and allocation of a Central Processor with backup temporary storage devices. Progress in all of these areas has been heartening, particularly those converted with the device and character recognition aspects associated with implementing graphical languages. Test results indicate that block-printed characters can, with a high degree of reliability, be identified without requiring either excess memory space or machine time, so long as the carefully-drawn input is correlated with a particular user's set. The approaches that proved successful with alphameric fonts of block-printed characters proved insufficient for the more subtle variations associated with larger, mixed fonts of cursive characters, and a broader approach encompassing both extremes has been derived. This new approach, while not yet reduced to software, has been simulated with experimental data and found to lead to results which should be quite adequate for a system with graphical feedback to resolve any uncertainties in identification. In the process, a much more stringent analysis has been made of the relationship between the appearance of symbols and the process of making them, as well as the limitations of an arbitrarily large and unrestricted font of characters.

As a prelude to designing specific, problem-oriented, graphical languages, feasible algorithms for translating into algebraic languages information that could be derived from graphical descriptions have been considered for electrical networks and boundary value field problems.

With respect to scheduling and allocation procedures and the re-organization of the MAD compiler, the requisite software is being tested, although portions of the hardware system, such as the disc storage, must still be simulated.

Finally, with respect to hardware and equipment, three on-line flexo-writers and a photoelectric tape reader have been operating satisfactorily, while the special-purpose, plotter control computer and three on-line plotters are still not satisfactory with respect to reliability, with primary difficulties centering in the associated disc storage unit. Software development to provide graph, contour plotting, and symbolic output is, however, continuing, and will be made available, via special statements in the MAD language. A graphical input device, based upon the original design and concept of this group has been

built by a government laboratory for their own uses, and is operating satisfactorily. We are now working on an advanced design for this device.

For the immediate future, work in each of the areas indicated above will continue. For the longer range, possible improvements to the over-all time-shared, real-time system, such as optical, read-only memory, small-sized CRT displays with small, inherent storage for rapid interrogation, and improved plotters to couple with the graphical input device, will be explored.

I. Graphical Input, Device

As has been mentioned in earlier reports, a principle for a graphical input device, capable of operating in real time with a special "pencil" and ordinary paper, had been found. The principle involved recognition of a stylus position by means of coded sequences of current pulses in a matrix of conductors below the plane of the paper. Experimental and theoretical work at that time indicated that the principle was sound, and has been confirmed by the performance of full scale device at another institution. As envisioned, the device would incorporate slope detection facilities (quantized into octants), and thus, if desirable, point information could be recoded into line segment information for easier computer assimilation. It would also be linked to a graphical output device (for feedback purposes) by means of a common supply of paper. Prototype development of this device has necessarily lagged due to other, more urgent problems, and all ramifications of this basic capability have meanwhile had to be considered in terms of its design characteristics, rather than actual measured performance.

The design characteristics of the device are that it provides, to a resolution of .01", the binary coordinates of the stylus whenever it is lifted or placed on the paper or whenever the direction of movement of the stylus crosses the boundaries of one of eight 45° octants extending over vertical, horizontal, and $\pm 45^{\circ}$ directions.

II. Graphical Input, Character Recognition

Recognition of a wide range of hand-printed characters is not a pre-condition for a feasible graphical language, such characters can also be entered into a computer via a keyboard, once their position on the surface has been located by the pencil. It is, however, an interesting problem in its own right, particularly since a much wider class of symbols in theory can be produced by hand than with a keyboard, with seemingly less distraction and attendant disruption of the users' attention. Also, since the graphical input device planned had characteristics which essentially eliminated the scanning problem, and the graphical output device could provide a necessary corrective feedback for error or uncertainty, it was felt that input of hand-drawn characters might be feasible without too many restrictions on the user. It was felt that rather than attempting to solve the general pattern recognition problem (which in the case of isolated hand-drawn characters taken from a very large font may not have any meaningful solution), a much more restricted problem of matching a particular sample to one of a font provided by a particular user would be more tractable.

There is little question that the user cannot be given the same freedom that he may choose to exercise in his normal writing and printing where he can always use context. The very fact that he is asked to define his own font is partly in recognition of this. Further, he cannot be allowed the normal overlap of characters (for example, single stroke "I's" or numeral ones), that he can resolve by context. At the other extreme, the constraints cannot be too binding or else he will soon forego the pencil for other alternatives.

From the characteristics of the input device, it could be assumed that the input device would be presented the character in the form of a sequence of x-y coordinate beginning and end points of line segments resulting from actions such as changing the slope or lifting the pencil off of the paper. Because of the manner in which the input was generated, the system chosen had to be, to a large extent, scale, position, and rotation insensitive, i.e., a user might reasonably be expected to write anywhere on the surface, in varying slants and scales. It further had to allow for a very wide class of symbols, perhaps up to two hundred. To be feasible, the recognition scheme had to be reasonably efficient and fast for each user, and require a minimum amount of program space for both the recognition program and for the specific font data of each user. For reasons of efficiency, problems of context (such as deciding where to segment connected characters) had to be avoided by design, i.e., the user would be required to segment his characters, either by making them with a single continuous movement without lifting the pencil, or else drawing the character in such a way that individual line segments crossed or touched line segments that had been drawn at worst no more than a few segments previously.

The approach taken was partly empirical and partly theoretical. From the basic input data (namely, a succession of line segments described by their end points and direction information) it should have been possible to reproduce characters which still looked the same as the original characters in context with the same person's particular font. Initially, this font was chosen to include block-printed representations of the entire alphabet and arabic numerals. These tests showed that there was no loss in intelligibility involved in approximating the block characters by derived, straight line approximations, unless the character was less than ten times the device resolution.

The same was largely true of cursive characters. Further work was also done with representation of characters by approximating line segments quantized in angle with a set of discrete lengths that were picked on an integer $\log_{\sqrt{2}}$ scale. This representation allowed studies of independent scale changes, rotations, and skewing. From this theoretical work, for example, it was found that most curved segments enclosing areas could be replaced by just horizontal and vertical segments without loss of identity, that independent X and Y scale changes and skewing of the slant of a character towards the vertical did not affect its appearance.

A particular writer could inadvertantly make any of these transformations in drawing a character without realizing it, and thus, a recognition scheme to be effective would have to take them into account.

To supplement the theoretical work, it was also necessary to develop programs to handle the recognition, and for this purpose, a more restricted font, namely block characters, was used to guide the development.

Block printing is a form of constraint that fits well into the type of data that the graphical input board can provide, so long as lines are drawn straight, and oriented nearly horizontal, vertical, or on a plus or minus 45 slant; curves are drawn full, and intersections are made at ends or middles of line segments, with right angle changes. Curved lines of the symbol can easily be discovered from the sequence of angular changes in the line segments making up the curved segment, with a moderate number of line segments of this sort, it is possible to construct an enormous vocabulary of symbols, of which those of the ordinary alphabet constitutes only a very small part. The problem, even in this extreme case is, however, to set up a fast separation scheme. To be fast, a separation scheme should ideally split up the font into a moderate number of equally populated subgroups at several classification levels from simple tests until the unknown is categorized. With a known character set, it is possible to minimize consideration of all the relationships between lines that might be present, to just those which are sufficient to make a separation among the subgroup.

In the case of block letters, such a classification can be made almost completely on the basis of the number of line segments of each type, using just a few relationships between lines to resolve the few cases that fall into the same categories. (The properties that have been mentioned for block letters have ignored the additional information that was present in the original line segment data, namely, the sense in which it was drawn, lengths, angular changes, etc.)

Recognition then procedes by recursive lookup, based upon the numerical results of a test to find the next test, etc., until the process either terminates in a successful result, an unknown, or an ambiguity. General programs have been written to test a variety of classification schemes, both to construct the tables appropriate to a given person's character set, as well as to utilize the tables for a fast search. In this approach, the first split is extremely important, and is generally chosen to give a large number of stable subgroups.

These programs have worked with upwards of 95 per cent correct recognition for carefully-drawn block letters. The computer time spent for such recognition has been in general on the order of ten's of milliseconds. As the characters have become more carelessly drawn, however, recognition results have deteriorated, in the sense of not being able to identify a larger proportion of unknowns, which is hardly surprising, since the original classification schemes for block characters are based on rather idealized assumptions for what constitutes a line, and a curve. To include block characters as part of the larger framework encompassing cursive letters, as well as to be far more forgiving with respect to block letters, we have had to consider more carefully the process by which letters are made, and the possible normal range of variations that are taken for granted, and are perhaps undetected by the person doing the writing.

As block letters become less constrained, the distinction between lines and curves becomes blurred; the placement of line segments becomes more haphazard as does their length, lines may slant more randomly, and curves encompass a far wider range of interpretation. The situation is worse in the case of cursive writing, or mixed cursive and printing, since a cursive letter can be considered as all one curve. As the work of Eden* and others would indicate, however, much of the variability in cursive letters comes about as a result of the nature of hand motions, and thus a recognition scheme could be based upon <u>how</u> a character is made, in terms of horizontal and vertical motions taken separately rather than upon what it looks like, since the latter can hardly be regarded as constant. Furthermore, because of the manner in which the character is scanned, (i.e., as a sequence of nearly vertical, horizontal, and slanted, directed line segments) the slant of an unknown character can be determined and skewed back to a vertical orientation for lookup purposes. A recognition scheme based on these principles to more broadly cover the probable range of variation has been checked by hand, and is now being incorporated into search and classification programs, but no test results are yet available.

III. <u>Graphical Languages</u>

Graphical languages are generally more of a convenience to the human operator, rather than an advantage for the machine. In addition to the problem of converting from a graphical description into a machine translatable form, there is the additional problem of converting the descriptions into efficient algorithms for machine processing, hopefully for useful purposes. Two investigations were undertaken in the areas of boundary value field problems and electrical network problems. In the former case, the investigation was concerned with transforming a graphical dimensioned description of a set of boundaries into a boundary condition, a partitioning of computer memory, and a set of difference equations to operate on the boundary and interior. The programming was done in the MAD language and has been partially tested.

The network analysis work concentrated on finding an efficient and memory conserving algorithm for complex matrix inversion, where some elements are partially symbolic. The methods, which seem to have met their objectives, are currently being programmed.

*Murray Eden and Morris Halle. The Characterization of Cursive Hand Writing, Fourth London Symposium on Information Theory.

IV. Allocation and Scheduling of Peripheral Equipment

Further consideration has been given to the usage of a limited number of peripheral facilities, such as magnetic tape and disc on a limited number of channels for the off-line output and temporary storage requirements of operating time-shared programs. Operating in real time will not eliminate the need for some "nice-to-have" outputs, such as program listings, punched outputs, and memory dumps, or volume numeric data. Since this information will essentially be generated interspersed in time with that of concurrent programs, to conserve tape facilities, such outputs will be placed in the same tapes for periodic demerging. No such simple solution is available in the case of so-called scratch tapes, which will have to be assigned to users over the duration of their need. The scratch tapes necessary for compilers can, of course, be returned when the compiler is finished with them. Programming to handle these functions is being written and tested. It would presently appear, however, that the availability of such tape units places an inherent constraint on the number of on-line users that can be accommodated on any single system.

V. Compiler Revision

The internal revisions to the MAD program for real-time use and improved debugging capability are nearly completed and tested. The changes have been primarily in the organization of internal tables and lookup procedures, as well as splitting the program into time-segmented portions.