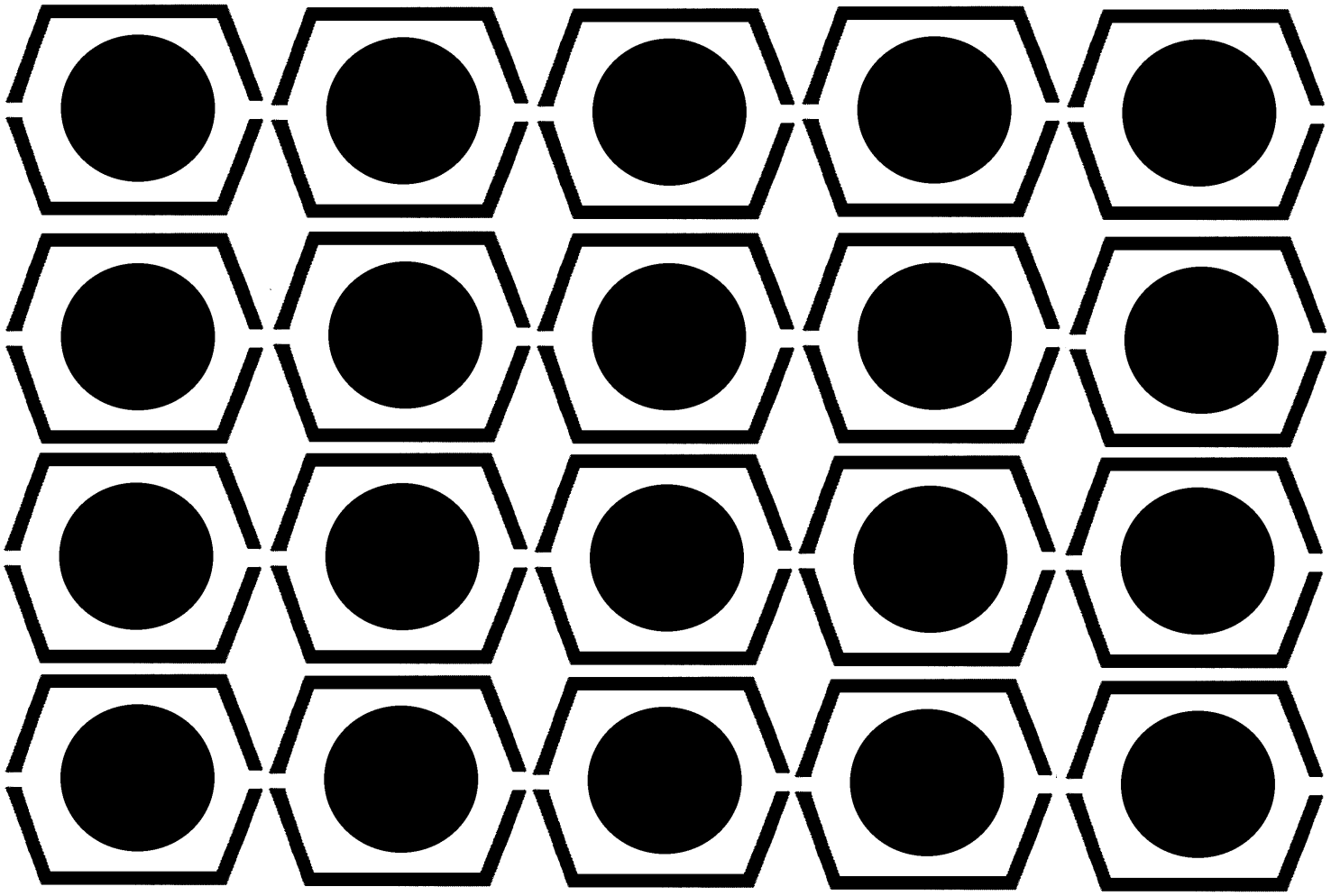


DECUS PROCEEDINGS

SPRING 1967

PAPERS AND PRESENTATIONS

of

Digital Equipment Computer Users' Society

Maynard, Massachusetts

SPRING
1967

PAPERS AND PRESENTATIONS

of

Digital Equipment Computer Users' Society

SPRING SYMPOSIUM

DISPLAY APPLICATIONS

APRIL 14, 15, 1967

Rutgers–The State University

New Brunswick, New Jersey

CONTENTS

* Abstract Only

PREFACE

The 1967 Spring Symposium was held on the campus of Rutgers - The State University, New Brunswick, New Jersey. The conference attracted approximately 200 participants from 25 states and 3 countries, other than the United States.

The success of any symposium, however, cannot be measured by the physical facilities available or the number of persons attending. It is directly and solely dependent upon the quality of the papers presented and the interest evoked by those papers. That the Spring Symposium was a success is a tribute to the participating authors whose papers are included in this "Proceedings." These papers are reproduced as received from the authors with no editorial changes. Reprints may be obtained from the DECUS office, Maynard, Massachusetts.

Special mention should be made of Applied Data Research Corporation, Applied Logic Corporation, and the Rutgers' Bubble Chamber Group, Physics Department. Each of these organizations made available their facilities for inspection and visitation by interested groups.

As the Meetings Chairman I would like to extend, on behalf of all the participants, our thanks to Mrs. Angela J. Cossette, DECUS Executive Secretary, and her staff for all of their efforts. Special thanks is also extended to Mr. Frederick A. Skove of the Department of Electrical Engineering, Rutgers, who was largely responsible for the physical arrangements.

Donald A. Molony
Meetings Chairman

SOFTWARE SUPPORT FOR A PDP-4/340 DISPLAY CONFIGURATION

H. Quintin Foster, Jr.
Department of Defense

Abstract

This paper will discuss a library structure on a PDP4/modified 340 display configuration. The library is of two parts .. an absolute library which consists of service routines for the user and a relocatable library which consists of subroutines for the PDP4 and the 340. The power of MIDAS, an assembly language, will show how the relocatable library was made possible. The method of editing both sections of the library will be discussed.

(In the oral presentation of this paper a movie demonstrates the method of retrieving programs from both sections of the library and the use of some of the service routines.)

Introduction

When the PDP4/340 configuration described in this paper was conceived it was obvious that we should expect little software support from DEC for our "off-brand" equipment. Since our configuration had no teletype, none of the debugging aids supplied by DEC would be of value to us. The modifications to the 340 also meant that no routines written by DEC to use a standard 340 display would run on our configuration. We were faced with the problem of supplying the average user with some help in programming this display system without becoming involved in each user's application.

Debugging routines would have to be written, subroutines to assist the user in building display files would be needed, computational subroutines would be necessary (most of these would be available from DEC), and some hardware interface routines would be needed to assist the user in his I/O problems.

Once these routines were written and debugged, the problem then became how to make these routines most accessible to the user. This paper will discuss one solution to this problem and in so doing will discuss some of the programs mentioned in the preceeding paragraph.

Hardware Configuration

The configuration of the computer that will be discussed is an 8K PDP-4 with paper tape reader, DECtapes, a modified 340 display with console, and a RAND tablet.

The 340 has a special subroutining option which adds three display instructions to the parameter mode instruction. The first of these instructions is a display jump instruction, the second a display subroutine jump instruction and the third a deposit save register instruction. This third instruction gives the capability of multiple level subroutines. Several other features have been added to the display but that is not the subject of this paper.[1] The main point is that it is not a DEC standard product.

Library Structure

A two part library residing on DECtape has been designed for the SCHOOLER* system. The first section of the library is a collection of service routines most of which, when in memory, reside between locations

*SCHOOLER is nothing more than a short name used to reference the hardware described above.

15000_8 and 17700_8. The routines can overlay each other by use of LIBCOR. The second section of the library is a collection of relocatable PDP-4 and display subroutines which can be called in and linked to user supplied programs by means of MRLL4. The structure of the two sections are different and independent of each other.

The structure of the absolute library is basically a collection of core images residing on DECtape, each program starting on a block boundry. (See Figure 1) The name of the routine and the DECtape block of the next program are the first three words of the first block of each program. The date of the routine is contained in the next two words. The structure of the core image is a first word address (FWA), and a last word address (LWA), followed by a LWA-FWA+1 words to be loaded starting at the address specified by FWA. If a program is to be loaded in several sections of core, then a first word address and last word address will be specified for each section. The end of the program is signaled by encountering a jump instruction rather than a first word address. The jump instruction is a jump to the start of the program.

The block pointed to by the last program will contain -0 as the first word to signal an end of file for the absolute library. The third and fourth words will specify the first and last block of the relocatable library. (This information was once used by the relocatable loader but is no longer needed.)

Now that the basic structure of the absolute library has been defined, the operation of LIBCOR, the program to load routines from the absolute library, will be discussed.

LIBCOR is a program which resides on DECtape blocks one through four as a core image. A small routine residing at 17700_8 through 17731_8 will bootstrap block one into locations 17363_8 through 17762_8. The first block is a bootstrap loader which pulls in the rest of LIBCOR from blocks two through four. When LIBCOR is loaded, a user may then type (using the special console) the name of any routine on the absolute library followed by a carriage return. LIBCOR will then start searching the tape starting at block 5 to find the named routine. The first three characters of the name are compared against the first word of block 5, the last three against the second word. If both words do not match, the name is stored in a buffer area (to be used later if the requested routine is not found) and the third word is picked up to get the block of the next program on the library. LIBCOR then searches forward to that block and compares the first two words. This procedure continues until the first two words match or the absolute library's end of file is reached.

If the names match, the two date words are skipped, the FWA and LWA are obtained, the number of words in the first section are calculated and loading starts at the address specified in FWA. When the number of words in that section are exhausted a new FWA or a jump instruction is obtained. If a new FWA is obtained the next section is loaded in a manner exactly like the first section. When a jump instruction is obtained, loading is completed and the jump instruction is executed after stopping and deselecting the library tape unit.

This entire operation is done in real time with respect to the DECtapes, therefore, there is no rocking motion of the DECtapes.

If the named routine is not found, then the names of all routines encountered in searching the tape are picked up from the buffer area and displayed on the 340. This incidentally is just one of the many reasons why a display is more advantageous in a debugging environment. The entire list of library routines can be displayed in a matter of milliseconds and the user does not have to wait a long period of time to see what the last routine on the library is, as would be the case with a teletype or typewriter.

Block five of the absolute library does not contain a routine even though it is structured as a routine. It is used as a pointer to the relocatable library. This eliminates the necessity for the relocatable loader to sequentially search down the tape for the absolute end of file.

The structure of the relocatable library (See Figure 2) is almost an exact copy of the relocatable paper tape produced by MIDAS (discussed later). The first word of each DECtape block is the negative of the number of words (obtained from a paper tape from MIDAS) in that block. Except for the last block each block contains 255_{10} words. At the end of the relocatable subroutines is a relocatable jump block which flags the end of the relocatable library.

Relocatable Input to SCHOOLER

All initial object input to SCHOOLER is via either paper tape produced by MIDAS running on a CDC 1604-A or the relocatable library residing on DECtape. The history of MIDAS is vague since no generally available documentation exists.* Our version of MIDAS originally ran on the PDP-1. Modifications were made to assemble for the PDP-4. Due to the inaccessibility of the PDP-1 to most SCHOOLER users, MIDAS was written to run on the 1604 (which is job shop operation).

MIDAS will produce paper tape in absolute format which can be loaded by the RIM loader or by a MIDAS absolute loader; or it can produce tape in relocatable format which must be loaded by the MIDAS relocating linking loader (MRLL4). The latter tape format is the only one which will be discussed in this paper.

When MIDAS is in relocatable mode it will produce four types of blocks on the paper tape output -- absolute, relocatable, library, and start. (All four blocks are of the form, first word address (FWA), FWA plus the number of data words in the block, data words, and checksum (See Figure 2). Thus the first two words specify the number of words in the block and, in the case of absolute or relocatable blocks, where the data is to be loaded.) The types of data words in the blocks are of the following forms -- absolute storage word, storage word to have relocation added or subtracted, and symbolic information. The symbolic information specifies entry points (absolute and relocatable) and external references to be added to or subtracted from the storage word.

A main program is composed of absolute and/or relocatable blocks and terminated with a start block. The absolute and relocatable blocks contain the storage words, definition of entry points and external references for the program. The start block contains a starting location as an external reference or as an absolute or relocatable location. However, the main purpose of the start block is to indicate to the loader that this is the end of the routine. User subroutines are of the same format as the main routine.

A library tape (whether actually on paper tape or on DECtape) has a library block followed by the absolute and/or relocatable blocks containing the program. There is no start block at the end of each library routine. Many library routines can be placed on the tape each headed by a library block. The library block contains, in symbolic form, the names of the entries in the program to follow, or the names of programs, which, when loaded, will reference entries in the program. A start block is placed after the last library routine to indicate the end of the tape.

LOADING RELOCATABLE PROGRAMS

With this brief background of the format of relocatable tape, the linking loader will be discussed. The linking loader is another program of unknown origin (to this author). Our introduction to the linking loader was via the PDP-1 and was probably originated at the same time and by the same group as MIDAS. The linking loader was translated for the PDP-4 using the PDP-1 version as a model.

The linking loader (MRLL4) reads paper tape images from DECtape and determines the type of block encountered. If the block is not a library or start block, then MRLL4 will load absolute blocks at the location specified by FWA and will load the relocatable blocks at a location defined by FWA plus a relocation factor specified by the user. As entry points are encountered, the absolute values of the entry points are saved in the loader table. As external references are encountered, the loader searches its table for the definition of the external and adds (subtracts) the value of the external to (from) the storage word presently being formed.

If a library block is encountered, the loader searches its tables for an unsatisfied request for any name in the library block. If one is found, the program following the library block is loaded and linked. If a request is not found, the program is passed and the next library block is obtained. Once a library block is encountered, MRLL4 searches the library tape until all requests have been satisfied or a start block is encountered.

---

*MIDAS was written by R. Saunders for the TX-0 at M.I.T. It was translated for the PDP-1 by D. Gross, A. Kotok, W. Mann, and R. Saunders.

Once the loading of a relocatable program and all its subroutines has been accomplished, the user might be interested in a map of his loaded program. Several routines are available to display the loader table as well as to modify it.

RLLMAP allows a user to see which external requests have not been fulfilled and at what locations the requests have been made. It also gives the address of each entry point that was encountered.

MAPNEW and MAPSRT both display the loader table but also allow modification to the table. Due to the method of displaying the loader table and the amount of information put on the display screen, these two routines have a very low refresh rate giving very bad flicker. However, for photographic purposes they are excellent.

MAPSRT gives the user the capability to sort the loader table into numerical order. It also allows the user to remove entries from the loader table thus reducing the amount of material on the screen and the flicker.

MAPNEW allows a user to correct errors in the loader table. For example if a user had mispelled the name of a routine in his source deck, he could use MAPNEW to change the name and rerun MRLL4 or give the name a value if the routine had already been loaded. This eliminates the necessity of reassembling a program just to correct a careless mistake. Another function of MAPNEW gives the user the capability of adding a new request into the loader table.

In general, MAPNEW and MAPSRT give the user means for intervention in the loading process. Thus, he is able to structure his program in special ways for whatever purpose he may desire.

Debugging Aids

After a user has loaded his program correctly his interests turn to saving the core image and debugging the program. Three programs to assist in this goal will be discussed -- DDTS, DIFDAR, and LIBRAY.

The first program, DDTS, should be familiar to all users of DEC computers. Therefore, only aspects of DDTS which are unique to SCHOOLER will be discussed. DDTS has nearly all of the capabilities of DEC supplied DDT. Its major differences are tnat it communicates via the SCHOOLER console and the display and does not have mnemonic output. DDTS can display twenty locations on the screen at any one time. A list is kept of the twenty (or less) locations that are being displayed. Rather than building a long display file which requires a considerable amount of memory, DDTS computes from the list the address and contents of that address to display on the screen and then displays them once. The full list is processed repetitively in this manner. This gives DDTS a "dynamic" appearance. When proceeding through breakpoints some specified number of times, the screen is refreshed once for each time through DDTS. Thus, if appropriate locations have been opened, the user can observe the progress of his program.

DIFDAR serves two purposes. It gives the capability of dumping onto and retrieving from DECtape in much the same manner as DEC supplied DECtape routines. It also gives the user the ability to get a derivative dump. A derivative dump compares a specified area of core with specified blocks on DECtape and displays only those words in core which do not match corresponding words on DECtape. This has proved to be most useful in tracking down very subtle bugs.

LIBRAY is a routine designed to manage DECtape for a user. It can be used both as a service routine and as an overlay handler. As a service routine, programs or files are referenced by name via the console. LIBRAY keeps a directory of all files and manages the block allocation for each file. The user may retrieve, add, or delete any file by name. In adding a file, the location of the data in memory must be supplied and a starting address to use upon retrieval may be given. This routine allows the user to construct his own personal library and/or data files.

As an overlay handler, LIBRAY has several subroutine entry points. The user can call LIBRAY from his main program to accomplish any function that can be performed as a service routine from button control. For example, he may ask LIBRAY to load any file, whether data or program overlay, and can allow LIBRAY to transfer control to the starting address declared upon filing; or he can have control returned to his main program by the normal subroutine return. In the case of data, the user can over-ride the declared target memory location and have the file reloaded in any position.

Library Editing

Editing the library tape is performed by two different routines. ALIBEDIT will perform an edit on the absolute portion of the library and copy the relocatable portion. RLIBEDIT will perform an edit on the relocatable portion and copy the absolute portion.

Both programs allow old routines to be deleted or replaced or new routines to be added. All "bookkeeping" is done by the routines. This makes updating the library a rather trivial task. Both edits force the editor to supply a date with any routine which is being added or changed. This eliminates many questions of "which version" of a routine is on the library.

ALIBEDIT will add a new routine onto the absolute library tape from the absolute paper tape of the routine or from a core dump of the routine on DECtape.

To assist in debugging and editing the absolute library routines, the linking loader, DDTS, and a version of the DECtape dump and retrieve program were assembled to load in core at locations 5000_8, 200_8, and 10000_8 respectively. (These programs will be referred to as system aids.) This allows all three programs to be in core simultaneously while one of the absolute library routines is being debugged or setup for editing. This has become a most useful method of operating, especially when one wants to write a routine to be incorporated into the absolute library but, in writing it, wants to make use of the relocatable library. This routine may be assembled as a relocatable program calling subroutines from the relocatable library tape and then debugged in core making full use of the absolute library routines. When the program is considered debugged, the system aids can be called into core, the relocatable version of the new absolute library program loaded into core starting at location 15000_8 and then dumped onto DECtape for the editor.

Contents of the Relocatable Library

For the most part the relocatable library can be considered to be broken down into four parts -- I/O subroutines, computational subroutines, display subroutines, and "display file building" subroutines.

I/O Subroutines

The two major I/O subroutines are the EXECUTIVE and MICROW. MICROW is a subroutine to read or write DECtape with the user specifying the unit number, the block, first word address, and last word address. The EXECUTIVE is the interrupt handler and interface to the console. It will process all interrupts and notify the user that the interrupt has occured if the user is interested in that particular interrupt. Otherwise, from the user's point of view, the interrupt is just cleared and ignored. The EXECUTIVE also reads all buttons (which are nothing more than switches) and knobs on the console and will notify the user when a button has been pressed or released if the user is interested in that button. Since the knobs produce pulses every 1.4 degrees of rotation and these pulses are used to drive a five bit up-down counter, the EXECUTIVE takes the five bit counters and keeps an accumulative count modulo some number specified by the user. For the user who doesn't want the full capabilities of the EXECUTIVE, there are subroutines on the library to handle only single parts of the console. The user may then use only those subroutines which interface with the parts of the console in which he is interested. Subroutines to start, stop, and save the state of the display are also available.

Computational Subroutines

The DEC standard multiply and divide packages are available to the user as well as some subroutines to perform shifting of the AC (ignoring the link). Matrix manipulations such as matrix multiplication and matrix movement are also available to the user.

Display Subroutines

The display subroutines are almost entirely character sets and their associated jump tables. Several different fonts and sizes of characters are available to the user. Special characters which might be of value to the user are included on the library. The philosophy of having the characters as subroutines means that only one copy of the code to actually draw each character is in core and display subroutine jumps to the appropriate character are performed each time one wishes to place a character on the screen.

"Display File Building" Subroutines

There are two major types of subroutines to assist the user in building his display files at execution time. The first allows the user to place different types of drawing instructions into a display file. The user may specify X-Y information to be placed in his file; he may specify delta X and delta Y and the appropriate vector instructions will be placed in his file; or he may pass a binary number to a subroutine which will place calls to characters to display the number as an octal or decimal integer.

The second group of subroutines allows the user to double buffer his display files. For example, if the computation of a display file takes a long time relative to the refresh rate of the file being displayed then it would be advantageous to display one file while the new file is being built to reduce the flicker rate to an acceptable point.

Conclusions

Although the software support for this display device has been sufficient to allow skilled users to make good use of the system, it has required that the user understand the hardware to a degree that should not be necessary. Just as FORTRAN and ALGOL (and other higher-level languages) have opened the field of programming and the use of computers to a wide range of people, a higher level graphical and console interaction language is needed to open the field of graphics to these same people. It should be realized that there are really two mechanisms involved: one is the language in which processes are defined; the other is the language in which processes are operated. With the software support provided for this system, suitable operational languages (for the second mechanism) can be created by a skilled programmer. However, at this installation, higher level process construction languages (for the first mechanism) have not been produced. Some work has been done in this field at other places, but very little has been published. It appears that these higher level languages are still quite a way in the future.

Author's Note: The work described in this paper was done by several people and is not solely the work of this author.

Reference

1. Ball, et al, "A Shared Memory Computer Display System", IEEE Transactions on Electronic Computers, Vol. EC-15, pp. 750-756, 1966.

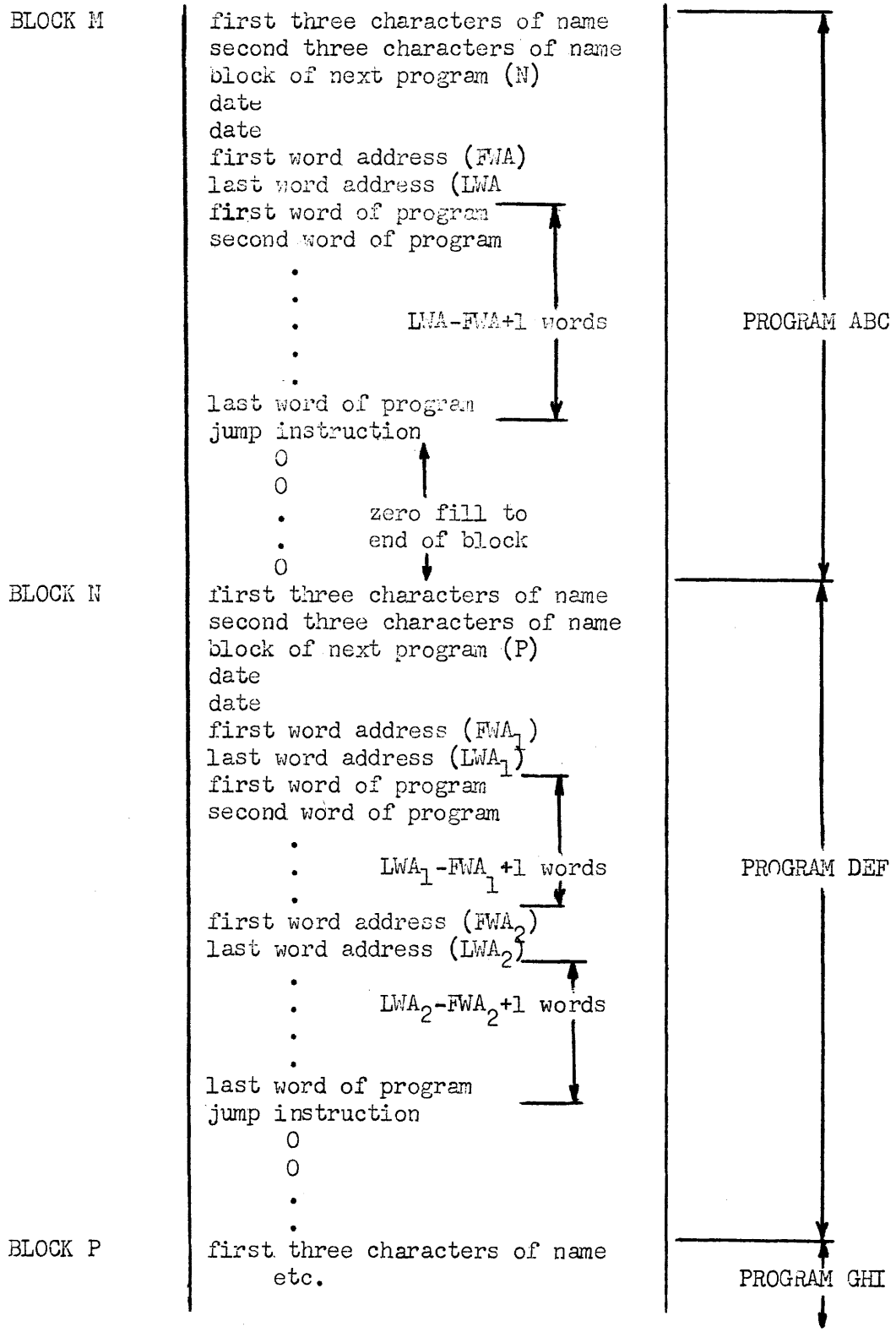


Figure 1 Structure of the Absolute Library

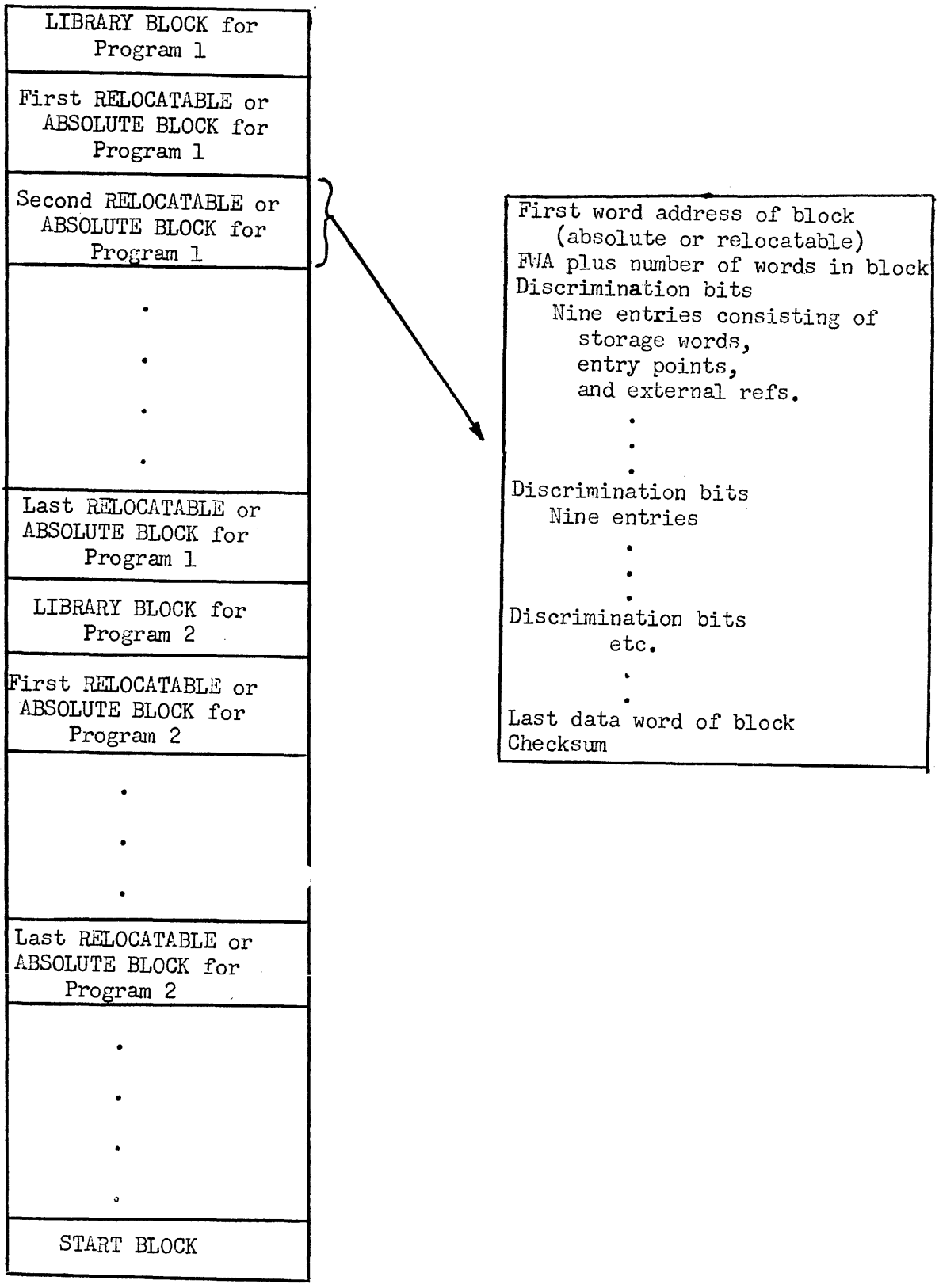


Figure 2 Structure of Relocatable Library

DISPLAYS FOR STUDYING SIGNAL DETECTION AND PATTERN RECOGNITION

T. Booth, R. Glorioso, R. Levy, J. Walter, and
H. Kaufman
University of Connecticut
Storrs, Connecticut

The use of a unique computer-controlled (PDP-5) CRT display system with light pen facility in the study of a wide range of signal detection and pattern recognition problems is described.

The displays used to study the capabilities of the human operator to detect signals embedded in noise are described and illustrated. In addition to illustrating displays to study the effects of various signal and display parameters, pre-processing and real-time, operator-directed (via light pen) processing are also shown.

The use of this system for tachistoscopic stimulus presentations to study basic human information processing capacities is also described.

I. Introduction

The Electrical Engineering and Psychology Departments at the University of Connecticut are engaged in an interdisciplinary research project to study how man processes visual information. A series of experiments are currently underway which involve presenting the human operator, in real time, various classes of visual displays and measuring his ability to process the information contained in them. This information may or may not be contaminated with background noise.

Because of the large amount of information which must be handled during each experiment, it is impossible to employ classical psychophysical display and measurement techniques to present these displays and collect data about the operator's reactions. This problem has been overcome by using a PDP-5, a high speed CRT display, and several special purpose I/O devices as a central data processing and control system. This paper will describe the various types of experiments which are currently being conducted and illustrate the advantages gained by using this automated display and data gathering system.

II. Basic Research Problem

The basic problem under investigation concerns man's ability to use and process various types of visual information. At present two specific types of problems are being investigated.

The first area concerns the ability of an operator to detect and/or recognize a pattern in a noise background. In this area a series of experiments are being conducted in an attempt to develop a model which describes the operator's detection and dynamic information processing capabilities. Figure 1 illustrates a typical display an operator might see during a given experiment. His task would be to decide which column in the display has the largest number of intensified points. Once he makes this decision, he is asked to identify the line by use of a light pen.

The second area of interest involves measuring the amount of information that an operator can extract and utilize from a noise-free display. In a typical experiment, the operator would be shown a series of patterns selected from a given class of patterns. After seeing each pattern he identifies the pattern that he thinks he saw by pushing the appropriate button on the operating console in front of him. Figure 2 shows one typical pattern that might be presented to the operator. By varying the number of component dimensions of the pattern, it is possible to obtain an idea of how much information the operator extracts when making his decision.

Traditionally, research of this type has been conducted by using a tachistoscope which is a small display device for presenting visual stimuli under controlled conditions of visual field, illumination, and exposure. An experimenter must be present during the whole experimental session to control the operation of the tachistoscope and to record the subject's responses. In addition, all of the displays must be fixed a priori,

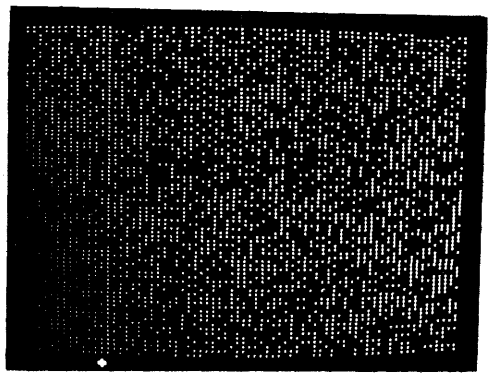


Figure 1
Signal Detection Display

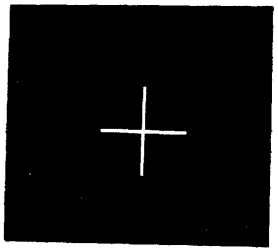


Figure 2
Information Transmission Display

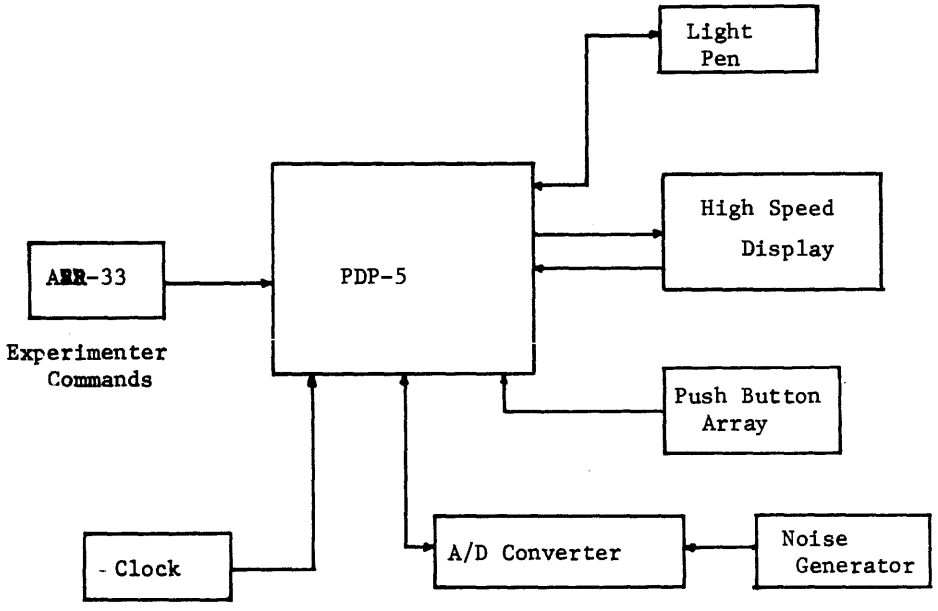


Figure 3
Block Diagram of Display System

since it is impossible to present dynamic displays to the operator. Such an experimental process suffers from several very severe limitations. Most notably, when dealing with problems of the type described above, it is not uncommon to require over one thousand different displays in order to gather statistically significant data.

The first major problem involves the mechanical process of generating, storing, and presenting a large number of displays when a tachistoscope is used. If a thousand different displays are used, the experimenter must generate each display and record it on a card. During an experimental session he must load each card into the projector and control the presentation of the card, which takes approximately 20 seconds. Therefore, a minimum of 6 hours would be required to run an experiment involving a thousand displays. For much of this time, the subject is waiting for the next display to appear; consequently, he quickly loses interest in performing his assigned task.

The physical task of generating a large number of display cards is also a major problem. It is possible to use a digital computer to generate the data required to specify each of the patterns needed for a given experiment. However, it is still necessary to transfer these computer generated data onto the display cards used by the tachistoscope. This is a tedious and time consuming task.

Even when the mechanical problems associated with preparing the experiment have been solved, the problem of collecting and processing the experimental data must be considered. In the types of problems under investigation, it is necessary to record parameters such as the subject's identifying response, the time he took to respond, the position of the pattern or patterns he responded to, and the statistical properties of the display itself. To collect this information using standard techniques can become a monumental, if not impossible, task. In some experiments, the experimenter must also have access to the display in order to make measurements on the display itself to determine which pattern the subject indicated. When this occurs, it not only disturbs the subject, but also increases the amount of time needed to perform a given series of trials.

Because of the above difficulties, many interesting problems concerning the way in which a human processes visual information have been difficult to investigate. In fact, without the use of the on-line system described in the next section, it would have been impossible to carry out some of the experiments which will be described in sections IV, V, and VI.

III. Computer Control System Used to Supervise Experiments

To overcome the experimental problems described in the last section, the computer controlled system illustrated in Figure 3 was developed. The heart of this system is a PDP-5 computer coupled with a special-purpose high-speed CRT display. This display system, which is described in detail elsewhere, can present flicker-free 72 x 85 dot patterns of the type shown in Figure 1 and Figure 2. These patterns, which are under direct program control at all times, can easily be modified to satisfy any desired experimental conditions.

For a given experiment, an experimenter can use any or all of the following I/O devices in conjunction with the CRT display system.

a) Light pen - The light pen is automatically tracked and controlled by logic built into the display system. Whenever needed, the position of the light pen can be read by the main program.

b) Noise generator - A wide-band Gaussian noise generator is available which can be sampled by the A/D converter under program control to produce a Gaussian distributed statistically independent variable.

c) Clock - An external program controllable clock can be used to measure total elapsed time and/or incremental changes in time.

d) Push button matrix - A 2 x 4 array of pushbuttons is available for the subject to indicate his responses to a given set of experimental conditions or to initiate a change in the display under light pen control.

To perform a particular experiment, all that an experimenter must do is write a program which will present the proper sequence of displays to a subject and measure the appropriate responses. During a given session the collected data may be stored in the computer until the end of the experiment. At that time the data is processed and the results are typed out on the teletype. The flexibility of this system is mainly limited by the skill of the

experimenter in designing his program and identifying the data he wishes to measure. Once the program has been developed, the actual running of the subject is handled almost entirely by the computer. Thus, once the experimenter has started the program, he is free to carry out other work while he waits for the end of the session.

In order to illustrate how this system is used, several of the experimental programs which have been run by this system will be described in the following sections. No attempt will be made to explain the reason for the experiments since this information is available in the papers cited in the references. The main emphasis will be upon showing how the experiment was programmed and illustrating the savings in time which were possible.

IV. Signal Detection Experiments

These experiments have been concerned with the ability of an operator to detect a target signal which is embedded in a noise background. Figure 4 illustrates a typical display shown to a subject in a given class of experiments. Under normal operating conditions the target path will not be as strong as shown in this Figure.

Two basic modes of operation have been considered. The first is the so-called "alerted operator mode" where the subject is told the location of a possible target and his task is to detect targets at that location. The second is the "unalerted operator mode" where the target location is not known and the operator's task is to locate a target, if present. Although these programs are similar, each has its own special requirements. They are alike, however, in the manner in which displays are generated.

To generate displays of the type shown in Figure 4, the system samples the external noise generator through the A/D converter. If the sampled value exceeds a predefined threshold value T_o, a one is stored in the display, corresponding to a dot in the display matrix at the appropriate location; otherwise, a zero, corresponding to no dot, is stored in the display matrix. The number of intensified points appearing in any part of the display matrix can be controlled by varying the value of T_o. To see how this is accomplished, it is necesary to consider the properties of the random number produced by the A/D conversion process.

Since the A/D converter operates from 0 volts to - 10 volts, the mean of the noise generator is set at -5 volts and the standard deviation is set at 2 volts. The output of the noise generator is Gaussian distributed and its spectral density is flat from d.c. to 100kcps. Under this condition the probability that the noise voltage will exceed the dynamic range of the A/D converter is .012. In order to insure that the noise voltage remains constant during the conversion time, a track and hold circuit is used between the noise generator and the A/D converter.

If a point is to be generated with a probability of intensification of 0.5, a threshold value of $T_o = 4000_8$ which corresponds to -5 volts is used. If a point intensification probability greater than 0.5 is needed, a constant C_o is added to 4000_8, or subtracted for a probability less than 0.5. For each trial during a given experimental session, a display is generated point by point and stored in core locations 6000_8 to 7247_8. A target path is produced by changing the point intensification probability for the points in the desired target path by adding the appropriate constant to the clipping level. Generation of the 6000 point display takes 1.8 sec.

The alerted operator experiments are usually conducted with straight line targets where the location of the target is indicated by an arrow as shown in Figure 4. The operator indicates the presence or absence of a signal by means of the push button matrix. After the operator's response, feedback may be presented as shown in Figure 5. An H corresponds to hit or correct response and an M corresponds to miss or incorrect response. A typical alerted operator experiment runs for 15 to 20 minutes during which 150 displays are viewed by the subject. The trial by trial data, as well as a compilation of the data after the run is complete, are usually printed and/or punched by the ASR-33. Previous alerted operator experiments using non-automatic techniques ran one hour and only 20 to 40 displays were viewed.

The display for an unalerted operator experiment is generated in essentially the same manner as was used for the alerted operator experiments except that the position of the target path is not indicated. Because of this, it is necessary to use the light pen as well as the push button matrix to communicate with the computer. A typical

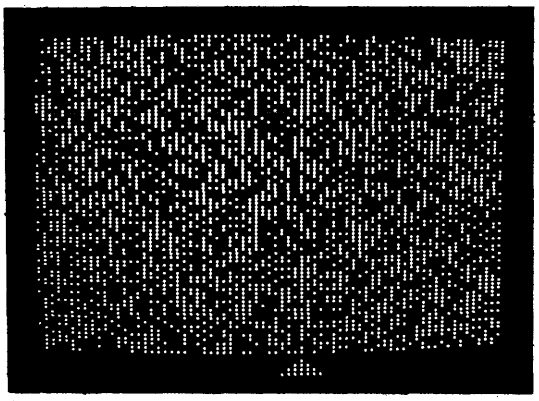


Figure 4
Alerted Operator Display

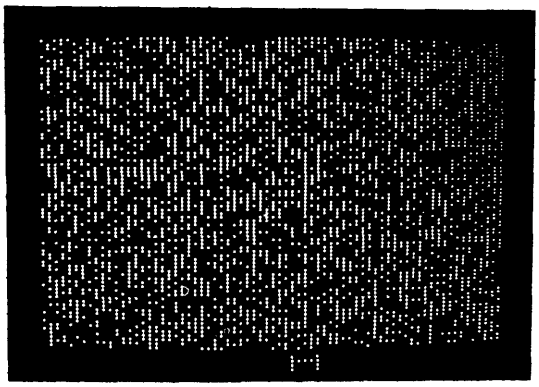


Figure 5
Alerted Operator Display With Feedback

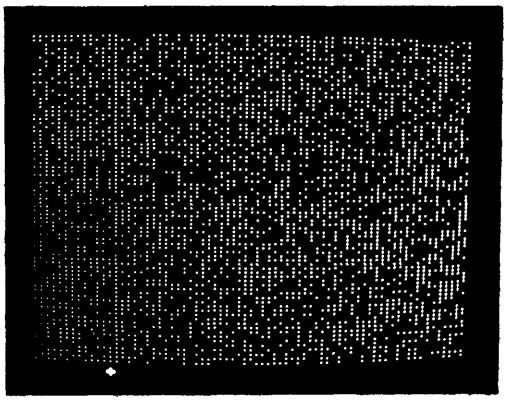


Figure 6
Unalerted Operator Display with
Light Pen Tracking

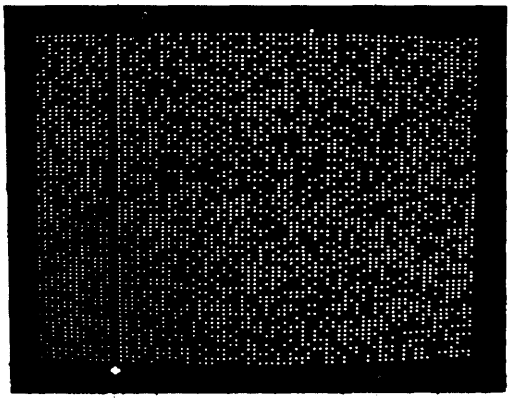


Figure 7
Column Elimination

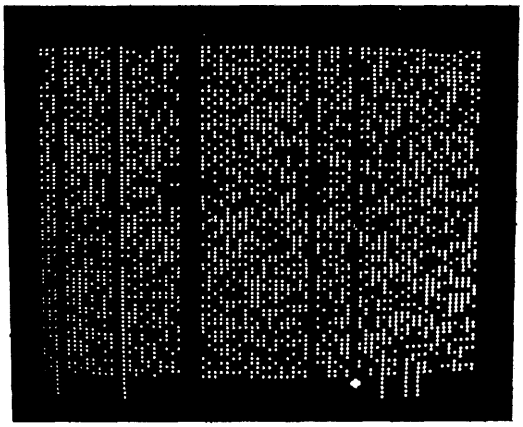


Figure 8
Column Check

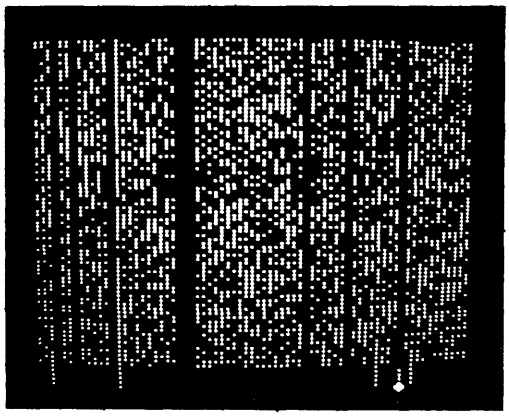


Figure 9
Columns Compared

experiment, which gives the operator control over computer processing options will be illustrated.

The subject is presented a display such as the one shown in Figure 6 and he is asked to indicate which column in the display is the target column. The dot cluster at the bottom of the display is the light pen tracking square which the operator can use to carry out one of the following program-display options:

1) Any column can be eliminated by placing the light pen tracking square under it and pushing the appropriate button.

2) Any column can be checked against two preset thresholds by using the light pen and pushbuttons: If the column strength is below the first threshold, the column is eliminated; if it is between the first threshold and the second threshold, it is left unchanged; and if it is above the second threshold, a marker is placed under the column.

3) Any two columns can be compared by placing the tracking square under one column and pressing the appropriate button and then placing the square under the other column and pressing the button again. The weaker column is then eliminated.

Figures 7 through 9 indicate the results of applying these program-display options to the display illustrated in Figure 6. Figure 7 shows the same display after the column above the tracking square has been eliminated. The columns with marker lines under them in Figure 8 have been checked and are above the second threshold. Figure 9 illustrates the effect of comparing the column above the square and the strong column on the left of the display.

The unalerted operator detection task can be generalized by asking the subject to detect targets with curved paths. A curved path is shown without background noise in Figure 10, and with background noise in Figure 11. In this type of an experiment, the subject indicates the path of a possible target by tracing it with the light pen. If the operator goes off the path or indicates an incorrect path, all points in the correct target path flash on and off and a line appears at the bottom of the display. If the operator indicates the correct path, all points in the target path flash after the tracking square is below the last row of the display.

Most unalerted operator experiments require trial by trial printing and/or punching as well as run compilation printouts. Usually 50 trials are run in one hour in these experiments. This kind of experiment is almost impossible to perform without a computer-display system.

Experiments which use these displays are described in references 2, 3, and 4.

V. Dynamic Displays

The detection problems discussed in section V concerned the ability of a subject to detect a target path in two dimensions. Such displays are common in radar and sonar systems. However, in systems of this type it is usually necessary to process three dimensional information consisting of two spatial dimensions, say X and Y, and time. This section will discuss a display capable of presenting information of this type to an operator. These displays are being used to investigate the ability of a subject to detect targets in a dynamic display situation.

The dynamic display presentation scheme uses a two dimensional display and time to present the three dimensions in such a way as to take advantage of the human eye-brain system's spatial and temporal integration capabilities.

Figure 12 is a representation of the three dimensions. At any given point in time, current X and Y information is assumed to be available from a device such as a sonar or radar receiver. This information can readily be shown on the two dimensional display tube; in this case, a signal from a target would not appear as a column, as it does in a two dimensional display, but as a cluster of cells in the matrix. As in the digitized displays shown in section IV, the difference between the target area and the background is that the former is generated by sampling from a distribution with a higher mean than that which produces the latter. Thus, the matrix containing a target will have a clustered subset of its cells within which the probability of a dot appearing is greater than in the rest of the matrix.

This current information could be put on the 2-dimensional display at time T=t and remain unchanged for one time interval (i. e., until T = t + 1) at which time it could be replaced with the new information.

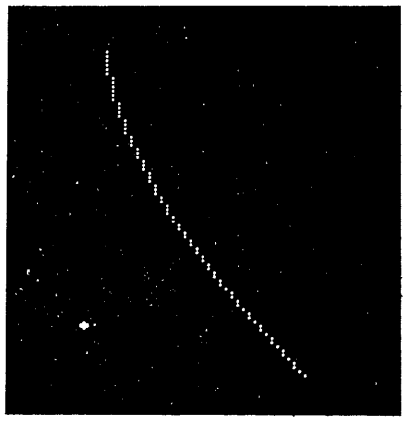


Figure 10
Curved Target Without Noise

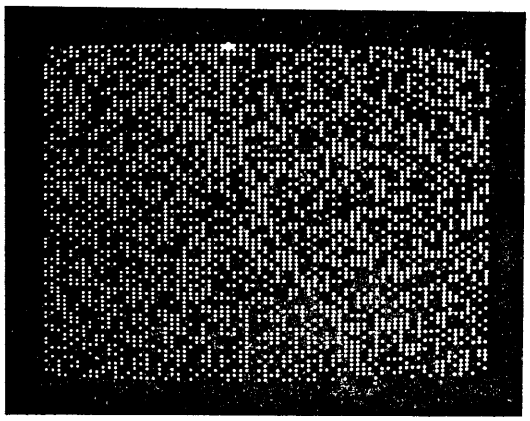


Figure 11
Curved Target With Noise

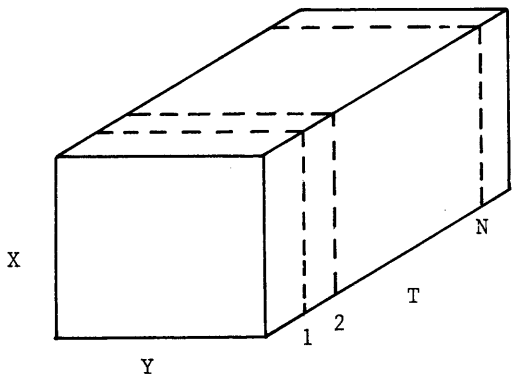


Figure 12
Representation of Three Dimensional Display

However, at time T = t_B, all the information from some previous time, t_A, up to the present can be stored in the computer memory. The number of time intervals between t_A and t_B, of course, depends on the number of points along the X and Y dimensions and the size of the memory. Having this information available makes the following possible: in the interval between t_B and $t_B+\Delta t$, all the successive X, Y displays from t_A to t_B can be shown in rapid sequence, rather than having a static display of the information from t_B alone. Similarly, in the interval $(t_B+\Delta t)$ to $(t_B+2\Delta t)$, the information from $(t_A+\Delta t)$ to $(t_B+\Delta t)$ is shown sequentially, etc. The last frame in each sequence is shown a little longer than the others while the frames are shifted up in the memory in preparation for the next sequence and the information from the present time is brought in. The effect is that of a moving window scanning the T dimension for some distance, jumping back to near its initial position, and moving on again, etc.

The human observer looking at this dynamic X, Y display sees a matrix of randomly flickering dots, but in the target cluster there are more dots; hence, they appear to flicker less frequently. In addition, the movement of the target cluster through the matrix, which would be hard to see in sequential static displays is made to stand out as smoother movement, as in movie film.

A program has been written for the PDP-5 which presents displays in this manner. Input parameters are: for the display: (t_B-t_A), Δt, frame rate; for the signal: matrix size, target cluster configuration, path, P (dot/noise), P (dot/Target).

This program requires about 350 core locations, leaving about 3000 for storage of display information. Thus, 250 12 x 12 matrices can be stored or 27 36 x 36 or only 5 of the maximum size 72 x 85. Because of these memory limitations, an expanded core or magnetic tape unit would make the program more versatile.

In operation, the sequence of events is as follows: First, the program generates as many frames (X, Y displays) as will be shown in one sequence and stores them. Next it transfers the first frame to the display area of core, hence onto the screen. Then it enters a wait loop whose length is determined by reading the switch register. After the wait loop, it transfers the next frame onto the display, etc. Thus, the frame rate is determined by the length of the wait loop and the amount of time required to transfer one frame; hence, the maximum frame rate decreases as the size of the frame increases.

After displaying all the frames in a sequence, all the frames in core are shifted up, as many frames as specified, and the new frames for the end of the next sequence are generated and stored. The program then jumps back and shows this new sequence.

A movie has been made with some examples of 12 x 12 and 36 x 36 frames shown in this manner. Figure 13 is extracted from this movie to illustrate a dynamic display when a very strong target is present, and Figure 14 shows a typical path which the cluster may follow in time.

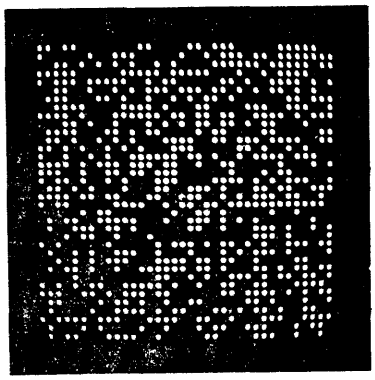


Figure 13
One Frame of a Dynamic Display

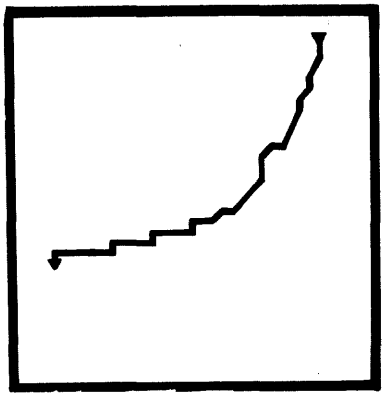


Figure 14
Typical Target Path as a Function of Time

VI. Pattern Recognition and Information Transmission Experiments

The use of the display system described in section III for studies of human pattern recognition and information transmission capabilities and characteristics has lead to great experimental flexibility and ease of data collection. Two research programs are currently underway in this area.

The first program is concerned with a problem related to pattern recognition, namely how do stimulus dimensions, such as size, hue, linear extent, and brightness, influence a subject's ability to identify a particular pattern from a set of possible patterns. Although, the operator can judge as alike or different 1000 or more stimulus values sampled from along one of these dimensions, he can only identify the equivalent of approximately 7 values. One only need witness the speed and accuracy with which the operator identifies the letters of the alphabet and other complex patterns to see that the number of dimensions which make up a pattern are an integral part of the operator efficiency at pattern recognition.

A requirement for studying how dimensions combine is ease of manipulation and generation of dimensions relevant to the particular research problems. In previous research on dimensionality [5], the dimensions of horizontal and vertical linear extent were used. Information transmission was compared for stimulus values along the single dimensions alone and for stimulus sets formed by combining values along both dimensions. Stimuli, which were prepared with black tape (approximately 2" x 3/32") on 8 1/2" x 11" white poster board cards, were presented with a Gerbrands tachistoscope. Because of manual presentation and interchanging of stimulus cards about 45 minutes were required to run 125 trials. From the subject's viewpoint, relatively little time is spent in his assigned task of identifying stimuli compared to the large amount of time idled away while the experimenter carries out his duties.

The CRT offers a means of displaying stimuli varying along these same dimensions of horizontal and vertical linear extent. The dots used to generate the patterns with other signal detection studies can be transformed so as to give the appearance of a solid line. When a single column and/or row is filled with intensified points, the gain on the X and Y axis controls turned all the way down (which gives a density of approximately 15 dots/cm.) and the focus and astigmatism adjusted appropriately, stable solid lines can be presented. The length of line is manipulated by the number of points displayed. In the present experiments, lines consisting of 33, 65, 67 and 69 dots are used. A third variable dimension, intensity, was obtained by putting 4 preset intensity levels under I/O transfer control. The levels of intensity are preset in the sense that the experimenter sets each level to any desired value within the dynamic range of the CRT. These levels are then available under program control.

Stimulus patterns are presented for short exposures (approximately .19 sec.) which leads to inherent problems. With short exposures, display duration must be calibrated according to full sweeps so that the display dies out uniformly. The exposure of .19 sec. corresponds to 7 full sweeps. Besides the flicker-free characteristics of the display system, the high frame rate allows relatively sensitive control and manipulation of the exposure times. Although a medium persistence phosphor is in use, it is essential to eliminate any persistence, since it leads to uninvited cues for stimulus identification during the task. An obvious solution to this problem is to have a high level of ambient light. A requirement for the ambient light, however, is that no reflections be cast off the front of the CRT. A reasonable level of ambient light is also desirable since it is perceptually unsound to have the subjects visual field switch, continuously, from light to dark as displays go off and on. Furthermore, control of the area of the subject's visual field would be advantageous since it would also prevent the use of extraneous cues in carrying out the assigned task. The requirements just outlined have been met by a 19" wide x 15" high x 22" long adapter that is easily hung over the face of the CRT.

A photograph of a subject's eye view of the face of the CRT through the adapter is shown in Figure 15. Non-reflecting lighting of the face of the CRT is achieved by a string of 7 watt lights placed around the front frame of the adapter. It would not be at all unreasonable to say that the CRT has been transformed to a super tachistoscope with all the advantages of on-line computer control.

The actual preparation of stimuli is under program control. On a trial by trial basis the appropriate horizontal, vertical, and intensity components of stimulus are either set up in the display area of core storage for horizontal and vertical dimensions or the I/O transfer pulse sent out. The operator pushes a button which exposes the stimulus and then pushes an appropriate button recording his response. Stimulus and response data are pre-processed and stored over a series of approximately 120 trials. These same 120 trials, including printout time, take approximately 10 minutes to run compared to the 45 minutes required to run 125 trials using the Gerbrands tachistoscope. Furthermore, changing stimulus conditions is extremely easy since the only input to the control program besides number of trials (the sequences of randomizations are prepared independently on each run by using sampling techniques with the noise generator) is a specification for each of the stimuli and the value along each of the component dimensions.

A second program is investigating the amount of information that a subject can retain and recreate after he has viewed various classes of displays. In 1956, George Miller [6] described means of remembering information in which the human takes a number of symbols to be remembered, groups them, and remembers the names of the groups. He then decodes the groups to the individual components when asked to recall them. Miller referred to this as "chunking". All of us use this procedure when we remember a 12-bit binary number by storing its octal equivalent. Miller's implication was that human memory capacity can be multiplied tremendously by taking larger and larger groups or chunks. However, all the examples he cited involved sequential presentation of the symbols and the time required to do the coding fits in between symbol presentations. Simultaneous presentation, however, is a different story. In this instance, Hyman and Kaufman [7] have shown that memory capacity is defined in terms of amount of information stored, not number of units, as Miller implied. Also, it has been shown that humans tend to adjust input and output rates so that the bit/sec. rate of throughput is relatively constant. An experimental program is currently underway which combines this type of research with simultaneous presentation by varying the information loading of the symbols used, the length of time they are presented, and whether or not chunking is used.

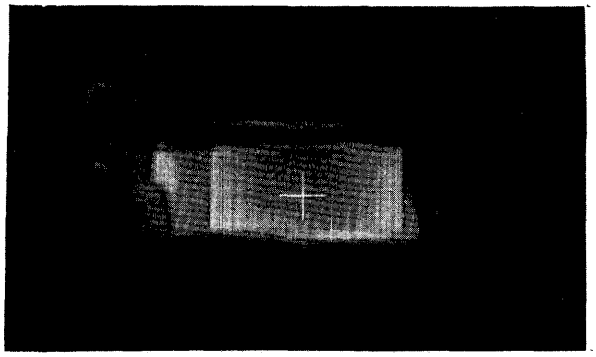


Figure 15
Subject's Eye View of CRT

The display used is a matrix of 64 symbols randomly chosen from 4 different ones, as illustrated in Figure 16. It contains 8 rows of 8 symbols each, each symbol being 13 dots arranged in the cells of a small 5 x 5 matrix. These have been shown to be readily discriminable. The pattern of dots in each row of each different symbol is stored in a number of locations in the core, these locations differing only in the rotation or position in the register that the bits occupy. This is done because each column in the 8 x 8 matrix corresponds to a different part of the word being displayed. That is, the first column corresponds to the left end of the first word in the row, the second column corresponds to the right end of the same word plus the left end of the second, etc. In generating the matrix, the program samples the noise generator twice to get a random number between 0_8 and 3_8 corresponding to one of the symbols. From this number and the position in the row into which the symbol will be put, the program generates the address of the location where the particular symbol is stored in the correct rotation. Having thus generated the 8 addresses required for the first row, it then transfers the symbols to the display area of the core, repeating this 8 times to get the full matrix.

When the operator is presented with this display, he tries to recall as many of the symbols as possible and pushes appropriate buttons in the response push button matrix. Each button generates a code between 0_8 and 3_8, so the program keeps track of whether each response is right or wrong and when the operator pushes a fifth button to indicate

he can respond no more, the program punches out the number of responses he made, the number that were correct, the number correct after correcting for guessing and the time required to make the responses. The codes for the responses and 64 symbols on that trial are also available. This program takes about 600 core locations and generates these display matrices as fast as the operator can push a button to request a new one.

The use of this system for this type of experiment has tremendous advantages over the use of a manual tachistoscope as was done in refs. 5 and 7. Once the program has been written, the actual symbols used can be determined or changed in a matter of minutes rather than the hours required to prepare cards with appropriate symbols as required by the tachistoscope. Also, the running of the subject is automated with a hard copy of the data available at the session's end. This frees the experimenter to do other work during the session, as opposed to continuously manipulating cards for the tachistoscope and writing down responses, as well as eliminating the need for later punching of data onto cards.

Acknowledgement

Acknowledgement is due the Office of Naval Research which supported this research through a prime contract NOnr 2512(00) with General Dynamics/Electric Boat as part of the SUBIC (Submarine Integrated Control) program.

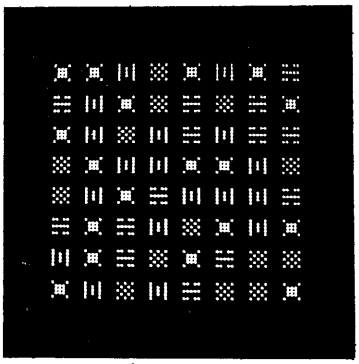


Figure 16
Short Term Memory Display

References

1. Brazeal, E. H. and Booth, T. L. 1965. "A High Speed Man-Computer Communication System." Spring. DECUS Symposium, May 1965. Cambridge, Mass.

2. Brazeal, E. H. and Booth, T. L. "Operator Noise in a Discrete Signal Detection Task." IEEE. Trans. on Human Factors in Electronics, Vol. HFE-7, No. 4. Dec. 1966, pp. 164-173.

3. Levy, R. M., Kaufman, H. M. and Walter, J. R. "Experimental Investigations of Operator Decision Processes: II. The Effects of S/N and Bearing Resolution" Presented Eastern Psychological Association Convention. New York, April 1966.

4. Glorioso, R. M. and Levy, R. M. "The Effects of Step Transitions in Signal to Noise Ratio and Feedback on Dynamic Operator Decision Behavior." Presented Eastern Psychological Association Convention. Boston, April 1967.

5. Levy, R. M. "The Effects of Stimulus Dimensionality on Information Processing" In Proceedings of the 73rd Annual Convention of the American Psychological Association, Washington, D. C. APA, 1965. pp. 42-43.

6. Miller, G. A. "The Magical Number Seven, Plus or Minus Two," Psychological review. Vol. 63. 1956. pp. 81-97.

7. Hyman, L. M. and Kaufman, H. M. "Information and the Memory Span." Perception and Psychiphysics, Vol. 1, 1966. pp. 235-237.

A STATUS REPORT ON SOME APPLICATIONS OF PROCESSOR CONTROLLED COLOR DISPLAYS IN SIGNAL ANALYSIS - 1957 TO 1967

Charlton M. Walter
Air Force Cambridge Research Laboratories
Bedford, Massachusetts

Abstract

On-line, processor controlled color displays have now been in use, on an experimental basis, for more than a decade. Experience with two systems, which were used primarily for work in the area of statistical signal analysis and on-line sensor data processing, is outlined. The potential of color in aiding the user to visualize the behavior of pattern recognition schemes and to obtain insight into complex sensor data processing operations is illustrated. Particular emphasis is placed on mistakes made, and lessons learned, both in hardware design and in modes of control required for the effective use of color.

1. Introduction

The use of on-line, processor controlled, color oscilloscopes has now covered more than a decade at the Air Force Cambridge Research Laboratories. Yet the general use of color, in on-line systems, still appears to be several years away. An examination of the causes of this rather slow evolution (at least in the context of general computer development) of a data monitoring capability which will eventually be almost universal, appears to be in order.

The relatively high cost of most displays, coupled with the substantial demand which they place on processor time, has been, of course, the key inhibiting factor to their general use. But part of this high cost has been caused by completely unrealistic requirements on beam positioning accuracy and deflection linearity. There are wide classes of problems in which the processor controlled scope is to be used strictly as a visual process monitoring device. And for most of these applications, absolute accuracy and linearity is of minor value, compared with the need for presenting large amounts of information, in context. Here also, the absolute intensity of points, lines and other graphic structures is not as important as the ability to produce a diverse spectrum of illumination, in order to provide the greatest potential for information transfer between the user and the processor.

In this survey I shall largely concentrate on applications of on-line color to problems in the area of signal analysis, sensor data processing, and to the investigation of pattern recognition and data compression schemes. These are all areas which require the portrayal of events and interrelationships which are inherently statistical and which involve the manipulation of large amounts of low precision data.

We shall only touch briefly on the use of on-line color in such important areas as tracing the flow of information through various kinds of networks and diagrams. For example, flow charts, PERT diagrams, city planning, architectual and machine drawings, chemical plant and utility networks, and so on; even though these applications may eventually account for some of the most extensive applications of processor controlled color. Also to be completely ignored are the potentially interesting applications of on-line color as a dynamic art medium, involving abstract symbolic painting, and the computer-aided translation of music and other art forms into visual form for artistic purposes.

2. Brief Description of the On-Line Color Systems

The first on-line, digitally controlled color display to be utilized in signal analysis investigations at AFCRL was originally procured from RCA under a contract begun in the early 1950's, to develop a digital radar color display for a prototype Base Air Defense System. Both the display, and a special digital computer, employing 600 KHz solid state magnetic amplifier circuitry, which was built as part of the prototype Air Defense System, became available for general research in 1955 and were impressed into use on several signal analysis studies by 1957.

Although 3 bits were allocated for color control in the display, the lack of convergence in the early three beam color tube resulted in the implementation of a purely on-off capability for red, green and blue. The deflection circuitry used 10 bit x-y

registers, with 6 bit Δx-Δy registers, in a configuration which had real promise, but which was rendered ineffective by the very slow speed (18 μsec word time) of the serial-by-decimal-word processor, having a 2000 word drum main memory, which had to carry out all of the arithmetic, logical control and display operations.

In the early 1960's DEC undertook the engineering of a new color display, which was delivered to AFCRL as part of an experimental multi-processor system, based on two interconnected PDP-1 central processing units. The original DEC color scope operated in a point plotting mode, with 9 bit x-y deflection registers and a plot option in the display instruction which allowed either a 50 μsec or a 200 μsec beam settling time. The longer delay time was required to achieve corner-to-corner deflection. Color was controlled by a 12 bit register with 4 bits (or 16 levels) used for each of the three primary colors. Very satisfactory color mixing was achieved, through effective beam convergence, in this system.

This display system was later modified by the E. W. Pughe Development Corporation to also operate in an incremental mode, with a basic increment instruction time of 1.66 μsec per 6 bit display instruction. The incremental display processor uses a drum buffer which is shared by the dual PDP-1 CPU's of the DX-1 system, and which also buffers a black/white oscilloscope system. The entire display system requires no central processor time when maintaining static pictures on the color scope, on dual b/w monitor scopes, and on the precision recording scope of the DX-1 System.

3. Color and Problems Associated with Its On-Line Control

Color provides essentially two other dimensions, hue and saturation, in addition to the intensity of illumination provided by the customary black and white oscilloscope. Both systems at AFCRL used an additive color scope system, based on the red, green and blue "primary" colors, of the current standard colored TV system. Digital control of the color, however, provides a level of hue and saturation consistency which does not exist in the domain of analog color TV.

The specification of intensity of each of the above primary colors provided the actual physical control in the DEC-built, on-line system operating at AFCRL. This rectangular coordinate system is, of course, highly unsatisfactory for the direct control of color by the human operator. A more psychologically satisfactory system is derived from the "color wheel" concept, and is based on a cylinderical coordinate system. Hue may be mapped onto polar coordinate angle, making use of the closure property of the perception of the color spectrum: red, orange, yellow, green, blue, indigo, violet, and back to red. If this is done in such a manner that colors on each side of the color wheel are complementary, in the sense that their mixture yields white, then we may define a mapping of "saturation" or "purity" on the radius vector from the origin, such that white is at the origin, and the appropriate pure spectrum is attained on the unit circle. If this is done in such a way as to achieve roughly equal intensity for all points on a particular wheel, then a series of color wheels can be defined, one for each level of intensity, thus defining a mapping of intensity onto the z-axis of the cylinderical coordinate system. Since saturation is not independent of intensity, and high purity cannot be perceived at low intensities, the coordinate system is actually more nearly conical than cylinderical.

Naturally, none of these mappings are unique, since the perception of color is a highly complex subject[1]. Thus the best approach to the desired effect seems to be through the use of appropriate manual controls, acting through a programming utility system having an on-line formula forming and interpreting capability, such as JOSS[2], for rapidly generating new mappings.

Looking toward the future, another useful degree of freedom in graphic control, not customarily considered in the usual black and white, line and curve generating display, is the controlled dispersion of the written points. This capability would provide a more efficient mechanism for the display of estimates of probability distributions and other fuzzy information needed for a more realistic picture of the true nature of raw sensor data and of the interpreted results obtained by filtering the data. While this feature might prove unnecessary on a very heavily buffered display, backed by an extremely powerful parallel processing system, its utility in cutting down on central processor time would be quite substantial, and is, in fact, the basis for a simple filtering system used in the PEPR system[3] for bubble chamber photograph analysis.

4. On-Line Applications of Color to Several Data Monitoring Problems Areas

The types of data process monitoring problems to which we have applied color have fallen into three broad, somewhat overlapping categories. The first category covers primarily statistical data processing operations carried out on large amounts of empirical data, obtained from sensors observing the real world environment.

Figure 1 (a black and white copy of several frames of a 16mm color film[4]) shows the use of color in a radar signal analysis problem utilizing the earlier RCA digitally controlled color scope. In the example shown, blue was used to exhibit the raw radar signal data at a given range, as the radar swept across a target; green was used to show the output of a detection filter under investigation, and red was used to exhibit the output of an azimuth position estimating filter. The beam convergence on this early RCA prototype shadow mask tube was too poor to obtain adequate color mixing, so we were constrainted to thethree primary phosphor colors: red, green and blue.

In spite of these deficiences, the most serious limitation of this sytem, for signal analysis purposes, was the very low speed of the serial digital processor with magnetic drum main memory.

Figure 2 illustrates the more modern (1961) DEC Color Console on the DX-1 System, and an experimental set-up used to input pressure cardiograph and other types of sensor data, in real time. Each source of data was assigned a color, and each signal was exhibited, both in its raw time series form at the top of the scope and as a single point in N-space (here N=25) projected on an arbitrary pair of "filter" axes. As large ensembles of signal vectors were collected and stored in computer memory, a statistical picture of each source was built-up as a fuzzy cluster of points in N-space. Each distribution being identified by its assigned color.

Various filters could then be generated, either analytically through a computer program, or input by "drawing" the filter structure on the scope face with a light pen. The stored signal data could then be refiltered and projected on various new sets of axes, as illustrated in Figure 3. Here we were working with a sample consisting of two sets of pressure cardiograph signals, H1 and H2 and two sets of spoken word signals, W1 and W2. The filters were orthonormal eigenfilters, based on the minimization of cross correlation between components of the original signal vectors.

Color serves to clearly distinguish the behavior of the different "clusters" of data, under various transformations, and help to provide insight into why some pattern categorization schemes tend to work so well on limited samples of data, belonging to relatively few categories, and often fail to work for the large sample size, many category situations.

Figure 4 illustrates the use of color in the display of raw and processed multi-channel sensor data. The top raster is a typical speech spectrogram of a sentence (with time running from left to right, frequency channels plotted vertically and energy level at a given frequency and time shown as intensity). The middle raster represents a transformed and greatly compressed version of the same sentence, from which the bottom approximation spectrogram can be resynthesized. Color can be used in a variety of different ways to emphasize the contribution of different filter structures in the resynthesis process.

This type of operation requires the storage of large amounts of raw and processed data having no extensive logical structure, hence puts a premium on economical buffer storage, and poses no severe limitations on the degree of resolution required of such display parameters as position, intensity, hue and purity.

Another category of monitoring operation involves graphic data generated by purely analytical procedures, such as the potential field around two oppositely charged flat plates illustrated in Figure 5. Diminishing levels of positive potential were assigned appropriate intensities of orange, and negative potentials were assigned intensities of blue. The equipotential "contour" lines were an accidental product of the 4 bit, or 16 level, quantization imposed on the available intensities by the original color system design. In a black and white display this induced contour line effect is actually an aid in interpreting the data. In the color version it is more of a distraction.

The requirements of this type of display on both position, intensity, hue and purity resolution, and on processor capability, are much more severe than those required for the monitoring of statistical data. Defects in the analytical procedures and "bugs" in the programs used to generate this type of display stand out very clearly. Consequently

such displays are very effective aids in both program debugging and in checking the validity of the data generating procedures.

A third category of on-line monitoring and data processing operation treats data which is partly specified empircally and partly generated by analytical or logical procedures. Meterological data, such as that illustrated in Figure 6, is typical of this type of problem area. Part of the pressure-temperature-wind velocity data may represent actual observations, while other data points and contours are obtained through both interpolation and extrapolation based on models which range from the extremely naive to the very sophisticated. Once again, color is a highly effective aid in overlaying the many different kinds of data needed to obtain insight into the underlying processes, without creating the excessive confusion so aptly illustrated by the black and white printed version of the original color display.

In Figure 7 we have the stylized version of a type of reconnaissance problem in which various sources of ground radio transmission are being located by triangulation from the bearing angle indications of direction finding sensors located on an aircraft. Once again, color is very helpful in indicating both the source characteristics and the degree to which one characteristic can be discriminated from another, in the presence of uncertainty.

The use of color in tracing the flow of control, and of data, through various kinds of networks and "flow charts" is so obvious that no further comments will be made on the subject in this paper.

One further application of color which deserves mention is its potential in the representation of various types of significance associated with alphanumeric data, as illustrated in Figure 8. The digits were assigned colors based on statistical measures of their probable error. In this way, both high and low estimates of significance can be presented and forced upon the attention of the user.

5. Some Remarks on the Resolution of Position, Intensity and Hue

Extensive operating experience with the DEC processor controlled color scope on the DX-1 System at AFCRL has indicated that the original allocation of a 4 bit (16 level) intensity control word, for each of the three primary colors, was inadequate for really effective control of the hue-purity-intensity attributes on data in a context involving smoothly changing levels of any of the three variables. At least 6 bits per primary color (64 levels) appears to be more a reasonable degree of quantization. Even 8 bit quantization would not be unreasonable for potential field type investigations, involving very smooth gradients, if a digitally induced contour line effect is not wanted. Our inability to easily distinguish the absolute intensity, or hue, or some color, taken out of context, to within better than 16 levels, is no indication of our abilityto make relative judgments concerning these attributes in a highly structured context.

Similarly, the choice of 9 bit quantization in x and y position coordinates, giving 512x512 discrete positions on a tube surface having a grid of 256x256 color phosphor triplets in the actual area used, proved inadequate for the representation of smooth curves. The ability of the eye to detect offsets of 0.5 mm in lines 1 mm wide at a typical viewing distance of 1 meter proved rather distracting in certain contexts. The tube used was an RCA 21FKP22, shadow mask type, with spacing of approximately 0.75 mm between color phosphors triplets, having a basically analog deflection system in which more than one color dot of a given color is usually excited, producing an effective minimum spot size of about 1 mm. Even with a spot this large, the beam requires positioning to less than 0.25 mm, or 10 bit precision, in order to adequately reduce the line offset problem. However, absolute accuracy, to the full 10 bits, is quite unnecessary for most monitoring applications.

6. Remarks on Display Buffer Storage Media

Displays used primarily for on-line data monitoring are subjected to a variety of conflicting requirements. In order to minimize flicker, it is desirable to use an erasable storage capability, such as a storage tube, or to regenerate the picture some 30 times per second. For color, the storage tube techniques poses serious development problems, and will not be considered further here.

The optimum scope display buffer appears to be a judicious combination of random access core, and circulating (disk file or drum) storage, accessable from the main computer central processor unit and interfaced to the display with a special display processing unit which can be as simple or sophisticated as the display load demands. Circulating storage is generally much cheapter

than random access core storage, and is reasonably good for displays which remain static while under examination. For the display of data in motion, or of data having a deep logical structure, which is to be manipulated on-line, rapid random access to the underlying data structures is essential, and the display processor itself should have a reasonable sophisticated micro-processing capability, if the central processor is to be relieved of many functions not requiring its full power.

7. Conclusions

The potential of color in wide classes of of on-line data monitoring operations is sufficiently great to offset the present rather serious limitations in resolution of the existing standard color tubes. In digitally controlled display systems which do not require ultra-high precision, the principle cost has been in the digital storage registers and decoding logic and not in the analog part of the display system. Under such conditions the additional logic required to control three primary colors, rather than just one "color", is only a small fraction of the total logic needed for positioning and moving the point of light under appropriate computer control.

Consequently, one cannot afford to not make increasing use of color, at least for many data monitoring functions. The absence of cheap means for obtaining standard size "hard" copy, in color, is still a limitation to the extensive use of color in scientific and engineering reports and articles, but rapid advances in color reproduction technology should quickly resolve this problem.

References

1. H. Yilmas and L. C. Clapp, "Perception", International Sciences and Technology, Nov 1963.

2. J. C. Shaw, "JOSS: Examples of the Use of an Experimental On-Line Computing Service," Sixth Annual Symposium of the Prof. Group on Human Factors in Electronics, IEEE, May 1965.

3. D. Friesen, "A Description of the PEPR System," Proc of the Digital Equipment Computer Users Society, Maynard, Mass., May 1965.

4. This illustration is a black and white copy of several frames of 16 mm color film of the old RCA prototype color scope, all other illustrations in this article are black and white copies of 35 mm Ektachrome slides of the DEC-built oscilloscope of the DX-1 System at AFCRL. Most of the b-w illustrations show the effect of overexposure needed to reproduce lines and points which were in low intensity red on the original transparencies, and bear little relation to the excellence of the original images, when reproduced in color.

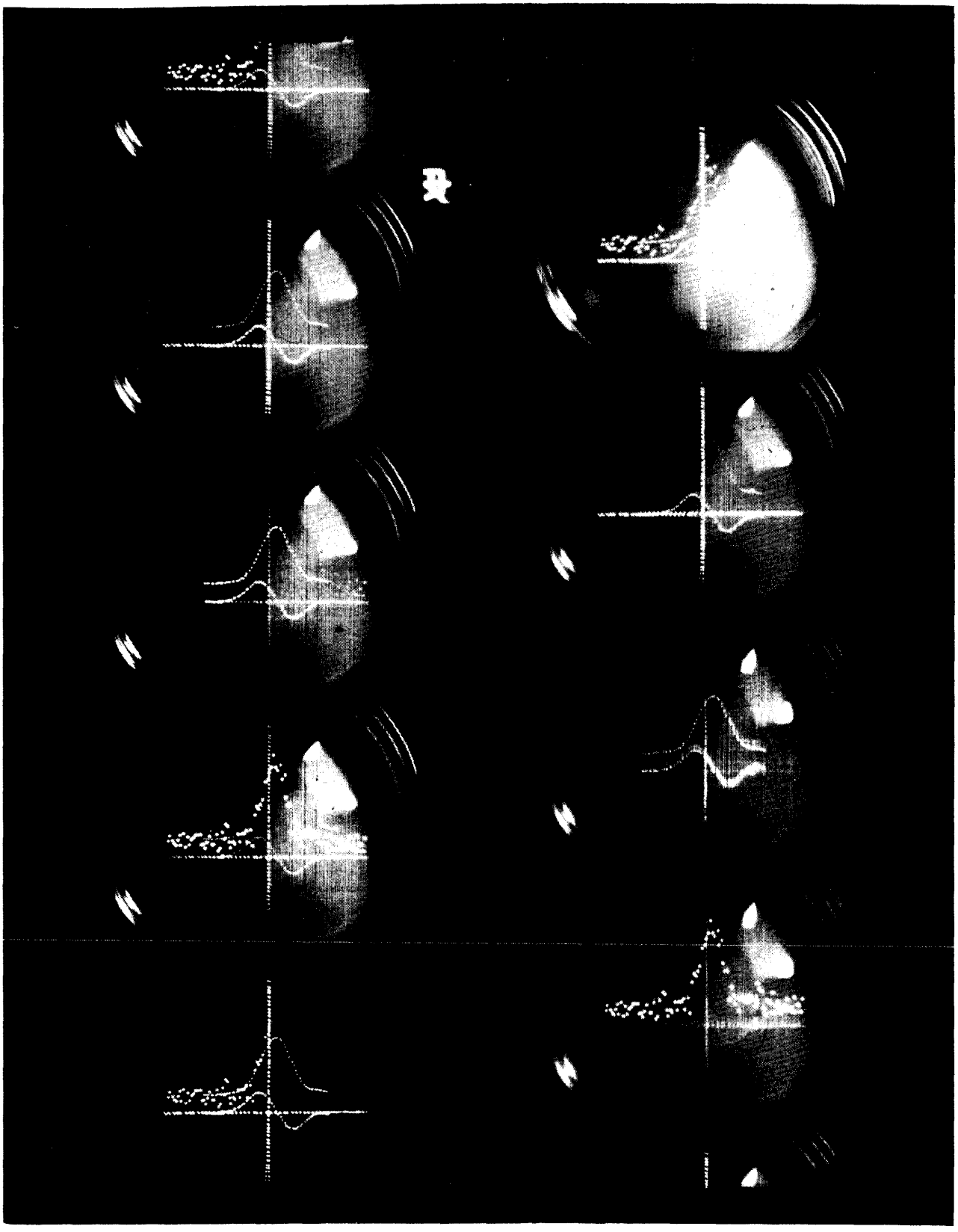


Figure 1. Radar Target Detection and Position Estimation Filters Operating on a Scintillating Signal in Noise Using Early RCA Prototype Color Scope.

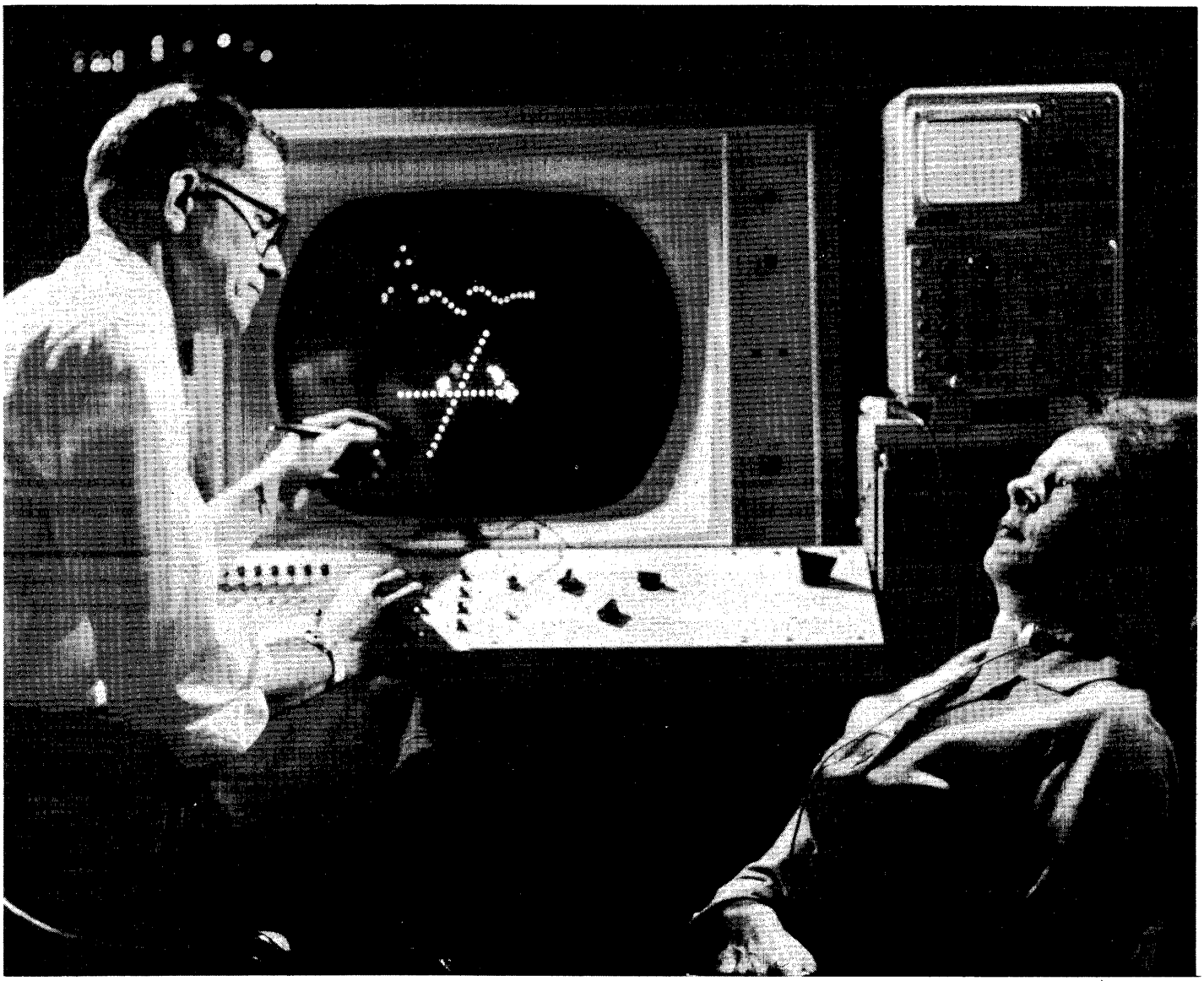
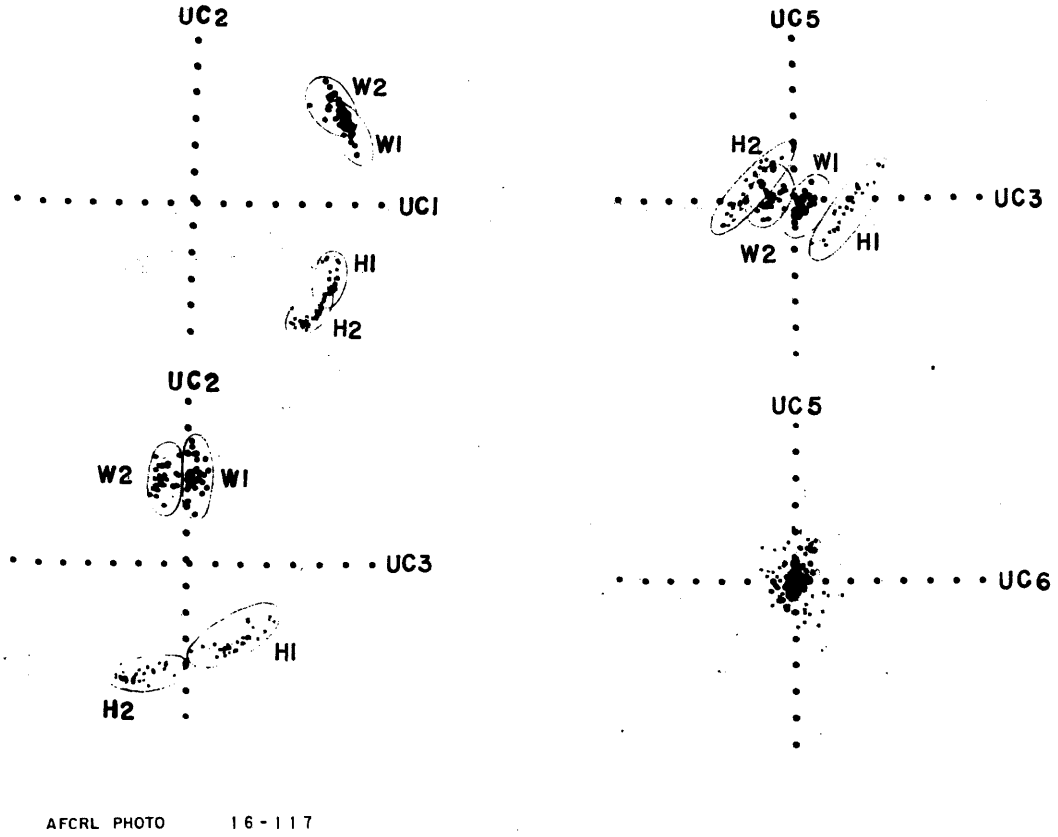


Figure 2. On-Line Inputting and Manipulation of Pressure Cardiograph Signals Using DEC Color Scope on DX-1 System.



AFCRL PHOTO 16-117

Figure 3. Use of Orthonormal Filters in Signal Categorization Investigation Involving Pressure Cardiograph and Spoken Word Data Clusters.

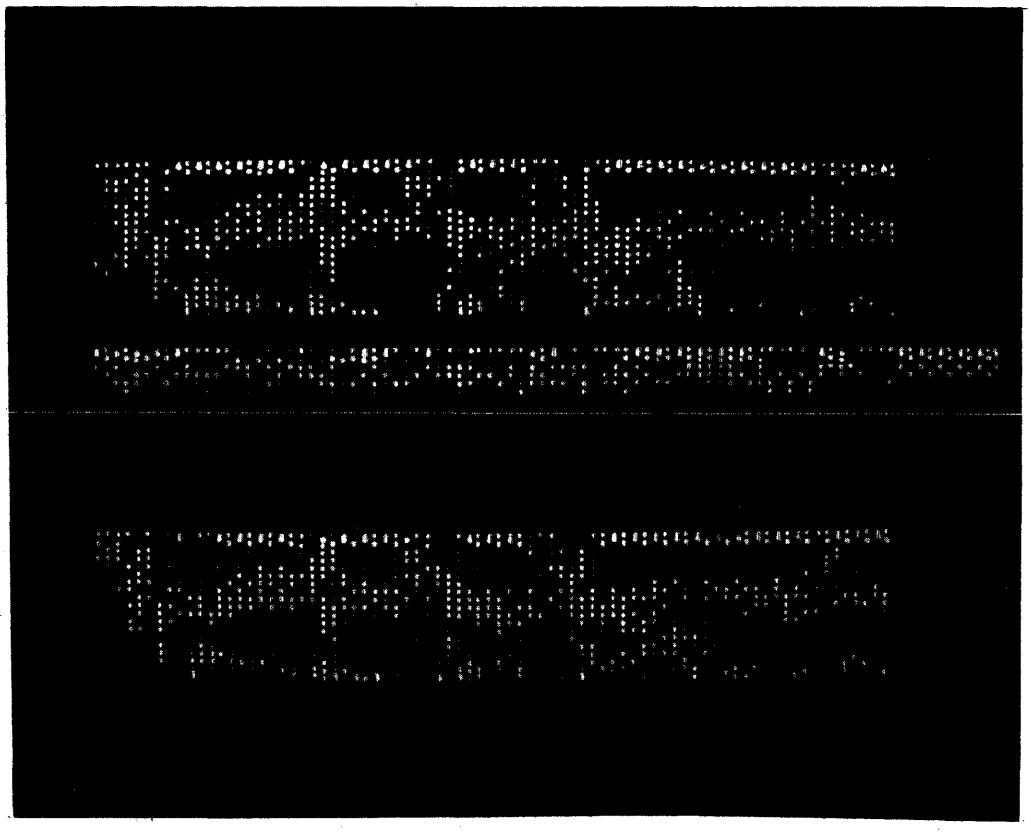


Figure 4. Illustration of Speech Bandwidth Compression Technique Showing Spectrogram of Sentence followed by Compressed Representation and the Resynthesized Spectrogram.

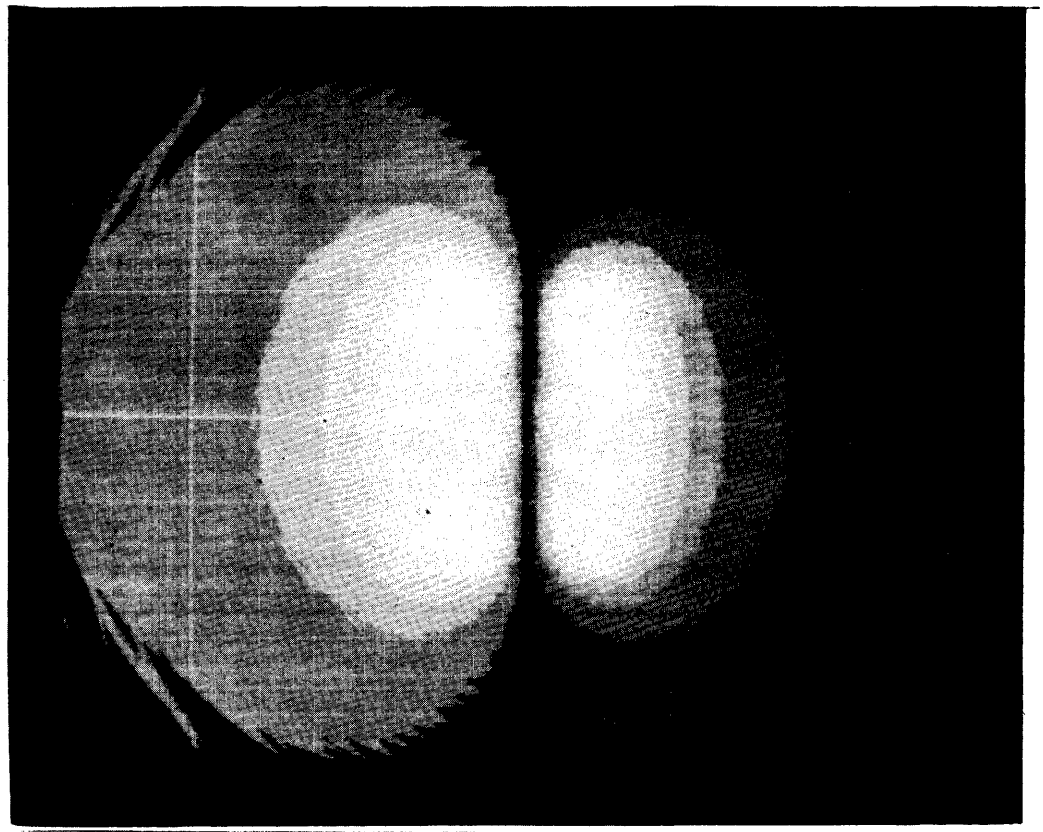


Figure 5. Utilization of Color in a Potential Field Investigation.

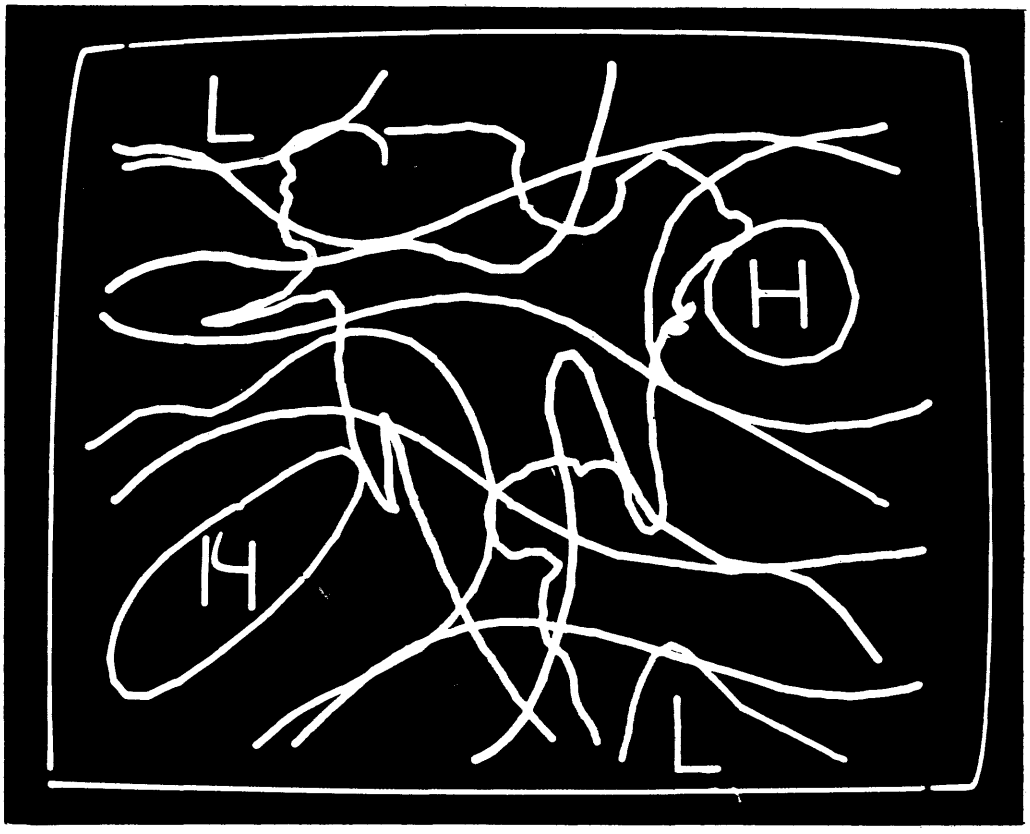


Figure 6. Use of Color in Distinguishing Isobar-Isotherm Data in a Meterological Investigation.

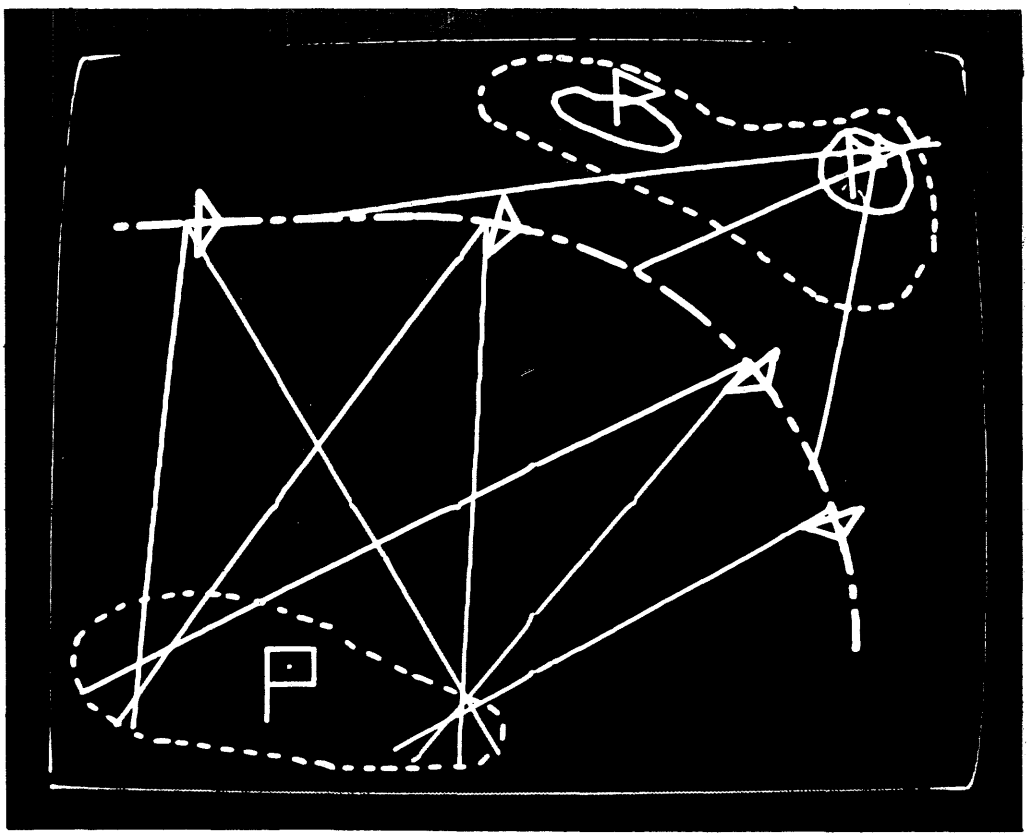


Figure 7. Identification and Location of Ground Radio Transmission by Bearing Angle Triangulation from Direction Finding Apparatus.

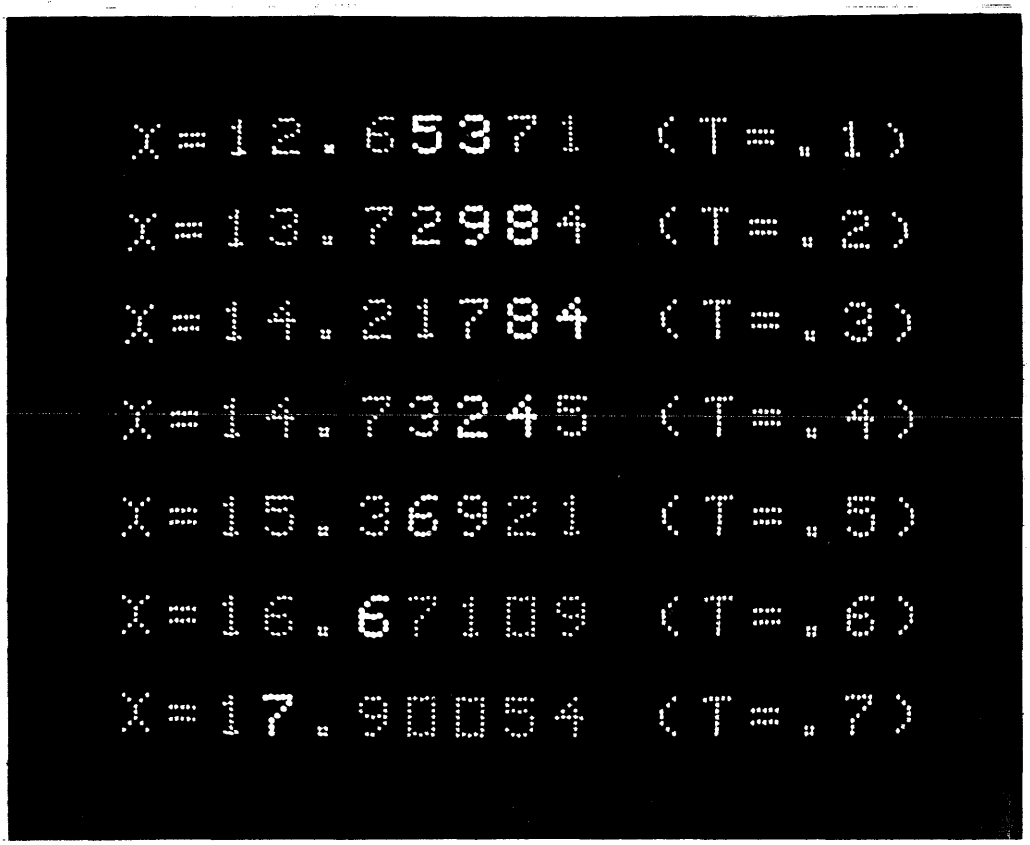


Figure 8. Use of Color in Specifying the Significance of Numerical Data.

SYSTEMS ANALYSIS OF DEC 338 PROGRAMMED BUFFERED DISPLAY

Stephen F. Lundstrom
University of Michigan
Ann Arbor, Michigan

Abstract

A semi-Markov chain model of the major states of the Digital Equipment Corporation (DEC) 338 Programmed Buffered Display is developed. The use of the model in determining best policies for graphics programming is described. In addition, the means for use of the model for comparison of effectiveness of various deflection logic hardware configurations is developed.

A PDP-8 Simulator which provides a supporting role in gathering statistics about the 338 Display programs is presented and discussed.

Introduction

This paper describes the system analysis of the DEC 338 Programmed Buffered Display (hereafter called the DEC 338). An introduction to the physical DEC 338 system and programming constraints is presented first in support of a summary of some of the typical problems which may be studied during the analysis of the system. The development and discussion of the semi-Markov model follows with a related discussion of the procedures which may be used for optimization of various display related parameters and evaluation of proposed changes in the display system design. A discussion of a means of gathering the statistics needed to analyze this system and some proposals for future extensions of the current work are presented in conclusion.

DEC 338 System Configuration

The DEC 338 Display consists of a PDP-8 Programmed Digital Computer and a 338 Display Control which includes a CRT with a 9 3/8" square usable display area, a pushbutton box with 13 pushbuttons for operator interaction with the system and a light pen which senses information displayed on the face of the CRT. The rest of this paper will concentrate upon the operation of the 338 Display Control which is essentially a channel which executes programs in a magnetic core memory shared with the PDP-8. The instructions which the 338 Display Control executes modify various control registers in the Display Control, control the positioning and display of information on the face of the CRT, interrogate the pushbutton box and effect interaction with the PDP-8 Central Processor.

The following is a short summary of the instructions available to the programmer when writing instructions for the 338 Display Control.

Control State Instructions - (Instructions which modify control registers)

1. Parameters - Set Scale, Enable the Light Pen, Set Intensity.

2. Mode - Stop the Display Control, Set the kind of Data State Instructions, Clear the low and/or high order display coordinates, Start Data State Instructions.

3. Jump - Direct jump to any of 32K locations or recursive subroutine call with much of the display status saved on a push-down stack.

4. Pop - Return from a subroutine. Options to defeat restoring of display status is provided.

5. Conditional Skip - A number of instructions which skip on defined logical conditions of the pushbutton register are provided.

6. Arithmetic Skip - A number of instructions to skip if the contents of the pushbutton register is equal to the value specified in the instruction are provided.

7. Misc. Skips - An instruction which may be microprogrammed to skip on the conditions of various pushbutton and other display conditions is provided.

8. Count - An instruction to count either or both the intensity and scale registers up and down is provided.

Data State Instructions - (Instructions which control CRT display)

1. Point - A two word instruction to plot a point.

2. Increment - An instruction which allows 1, 2, or 3 moves in one of 8 primary directions. Two of these instructions are packed in one word.

3. Vector - A two word instruction to plot a ΔX, ΔY vector.

4. Vector Continue - Same as Vector except the vector is drawn in the direction specified until the edge of the "screen" is encountered.

5. Short Vector - Same as Vector except that shorter vectors only may be drawn and only one word of memory is required.

6. Graph Plot - An instruction which allows either coordinate to be incremented and the other to be plotted with the coordinate given in the one word required.

A number of options such as character mode in the data state and the set slaves in the control state are not considered here as they are optional equipment. For a more detailed description of the PDP-8 and the 338 Display the PDP-8 User Handbook, F-85 and the Programmed Buffered Display 338 - Programming Manual DEC-08-G61A-D should be consulted (both available from Digital Equipment Corporation).

Now consider the operation of the 338 Display Control in more detail. The Display Control enters a number of major states during the execution of the instructions described above. Each state is begun by the access of one word in the magnetic core memory. The appropriate action is then taken and the Display Control enters another state to fetch the next word from storage. The amount of time spent in each state varies for a number of reasons. Initially, the Display Control must wait for its turn for a storage access cycle and the wait for the access cycle itself to be completed before decoding and executing of the accessed word can begin. The control state instructions require enough time to modify control registers as instructed before the next major state is entered. The data state instructions not only require the time for control register modification, but also require waiting for beam positioning (and intensification if requested). It should be noted here that vectors are drawn by point plotting. Thus the time per vector is the time required to plot the number of points required to draw the vector. A summary of the major states of the 338 Display Control follows.

Control State

Cycle 1 - Access and execute the parameter, mode, conditional skip, arithmetic skip, count, or pop instructions. Access the first word of the two word jump or push jump (recursive subroutine call) instruction.

Cycle 2 - Access the second word and execute the jump or push jump instruction.

Push 1 - Save the first status word on a push jump instruction.

Push 2 - Save the second status word on a push jump instruction.

Pop 1 - Restore the second status word on a pop instruction.

Pop 2 - Restore the first status word on a pop instruction.

Skip 1 - Skip the first word on a successful test during a skip instruction.

Skip 2 - Skip the second word on a successful test during a skip instruction.

Data State

S1 - Access and execute the increment, short vector, or graph plot instructions. Access the first word of the two word point, vector or vector continue instruction.

S2 - Access the second word of and execute the point, vector or vector continue instruction.

One of the major problems confronting the user of graphics terminals such as the 338 Display is the amount of information which can be displayed on the CRT and not irritate the viewer with "flicker". The flicker effect is caused by the reduction of the rate that the entire information display is presented on the face of the CRT to the point where the human operator can detect the separate presentations. The rate of information display presentation is clearly inversely related to the total amount of time spent in all of the above major states of the 338 Display Control. Graphics system users and programmers are thus interested in determining ways of minimizing the amount of time required to present the required information display. The systems analysis described in the rest of this paper was initiated in order that some of the possible ways of optimizing the rate of information display could be determined, at least as far as the "average" user is concerned.

Systems Analysis

The 338 Display Control systems analysis is described below. The development of the semi-Markov chain model is presented first with related definitions of various parameters in terms of the physical configuration. A discussion of the use of the model for the analysis of system performance, especially in terms of the questions posed at the end of the previous section, is presented to conclude this section.

Since the 338 Display Control executes programs written by human programmers, it is reasonable to assume that the occurrence and order of the various instructions are random variables when considered over the ensemble of programs it executes. Because the occurrence of instructions is a random variable independent of the previous instructions, it is thus possible to say that the occurrence of any one of the major states of the 338 Display Control is dependent only on the previous state. Thus, the major states of the Display Control may be modeled as a Markov chain. However, as mentioned before, the amount of time required to complete the required service in any state is variable. Since the time required is dependent on what points are plotted, how long the vectors are and how much time is required before the execution of the memory access cycle, the service times may also be considered to be random variables. Thus the model is a semi-Markov Chain.

The states of the Markov chain and their relationship to the major states of the 338 Display Control are shown in Table 1. The summary given previously relates the major states to the instructions being executed. The state diagram of this model is shown in Figure 1. The arrows between the states represent the corresponding transition probabilities. For example, the arrow between 2, CYCLE 2 and 1, CYCLE 1 represents P(2,1). The transition probability, P(i,j), is the probability of going to state j if it is known that the current state is state i.

The class of all states is closed, irreducible, positive recurrent, aperiodic and communicating. Thus it can be considered to be an ergodic Markov chain possessing a stationary long range distribution. Considering the states at the times of transition, the long term probability, Q, of being in

any state, i, can now be given in terms of the transition probabilities.

$$Q(I) = \sum_{J=1}^{6} Q(J)P(J,I) \quad ;$$

$$\text{for } I = 1,2,\ldots,6$$

$$\sum_{I=1}^{6} Q(I) = 1$$

The P(J,I) transition probabilities are those defined by the state diagram. The long term probabilities, Q(I), then are the solutions to the above set of equations. The system of equations can now be written out and solved.

$$\begin{aligned}
Q(1) &= Q(1)P(1,1)+Q(2)P(2,1) \\
&\quad +Q(3)+Q(4)+Q(5)+Q(6) \\
&\quad P(6,1)+Q(7)P(7,1) \\
Q(2) &= Q(1)P(1,2) \\
Q(3) &= Q(2)P(2,3) \\
Q(4) &= Q(1)P(1,4) \\
Q(5) &= Q(1)P(1,5) \\
Q(6) &= Q(1)P(1,6)+Q(6)P(6,6) \\
&\quad +Q(7)P(7,6) \\
Q(7) &= Q(6)P(6,7) \\
Q(1)&+Q(2)+Q(3)+Q(4)+Q(5)+Q(6) \\
&+Q(7) = 1
\end{aligned}$$

Solving in terms of transition probabilities:

$$\begin{aligned}
1/Q(1) &= P(1,1)+P(1,2)(1+P(2,1) \\
&\quad +2P(2,3))+2P(1,4) \\
&\quad +2P(1,5)+P(1,6)(1 \\
&\quad +P(6,1)+P(6,7)(1 \\
&\quad +P(7,1)))/(1-P(6,6) \\
&\quad -P(6,7)P(7,6)) \\
Q(2) &= Q(1)P(1,2) \\
Q(3) &= Q(1)P(1,2)P(2,3) \\
Q(4) &= Q(1)P(1,4) \\
Q(5) &= Q(1)P(1,5) \\
Q(6) &= Q(1)P(1,6)/(1-P(6,6) \\
&\quad -P(6,7)P(7,6)) \\
Q(7) &= Q(1)P(1,6)P(6,7) \\
&\quad /1-P(6,6)-P(6,7)P(7,6)
\end{aligned}$$

Thus Q(1) is now defined in terms of the transition probabilities and the other transition probabilities are defined in terms of Q(1) and the transition probabilities.

As mentioned previously, the long term probabilities, Q(I), defined above are with respect to the state transition times. The actual time of each transition is also a random variable. The reason for this is that the expected waiting time or expected service time, T(I), in each state is a function of how long a wait is required for a core cycle, how long a vector is drawn as well as many other factors. It now remains to take into consideration the variations in the service time in each state. In general, let T(I) be the mean length of time required to service state I for I = 1,2,...,6. Then in terms of the actual discrete equal time intervals, the long range distribution is given by:

$$R(I) = Q(I)T(I)/\left(\sum_{J=1}^{6} Q(I)T(J)\right)$$

$$\text{for } I = 1,2,\ldots,6$$

For this system of equations to have any meaning, the T(I)'s must be defined for all I's. As an introduction to the definition of T(I) for each state I, the following terminology is introduced.

Continuous Variables

- L(C) = Length of one core cycle
- L(IV) = Length of time required to plot one point in an intensified vector which is "on screen"
- L(NIV) = Length of time required to plot one point in a non-intensified vector which is "on screen"
- L(ICP) = Length of time required to plot "close" intensified points
- L(IFP) = Length of time required to plot "far" intensified points
- L(NCP) = Length of time required to plot "close" non-intensified points
- L(NFP) = Length of time required to plot "far" non-intensified points
- L(VOF) = Length of time required to plot a point in a vector which is "off screen"
- W(B) = Expected waiting time to beginning of storage access cycle

Discrete Variables

- N(VON) = Expected number of points plotted in the vector which are on the screen

N(VOF) = Expected number of points plotted in the vector which are off the screen

F = 0 if "close" point
  = 1 if "far" point
I = 0 if non-intensified
  = 1 if intensified
C(E) = 0 if the first part of increment word is an escape
  = 1 if the first part of increment word is not an escape
T = 0 if skip test is false
  = 1 if skip test is true
E = 0 edge flags enabled
  = 1 edge flags inhibited
E(V) = 0 edge not violated in vector
  = 1 edge violated in vector

Where F, I, C(E), T, E, E(V) are to be considered to be two-valued, discrete random variables.

Using the above defined constants and variables, the expected waiting time to get an instruction through a storage access cycle, W, the expected time required to draw a vector, V, and the expected time required to plot a point, P(P), can be defined as follows.

$$W = L(C) + W(B)$$

$$V = (L(IV)I+(1-I)L(NIV))N(VON)+E(V)EL(NFP)$$

$$P(P) = F(L(IFP)I+(1-I)L(NFP))+(1-F)(L(ICP)I+(1-I)L(NCP))$$

Also note that the total number of points expected in a vector, N(V) is given by

$$N(V) = N(VON)+N(VOF)$$

The values of T(I) for each state can now be summarized as shown in Table 2.

A single value for T(6) and T(7) must now be defined.

$$T(6) = W+P(SHT\ VCT)V+P(GPH\ PLT)P(P)+P(INC)V(1+C(E))$$

where $P(PNT)+P(INC)+P(VCT)+P(VCT\ CNT)+P(SHT\ VCT)+P(GPH\ PLT) = 1$

$$T(7) = W+(P(PNT)P(P)+V(P(VCT)+P(VCT\ CNT)))/N$$

where $P(PNT)+P(VCT)+P(VCT\ CNT) = N$

Note that the expected amount of time spent drawing information on the CRT screen is the expected time spent in states 6 and 7.

Uses of Model

The major problem in the development of new display systems and the improvement of existing display systems is how to get the maximum amount of information on the display. In the case of the 338 Display Control, this means two things. The probability of being in one of the two data states must be maximized. That is, the display must spend as much time as possible displaying information on the CRT screen and as little time as possible modifying control registers. However, a high probability of being in data state is not sufficient to insure presentation of the largest possible amount of information. The expected waiting time or expected service time in the data states must be minimized. These two parameters can be studied through the use of the semi-Markov chain model developed previously. T(6)+T(7) is the expected waiting time, EWT, or expected service time in the data states while R(6)+R(7) is the probability of being in one of the data states, P(DS).

One should note that the EWT and P(DS) are functions both of hardware and of software parameters. Thus, optimization of EWT and P(DS) can be carried out on software parameters or on hardware parameters. For example, one might optimize the software parameters by assuming a display with vectors of random lengths and random starting positions and determine what mix of point plot and/or vector instructions should be used to meet the optimization criterion. Some of the hardware parameters to be considered are the basic core cycle time, the type of vector generation (which affects length of time required to draw the vector), or the characteristics of the CRT deflection system. If this model is used

to show possible future areas for display development, a technique of seeing which variables have the most effect on optimization of EWT and P(DS) would probably be most useful. Perhaps an appropriate procedure would be to plot percent improvement of EWT and P(DS) vs percent change of each of the hardware and software parameters. Such a family of curves could show quite clearly which areas would prove to provide the most improvement in results for "equivalent" efforts.

Statistics Gathering

In order that the above studies may be carried out, some means of gathering statistics to define typical values of the lengths of vectors, what points are most often plotted and the different transition probabilities must be provided. To accomplish the task of statistics gathering, a PDP-8 Simulator has been programmed which operates on the PDP-7. This simulator can simulate any programs which can operate in a standard 4K PDP-8. A more detailed report on the PDP-8 Simulator will be available at a later date. It will be sufficient here to point out that during simulation, the section which handles IOT (Input-Output Transfer) instructions monitors for display oriented IOT's. When the display start IOT is executed, control is passed to a 338 Display Control Simulator which simulates the execution of the display program, within limits, gathers relevant statistics and summarizes them on a teletype printout upon completion of the simulation. A summary of the statistics which the program will gather and what simplifying assumptions have been made will be given here as a conclusion.

The probability transition matrix is expanded to one more state by removing all skips from Cycle 1 and giving them an initial state of their own (called SKIP TEST). This is done because the skip instructions are the only instructions whose transition probabilities to the next step are related to the probability of operator interaction. The simulation simply follows all branches at skips in all possible directions and keeps a count of the state transitions accordingly. The simulation is deemed complete when all skips have been followed accordingly and either instructions are encountered which were previously recorded or the display executes the halt command. At the completion of the simulation, the probability of transitions for the added state must be multiplied by the probability of operator interaction to cause a successful skip condition. When this has been done, the state may be recombined with Cycle 1 and analysis will proceed as described previously. In addition, the types and number of each skip instruction are tallied as are the number of times each point was plotted with how many times it was intensified and how many escapes were made. Also, similar data is kept for every length of vector. In addition to this tabular data, a count is kept on each of a number of other variables of interest. These variables are the number of "far" points plotted, how many increment instructions were executed, how many of those escaped and how many of those escaped on the first byte of the word, how many edge violations were encountered and how many illegal instructions (or instructions which were ignored) were seen. The simplifying assumptions made are that the minimum paper size is used, edge flags are inhibited, there are no slaves or character generator, the pop instruction cannot enter the data state and there are no vector continue instructions.

It is anticipated that the continued development and use of this program will generate a sufficient number of program summaries to be useful in performing some of the analyses suggested at the conclusion of the previous section.

Acknowledgements

The work reported here was originally done for partial fulfillment of the requirements for EE 673, a course in Large Scale System Design at the University of Michigan. Dr. Keki B. Irani provided necessary guidance, counseling, and stimulation. The original work on the PDP-8 and 338 Display Control Simulators was done on a PDP-7 which is part of the CONCOMP project at the

University of Michigan. The CONCOMP project is supported by the Advanced Research Projects Agency of the Department of Defense (contract number DA-49-083 OSA-3050, ARPA order number 716 administered through the Office of Research Administration, Ann Arbor, Michigan). Further development of the simulators is being done with the support of the CONCOMP project.

CORRESPONDENCE OF MAJOR STATES

MARKOV CHAIN MODEL	338 DISPLAY CONTROL
CYCLE 1	CYCLE 1
CYCLE 2	CYCLE 2
PUSH	PUSH 1
	PUSH 2
POP	POP 1
	POP 2
SKIP	SKIP 1
	SKIP 2
DATA 1	S1
DATA 2	S2

Table 1 Correspondence of Major States

EXPECTED WAITING TIMES, T_i

i	STATE	SUBSTATE	T_i
1	CYCLE 1		W
2	CYCLE 2		W
3	PUSH		2 W
4	POP		2 W
5	SKIP		2 W
6	DATA 1	POINT	W
		INCREMENT	$W + V + C_e V$
		VECTOR	W
		VECTOR CONT.	W
		SHORT VECTOR	W + V
		GRAPH PLOT	$W + P_p$
7	DATA 2	POINT	$W + P_p$
		VECTOR	W + V
		VECTOR CONT.	W + V

Table 2 Expected Waiting Times, T(I)

STATE DIAGRAM

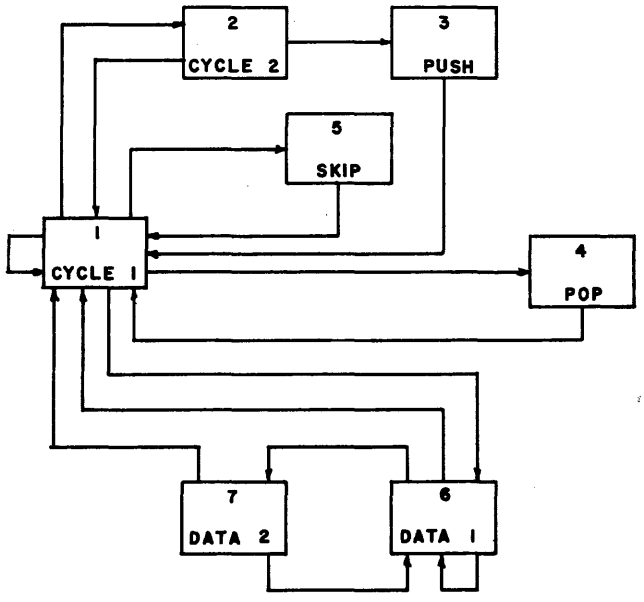


Figure 2 State Diagram

NUMERICAL CONTROL GRAPHIC SYSTEM

J. A. Snow
The Boeing Company
Seattle, Washington

Abstract

A research project at The Boeing Company was established to explore the feasibility of using a computer with a cathode ray tube (CRT) display to generate machining instructions for numerically controlled machine tools. The part programmer would input part geometry and observe a display of this input on the CRT. From the displayed part geometry, a cutter path would be generated, displayed, and then processed and transformed into a punched mylar tape.

The results of this project showed that an accurate part could be machined from information generated on a CRT coupled to a computer. This research also showed that there could be a significant reduction of flow time from part definition to control tape by using a graphic system in place of the conventional numerical control computing system.

Introduction

In July, 1965, the Numerical Control Computing Group (NCC) in the Aerospace Group of The Boeing Company began participating in Computer Aided Design Research Effort (CADRE). For the NCC group, this research effort was to be directed towards the development of an assemblage of computer programs that would speed up the part programming phases of part machining. These programs were to reside in a special purpose computer, equipped with a CRT, teletypewriter, and a function button box (push buttons under program control). The equipment chosen was a Digital Equipment Corporation (DEC) PDP-7 computer with a Model 340 scope (CRT).

The computer installation was to be maintained by the CADRE group, under the supervision of Jerry Kotulan. This group was to be responsible for the computer executive system, support library, and maintenance of all DEC supplied software.

The job facing the NCC group was to implement a numerical control computing system, totally new in concept, in which the part programmer would cause the part to be displayed on a CRT, and then generate cutter information from that display. The cutter information would then be converted to APT III[1] CLTAPE format and post processed. However, this system would not use APT per se.

[1] Latest version of the Automatically Programmed Tools language.

This project was motivated by the following considerations:

1. The total time spent in preparing an APT run of some part to be machined and in the acquisition of the punched control tape for the desired machine tool can be greatly shortened if the part programmer is placed directly within the computing loop. That is, the part programmer inputs information or commands into the computer on an on-line, real-time basis, and receives immediate and visual verification or acknowledgement of his actions from the computer.

2. Computer equipment needed for graphic part programming was available and at a reasonable cost (for a research effort).

3. Results were judged attainable within a year.

4. Extension of graphics into other areas was readily discernable, specifically, the whole design process, from definition of the part by the design engineer to the machining of the part.

5. The long range return was deemed very substantial in terms of dollars saved by the significant reduction in the process flow time from the designer's drawing board to the machine tool control tape.

6. Parts that could be programmed on a gra-

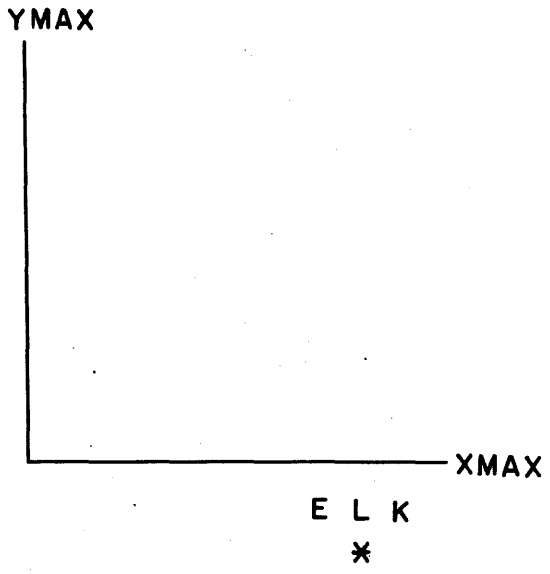
phic system comprised the bulk of all machinable parts. That is, two-dimensional parts consisting of lines and circles accounted for approximately sixty percent of all machinable parts at Boeing. This figure is substantiated by several other aerospace companies.[2]

The computer programs, hereafter called the NC-system, developed under CADRE are considered to be only the first plateau of effort in implementing a graphic numerical control computing system. Therefore, the capabilities included in the system were limited to those essential to the part programming of a part. There are other limitations due to ignorance, since there were very few guideposts indicating the proper direction in which to take in system development. However, the NCC group plans to develop a second generation system free from the limitations in the existing system plus any that can be foreseen. This second generation system would contain any additions needed to make the present system more flexible. Flexible not only from the part programmer's point of view but, also, in the ease in which capabilities can be added as the need for their inclusion into the system arises.

System Description

The NC-system is composed of six parts. These are:

1. INITIALIZATION - The part programmer establishes the transformation constants between part and scope coordinates by using the teletype to input the maximum dimension of the part. The coordinate axes are automatically displayed with the maximum part dimension located at the end of each axis. The control indicators, "E", "L", and "K" with an "*" under the "L" are also displayed under the abcissa axis. (See Figure 1.) The part programmer specifies one of three methods of information input with a function button push. The computer, under program control, will sometimes automatically select one of the three methods by which the part programmer is expected to input information. The asterisk moves under the proper control indicator. E-control allows information to be extracted from a previously defined part element. L-control (light pen) is the normal control mode unless another is specified. K-control allows information to be keyed in on the teletype.



NOTE: XMAX = YMAX

Figure 1 Scope display from Initialization section of system

2. DEFINITION - In this section of the system, the part programmer has access to the various geometry construction routines with which to establish the part display as well as the mathematical description of the part. Master Dimensions' software is used to mathematically define each element as well as calculate the points needed for the display of each element.

   These MD routines are computer adaptations of the mathematical techniques that convert a set of related numerical quantities into another set of numerical quantities, such that the second set is referred to as the cannonical form of a T-Curve.[3]

   Specifically, the input set of data consists of the endpoints and corresponding tangents of each element. These tangents, generally, are of arbitrary length.

[2] Lockheed, Georgia; McDonnell; North American; and Douglas

[3] The T-Curve is a generalization of the F-Curve, which is the designation of a parametric, three dimensional curve. The F-Curve and the T-Curve were developed by James Ferguson and Marion Rowin, respectively, both of The Boeing Company.

This data is processed by the T-Curve Conversion routines. The output set of data contains the element endpoints (unchanged), and re-calculated tangent lengths, depending upon the element type with which they are associated (line, circle, parabloa, etc.). Another term, called the "rho-value", is outputted, being a function of the endpoints and the tangents.

The reasons for using the T-Curve as the method for mathematically defining each part element are as follows:

a. Software required to generate the intermediate points between the endpoints of an element is kept to a minimum because the mathematical definition of each element is identical in format.

b. A T-Curve is a three-dimensional curve, although part elements are essentially two-dimensional with the third dimension set to zero. However, the part element definition can be expanded to three dimensions with no change in the T-Curve mathematics.

c. System expansion by including additional MD capabilities can be accomplished without a major overhaul of the system.

d. MD software (T-Curve) was used to see if it would perform as desired in the NC-system. With minor alterations to compensate for PDP-7 characteristics, the MD software performed very satisfactorily. Thusly, an MD-NC interface has been created that could have applications in areas other than graphic part programming.

3. CLDATA - This is the section of the system that generates cutter offset information. The part programmer establishes the machining and drilling sequences of the part. That is, he specifies the machining order of each element making up the part as well as the holes that need to be drilled. The part programmer has control of the Z-Height[4] of the cutter at all times.

   The cutter (or drill) can be returned to the set point at any time during the machining operation and the machine tool stopped. Consequently, the cutter or drill can be changed to another one of different size, and then the machining operation can be resumed. With this capability, multiple cutting passes can be generated. In drilling, a drill-bore-ream sequence could be established as well as the drilling of different sized holes. Therefore, the part programmer can control the machining of a wide range of part sizes, shapes, surface finishes, and drilled-hole sizes and finish.

4. 1108 - The original plans of the NC-system called for the cutter information generated in the CLDATA section of the system to be transmitted by Tel-Pac A lines to the UNIVAC 418 and then stored on a FASTRAND drum. The 418 is connected to the 1108 and both computers have access to this particular drum unit. The 418 would alert the 1108 of the presence of PDP-7 data. In due time, the 1108 would bring in the NC 1108 system that would convert the transmitted data into APT III CLTAPE format. The proper post processor would be called in to convert the CLTAPE data into machine tool instructions. The post processor output would be written on a magnetic tape. The data on the tape would then be converted to a control tape (punched mylar) on an IBM 360-30.

   As of August, 1966, the hook-up of the PDP-7, 418, and the 1108 computers was not fully operational. Therefore, the NC-system was modified to bypass this roadblock. Cutter information is punched out on paper tape at the PDP-7. The tape is converted to punched cards on the 360-30. These cards are then used as input data to the NC-Simulator,[5] which is runnable on either the 1108 or the IBM 7094. The simulator converts the cutter data into APT III CLTAPE format and then calls the post processor for the specified machine tool. The post processor output goes to the 360-30 to be converted into punched control tape for the machine tool. (See Figure 2.)

[4] Z-Height of cutter refers to the distance of the cutter tip above the zero position on the machine tool.

[5] Originally developed to check out post processors without going through APT to obtain CLTAPE data.

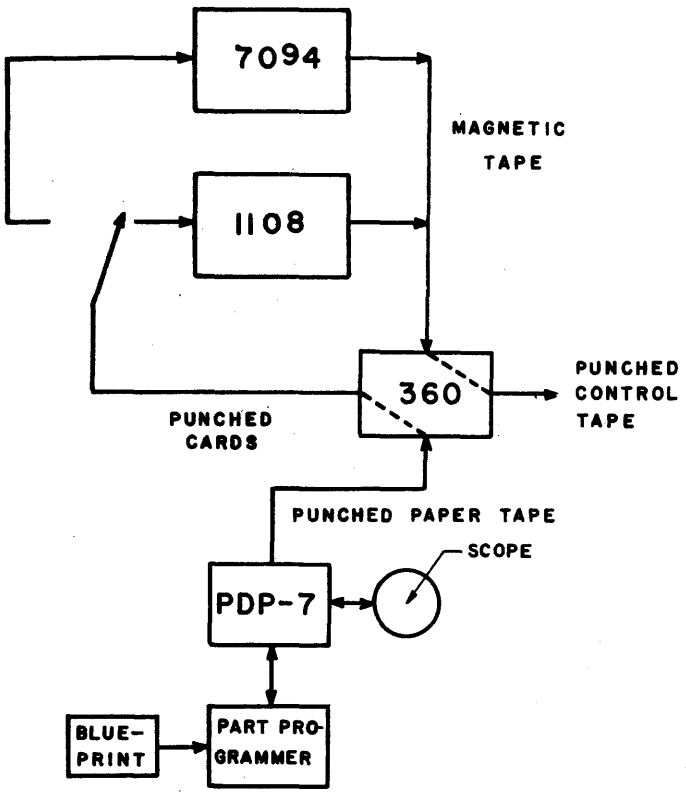


Figure 2 Information flow through NC-system

5. SIGN OFF - This section of the system serves only to reset the display counters and to zero out the display memory. This action erases everything being displayed except for the system control words at the bottom of the scope. After calling SIGN OFF, the part programmer could then start working on a new part.

6. UTILITY - Intersection of two lines. In many cases, the shoulder point, or the common point of the two intersecting lines for circular arc generation is not readily obtainable from a blueprint. With this section of the system, the part programmer can determine that point.

   The part programmer types in the X- and the Y- coordinate values of any two points on each of the two lines. The intersection point is calculated and the X- and Y-coordinates of that point are automatically typed out. This procedure may be repeated any number of times until all such points are obtained. (See 1.c. under System Operation.)

System Operation

1. Element Construction (Definition Section)

   a. Line - A line requires two points for construction. These points may be inputted either by light pen action only, typewriter, extraction of the endpoints of previously defined elements, or any combination of these three methods.

   b. Circle - A circle requires the center point and radius for construction. The center point can be inputted by any method mentioned in (a) above. The radius is keyed in. The circle is used only to indicate the location of a hole. The center of the hole is also displayed. Cutter motion cannot be directed around a circle defined in this section of the system.

   c. Circular Arc - A circular arc is inscribed between two intersecting lines, with the line segments terminating at the intersection point. (See Figure 3a.) Therefore, a circular arc requires the existence of two intersecting lines and the radius of the arc. The lines are extracted from storage by means of the light pen under E-control. The radius is keyed in as well as the number of points along the arc. The "V" formed by the intersecting lines on the outside of the arc is deleted. (See Figure 3b.)

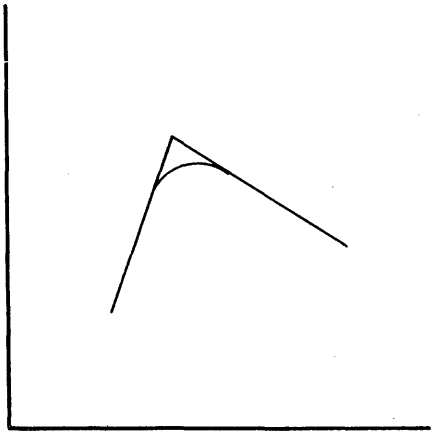


Figure 3a Circular arc construction

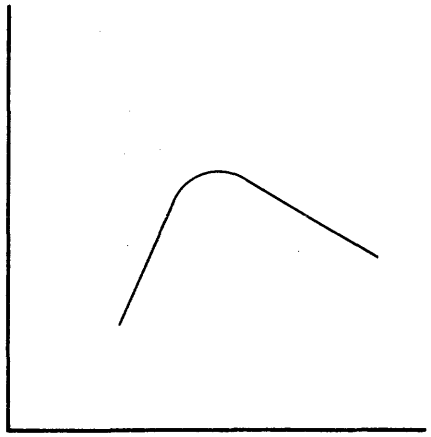
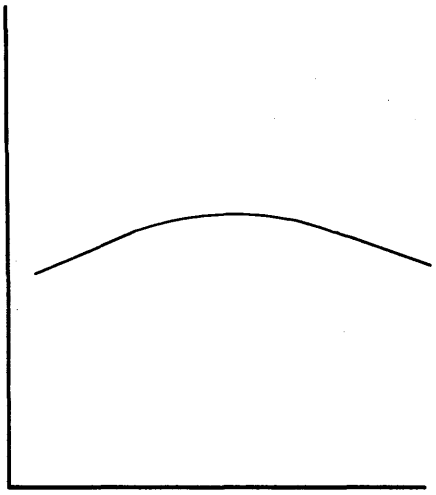


Figure 3b Circular arc construction

d. Parabolic Arc - A parabolic arc requires that two intersecting lines, but not displayed as such, be previously defined. The parabolic arc is placed between the lines by extracting these lines from storage and keying in the number of points along the arc. (See Figure 4.)



(Parabolic Arc from Test Part, Fig. 6)

Figure 4 Parabolic arc construction

All element construction must be maintained in the same general direction. Lines and arcs must intersect in a tangency condition.[6]

[6] Exceptions to these restraints are discussed in the section on Cutter Offset.

2. Cutter Offset (CLDATA Section)

This section of the system has been designed such that the part programmer is directed in a way as identical as possible to the way in which he might normally establish the cutting sequences of a part. The system requests that the X-, Y-, and Z- coordinates of the set-point and the radius of the cutter be keyed in by the part programmer. The part programmer selects the first element of the cut sequence with the light pen. He keys in the lead-in distance to the first element as well as the cutter height at this point. The cutter path to this point is displayed. (See Figure 5.) When the part programmer selects (with the light pen) the next element to be machined, the center path of the cutter is displayed along the first element. The part programmer touches each successive element with the light pen until the part is machined.

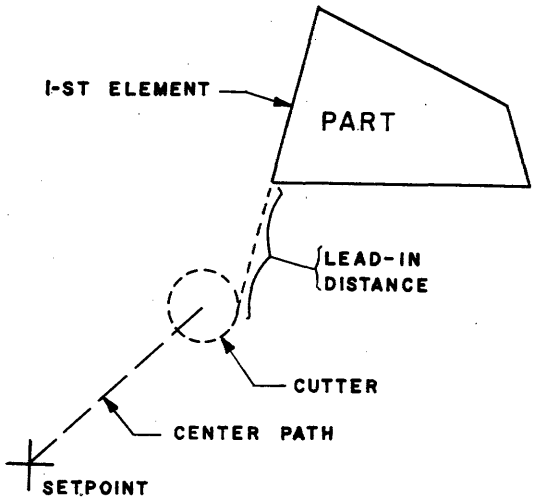


Figure 5 Start of a cut sequence

At any time between element cuts, the part programmer may alter the cutter Z, either in a step change or a taper change. He may also cause the cutter to be returned to the set point. To restart the machining operation, the part programmer must restate the set point, cutter radius, lead-in, and designate the first element of the cut. This means that multiple cutter passes at the part may be made. With this in mind, some of the construction restrictions may be relaxed. That is, lines and arcs need not be tangent if a line and an adjoining arc are not machined in the same cutter pass. The direction of element

construction must be in the same general direction only for elements grouped together in the same cutting sequence. By relaxing these construction restrictions and CAREFULLY paying attention to the grouping of part elements, the part programmer can define and machine a wide range of parts without committing any fatal mistakes.

3. Drill

The drill section allows the part programmer to cause the drilling of virtually an unlimited number of holes in a part. To drill holes of different diameters, the part programmer returns the drill head to the set point. A machine tool STOP command is automatically included with the outputted drilling information. (At this point, the machine tool operator can change the drill.) This operation can be repeated for each hole, if necessary.

To cause the drilling of a hole, the part programmer keys in the Z-value of a plane (called the "drill plane"[7]) parallel to the X-Y plane of the part. The depth of each hole is then measured from this plane. Drill head motion between hole center points takes place with the drill Z identical to the Z of the drill plane. This is done to prevent the drill from interferring with the fixtures holding the part and/or the part itself. To select the proper hole, the part programmer merely touches the dot at the center of the desired hole with the light pen. He then keys in the hole depth. The system processes these inputs and then recycles to the beginning of the drill sequence. The drill path from center point to center point is displayed, showing the part programmer which holes he has drilled.

4. Test Part

On August 25, 1966, a template for production part checking was machined from cutter information generated by the NC-system. This part is shown in Figure 6. It is comprised of straight lines, circular arcs, one parabolic arc, and one drilled hole. A total time of approximately three hours was needed from start to finish to obtain the punched paper tape (at the PDP-7) containing cutter offset information. The time breakdown for the various operations is as follows:

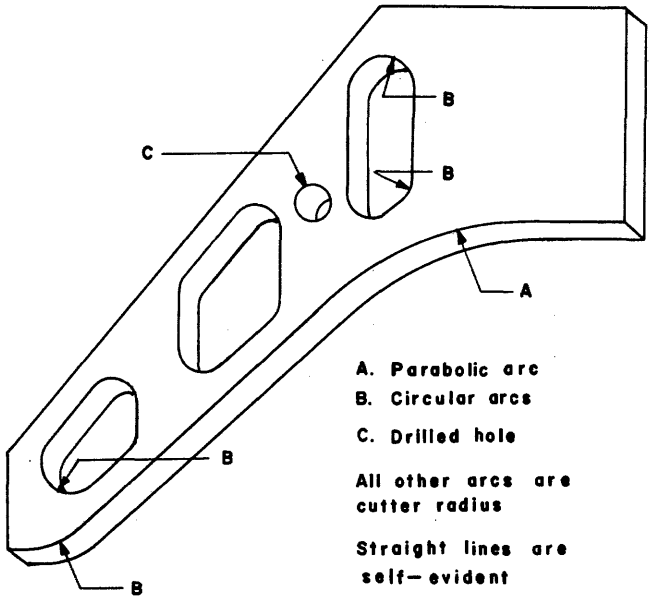


Figure 6 Test part

a. 2.0 hours - part familiarization, obtaining shoulder points for the circular arcs plus other preprocessing calculations.

b. 0.4 hours - generate the part display.

c. 0.4 hours - generate cutter offset information.

d. 0.2 hours - punch out the tape.

These times are all approximate. Overall flow time from start to machine tool control tape was one day.

By comparison, an experienced part programmer took the blueprint of this test part, wrote an APT part program and submitted it for computer processing. A total time of approximately eight part programming hours was needed from start to finish to obtain a machine tool control tape. The time breakdown for the various operations is as follows:

a. 7.5 hours - part familiarization and part programming.

[7] In part programming language, this is called the "clearance plane".

b. 0.3 hours - correction of part program after first computer run.

c. 0.3 hours - correction of part program after second computer run.

Three computer runs were required for successful execution of the part program. Overall flow time from start to machine tool control tape was four days.

Keypunch time was omitted since an accounting of this was not obtained; however, it would be approximately one hour. Another omitted item was preparation time for the machine tool operator's document. Since this operation is necessary for all jobs, then it would be a common factor in this comparison. Therefore, it was omitted.

System Evaluation

During the months of September and October, 1966, five part programmers each spent approximately twenty-four hours in using the NC-system. The programming experience of these men ranged from part programming school to over seven years. The consensus of ALL these programmers was as follows:

1. The NC-system was too limited for production use.[8]

2. The NC-system was easy to use, once they had become accustomed to how it worked.

3. Within the framework of its limited capabilities, this NC-system was superior to the standard APT system because

   a. there was no waiting for a job to be processed, since the part programmer ran his own job;

   b. the computer responded immediately to part programmer input and presented a visual display of the part geometry and cutter path; and

   c. errors were detected when they occurred, allowing immediate correction.

4. A production version of the NC-system should be implemented for part programmer use.

These same part programmers plus others in the part programming group established a list of specifications for a production version of the NC-CADRE system. When the NCC group implements a production level graphic part programming system, the structure of this system will be according to the specifications supplied by the part programming group.

An accurate comparison of the value of a graphic system and that of conventional methods is possible only when the total picture is considered. Therefore, in conventional methods we will not only include part programming in APT, using card input to the computer, but also an on-line APT system even though such a system does not exist. This is because statements have been made to the effect that if a part programmer had instant access to a computer, the flow time required for producing a control tape through APT would be reduced, such that any benefits derived from a graphic system would be greatly, if not totally nullified. On the basis of this observation, it is necessary to include an on-line APT system in the comparison to clearly demonstrate the superiority of the graphic concept.

Figures 7, 8, and 2 show the information flow from part programmer to control tape for the present day APT system, an on-line APT system, and the NC-system respectively. The man-hours required for producing the test part control tape with the present day APT system was eight. Overall flow time was four days. With the on-line APT system, the man-hours would be about the same, but the overall flow time would be reduced to less than two days (assuming no carryover into another work shift). With the CADRE system, three man-hours were required to produce the machine tool control tape for the test part. Overall flow time was one day.

[8] See final paragraph in "Introduction".

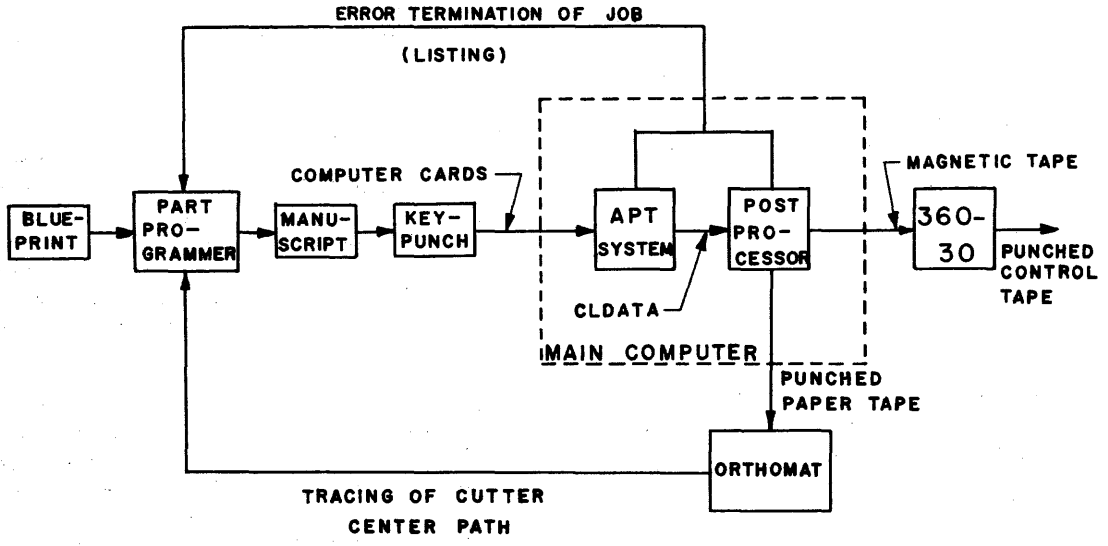


Figure 7 Information flow through standard APT system

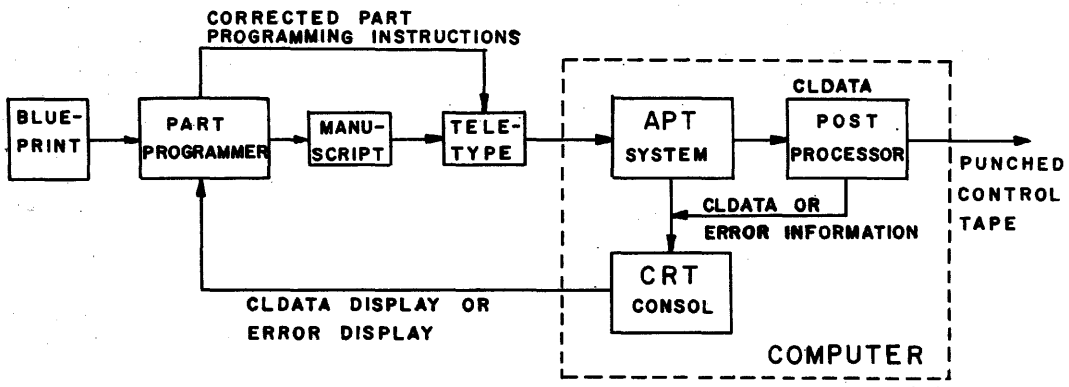


Figure 8 Information flow through an on-line APT system

Conclusions

With the limited capability NC-system, the time spent in part programming the test part was substantially less than part programming with present day methods using APT or an envisioned on-line system using APT. Clearly, the graphic system provides capabilities to part programmers that no other type of system can provide; namely:

1. Immediate visualization of all geometric construction and cutter path generation;
2. Immediate correction of errors, causing the probability of errors in the control tape to be almost zero;
3. A radical decrease in overall flow time for control tape generation; and
4. An increase in part programmer productivity by about a factor of four.[9]

When the NC-system is expanded to production status, part programming parts in APT that can be part programmed on the graphic system will be an out-moded and costly practice.

[9] This conclusion can be taken only for parts similar to the test part. Parts differing in complexity might indicate a different productivity ratio.

ENHANCEMENT TO A TIME-SHARED OPERATING SYSTEM

Richard N. Freedman
Laboratory for Nuclear Science*
Massachusetts Institute of Technology
Cambridge, Massachusetts

The PDP-6 work at LNS is to develop a system for recognition and measurement of tracks in bubble chamber pictures with a flying-spot device on-line to a small computer. The project was begun on a PDP-1, and was switched to a PDP-6 to attempt to take advantage of the speed of the flying-spot device. But with this background, we have favored and tried to retain the methods of program development originally suitable only for small computers.

Effective computer use in this fashion, however, requires more than the standard time-sharing of the processor, but similar sharing of the input-output devices. Our intent has been to give each user as much I/O capability as he can use, but no more - that is, to allow a secondary user every capability which the primary user does not need. And we have observed and implemented features which should be part of any small time-sharing system.

The installation at LNS consists of a PDP-6 with fast accumulators, 32K of two microsecond memory, 16K of five microsecond memory, a line printer, a card reader, a TYPE 3Ø display scope, four DECtapes, three magtapes, and two teletypes, all in one room. The social aspects of time-sharing are thus simplified, in that if the user three feet away is not using the printer, it must be available. But some software may be needed for the operating system to know it is available.

For example, consider the development of a program, which, in production, uses the line printer heavily. In the debugging phase, it may suffice to see, by DDT, whether variables are given plausible values or whether a section of coding runs to completion. Rather than suppressing the output commands, we merely specify a different device. For this purpose we have added a software device, called the NUL device, to our configuration. This device is like an invisible line printer with wastebasket attached. Data sent to this device is never seen again. But one of the best features of the DEC Multiprogramming System is the device-independence of I/O programming. So the program being tested with the NUL device is given a true test, and the alternate device remains accessible to other users.

A related problem in accessibility occurs when a program reads a dozen cards and then computes for an hour. The FORTRAN Operating System does not know that no more cards will be read, and is willing to keep the device tied up. But the programmer sometimes knows when he has read his last card, and should be able to release the device easily. With a slight change to the operating system, END FILE, for a non-tape device, can be made to release the device; or in general a subroutine call is available, which, for any device, ends the file and releases the device. Thus the card reader, in this case, becomes available to other users.

An opposite situation may occur, with a different solution necessary. After computing for an hour, a program may be ready to print results. In accordance with our principle, however, we have been using the printer for other jobs in that hour, and an attempt to print will cause an error exit. Clearly a time-sharing system should have some way of requesting a device. We have a subroutine which checks on the availability of a device. If it is available, the subroutine returns and the caller can confidently initiate output on the device. If it is in use, a message is typed. The user then either persuades the other user to relinquish the device or specifies an alternate device to be used.

Finally, one minor feature is useful to an installation with any special I/O equipment. We have the film scanning apparatus, and run our TYPE 3Ø scope as a special device. To reduce the system overhead time on such very fast devices, it is

convenient to be able to execute I/O instructions in user mode. Restricting this privilege to a particular job is inconvenient in a small group, because one person may be using the scope while another is compiling; but at any time the person compiling may have loaded and want the scope while the other has finished with it. So in our monitor any job, although only one at a time, may request to run in this privileged mode.

Another class of problems deals primarily with a large FORTRAN-oriented system, not necessarily time-shared. The scanning system consists of about a hundred subroutines, mostly in FORTRAN IV, with thirty or forty blocks of labelled common. Keeping this set up to date when one or two changes are made to common becomes quite a bookkeeping problem, which has been attacked in two ways. First, a magtape is used instead of six DECtapes, to keep the current sources and relocatable binaries. A filing system has been developed, which writes files containing both (or either) ASCII and binary versions of each program. Either or both parts of the files can be extracted at will, for loading, editing, listing, etc. The file name, date, and composition of each file is indicated in the three-word header on each record, to simplify the extraction procedure. This particular program, now available through DECUS, is not the only approach to the problem, but some massive storage device is necessary.

Another bookkeeping problem can occur in the changing of a common block. When twenty or thirty subroutines need to be changed to agree with a changed common block, it is desirable to minimize the work involved. A new feature of FORTRAN has been implemented for this problem, similar to a MACRO instruction in assembly language. The common block name is defined to compile as the normal full block specification, and only the system definitions need to be changed before a recompilation. (This version of the compiler, with the cliche feature, as it is called, will be available shortly from MIT.)

*This work has been supported in part by the United States Atomic Energy Commission under contract number AT (30-1) - 2098

THE COMPUTER DISPLAY IN A TIME-SHARING ENVIRONMENT*

Thomas P. Skinner
M.I.T. Project MAC
Cambridge Massachusetts

Abstract

Current trends in the computer field seem to indicate that in the future we will have very sophisticated graphical display consoles and much more powerful time-sharing systems. This paper discusses the numerous basic display configurations that are possible in a time-sharing environment. The display console itself, as well as the various communications systems available, are integrated in the discussion of possible software systems. Finally, a system is discussed which is a preliminary attempt at developing a terminal using the DEC 338 Display Computer connected to the MULTICS Time-Sharing System now under development at Project MAC. Connection is made by means of a 2400 bit per second full duplex dataphone.

Why Time-Sharing

Graphical terminals are now being used as input and output devices for the solution of very complicated problems. These problems may require a considerable amount of computing power. If it were not for the interactive nature of the computer display, the logical solution to the problem would be a larger computer. The large stand alone computer becomes somewhat of a financial burden when throughput is limited by the person using the display terminal. We must look to other schemes unless we have an unlimited budget. Time-sharing systems present an alternative to the problem.

The first level of time-sharing systems are actually batch processing monitors. A high level of sophistication in batch monitors has caused a number of them to be considered time-sharing systems. For example, input-output overlay processing and job scheduling is actually a form of time-sharing. Some display systems have used this sort of monitor to gain access to large machines for raw data reduction. GRAPHICS I at Bell Labs is an example of this form of time-sharing system.

*Work reported herein was supported by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01).

In the more conventional sense we have the time-sharing systems in which time slicing is employed and the user accesses the machine from remote terminals. CTSS at Project MAC is a prime example. Time-sharing systems of this nature provide a much better response for the time-sharing display. To a certain extent, real-time interaction is possible and the user has control over his programs.

Above the level of CTSS is the new class of time-sharing systems being developed. TSS 360 by IBM for the system 360 and Multics at MIT for the GE 645 are both new and extremely flexible systems.

Display Terminals

Display terminals fit into three basic categories:

A. Video typewriters
B. Directly driven displays
C. Remote processing displays

The video typewriter is little more than what its name implies. It can be faster than a hard copy device but offers little to extend graphic capability. Some clever things have been done with these consoles but they fall into the same category as graph drawing using characters and print positions on conventional printed output.

The directly driven display can be quite useful, but has the restriction that it requires a large amount of central processor time or a dedicated data channel to maintain the display. Usually the terminal must be located close to the processor because of the wide bandwidth of the communications channel required. The dedicated core memory needed to drive the data channel is also a big problem.

The remote processing display seems to offer the most flexibility since any bandwidth communication system may be used without needing a large bandwidth merely to maintain the display.

We now turn our attention to the class of communications systems that use the various data transmission devices provided by the telephone company. Both ends of the transmission path, whether it be a private wire or the dial network, must have a data set or dataphone. There are a large variety of dataphones differing in transmission rates and modulation methods. They all behave in a similar manner, however. A serial bit stream is transmitted at one end and received at the other. If two paths are provided simultaneously, such that both transmission and reception can occur concurrently, the communication system is said to be full duplex. Half duplex is merely one line that can only transmit in one direction at a time.

There are two major classes of dataphones:

1. High speed datasets which usually link two computer-like devices together. These datasets usually operate at speeds higher than 2,000 bits per second. Some new devices operate at up to 40,000 BPS.
2. Teleprinter rate lines usually ten to fifteen

characters per second connecting typewriter-like terminals together or to a computer.

High speed dataphones are best suited to the needs of the time-shared display. Large amounts of data can be sent in a relatively short amount of time. Display lists can be sent back and forth from the time-shared computer to the display. It is not too likely that large data bases other than display lists would need to be transmitted to the display console since the amount of computing power at the display terminal is usually minimal and would defeat the purpose of using the display in a time-shared environment. The low speed dataphone is a severe restriction on the interactive capability of the remote display. Data transfers of any proportion may severly hinder the response time of the system to real time interactions from the user.

Display Software

Display software is perhaps the limiting reagent in the graphics experiment. Software development on almost any scale is quite painful. Many display software systems in use today are extremely applications oriented. The Electronic Systems Laboratory at MIT has developed a display software system for their display console at Project MAC. It is a general purpose system with procedures to generate graphic constructs such as lines, circles, and subpictures.

With a remote processing configuration, such as the DEC 338, and a time-sharing system, such as Multics, we have a great deal of flexibility at hand. The proposed Multics graphics system will have software of a very general nature with provisions for letting the user define his own subsystem or particular real-time procedures such as light pen tracking or push button manipulation.

The Multics system graphical philosophy is that we maintain at the remote terminal processor only those procedures that are of a real-time nature. All other procedures reside in the central computer. A model of the display list can be kept in the central computer and the actual list at the remote terminal. Any changes are made to both the model and the actual display list.

At both the display end and at the Multics end of the communications line is a communications management module. This module is responsible for the maintenance of the communications link. We might then think of the central computer and the terminal as independent processes that are exchanging data by means of the communications link. There are numerous interlocking problems that this independence creates, but this type of system offers us much more freedom than the system in which the remote terminal is merely a slave the central computer.

The initial Multics graphics skeleton system will probably be much like other graphics systems but with the major difference that the user will be able to restructure any part of the system he desires. A

library of procedures and subsystems will enable the unsophisticated user to configure the system to his particular applications.

Current Hardware

The present display terminal connected to the Multics system consists of a DEC 338 display and PDP-8 computer with EAE and an additional memory module. The 338 includes the programmable character generator and search logic. Other peripheral hardware includes DECtape and DM01 multiplexor.

Special hardware was also designed by Project MAC and added to the installation. This includes a dual channel digital to analog convertor and a real-time clock. The dataphone used is a 201B and a specially designed interface to the PDP-8 that uses the three cycle data break facility. This interface allows full duplex operation of the 201B which is a 2400 bit per second modem.

The Multics system itself includes a dual processor GE 645 with 256K of memory. All communications line I-O to the GE machine goes through the GIOC (Generalized Input and Output Controller) which is very much like a data channel. A high speed drum and several low speed secondary storage devices are also included in the 645 installation.

Since at present the Multics system is not fully operational, it is not possible to develop the necessary software for the 338 terminal. The 338 terminal is presently being used as a debugging device for the Multics system. A system known as XRAY has been developed by which system programmers are able to investigate the contents of GE 645 memory and monitor certain data bases on the 338 display. This debugging device has the unique feature in that it does not require any 645 processor time to operate and can thus remain a true "invisible" debugging tool. It is anticipated that much more elaborate applications of the 338 terminal to debugging will be made once the Multics system is fully operational.

BIBLIOGRAPHY

1. Skinner, Thomas P., "Flow-Debug, A Graphical Debugging Aid for Online Computer Operations," S.B. Thesis, M.I.T., June 1966.

2. Digital Equipment Corporation, Programmed Buffered Display 338 Programming Manual (DEC-08-G61A-D), Maynard, Mass., October 1966.

3. Ninke, W. H., "Graphic 1 - A Remote Graphical Display Console System," Proc. FJCC 1965.

4. Stockham, T. G., Jr., "Some Methods of Graphical Debugging," Proc. of IBM Scientific Computing Symposium on Man-Machine Communication, May 1965, to be published.

5. New B-Core System for Programming the ESL Display Console, Memo 9442-M-122, Cambridge: Electronic Systems Laboratory, M.I.T., April 30, 1965

X-Y GRAPH PRODUCTION AND MANIPULATION

J.W. Brackett and Roy Kaplow
Department of Metallurgy and Center for Materials Science and Engineering
Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract

A number of user-oriented programs have been developed for forming x-y graph displays on the M.I.T. Compatible Time-Sharing System. A text input which describes the graph produces output on either the Project MAC interactive display facilities or remote storage oscilloscope displays, depending on the user's location. Both axes of the graph may be either logarithmic or linear; production of annotated graph "paper" is automatic unless ranges are set by the user. Up to three single- or multi-valued functions can be displayed simultaneously with a single request, any of which may be either point or line-connected plots. A specification of precision is used to select data points for line-connected curves to minimize buffer requirements and/or transmission time. A light pen, if available, may be used to reference any one of the curves on a multi-curve plot. The subsystem is conversational and will request any necessary information which the user omits or supplies incorrectly.

Introduction

Graphical display of information, always an important aspect of scientific and engineering analyses, is central to the concept of on-line computers, in which environment rapid and lucid communication between the machine and its users is perhaps the single most important requirement. Of the many forms which graphical display may take, x-y graphs are one of the more readily achieved and probably the most generally useful and understandable. Under this general heading, for example, are included the representation of histograms, multi-dimensional vectors, the direct relationship between two functions or vectors, (i.e., the generalization of Lissajous figures), as well as single- or multi-valued functions of one independent variable - with or without the variation of subsidiary parameters. Particularly because of their general applicability, it is desirable that a computer facility provide a means for the construction of such graphs to the least sophisticated of its users.

*This work was supported, in part, by Project MAC, an M.I.T. research program sponsored by the Advanced Research Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(02).

In this paper we describe some "commands" which have been written for this purpose, which are used within MAP[1,2], a conversational sub-system for mathematical analysis, and more generally, within the M.I.T. Compatible Time Sharing System (CTSS)[3]. Certain essential attributes of these commands are listed below; some of these will be discussed in detail in subsequent sections of the paper;

a) The user need know none of the details of the hardware.

b) The commands are usable in essentially invariant form, regardless of the particular hardware at the user's location.

c) The user need not write any programs. In fact, he need not know any details in advance, since all the information which the command requires is solicited by questions if the parameters are not provided on the command line.

d) Several superimposed curves may be displayed simultaneously.

e) "Graph paper" is generated automatically, in any desired linear-logarithmic combination. This includes the determination of the ranges

required for both ordinates (or the use of optional user specified ranges), the selection and labelling of suitable graph paper scales using "round-number" subdivisions, and the mapping of the data values into display scope coordinates.

f) For curves which are to be constructed with straight-line segments drawn between the data points, as is usually the case, data points which are either unnecessary or most dispensable are discarded, in light of hardware resolution, available buffer space, or transmission time requirements.

g) The user may interact with existing pictures, using a light pen if there is one and the typewriter, to request identification of curves at specific points and to request the deletion or addition of curves.

The Terminals

There are three different display terminal configurations attached to CTSS at present: a) the ESL Display Console[4] which uses a PDP-7 as a display buffer computer and which is connected to the IBM 7094 on a direct data channel; b) a PDP-7 with a 340 display connected by a 1200 bits/second telephone line, and; c) storage oscilloscope display[5], also connected by a 1200 bits/second telephone line. The first two types of terminals have light pen and pushbutton attachments for graphical input in addition to typewriters and also possess built-in character generators. All input for the storage tube display must pass through the associated typewriter console and all characters are presently drawn with vectors, but a low cost character generator is under development.[6]

The ESL and type 340 displays both have 21 inch screens, and a resolution of 1 part in 820 within the area used for the graph paper; both are subject to buffer space limitations and transmission time may be a consideration for the latter. The storage display has a 5 inch screen, and a resolution of about 1 part in 250; it has no buffer space limitation but transmission time may be a consideration. The single essential difference between the terminals, from the user's point of view for x-y graphs, is that displayed curves or specific spots on curves may be referenced with a light pen on the dynamic displays, while typewriter "by-name" reference must be used on the storage units. Although the actual programs for the commands obviously vary considerably, the user is shielded from the differences even if the command is incorporated (as a subroutine) in a program which is to be used interchangeably among the terminals. The proper procedure is loaded automatically, depending on the terminal the user plans to employ. This concept, to which relatively little attention has been paid in the past, is important to the maintenance of flexibility in a facility, particularly one with experimental attributes.

Format of Graphs

If the user specifies no format information other than indicating the data to be plotted, a graph with linear axes and with line segments connecting the data points will be generated; a suitable "graph-paper" is used, depending on the ranges required by the data. The user may optionally specify ranges to be used for either or both ordinates; if these are smaller than is implicit in the data the display will be a magnified representation of a limited region. Figure 1 is a typical display of two functions, one represented by a line segment plot and the other by the data points. The lower left corner of the graph paper is labelled with complete numerical "E" format values. Subsequent divisions on the graph paper are specified by the appropriate least significant digits. Only integer labels are used for logarithmic axes and these then designate powers of 10.

Sampling of Data

Since the user may wish to plot several functions simultaneously, each of which may contain a thousand or more data points, the buffer space may be insufficient or the transmission time excessive for reasonable service. If line segment curves are to be drawn, it is often possible to satisfy a maximum number of points criterion by eliminating only those points which are redundant in terms of the hardware resolution. Given a resolution of δ, the following algorithm is used to select only the necessary (or most important) points:

a) The first data point is always selected.

b) An imaginary line is extrapolated from the most recently selected point, i, through and beyond the next point, i+1.

c) The vertical ordinate difference, Δy_j, between the jth point and the extrapolated line is compared to δ, for j = i+2, i+3, etc. until,

d) Δy_j is greater than $\delta/2$, then the j-1

point is selected (and the process continues from step b).

e) The last data point is always selected.

f) If the resolution δ is insufficient to reduce the total number of points to a required level, δ is multiplied by the ratio: number remaining/ number allowed, and additional passes taken, as required.

If more than one curve is to be plotted, points are selected for all of them with a common δ . Thus, if a rapidly oscillating function is displayed simultaneously with a smooth function, the bulk of the allowable points are utilized for the former. Although the procedure is not a true selector of "most important points", it is quite effective and rapid in execution; moreover, when a single pass with hardware resolution is sufficient, as often is the case, the user is not even aware of the sampling process.

Figure 2 illustrates the use of the sampling procedure in a practical case; the plotting of $\cos(x^2)$ for $0 \leq x \leq 6$. The original function, displayed in Figure 2a, was represented by 601 data points, tabulated at equal intervals in x with a Δx of .01. The sampling algorithm, however, does not assume that the function is given at equal intervals nor does it assume that the x coordinates of the data points monotonically increase. With hardware resolution tolerance, (δ = y axis range/820 = .0024), the procedure discards 180 (30%) of the points without any perceptible effect on the display. With the tolerance increased to .02, the number of points was reduced to 183. Figure 2b shows the display formed from the 84 points remaining after a pass with the tolerance set to .10. Although only about one seventh of the original points remain, the positions of the maxima and minima are unchanged and the display is usable for many purposes.

The same type of sampling may be used on graphs having logarithmic scales for one or both of the axes. Figure 3 is a plot of $e^{-(x-6)^2}$ for $1 \leq x \leq 11$ on 12 cycle semi-log graph paper. Only 18 points, selected by sampling with a tolerance of ten times the resolution criterion, have been used; the resolution criterion requires 81 points.

User Communication with the Plotting Procedures

The plotting commands, as incorporated within the MAP System, allow the user to display up to three functions by simply providing their names, which follow the usual function or vector notation, e.g., g(x) or h(i). If the function is tabulated at equal intervals in the independent variable, the MAP function definition includes the range and the interval in the independent variable for which the function is defined; for plots of such functions the horizontal ordinate is thus specified implicitly. If the user types only the MAP command "plot", all of the required or optional information, including the names of the functions, will be requested; the details of the interaction are available elsewhere[1,2]. If he follows the word "plot" with the names of the functions to be plotted, the graph is immediately constructed, with all the default options assumed, as previously described. Various combinations of options may be specified in a format-free fashion, as in the following example:

```
plot g(x) lines h(x) points log-linear 3.2 6.8
```

which specifies a graph of g(x) and h(x) for $3.2 \leq x \leq 6.8$ on semi-log paper, with g(x) drawn as a continuous curve and h(x) indicated by the data points alone.

A graph for the storage tube requires the companion command, "plot storage"; all of the parameter information can be specified in identical fashion.

The relationship between sets of tabular data, not necessarily equal interval functions of an independent variable, but related through a subscript correspondence, can be displayed using the MAP "compare" command, e.g.,:

```
compare ind(i) a(i) b(i) c(i) linear-log
```

will plot the values of a(i), b(i) and c(i) against the logarithms of the values of ind(i).

The facilities of the MAP compare command are available to programmers using CTSS through a subroutine named PLOTIT[7] . The argument to PLOTIT is a pointer to block of text of the same form that would be provided with the MAP command. As with MAP the required information will be requested

if any errors or omissions in the text are detected during execution of the procedure. In Figure 4 a sample interaction is shown in which an error was detected and the required information requested from the user. Up to three functions may be plotted on the same graph with one request; the data is assumed to be stored in the CTSS disk file in a standard format, for which purpose matching disk writing and reading subroutines are available.

With both the MAP commands and PLOTIT, the user is given the opportunity to interact with the displayed graph, i.e., the programs specifically ask for any desired further interaction. The user can cause the names of the functions or disk files to be printed on specific curves. If a light pen is available it can be used to point to a definite spot on the curve for the label, or to a curve to be removed; otherwise the curve is referenced by name through the typewriter. Curves can also be added to an existing graph (up to a net total of four).

For more sophisticated programmers, interested in exercising the greatest possible control over the displayed output, the basic subroutines used to generate the display for the MAP commands and PLOTIT are available. In particular, the use of the basic routines allows greater flexibility in utilizing the light pen. With only slightly increased programming complexity, for example, the user can indicate a function or disk file on which any further calculations are to be done by pointing at its representation on an existing graph.

Acknowledgements

The graph "paper" routines used in these procedures are only slight variations of those developed by Dr. Thomas G. Stockham of M.I.T. Lincoln Laboratory, which were included in his plotting routines which were available in early 1965 at Project MAC. We also appreciate Dr. Stockham's continuing interest in this work.

References

1. Kaplow, R., Brackett, J., Strong, S., Man-Machine Communication in On-Line Mathematical Analysis, Proceedings of the Fall Joint Computer Conference 29, 465-477, Spartan Books, Washington, D.C., 1966.

2. Kaplow, R., Strong, S., Brackett, J., MAP, A System for On-Line Mathematical Analysis: Description of the Language and User Manual, Technical Report MAC-TR-24, M.I.T., 1966.

3. Crisman, P.A., Editor: The Compatible Time-Sharing System: A Programmer's Guide, second edition, M.I.T. Press, Cambridge, Massachusetts, 1965.

4. Stotz, R., Ward, J., Operating Manual for the ESL Display Console, Electronic Systems Laboratory Memorandum M--129, M.I.T., 1965.

5. Cheek, T., Ward, J., Thornhill, D., Operation and Programming Manual for the ARDS-I Experimental Dataphone - Driven Remote Storage-Tube Display, Project MAC Memorandum M-336, 1966.

6. Cheek, T., Design of a Low-Cost Character Generator for Remote Computer Displays, Technical Report MAC-TR-26, M.I.T., 1966.

7. Brackett, J., Subroutines for Automatic Plotting, M.I T., Computation Center Memorandum 270, 1967.

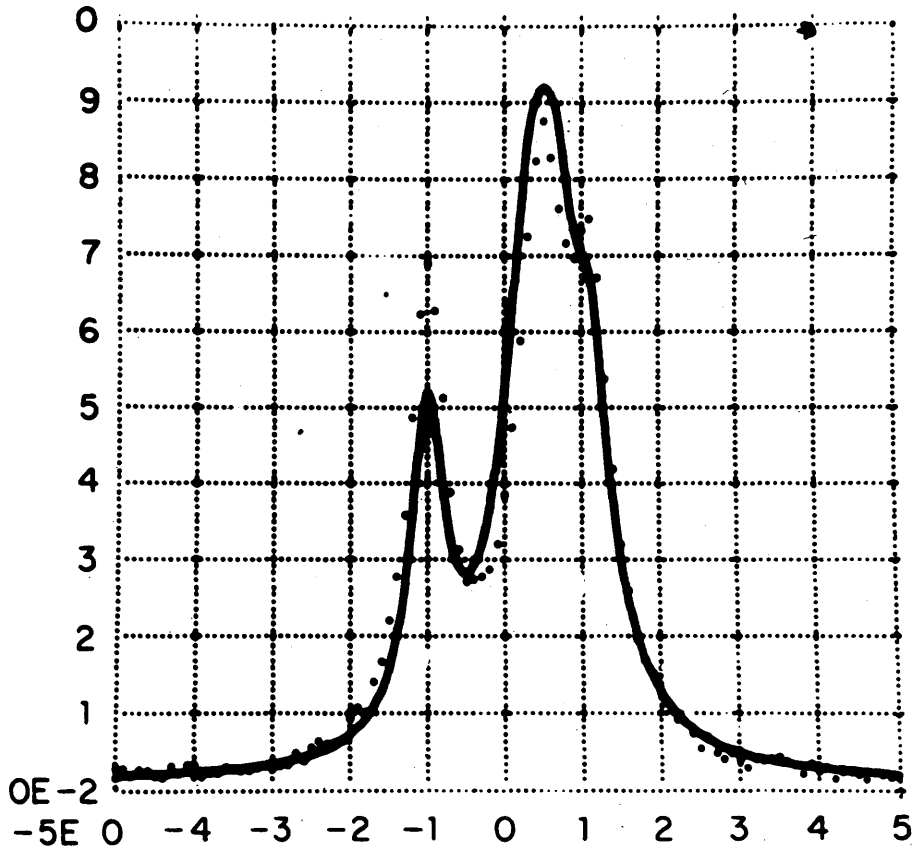


Figure 1: Simultaneous display of two functions, one as a continuous curve and the other as data points only. The figure is in reverse of the actual display.

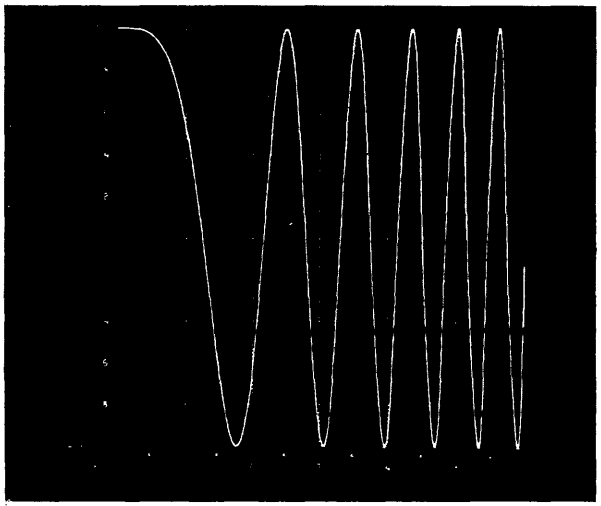


Figure 2a: Plot of $\cos(x^2)$ as represented by 601 points.

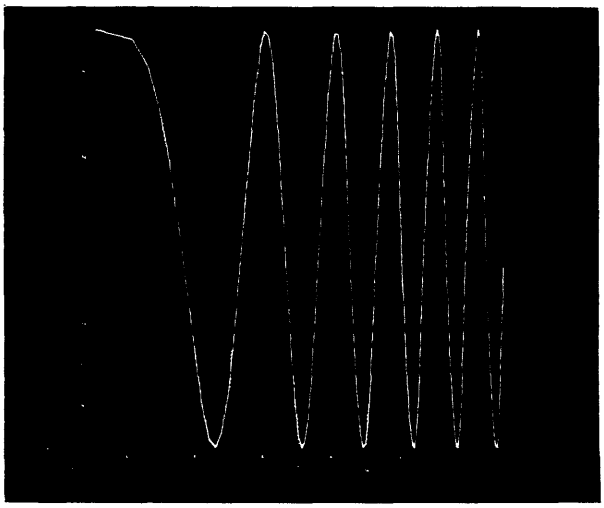


Figure 2b: Plot of $\cos(x^2)$ as represented by 84 points selected by the sampling algorithm.

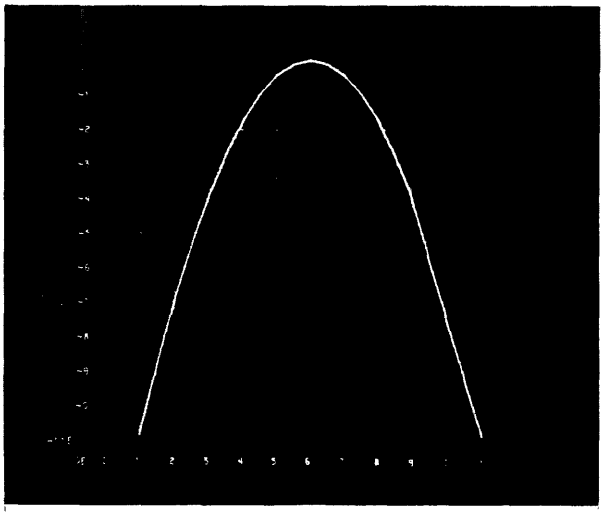


Figure 3: Plot of $e^{-(x-6)^2}$ as represented by 18 points and displayed on log-linear graph paper.

```
AN ERROR IN THE ARGUMENT TO PLOTIT HAS BEEN FOUND.  THE INFORMATION NECESSARY TO
PREPARE THE GRAPH WILL BE REQUESTED.

WHAT IS THE NAME OF THE DISK FILE CONTAINING THE VALUES WHICH SHOULD BE USED TO
SPECIFY THE X AXIS.      dist x

WHAT ARE THE NAMES OF THE DISK FILES CONTAINING THE VALUES WHICH SPECIFY THE Y AXIS.
exp x cosine x


SHOULD THE PLOT BE LINEAR, LOG LOG, LINEAR LOG, OR LOG LINEAR.          linear

IF YOU WANT A POINT PLOT OF ANY OF THE FUNCTIONS, TYPE THE NAMES OF THE APPROPRIATE
DISK FILES.  OTHERWISE GIVE ONLY A CARRIAGE RETURN.

IF YOU DO NOT WANT ALL OF THE POINTS OF THE FUNCTIONS(S) PLOTTED, TYPE THE DECIMAL
VALUES INDICATING THE DESIRED RANGE FOR THE X AXIS.  OTHERWISE GIVE ONLY A CARRIAGE
RETURN.
2.5e01 4.0e01
```

Figure 4: A sample interaction using PLOT IT.

COMPUTER IMAGE PROCESSING OF BIOLOGICAL SPECIMENS

Stephen Lorch
Massachusetts General Hospital
Boston, Massachusetts

With the rapidly falling cost of computers and bulk storage the feasibility of building inexpensive flying spot scanning devices interfaced to a small computer has markedly increased in the last two years. The major problem of pattern recognition and how much this function can be automated is still in large part unresolved, but as more flying spot scanners appear and more investigators become conversant with the techniques, their advantages and limitations, some of these questions should soon be answered.

We have interfaced a flying spot scanner to our PDP-7 as inexpensively as possible. The optics consists of an old Kodak slide projector which has special mounting screws to permit the replacement of the projector bulb by a photomultiplier tube. The 30D scope or Fairchild 347 is used to generate a scanning raster and the signal from the photomultiplier is processed either through the A/D converter (type 138E), or the interrupt, using Z axis modulation. The scanning is accomplished with a TV type scan whose resoltuion can be preset in the accumulator switches by the operator. Intermediate images are stored on DEC tape and after the scan is complete are automatically played back onto the screen. A light pen feature is also included which allows a portion of the image to be chosen on playback and scanned at high-resolution (1024 points) and the image also stored. The program then allows either image to be played back, either as scanned or differentially by shades of gray. In addition, the program allows the negative of the scan to be played back, or all but certain shades of gray, and it will also perform edge detection.

Importance of Scanning to Biology and Medicine:

Much data collected by biologists or physicians are presented in some analog form, frequently in the form of a picture. In fact, many of these pictures contain information in the form of shape and/or shades of gray, such as brain tissues, cytological smears, and X-rays. The quantification of this type of data has been extremely difficult, and the importance of various constellations of patterns has been transmitted from generation to generation on an apprenticeship basis. The development of scanning techniques; the ability to store these images and therefore quantitatively detect unique features associated with certain pathological states permits more systematic investigation of the information carried in these images than was previously possible. In addition, transformations of a digital computer such as contouring, edge detection, display of specific shades of gray, may enable the investigator to make more discernible the heretofore intuitive patterns he has recognized in the specimens. In addition, the possibility of automating such routine procedures as chest X-rays, or papanicolar smears would add immeasurably to the preventive medicine and public health programs which are now in progress.

In our laboratory we have been concerned more with problems in the cellular structure of animals, specifically in visual central nervous system. For instance, a typical question one would like to answer is, "does the neuronal network of cells found in the visual system develop by the experience of seeing or is it determined genetically"? This type of investigation has always been extremely difficult because of the magnitude of interconnections one finds in the visual system. However, with the scanning of brain tissue into the computer it may be possible to have the computer make a histological analysis of the richness of interconnection of cells in this area, between animals who have been able to see pattern and animals who have been brought up in a patternless environment. We do not believe that the computer at this point is able to discern which of these are truely neurons as opposed to artifact or "noise" in the staining technique, We therefore have devised our program to allow interaction by a human via light pen to caricature the interconnections and to edit out most of the artifact. In this way the image scanned into the computer directly can be cleaned up before the computer is asked to process it In addition, it is conceivable to have the computer programmed heuristicly to recognize some of this artifact, and in a sense

be rewarded by the operator or his light pen so that in the future this type of editing could be done in a more automatic fashion.

Corollary to the study of cellular behavior would be the study of more gross anatomical structure as demonstrated by a section of monkey brain. A typical brain tissue slide may be stained to show the density of myelin sheathing around fibers running in the brain. The anatomy of certain of these structures is of interest to the biologist and neuropathologist and certain tracts are followed after lesions have been made in the brain to see how these fibers degenerate because of the lesion. It is extremely difficult to differentiate certain shades of gray in any particular slide. However, if we take a slide of this type, scan it and then play it back using differential shades of gray we are now able to see much more clearly the fiber tracts of interest. In addition, the light pen feature which allows us electronically to blow up certain parts of the section allows even more careful investigation. Another interesting transformation is that of contouring, or doubling or tripling the gray scale in the original image. Although it is difficult to say exactly what contouring does, since the shades of gray represented on the oscilloscope are not true, one feature is the compensation by the computer for the non-linearity of the receptors in the human eye. In other words, as one reaches more and more intense levels of shading the human eye can no longer differentiate between gray shades as easily as at less intense shades. Thus contouring or sliding these shades of gray down the scale to the less intense range allows one to see much more detail. This particular feature may prove extremely valuable in looking at fine structures which otherwise gets hidden in too much signal.

Another application would be the development of a very accurate stereotaxic atlas from a brain section such as this. Stereotaxic atlases are essentially maps of the brains used by biologists and neurosurgeons in performing surgery. They are characterizations of the brain made section for section and accurately measured along XYZ axes to allow proper implantation of electrodes in certain structures. For example, if one would want to place an electrode in a certain part of a monkey brain, one would go to a monkey atlas for that species, look up its XYZ coodinante, place a stereotaxic grid on the monkeys' head which would be zeroed to some particular reference point and then eventually place the electrode at the coordinates found in the atlas. The characterization of the stereotaxic atlas has always been done by neuroanatomists, contouring the different shades of gray found on the monkey slide by eye. However, if we now scan this slide into the computer and pick the gray level which differentiate the boundaries of these various structures we could have the computer make the stereotaxic atlas much more accurately than the neuroanatomist. In addition, since all brains are not uniform the ability of the computer to make a stereotaxic atlas which could then be scaled up or down according to some measurement made on the particular brain being looked at is of extreme importance. In addition, the scanning of X-rays of electrode placements in order to have the computer measure the position of the electorde on the X-ray in relation to certain fiduciary marks and correlate the placement with the target siet in the stereotaxic atlas would be invaluable.

Another field of medicine and biology where scanning has equal importance is the field of radiology. It is conceivable that tumors and other pathology have unique densities which the computer could look for on a scanned image or enhance for the radiologist to see. The same is true of looking at pathological sections, particularly cytological smears. Cells are usually distinguished by their size, shape and characteristics of their boundry. These characteristics could be fed into the computer and refined until the computer although not able to diagnose disease accurately could certainly point out any pathology present on a certain slide in order to have a human look further into whether the report is really true or not.

In conclusion, I would like to discuss the possibility of using these techniques to model a brain in a computer three-dimensionally. It is now possible to scan at resolutions of approximately 35 micron. This means that a monkey brain section could be determined by approximately 1 million points. If we now take a series of frontal sections of monkey, cut at 35 micron and scan each one into the machine serially from the front of the brain to the back we would essentially have stored away approximately 1000 images

for this one brain. However, if we now think of this image stored away as a three dimensional matrix we can now think about rotating this matrix in any conceivable way and pull out sections which were not frontal but orthoginal to the original plane of scanning, or a diagonal, or any angle the investigator would desire. This would be of extreme significance in studies of neuroanatomy. At the present the neuroanatomist looks at fibers serially from section to section and tries to follow them through the brain in this discrete way. With the ability to now cut the brain up, scan it into the machine, put it all back together, and then recut it at any angle he desired the neuroanatomist could follow structures from beginning to end independent of their angle. In addition, to this type of storage for neuroanatomical structures the application for neurosurgery and proper placement of electrodes during procedures of this type will also have extreme significance in furthering man's understanding of the brain, how it works and how it is put together.

HIGH PRECISION CRT SCANNING SYSTEM*

C.A. Bordner, Jr.†, A.E. Brenner, P. deBruyne,
B.J. Reuter, and D. Rudnick
Harvard University
Cambridge, Massachusetts

Abstract

A high-precision, relatively inexpensive CRT scanning system controlled by a PDP-1 computer with a resolution capability of approximately 1 part in 30,000 has been developed. Although primarily designed for the automatic scanning and measuring of photographs taken of spark chambers used in high-energy physics experiments, it has wide range capabilities. The method and details of electronic implementation, test measurements, and the software required will be described.

System Description

We describe here a precision programmed CRT scanning system designed primarily for scanning and measuring of wide gap spark chamber photographs. Conventional spark chambers have found extensive use in high energy physics experiments for the past few years and continue to do so today. Now wide gap spark chambers are beginning to be used in addition for the same purposes. These chambers are usually parallel plate capacitors with surface areas up to several square meters and with gaps up to one half meter containing neon or a neon-helium gas mixture. The application of a fast rise high voltage pulse of several thousand volts/cm. causes electrical breakdown which will track with great precision the path of an ionizing particle traversing the chamber. Precision measurements of stereo photographs taken of the subsequent discharge give detailed information regarding the direction and energy of the particles traversing the chamber. During the course of a single experiment normally the number of photographs taken ranges in the hundreds of thousands. Figure 1 illustrates a typical wide gap spark chamber photograph.

To scan and measure photographs of this type rapidly and to the required resolution of 1 part in 30,000 (equivalent to about 2 microns on 70mm film), we have developed some comparatively inexpensive hardware. The system, called SPASM, for Spark-Chamber Precision Automatic Scanning and Measuring, makes use of some ideas introduced in existing devices designed for automatic measuring of bubble chamber photographs [1,2] and also the low resolution scanning and measuring of conventional spark chamber photographs [3]. The complete system consists of a precision cathode ray display tube, an optical system which images the light produced on the CRT face onto the film, a detector a film transport and the SPASM logic unit. It is controlled by a time-shared PDP1 computer [4] with a total memory of 32,000 words. There are 4096 x 4096 addressable points on the CRT face which define the starting point of a short unblanked vector which is generated during a fixed time interval. The components of the vector $\int x$ and $\int y$ may both simultaneously range from 0 to ± 2mm in 31 micron increments. The 2mm sweep is used in the scanning mode

and the shorter sweeps - primarily the 125 micron sweep - are used for precision measuring.

A programmable discriminator (8 levels) defines the threshold for information signals, or "hits". The $\int x$ and $\int y$ sweep amplitude buffers and direction flags, the discriminator level buffer and a high voltage enabling flag are all part of an Instruction Register, which is set by a single IOT strobe. The format for this word is given in Fig. 2a. Every time a display instruction is executed by the computer, the vector is plotted, starting at the coordinates set in x and y buffers and in a direction given by the value in the $\int x$ and $\int y$ buffers. During the time of the sweep an interpolation scaler counting at 10 MHz divides the sweep length into 128 elements. It is this interpolation procedure, while maintaining the required short term stability, which allows measurements with a resolution an order of magnitude better than the 4096 addressable points of the display.

In addition to the interpolation count of the center of the hit, the hit buffers contain the width of the absorbing region above the discrimination level in interpolation count units, the peak brightness of the hit in discriminator level units, as well as overflow flags as indicated in Fig. 2b. A 1 in bit 0 incidates an additional hit buffer is filled and a 1 in bit 1 indicates an incomplete hit measurement. After each display the program reads the hit buffers to see if there have been any hits. Successive reads bring in successive hits (up to three) registered in a single sweep.

Various tests of the performance of the hardware have been made and these indicate that the system indeed performs at the required level. The short term stability is demonstrated in Fig. 3, which is a display of the interpolation count for 1000 sweeps across a grid line with the total sweep length set at a 125 microns. The scale in this figure is 1 micron / division indicating a stability of approximately ± 1 micron. Similar tests spaced ~5 minutes apart indicate drifts over that period of time of about 1 micron.

Although the magnetic hysteresis of the deflection system is down at the 1 micron level, the settling time for a large deflection due to induced currents in nearby conducting elements is a more serious problem, necessitating delays of up to two milliseconds. Figure 4 illustrates this effect with the plotting of a point moved from the top of the CRT to the bottom and then successively plotted in place. This additional required time delay for large excursions does not seriously affect the overall scanning time.

Much of the basic software packages required to run the system have been implimented. These include calibration routines for the vector sweeps and routines to analyze the distortions inherent in the cathode ray tube. Track following and measuring routines have been written, which follow both straight and curve tracks. The remaining routines required to make the system operational are presently being developed.

Due to the inherent efficiency of the vector scan in the scanning mode and the continuous nature of the tracks, only a few full scan lines must be made across each spark chamber image. The time required to perform a single complete scanning sweep across a 70mm wide image is between 5 and 10 msec. With several scanning sweeps across each chamber view and the precision measuring of these tracks, the estimated time is of the order of 100 msec per chamber stereo pair.

The cost of the system is also quite attractive. The CRT tube and coils together cost just over $15,000. The SPASM logic itself, apart from engineering, is about $5000, and the required power supplies add an additional $5000. The optics, film transport and mounting hardware cost approximately $10,000. Finally, the display logic and amplifiers--essentially a standard option on the PDP1--add an additional $15,000 to the cost of the system. Thus the total estimated cost, exclusive of the computer itself, is approximately $50,000.

Finally, we should point out that although the system has been designed with spark chamber photograph scanning as a goal, the hardware possesses a quite general purpose capability both for the rapid scanning of material as well as for precision measurement.

Hardware

Vector Generation

SPASM is capable of a short sweep in any direction from any of 4096 x 4096 positions on the 5 inch C.R.T. A slightly modified D.E.C. type 30 display amplifier is used together with improved 12 bit digital to analog converters. The spot size obtained from the tube, a Dumont KC 2347 PBQ, with CELCO deflection coils, [5] is about 30 microns. The spot is imaged with a 1:1 magnification on the film using an f1.9 Elgeet lens with a focal length of 15 cm. There are two sets of coils for deflecting the spot, those for the sweep starting position and those used for executing a short vector sweep from that point. Before the sweep command is given the main deflection buffers are filled with the X and Y position words, each of 12 bits, and the Instruction Register is set controlling the SPASM vector generation.

The Absorption Detector and Sweep Generator

Figure 5 is a block diagram of the system. After the sweep has been triggered by the computer the beam is unblanked and the vector is plotted. A brightness reference signal is obtained using a photomultiplier viewing the light from a 25 micron thick half silvered mylar mirror placed at a 45 degree angle behind the objective lens. This brightness reference signal is subtracted from the absorption signal of a photomultiplier placed behind the film. The photomultiplier voltages are adjusted so that the combined output is about zero at positions of absorption (black film). We hence have an output which primarily depends on absorption in the film and the influence of brightness variations or dirt on the screen is very small. The subtraction is carried out by connecting the dynode of the reference PM to the anode of the absorption PM and summing the currents in a grounded base amplifier.

When moving to various positions on the film frame, the fraction of light reaching one photomultiplier changes with respect to the fraction reaching the other one. This is due not only to data on the film but also to changes in background density and imperfections in the light gathering system. A simple difference signal is thus not always a good measure of data absorbtion.

With spark chamber pictures it is better to refer this difference to one obtained in the immediate vicinity, showing a minimum of absorption. In practice this can be accomplished by a peak stretcher which stores a voltage corresponding to the peak transmission of previously scanned points. The absorption detector logic[6] of Fig. 6 shows how a threshold is then set at some fraction of this voltage and the instantaneous value of the absorption signal continuously compared with this level in a linear discriminating amplifier. The output of this amplifier is compared with a sensitivity level previously set up in the Instruction Register. The output of this last discriminator is a logic level which defines a hit.

In order to sample the peak brightness before the active part of the sweep starts, the spot is actually backed off 10% of the sweep length and then allowed to sweep forward through the initial position, up to its maximum value, in one continuous sweep lasting about 18 microseconds. The beam is unblanked before the position of zero deflection is passed so that the peak detector will retrieve a reference peak brightness voltage before the active part of the sweep begins.

The above system allows a uniform high sensitivity over the whole film in presence of large CRT brightness variations varying film densities and unequal lens vignetting of the two photomultiplier channels.

Hit Buffers

During the active part of the sweep which lasts for 12.8 microseconds a 10MHz clock counts into a 7 bit scaler.

When the discriminator output starts to show a film absorption, the contents of the scaler are transferred into a half speed scaler counting at 5 MHz. During the time that there is an absorption level a width scaler accumulates a number of counts at 10 MHz. When the spot has finally passed the absorption region the width and half speed scalers will stop. The width scaler will then register a number depending on the sweep velocity and width of the mark on the film. The half speed scaler will register the center of this mark with respect to the starting position, measured in clock counts. Figure 7 illustrates a 2mm sweep ramp resulting in 3 hits. It also shows the discriminator output and the strobe pulses which activate the half speed scaler.

As the sweep length is known and corresponds to 128 clock counts subsequent retrieval of the position can be easily handled by the computer program. The degree of absorption is also recorded for each hit. An analog to digital time gate generator is connected to the output of the discrimination amplifier and provides an output of 0 to 0.8 microseconds duration depending on the level of absorption (brightness of the spark). A 3 bit brightness counter is allowed to count at 10 MHz for this time allowing 8 brightness levels to be detected.

The half speed counter, width and brightness counter act like a buffer after the absorption level has ended. The contents of this buffer are allowed to transfer into a second and subsequently into a third buffer provided they were not already in use. A total of 3 separate absorptions may thus be recorded in a single vector sweep. They form a queue in the buffer and are read out by the computer one at a time via cable drivers connected to buffer 3. Each buffer has a control flip flop showing whether it is filled or not. When the first word has been transferred, the computer I.O.T. return pulse will clear buffer 3 allowing the information in buffer 2 to take its place. This may be repeated once more if there were 3 absorptions during that sweep allowing all three words to be transferred.

Calibration

The calibration of the D to A convertor of the major deflection system is carried out by adjusting the weight potentiometers of the X and Y address bits in such a way that each of the 4096 increments in X and Y are of equal magnitude. The greatest errors are of course to be found in the most significant bit. By incrementing the starting position in unit steps, around the center position, we can observe whether these steps are of the same size to better than 1.0 micron. The other weight potentiometers are adjusted similarly.

We can calibrate the sweep velocities by sweeping across 2 absorptions of known distance apart. The $\int X$ and $\int Y$ velocities are adjusted individually on the 1 or 2 mm sweep length positions. The positions of zero velocity are adjusted using the operational amplifier offset adjustment. Intermediate velocities will be correct because a high quality D to A converter is used.

The duration and velocity of the back off sweep are first adjusted to their approximate correct values by inspecting the waveform of the sweep current. Final adjustments are made by scanning a graticule with such sweep amplitude that 2 hits are obtained. After reversing the sweep direction using computer controlled relays, starting at the same x,y position two other hits are obtained. The number of clock counts for the first hit recorded in the two directions are added, and the sum compared with the difference between the two hits obtained in one direction. It may hence be seen whether the starting position of the active portion of the sweep is independent of direction. Final adjustments to the "back up" sweep velocity may be made to assure the proper starting position.

References

1. P.V.C. Hough and B.W. Powell. "A method for fast analysis of bubble chamber pictures", Nuovo Cimento, 18, pp. 1184-91, August 1960.

2. B.F. Wadsworth, "PEPR A developmental system for rapid processing of bubble chamber film". Conference on Filmed Data and Computers. June 1966, Boston, Mass., Proc. Soc. Photogr. Instrum. Engrs, IX, pp.1-14.
I.A. Pless et al., "A precision encoding and pattern recognition system (PEPR)", Proceedings, XIIth International Conference on High Energy Physics, Dubna, U.S.S.R., 1964.

3. H. Rudloe, M. Deutsch and T. Merill, "PIP. A photo interpretixe program for the analysis of spark chamber data", Commun. Assoc. Computing Machinery, 6, No. 6, p. 332, June 1963.

   M. Deutsch. "A spark chamber automatic scanning system", Proceedings of the Conference on Photon-interactions in the BeV Energy Range, Massachusetts Institute of Technology and Laboratory of Nuclear Science, 1963. (Unpublished)

4. A.E. Brenner, Time-Shared Multi-Experiment Use of a Small Computer, IEEE Transactions on Nuclear Science, NS-12 241 (1965)

5. Celco; Constantine Engineering Lab., Co., 70 Constantine Drive, Mahwah, N.J. This includes the HADS428P560 Deflectron and associated mounts and coils.

6. P. de Bruyne, "A light absorption detector for a computer controlled film scanner", The Radio and Electronic Engineer, 29, No. 5, p. 325, May 1965.

* Supported in part by the U.S. Atomic Energy Commission

† Present address: Department of Physics, Colorado College, Colorado Springs, Colo.

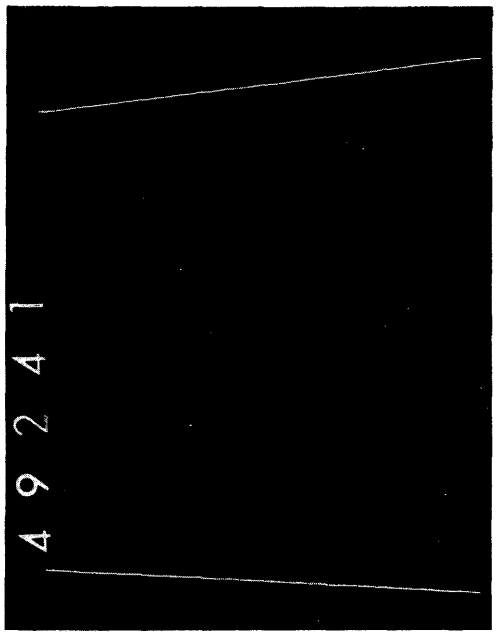


Figure 1 A typical wide gap spark chamber photograph with two sparks. The sparks are 10 inches long.

SPASM I-O WORDS

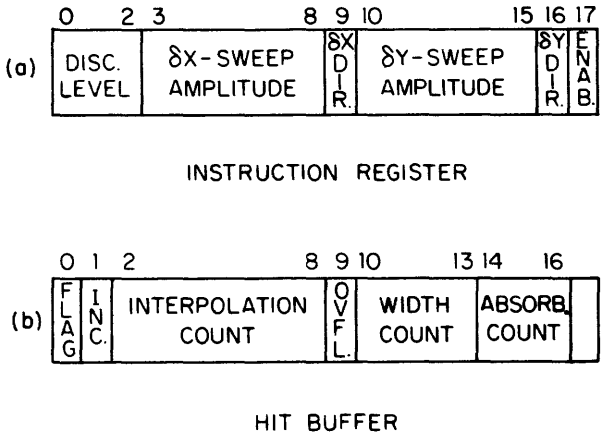


Figure 2 a. Instruction Register format

b. Hit Buffer format

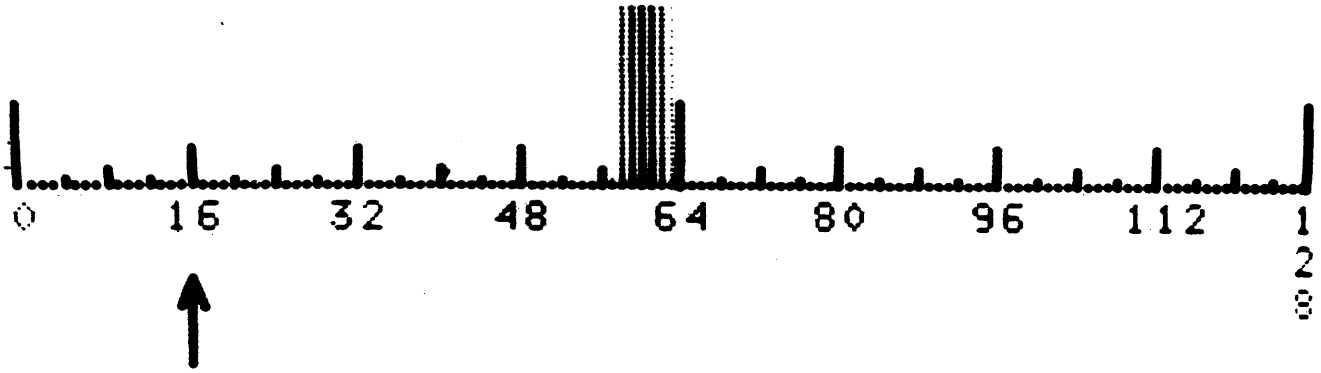


Figure 3 Interpolation count of 1000 scans of a grid line. Each unit represents 1 micron on film.

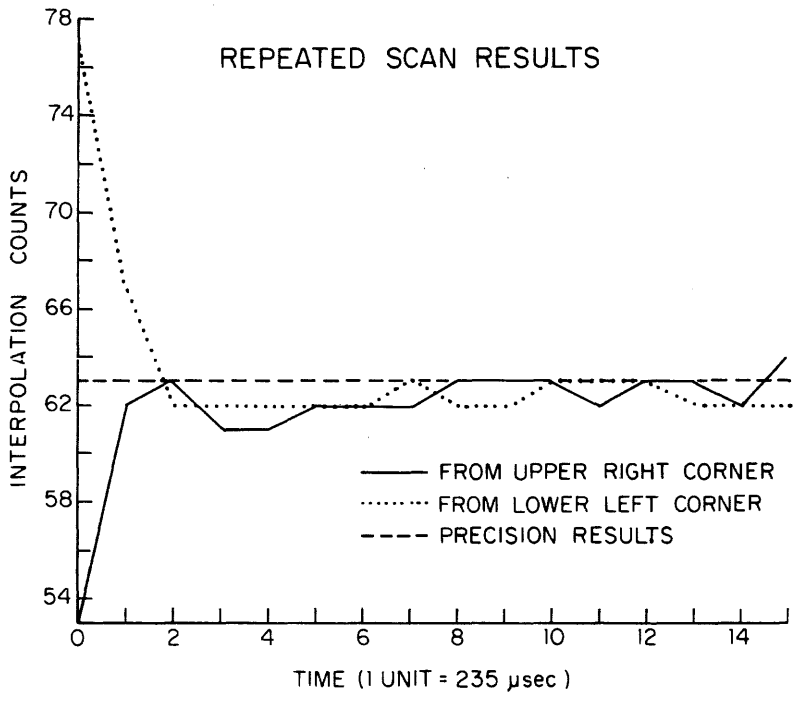


Figure 4 Short term settling time study for large scope coordinate changes. Each interpolation count is 4 microns on film.

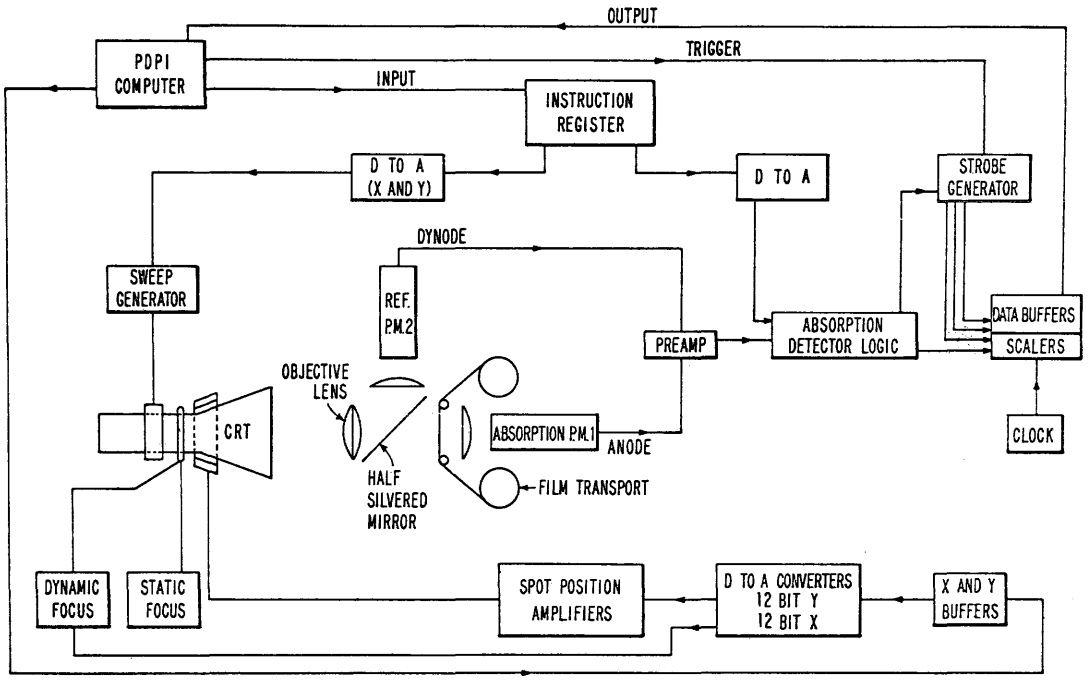


Figure 5 Block diagram of the SPASM system configuration.

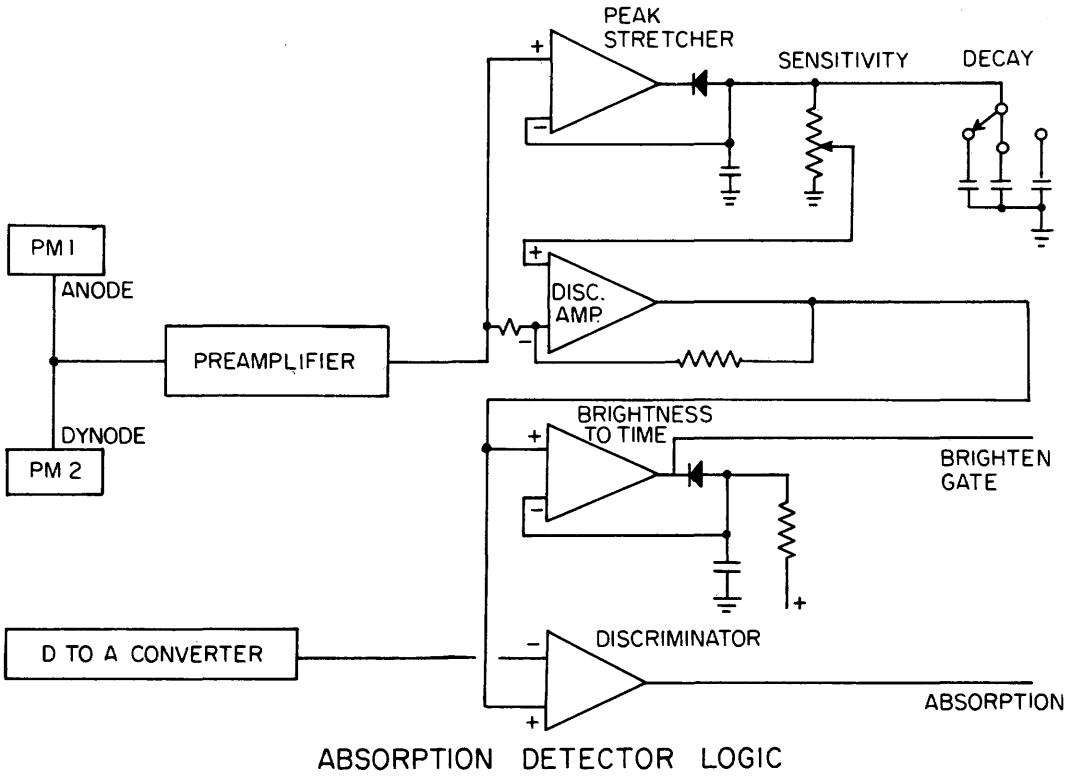


Figure 6 A fraction of the slowly decaying output of a peak stretching amplifier is compared with its input in a linear descriminating amplifier. The output of this is compared with a programmed level. It is also converted in the brightness to time converter to a gating level for the brightness scaler.

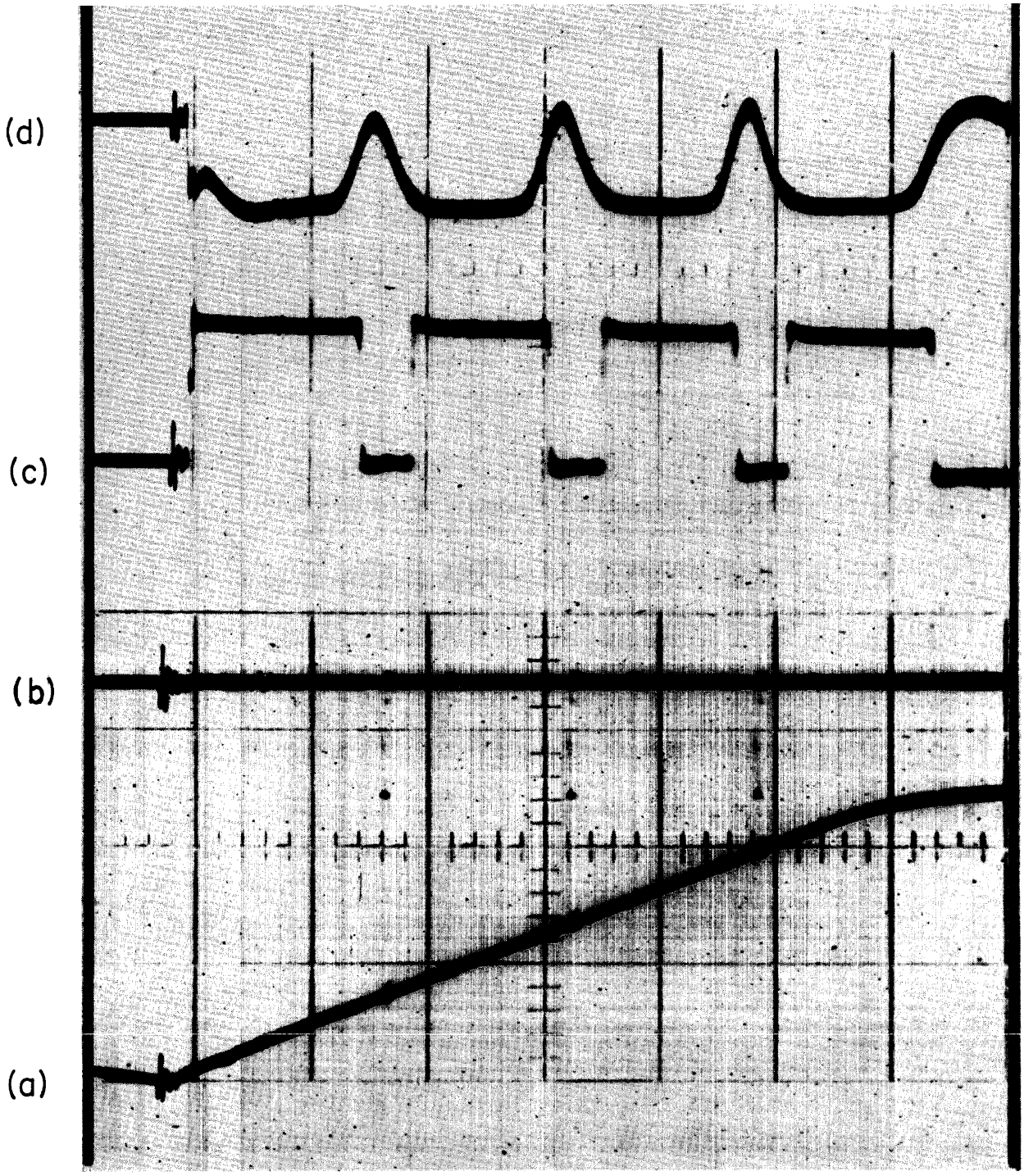


Figure 7 a) The current through the sweep deflection coils. 0.5 amp/cm vertical, and 2.6 microseconds/cm. horizontal scale.

b) Strobe pulses transferring a hit into the half speed scaler; 2V/cm vertical scale.

c) Output absorbtion detector; 2V/cm vertical scale.

d) Output photomultiplier summing amplifier; 1V/cm vertical scale.

A GENERAL APPROACH FOR ACCOMPLISHING EFFICIENT DISPLAY SOFTWARE

J. Richard Wright
Wolf Research and Development Corporation
West Concord, Mass.

Abstract

A specific requirement is used to illustrate the effieient methods for the development of display software. These will be developed in the areas of display byte structure, problem definition, display areas and color selections. This work was done under Air Force Contract No. AF 19(628)-67C0095.

Introduction

A specific problem is used to illustrate the methods developed in this paper. These methods are a direct result of over a year's work with a group of programmers and analysts in the DX-1 Laboratory of Air Force Cambridge Research Laboratories and 10 years previous computing experience. Graphics were generated for the first typewriters, teletypes and line printers (Christmas trees, girls' figures, etc.). Some were generated on the computer and punched on cards for the line printer where the display was produced. Many years ago subroutines to plot up to 8 variables on a printer were available in the SHARE library. Several years ago we used the technique of plotting curves on the printer as bar graphs at time increments. To view them, we flipped through the pages like the old-time peep shows to simulate movies. The Stromberg-Carlson 4020 is not new. The Whirlwind computer had CRT's 15 years ago. Today at the DX-1 Laboratory we have a 5 inch precision scope, two 16 inch scopes and a 21 inch color CRT. We have two PDP-1 processors with information exchange, one 1.2 million bit drum and three magnetic tapes on each, one tape being switchable. The processors have an 18 bit word; one processor has 8K memory and the other 12K memory. We also have an experimental graphic console or consoles, depending on how they are plugged in. Until recently they have not been used to the fullest extent graphically. The primary purpose of the processors was to test and evaluate techniques in the reduction of bandwidth in communications. The graphics capability was implemented primarily as a tool to aid the experimenter in the expeditious evaluation of the computed data. The investigations of communications bandwidth reduction techniques have indicated areas that can profitably be studied in more detail. The use of graphics has become more important as these investigations are continued since real, as well as simulated, data is being used.

One area of investigation requires the display of a sentence or more of speech data at a time. We wanted to show the speech data, and after compression, the resynthesized data on the scope at the same time. We wanted to see it as nearly as possible in real time. This is basically the problem that motivated us to develop this present graphic software for our computer configuration.

Graphic software must be considered the same as any other software. This implies that good methods and standards for other classes of software will apply to graphic software. From my experience, I must say that the most important procedures that hold absolutely true for graphic software are: (1) good problem definition, and (2) good documentation.

There is no intent here to minimize a third one, good programming, but this is a topic in itself. Consequently, I have not concerned myself with the human element at this time.

Problem Definition

The question has been asked many times, what is meant by good problem definition? The answers vary with the type of processing required. However, in software generation it is the programming that is the problem. The programming is the most important part of the problem and the best way to attack the problem definition for any

non-trivial software is from the standpoint of programming.

Our problem, as it was originally stated, was to display a sentence of speech data and the same amount of resynthesized data on a color scope. It was desirable to do this in real time and also be able to play it back at slower speeds. The scope is an RCA TV color tube with logic added and is drum driven. It can display 512 × 512 points on a 12 inch by 12 inch useable screen face. Using vocoder data we represent speech graphically by its amplitude on the 18 frequency channels plus average amplitude, voicing and pitch. This amplitude value would be represented by different colors. This information was recorded at 50 cycles per second. The problem definition was originally divided into logical blocks from the programming point of view on the basis of extensive previous experience. This problem definition was divided as follows.

1. Multiple displays
2. Movement, real time
3. Multiple input
4. Data structure
5. Color tables

After preliminary analysis, it was further divided as follows.

1. Display byte structure
2. Display format
3. Movement
4. Color
5. Data structure
6. Input conversion

The approach to the solution of the problem definition problem was first the assignment of priorities. Top priority was assigned to display byte structure and to display arrangement. These could be solved separately, but they interact.

The display format was to be one of those shown in Figures 1 and 2. At this point, a number of questions developed. The first question, did the scope, which was originally a TV picture tube, have different characteristics in horizontal movement from its characteristics in vertical movement? A program was written to test this and no perceptual difference was found.

The second question, was there sufficient drum storage to drive a display that could cover a large area of the scopeface with different colors? A program was written to test the maximum display capability. This program had the capability of displaying from one display byte structure at a time, but under sense switch control would change the display byte structure. Then when a good color change and a large number of points were displayed compactly, the byte structure could be dumped. A "Joss" type computer program to simulate the analog beam drivers was used to develop graphs of the effects of the incremental logic. The combined results indicated that the beam must be kept moving and turned off and on, on the fly, in order to get maximum coverage. It also gave us the best byte structure for doing this.

Now the information was at hand to make a decision between Figures 1 and 2. These show three similar areas on the scope at one time. The first area displays vocoder data, the second area displays resynthesized data, and the third area displays the transformation vectors. Either figure was equally difficult based on drum logic, color tube dynamic characteristics, and storage availability. The investigator in charge decided on Figure 1 as a personal preference.

The next problem to be solved was the drum driving pattern needed to generate this display. We have a choice of four patterns (see Figures 3-6). Figures 5 and 6 were discarded on the basis of previous experience. They require inefficient beam movement which leads to an excessive number of display bytes. They would have simplified programming but would have drastically reduced the amount of information that could be displayed. This left the choice between Figures 3 and 4. Figure 4 has advantages for real time input, in that each vector in each of three displays represents the same instant in time and would be received in that way. However, the alignment of vectors in each row was much more difficult than would at first be suspected. Programs were written to test these two patterns. They showed that it didn't require any more display bytes to use Figure 3 over Figure 4. Moreover, it was easier to calculate the slowdown and to keep the vectors aligned than it was to calculate the speed variations necessary to maintain an apparent constant speed and alignment for longer vectors. Therefore, the Figure 3 pattern was chosen for the display format.

While this work was proceeding, we attacked another problem that was independent of the first ones; namely, color. We programmed test displays of several colors that could be varied at the color console for determining the final color scale to be used. Apparent color is dependent on adjacent color, so that this testing had to be done using

several colors at a time. At the present time we have three different color tables for the displays and a fourth color table which is symmetric about zero for use with correlation matrices.

Early in the programming effort we realized the need for a new utility program. The color display byte technique, and the format problem, all had a commom programming problem. Each drum sector could be typed out in about 7 minutes and punched out on paper tape in about 4 minutes. The punched output still had to be typed on an off-line typewriter. Since we debugged on line and often needed to see more than one drum sector, both methods took too long. This problem was a programming problem and was solved by using some knowledge gained from the first part of our work. We wrote a program to display one whole sector of 334 words of 6 octal characters on the black and white scope. This took 14 seconds for the first sector to be displayed since one head group of 20 sectors was saved on tape so that the drum could be restored to its original condition. After the first sector had been displayed, we could look at another in less than 2 seconds, except where tape movement was involved and then it only took about 5 seconds. This one tool was a great saving on our debugging time.

Basic data structure was decided upon for ease of programming. Data input was considered to be strings of numbers representing a vector, either in floating point or fixed point from an external source and each vector was to be preceded by a sequentially numbered ID. Data structure was considered next, but input conversion routines were left until last. Many data structures were studied, as well as ID tables, pointers, rings, etc., to see if there were any new ideas that would help us. It would have been convenient to have had enough core storage available for the whole display in drum bytes. However, this would have required over 13,000 words, so the total display is assembled in four parts. The previous statement about core availability also applies to the data storage from which the display is created. The number of discrete points possible in the display seemed to be about 6,400. Input storage was only available for 4,000 fixed point or 2,000 floating point words. Therefore, input was converted to fixed point and stored in a list with an end-of-list indicator word and could be segmented. Also necessary was the input characteristics for each display which included input record size, number of words and format. We also needed the following parameters for each display: the number of points/line; number of lines/display; space between lines, and color table to be used. The program would then determine the starting location for each display and all the other constants and variables.

For real time display, ease of programming was again the determining factor. A simple scheme was devised--that part of the total data that had just been displayed would be saved on a drum head group with one drum write command. Next time this data would be read back into core and n vectors of new data tacked onto the end of it. The display routine redisplays the data starting n vectors from the beginning. Then this data is written back onto the same head group of the drum, starting n vectors from the beginning.

Documentation

Documentation of graphic software must be complete. Each of the test programs was documented and so were all the changes as they were made and what effect they had. Reference to these program documents saved 6-12 man weeks of unnecessary coding and testing. Most of the ideas brought up were found to have been investigated during these test programs and we were able to say why they would or would not work. The best way to accomplish this kind of documentation is to collect and review it each week.

Miscellaneous

I have tried to show that programming is the most important aspect of software and therefore the software design should be approached from the programming point of view.

Other procedures that were used in this effort were: (1) complete problem definition, (2) all possible program functions were made subroutines or a macros, (3) almost all variables were kept in a common storage, (4) week-by-week documentation.

Another important method not previously mentioned is the testing of all subroutines by themselves, since subroutines were often nested to several levels. This is both a challenge and an opportunity to greatly improve debugging efficiency.

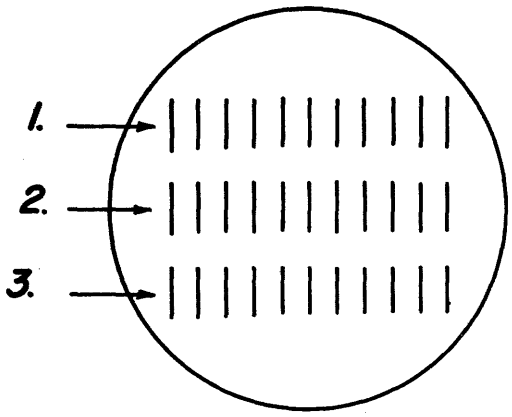


FIGURE 1

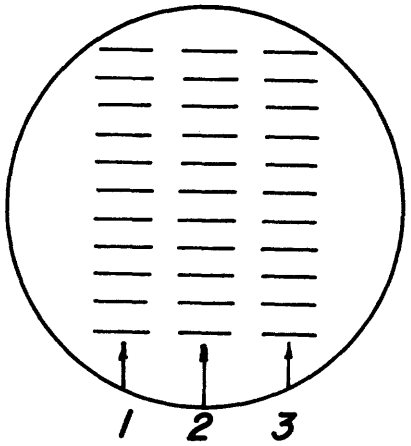


FIGURE 2

1. Vocoder raw data. *2. Resynthesized data.* *3. Transform vector.*

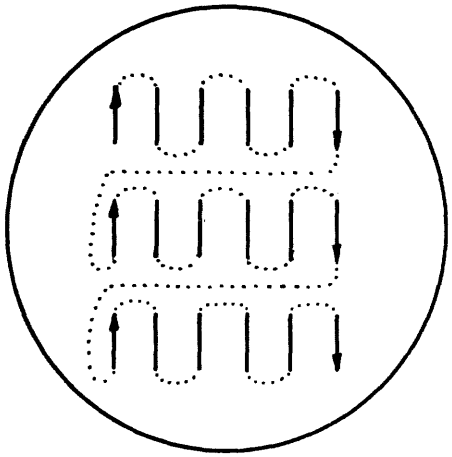


FIGURE 3

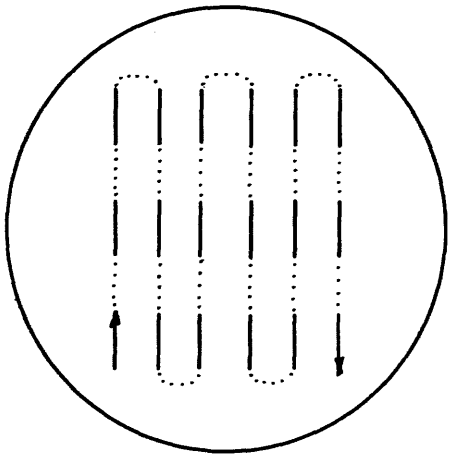


FIGURE 4

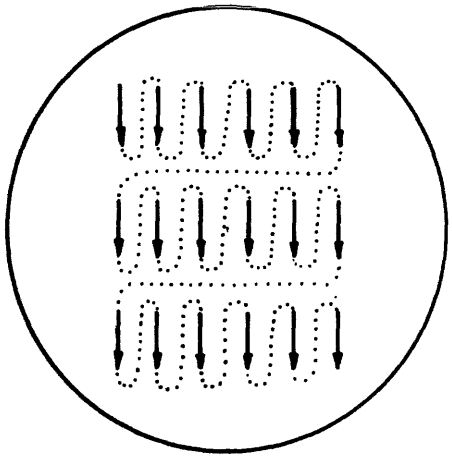


FIGURE 5

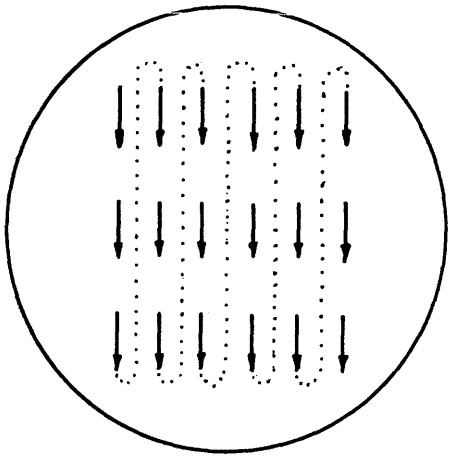


FIGURE 6

TracD - AN EXPERIMENTAL GRAPHIC PROGRAMMING LANGUAGE

Barry Wessler
Digital Equipment Corporation
Maynard, Massachusetts

Abstract

A graphic programming language was designed and implemented on a DEC Type 338 Programmed Buffered Display system. A host language, TRAC, was chosen and extended to include a set of graphic primitives. The language retains both data and programs in textual form and executes them interpretively. The language was an attempt to create a system-independent structure for graphic processing.

Graphic systems can be broken into three logical groups: the display programming language, the user system, and the analysis program. The programming language is usually a higher-level language with special commands added to communicate with the display. The language used varies depending on the computer system. The Graphic I console at Bell Laboratories uses a language called GRIN whose basic structure looks like FORTRAN. The ESL Display at Project Mac uses the B-core system which is written in AED.

The user system is a program written in the display programming language. It is the program that listens to the user requests, creates the picture data base, and maintains the display. The user system may be either designed for a specific application such as Sketchpad. The former has the advantage of being problem specific and, therefore, being more simple to design and program. The data base can also be tailored to the problem at hand. The latter has the advantage of being available for the user who does not want to bother with programming the display. The generalized system, however, tends to be a very large program. For example, the Digigraphic System requires 25,000 words of storage for the user system program. The designers used a memory overlay technique with a disk as the mass storage element in order to conserve the amount of core memory required.

The analysis program computes the data base created by the user system. It may output the solution on the display through the user system, or on another external device. The analysis program can be written in the display programming language or another language. The choice of language will be dictated by the capabilities of the display language and the hardware configuration.

Specifications For A Display Language

The following nine specifications were decided on as design criteria for a graphic language. The first six are criteria for the host language, and the last three are for the graphic primitives:

1. The language must be able to communicate to the user in real time. It must have the facility to accept input from the user through a typewriter in any format and output necessary information to him.

2. The implementation of the language must be compact. Due to the limited amount of memory space in the PDP-8, most languages had to be discarded. The FORTRAN system provided with the machine, for instance, did not lend itself to expansion and did not allow room for enough statements to do meaningful graphics.

3. The language must be suited to tasks such as storing block data,

naming items, searching and calling back the data. The graphic elements would be stored as data base within the language as blocks of information pertaining to element. This data would have to be stored dynamically as the user developed the picture. The data must be labeled so that it could be called back by the user for modification or deletion.

4. The language must have elementary arithmetic and Boolean capabilities. In graphics the value of sophisticated arithmetic functions, such as floating-point operations, is not great for most applications envisioned. The ability to add such capabilities, should the need arise, is important.

5. The language need not be capable of analyzing the data except where it is necessary for the organization and manipulation of the data base. The more powerful the language, the less reliance it will have on the large computer. The problem of memory space for the SCD programs is again the limiting factor. If the language is more powerful, it will also undoubtedly take up more room in the SCD leaving less room for programs.

6. The language must be capable of communicating with other processors to send the data base and to accept new programs.

7. The language must be capable of interpreting the data base into a display file in real time. The display file is the list of instructions that the display device executes to produce the picture on the CRT. The display file must be updated each time the data base is changed so that the picture corresponds to the data.

This job could most efficiently be done by making the data base executable functions within the language that produced the proper display-file code. The data base need only be re-executed when the data is changed.

8. The language must handle display-oriented inputs from the light pen, push buttons, Rand Tablet, etc. Both light-pen functions, pointing (at an element) and positioning (using a tracking mechanism) must be implemented in the language.

9. Finally, the language should be machine independent. The language should be simple enough to be implemented on any general-purpose computer. The graphic primitives should also be kept simple and general so that they can produce a display file for any reasonable display device. The language should be externally similar independent of the implementation.

Choice of Language

The first three specifications suggested that a type of text editing language be used. The idea of a picture being equivalent to a string of text is not new although some researchers have argued that a two-dimensional representation is preferred. Editors are usually compact and well suited to tasks as storing and re-calling the data. They are usually lacking, however, in logical and arithmetic capabilities.

A more general text handling language was sought and two were found, SNOBOL and TRAC. The former, though logically more powerful than the latter, did not meet the second criteria of being compact. The latter, on the other hand, was already partially implemented on the PDP-8 in two thousand (octal) locations. TRAC was originally designed for communication to the user through the typewriter, so that the input-output problem was already solved.[1]

The language also lent itself well to the graphic extensions as primitive functions.

TRAC stores both its programs and data in a textual representation. The programs are therefore, machine independent because they are stored in the same format independent of the machine. The programs are executed interpretively and can create new textual forms which can be either data or new programs.

The TRAC language, meeting all the necessary specifications, was extended to include the graphic primitives and renamed TracD. This document

describes the work and the thought involved in that project.

Display-Oriented TRAC Primitives

The intent was to use the TRAC language as the basis for a display programming language. Therefore, the set of TRAC primitives was extended to include the facility to communicate with the display device. The display primitives were designed to be simple enough so that they could be implemented on a simple display system. This was done in order to maintain the concept of device independence stated above. There are occasional references to the particular hardware used in order to clarify a specific function.

The display is both an input and an output device. The output is the information presented on the cathode ray tube (CRT). In order to present the information, a file of display instructions are interpreted by the display logic to draw lines, points, and characters, to do control functions like skips, transfers, subroutines (push, jump, and pop), and to modify registers. Display inputs are accepted through the light pen and the push-button box.

The new primitives added to the system are broken into three major categories:

1. Display Elements - Four primitives that create the display instructions to put information on the CRT.

2. Display-File Control - Four primitives that control the naming, skeletal production, and maintenance of the display files.

3. Display Input - Three primitives that provide information about the status of the light pen and push-button box.

Display Elements

There are four display-element primitives in the present implementation of TracD. Three of these are "Basic Elements" which produce the appropriate display code for its element. The elements are points (PT), line (LI), and text (TX). These elements may be combined to form a symbol that can be called later with the symbol (SY) primitive. A symbol may be composed of all four types of elements and may call any defined symbol except itself.

The set of elements chosen can be extended at a later date. Circles and arcs, which are simple to generate, may be added in the near future. Generalized second-order functions, ellipses, parabolas, etc., could also be added. The line, point, and circle elements could then be removed since they are special cases of the second-order curve.

Within the argument string of all the present elements, there is an argument reserved for naming that element. This argument is the last argument and is optional. The argument will be represented as NAME* in the function calls given below. NAME* is the value left when the light pen pointing primitive function is given and, therefore, may be a simple text string or another function.

Another entity common to all display elements is the x and y coordinate pair. It is used to indicate the starting point, origin, or end point of that element. The x coordinate preceeds the y and is separated by a comma; i.e., they are two separate arguments. The range of the coordinates are from 0 to 7777 octal, with (0 0) normally being the lower left corner. Only 2000 octal points are visible on the CRT at one time, but picture translation is possible.

POINT

(PT, x, y, TEXT, NAME*)

The PT element does not intensify a single point at the coordinate pair but rather puts up the string TEXT (the fourth argument, whatever the string happens to be). PT is, therefore, not designed for curve generation or the like but rather as a means of spotting a particular coordinate pair. This type of function is necessary for making logical connections in a topological structure.

LINE

:(LI, x_1, y_1, x_2, y_2, NAME*)

Straight-line drawing capabilities are called by the LI primitive. The two coordinate pairs specify the starting point (x_1, y_1) and the end point (x_2, y_2) of the line. The display hardware automatically generates lines, given the incremental movements delta x and delta y in the display list. The routine that executes the LI primitive must therefore generate the display code to move the beam to the starting point (x_1, y_1) and then calculate the incremental vector:

$$\Delta x = x_2 - x$$

$$\Delta y = y_2 - y$$

and place it in the display list.

The above method of representing lines was chosen over the more natural (for the present hardware) incremental representation.

:(LI,ΔX,ΔY, NAME*)

The incremental representation was felt to be too sequence dependent. Deleting a single function could change the entire orientation of the picture. It was felt that the form required to compensate for line deletion in the incremental representation would be excessively complicated. The major advantage of the latter representation, the ability to relocate parts of the picture by changing a set point, is offered in the symbol function.

TEXT

:(TX, X, Y, STRING, NAME*)

This function provides the facility for displaying strings of alphanumeric characters. The (x,y) coordinate pair is the starting point of the original x, in order to format the text properly. There are 55 printing characters in the present display character generator, but the full 96 printing ascii set could be implemented with no difficulty.

SYMBOL

:(sy, x, y, NAME, NAME*)

The SYMBOL primitive causes the display to execute previously defined display code. The code was generated by calls to the elements PT, LI, TX, and SY, and defined with the name NAME. The details of the procedure for defining code as a symbol is given in the Display File Control section. The symbol is relocatable on the screen with its starting point being the x-y coordinate pair.

A symbol is considered local rather than global in nature. The internal composition is maintained only in the display file, not in the data base used by the TRAC forms.

There is no way of distinguishing different pieces of a single symbol. The light pen function will yield NAME* only, independent of which part of the symbol was pointed at and to what depth the sub-routining has gone.

The justification for this approach lies in the original reasons for having symbols. Symbols were designed to be collectors of unwanted data base information. Providing such a garbage collecting technique was considered a necessity because of the limited space available within the TRAC form storage area. It is also an asset because of the increase in system response time due to the removal of unnecessary information. If global symbols are required, it is possible to create such a structure within the TRAC framework.

Display File Control

The functions of the Display File Control primitives are to name, store, and delete symbols and to store and execute a picture. There are four primitives in this group: the first names and indicates the beginning of a symbol or a picture (PN); the second and third indicate the end of a symbol (FS or picture FW); and the fourth removes all symbols from storage (CE).

PICTURE NAME

:(PN, NAME)

This function has two arguments and null values. It creates the skeleton structure needed to start a display file. It enters NAME, in the appropriate place and sets the first pointer to the first location available for the display file. It also stops the display, if it is presently running, while the new display code is being generated. The function also checks all the symbols names presently in core. If the same name is found, TRAC will respond with an error code "S!" typed on the printer.

FINISH WORK AREA

:(FW)

The (FW) function indicates the end of element functions for the picture desired. It adds push jumps to two permanent sub-routines for cross-tracking and translation. It also adds a display jump to the beginning of work area. The display, which was turned off with the (PN) function, is reinitialized at the beginning of the work area.

FINISH SYMBOL

:(FS)

This function calculates and inserts in the display file a vector to return the x-y coordinates back to their original value. This makes the symbol closed; i.e., it has the same start and end point. A pop (return from sub-routine) command is also put into the display file. The work area pointers are changed, so that the work area begins after the new symbol.

CLEAR CORE

:(CE)

The Clear Core function removes all the symbols that have been defined. Initially, a selective symbol delete function was anticipated. Problems arose because of the interdependence of one symbol on another. It was felt that this function should be handled by a TRAC form. Therefore, to delete a single symbol all symbols desired should be redefined.

Display Input

The third class of primitives is concerned with introducing input data into TRAC. The two devices presently handled by TracD are the push bottons and the light pen. The light pen input is separated into two distinct functions; that of indicating a particular coordinate position on the screen (DC), and that of pointing at an element on the display (LP).

READ PUSH BUTTONS

:(RB)

In the present hardware, there are twelve push buttons with lights behind each indicating the present state of the button. Pressing the button complements its state. The function (RB) has the value of the state of the buttons represented as four octal digits.

DISPLAY COORDINATES

:(DC)

The value of the (DC) function is two four-digit octal numbers separated by a comma. The two numbers represent the present x and y coordinate position of the tracking cross. The cross is displayed whenever the display is running. If push button zero is on, the cross will stay under the center of the light pen. By moving the cross to the desired position and giving the (DC) function, the coordinate pair for that point can be got and placed directly in an element function.

LIGHT PEN

:(LP)

The light pen function has one argument and its value is the contents of the light pen name buffer (LPNB). The LPNB is always cleared after being read, so that its contents can only be read once. If the buffer is empty when the (LP) function is given, the function will have null value.

The purpose of the LPNB is to hold the light pen name, NAME*, of the last element seen by the light pen.

Evaluation

The advantages of TracD are dependent on the type of system the display is in. In the batched processing computer, the TracD system allows the user to develop the problem formulation. Only when the LC is needed will it be requested; most real-time operations will be handled by the SCD-TracD system. In the time-shared environment, the TracD system aids in reducing the amount of expensive computing time and improves response time to most real-time requests.

In a single computer system, it is possible to have more than one type of display system. For instance, at Project Mac, there are three different SCD systems; namely, the PDP-7 Kludge, the PDP-7-340 system at the Science Teaching Center, and the 338 system. Unfortunately, it is not possible to take programs written for one system and run them on another without re-programming. The concept of display compatibility is, therefore, very important if the art of computer graphics is to advance rapidly. Since display manufacturers are not likely to agree on a single, compatible instruction repertoire for their displays, it is desirable to develop a common representation or meta structure, for compatibility. One of the important design criteria is that it satisfies the compatibility constraint.

The TracD system satisfies the compatibility criteria by creating a textual representation of the picture. The text is stored as seven-bit ascii characters whether the SCD has twelve, sixteen, or eighteen-bit word length. The graphic elements, although similar externally, must produce the display code in the format for the particular display device. The element functions were kept simple to make the code generation a simple task in order to maintain the concept of device independence.

Functions in TracD are also stored in a textual representation. Functions or a system of functions written for one SCD can be run on any other that has TracD implemented. Thus, the dataphone communications in both directions; i.e., the functions from the LC and the data to the LC are compatible independent of the type of SCD used as a terminal. The LC, therefore, need not know which terminal it was talking to since they all look identical.

The concept of compatibility can be extrapolated one step further to include terminals not belonging to the original LC complex. Suppose a user at a display terminal at MIT's Lincoln Laboratories, normally connected to their IBM 360 model 67, wished to process graphic data with a program written for Project Mac's Multics System. He would have three alternatives: 1) He could take the program from Mac and re-program it for his machine; 2) He could come down to Mac and use a terminal here; or 3) If he had a TracD system for his terminal, he could call Multics and use the program at his own terminal. If the latter alternative were permitted to everyone, a user would not have to spend valuable time reconstructing an existing system.

REFERENCES

1. TRAC - A text handling language.

   Calvin N. Mooers & L. P. Deutsch
   20th National Conference, ACM
   August, 1965

2. New B-Core System for Programming the ESL Display.

   Charles Lang - Project Mac
   Memorandum Mac-M-216
   April, 1965

3. Operating Manual for the ESL Display Console.

   R. Stotz and J. Ward - Project
   Mac Memorandum Mac-M-217
   March, 1965

4. AED-0 Programming Manual,
   D. T. Ross

5. Sketchpad: Man-Machine Graphical Communication System.

   I. E. Sutherland
   AF/IPS Conf. Proc. 23, 1967

6. Programmed Buffered Display 338 Programming Manual

   Digital Equipment Corp.
   September, 1966

HELP - AN INTEGRATED DISPLAY SYSTEM FOR PROGRAM DEVELOPMENT

D. Friesen and J. Taylor
Massachusetts Institute of Technology
Cambridge, Massachusetts

Abstract

A set of display programs has been written as an integral part of the PEPR film scanning and measuring project. This display system, HELP, serves as a major tool in the development of other programs for the film scanning project. Included in HELP are real-time displays of program operations, and graphic displays of numerical data resulting from operations. Programmer interaction via light pen and teletype permits direct investigation and control of operating programs.

* This paper was not received for publication.

CHARACTERISTICS OF SPEECH DISPLAYED BY PDP-8

Morton M. Traum and Edward Della Torre
American-Standard Research Division
New Brunswick, New Jersey

Abstract

The frequency and amplitude characteristics of live or recorded speech are displayed as digital patterns at the teletypewriter. By interfacing an Audio Spectral Analyzer with the PDP-8, digital data produced while speaking into a microphone is loaded under program control into the computer memory.

A direct visual display at the teletypewriter of the stored data is possible, or reduction and calculation of the data according to software instruction may be performed prior to print out. The patterns produced serve as an important research tool for the analysis of speech.

Introduction

Conventionally, devices which analyze speech are limited by a priori hardware assumptions as to which characteristics of speech are significant for word recognition. Hardware reduction of speech information lowers high rates of data production to values within the capacity of the recording apparatus. The recorded data is often fed to large computers for compilation and analysis. Major hardware modifications are commonly required for altering the basic assumptions which are found unsuitable.

The approach taken here was to minimize a priori hardware limitations on the information obtained from speech, and to use the computer itself as a rapid data-logging device. In addition to displaying the stored information directly as recorded, the computer may be used to analyze and reduce the data according to arbitrary software instructions. The flexibility afforded by software logic makes the system an ideal research instrument for speech analysis.

It was found that the fast, but relatively small PDP-8 computer could be used satisfactorily to present the data in extremedetail or to derive any programmed function of the raw speech information. Analysis of the results showed patterns in the data which were characteristic of the spoken word but relatively independent of the speaker.

Hardware

Equipment Configuration

The basic hardware consists of a microphone and a magnetic tape recorder for voice input to a digital spectral analyzer, which operates on-line with a PDP-8. The computer is equipped with an Extended Arithmetic Element type 182 for rapid calculation. An ASR-35 teletypewriter and punch and a 3500 high speed paper tape reader are also interfaced with the computer. Figure 1 shows a block diagram of the equipment configuration.

Spectral Analyzer

Operation - The spectral analyzer uses a bank of sixteen frequency bandpass filters to sample the voice input every millisecond. During each sampling period the output of each filter is represented by one of eight discrete levels, 0 to 7. Thus three bits characterize the energy in each frequency band, or "channel". The group of sixteen channels is divided into four subgroups of four channels each, and, in consecutive 250 μsec subperiods, the output of each successive subgroup is available as a twelve-bit-binary number for transfer in parallel into the PDP-8 accumulator.

Control - Although the spectral analyzer is continuously sampling the input and making data available to the computer in digital form, transfer of information only occurs upon computer command. Associated with each four-channel subgroup is a special flag to indicate when that subgroup is ready for parallel transfer. There are four skip-on-flag instructions and two load instructions, which allow the computer to "or" with the contents of the accumulator, the contents of any selected subgroup during its respective 250 μsec availability period. The contents of any one channel of the subgroup may then be isolated by masking the other nine accumulator bits.

Software

Modification of PAL III

Using the EXPUNGE and FIXTAB facility of PAL III, the speech flag-sensing and channel-loading instructions were given mnemonics and incorporated into the permanent symbol table. Examples of the new instructions are: SSF1, skip on speech flag of subgroup 1; SL13, load the contents of speech subgroup 1 or 3. Thus we write

```
CLA
SSF1
JMP .-1
SL13
```

to load channels 1 through 4 of subgroup 1.

Real-Time to Paper Tape

A program has been written to sample the spectral analyzer output at the maximum rate until the computer memory is filled. The stored channel contents are then punched out on paper tape in a special binary format. The tape serves as a fixed record of the spectral information, and can be used in place of live speech data as a deterministic input to other speech analysis programs. New programs can be tested using this facility, and by comparing the results of paper tape input with those of recorded speech, the electronics can be calibrated and checked for repeatability. The data of up to one second of continuous speech can be transferred to tape, using 4K of memory.

The Phoneme Approach

Theory

The contents of the sixteen channels in any one millisecond can be considered to be the components of a position vector in a 16-dimensional space.

Extensive data analysis revealed that successive spectral analyzer outputs were mostly redundant. That is, the speech vector was observed to remain localized about various points for relatively long time intervals during the spoken word, while its transition between two stable points was smooth and rapid.

The redundancy in human speech allows for a more efficient means, than direct transfer from spectral analyzer to memory, of recording the channel contents as a function of time. By noting the points of stability and when and how long the speech vector resides at them, a significant data reduction is achieved without much loss in information. The points of stability are called phonemes, and the measure of localization is a distance function called the correlation difference.

Data reduction by the phoneme detection method does not invoke a priori assumptions as to the significant speech characteristics, since it follows from analysis of detailed unreduced data. Furthermore, since the method is entirely specified by software, the system is easily subject to variation.

Procedure

Under the control of a program entitled QPS No. 7 (The 7th in a series of programs using the phoneme approach), the system allows the enunciation of a word at the microphone, at the tape recorder, or from a punched paper tape at the high speed reader. The computer detects phonemes during the course of speech by determining when the correlation difference between successive spectra (speech vectors) falls below a given bound (localizes). A spectrum typical of each phoneme is stored in memory, and when a given number of phonemes have occurred, or when a switch register bit is changed, the stored spectra are printed out at the teletypewriter.

A sample of the spectra display of QPS No. 7 is shown in Figure 2. The word spoken was "six". The spectra are shown as a series of lines of data; each line is broken into four groups of four digits each. Each digit represents a channel content of value 0 to 7. The frequencies of the channels represented increase from left to right, ranging from 30Hz to 17.5KHz.

The spectra are arranged so that descending lines correspond to increasing time. The column of numbers to the right of the spectra count the number of phonemes (steady states in speech) that have occurred, and the number on the far right of a spectrum indicates the magnitude of the correlation difference between the first two spectra of the steady state just entered.

Observations

The QPS No. 7 program fails to keep track of the duration of a phoneme; it does not consider slow drift of the speech vector beyond the localization bounds as a movement from one phoneme to another, and it does not scan the spectral analyzer any faster than once per three milliseconds because of time needed to compute the correlation difference.

Nevertheless, enough significant information is retained in the QPS No. 7 display to demonstrate many characteristics of a spoken word which are common to different speakers.

Figure 3 shows the display of another enunciation of the word "six" by the same speaker. Printout of zero energy content was suppressed for clarity. Comparison with Figure 2 shows a strong pattern correlation between the displays. Indeed, voicings of the same word many times by a large variety of speakers yielded patterns easily matched with those shown in the figures.

Further Reduction

Based on the results of QPS No. 7, QPS No. 8 was written to further reduce the characteristics displayed to a few that seemed to most favorably characterize the spoken word.

It can be seen from Figure 3 that the highest energy contents localize about various channels during the course of the word. By tracing the peaks of energy as a function of time, a simpler but still significant picture of the word is obtained.

The exact location of a peak is depicted by numbering the channels from left to right, weighting each channel by its contents, and finding the mean channel number of each peak in the spectrum.

In place of computing a correlation difference to measure the steady state, the computer averages the channel contents over a given number of successive spectra. The average spectrum so produced is scanned for peaks and then compared with the previous average spectrum. A count is made of how often average spectra repeat.

A third voicing of the word "six" is shown as a QPS No. 8 display in Figure 4. The phonemes are numbered at the left, and the peaks are shown as the average channel numbers multiplied by 10. The number to the right of the colon indicates the number of times the average spectrum repeated.

Conclusions

The PDP-8 equipped with only 4K of memory and an EAE adequately served in the system for analysis and presentation of speech data. The methods of data reduction chosen were not optimal; the software was written to test the system and to show its feasibility as a useful research instrument.

Even in this quick attempt to determine the system capabilities, data emerged which proved essential in identifying some invariants of speech. A successful software speech recognition system has already been constructed, based just on the limited data obtained from the output of QPS No. 8.

Acknowledgments

The authors wish to thank S. Tannenbaum for his excellent technical assistance in constructing and maintaining the equipment, R. S. Ward for his unfailing first aid to electronic difficulties and his expert advice on software matters, S. Orsen for his background work in other approaches to speech analysis, and those members of the research staff who offered many helpful comments and who patiently gave their voices for analysis.

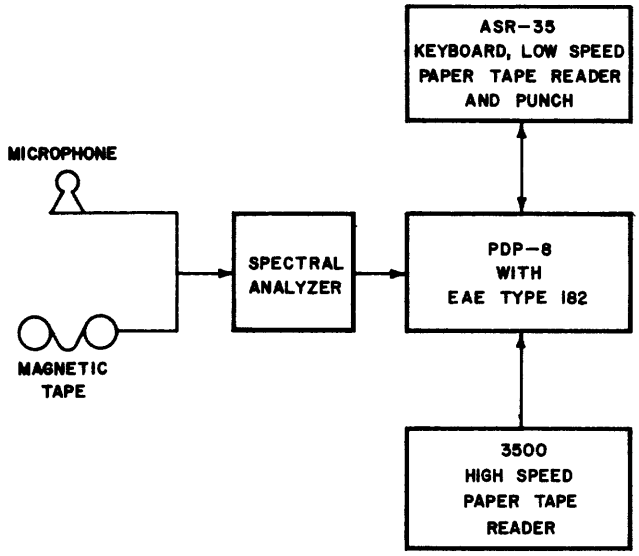


Figure 1 Equipment Configuration

```
WORD SPOKEN: SIX (6)
POWR=1
PHNO=30
FLAG=0
CDBN=2
0000
0 0 0 0  0 0 0 0  0 0 0 0  4 5 0 0  0000 0000
0 0 0 0  0 0 0 0  0 0 0 0  5 7 2 0  0001 0001
0 0 0 0  0 0 0 0  0 0 0 1  5 7 3 0  0002 0000
0 0 0 0  0 0 0 0  0 0 0 0  4 7 3 0  0003 0001
0 0 0 0  0 0 0 0  0 0 0 0  5 7 2 0  0004 0001
0 0 0 0  0 0 0 0  0 0 0 0  5 7 3 0  0005 0001
0 0 0 0  0 0 0 0  0 0 0 0  4 7 3 0  0006 0001
0 0 0 0  0 0 0 0  0 0 0 0  5 7 3 0  0007 0000
0 0 0 0  0 0 0 0  0 0 0 0  4 7 3 0  0008 0001
0 0 0 0  0 0 0 0  0 0 0 0  4 7 3 0  0009 0001
0 0 0 0  0 0 0 0  0 0 0 0  4 7 3 0  0010 0001
0 0 0 0  0 0 0 0  0 0 0 0  0 4 0 0  0011 0000
0 0 0 0  0 0 0 0  0 2 0 0  0 0 0 0  0012 0001
0 0 0 0  0 1 6 2  1 6 3 0  0 0 0 0  0013 0001
0 0 0 0  0 0 4 0  0 4 3 0  0 0 0 0  0014 0001
0 0 0 0  0 0 4 0  0 3 2 0  0 0 0 0  0015 0000
0 0 0 0  0 0 3 0  0 1 0 0  0 0 0 0  0016 0001
0 0 0 0  0 0 0 0  0 0 0 0  2 6 0 0  0017 0001
0 0 0 0  0 0 0 0  0 0 0 0  5 6 1 0  0018 0000
0 0 0 0  0 0 0 0  0 0 0 0  5 7 2 0  0019 0001
0 0 0 0  0 0 0 0  0 2 0 2  6 7 2 0  0020 0001
0 0 0 0  0 0 0 0  0 0 0 0  5 6 1 0  0021 0001
0 0 0 0  0 0 0 0  0 0 0 0  4 6 1 0  0022 0001
0 0 0 0  0 0 0 0  0 0 0 0  3 6 0 0  0023 0001
0 0 0 0  0 0 0 0  0 0 0 0  0 2 0 0  0024 0001
```

Figure 2 A Phoneme Picture of the Word "Six"

```
WORD SPOKEN: SIX (6)
POWR=1
PHNO=30
FLAG=0
CDBN=2
0000
.         .         .4 5      0000 0001
.         .         .5 7 2    0001 0001
.         .       2 .5 7 4    0002 0001
.         .         .5 7 3    0003 0001
.         .       2 .5 7 3    0004 0001
.         .         .4 7 4    0005 0001
.         .         .4 7 3    0006 0001
.         .         .4 7 4    0007 0001
.         .         .5 7 3    0008 0001
.         .         .4 6 2    0009 0001
.         .         .4 7 3    0010 0001
.         .         .4 7 4    0011 0001
.         .         .4 5      0012 0001
.         .         .2 5      0013 0001
.  2 6    .2 7 4    .         0014 0001
.  1 5 1  .2 7 4    .         0015 0000
.    4    .  4 3    .         0016 0000
.    4    .  3 2    .         0017 0001
.    3    .  2      .         0018 0001
.    3    .         .         0019 0000
.         .         .         0020 0001
.         .  4      .  3      0021 0000
.         .  1      .4 5      0022 0001
.         .         .4 6 1    0023 0000
.         .         .3 6 1    0024 0001
.         .       1 .4 7 3    0025 0001
.         .       3 .7 7 2    0026 0001
.         .         .5 7 2    0027 0001
.         .       3 .6 7 2    0028 0000
.         .         .4 5      0029 0001
```

Figure 3 Repetition of the Word, "Six," Zero's Withheld

```
WORD SPOKEN: SIX
SPEC=4
DELT=3
MAXN=250
ENDN=50
FIRS=1
LAST=16
TAPE=0
1. 0
2. 137  : 14
3. 141  : 7
4. 136  : 1
5. 100  : 2
6. 70-103  : 1
7. 70-104-130
8. 70-104  : 6
9. 70-100
10. 70  : 4
11. 0  : 15
12. 103
13. 0  : 1
14. 135  : 29
15. 140
16. 0  : 51

REPT=
```

Figure 4 Third Voicing of "Six," Peak Tracing Approach

AN ELECTRONIC SPEECH RECOGNITION SYSTEM

Morton M. Traum and Edward Della Torre
American-Standard Research Division
New Brunswick, New Jersey

Abstract

A speech recognition system for arbitrary vocabulary and speaker has been constructed using an Audio Spectral Analyzer interfaced with a PDP-8. The speaker enunciates a vocabulary of arbitrarily chosen words into the microphone while the operator types an identification of each at the keyboard. The computer automatically recognizes the subsequent enunciation of any vocabulary word and displays its previously inserted identification at the teletypewriter

Using only 4K of memory, a satisfying accuracy of recognition has been achieved. The software resulting from this initial research is subject to optimization which, it is felt, will render the system highly comparable to the far more elaborate systems common today.

Introduction

A system for displaying and analyzing the characteristics of speech[1] has been constructed using an audio spectral analyzer interfaced with a PDP-8. Speech is introduced to the spectral analyzer via a microphone or magnetic tape recorder, and upon computer command, digital speech information is transferred in parallel from the spectral analyzer to the PDP-8 accumulator. Reduction of the data is performed according to software instruction and the essential information stored in computer memory.

Initial research entailed displaying the stored information at the teletypewriter for analysis and for exploring the potential of the speech system. The results of these relatively crude first investigations presented enough of the invariant characteristics of a spoken word to warrant a simple attempt at recognition.

Accordingly, a program was developed to compare the data produced while voicing a word, with the previously stored data of a vocabulary of words. A match of data patterns was used as the recognition criterion, and a working recognition system was achieved.

Real-Time Data Handling

Raw Data

The spectral analyzer provides eight levels (0 to 7) of measuring the energy content of each of sixteen frequency bandpass filters. The voice is sampled at a one-millisecond rate; the contents of the sixteen filters (often referred to as the "channel" contents) during a millisecond period are collectively called a spectrum.

Reduced Data

During speech, the computer is obtaining raw data from the analyzer and averaging the channel contents over a given number of successive spectra. The channels of the average spectrum so produced are scanned for peaks in energy value, and the mean channel location of the peaks are finally stored in memory.

A program[1] entitled QPS No. 8 displays the data at this point. Figure 1 shows a sample of the display resulting from the spoken word, "six". The average spectra are numbered at the left in the order of their occurrence. The peaks are shown as the average channel numbers multiplied by ten, where the channels have been numbered from 1 to 16 in order of increasing frequency. The number to the right of the colon indicates how often the average

spectrum repeats successively in time. The values shown above the spectra govern parameters such as sensitivity and timing of the acquisition-reduction process.

The QPS No. 8 program has been incorporated into the recognition system, but the parameters have been fixed at reasonable values and the teletypewriter display was eliminated. Instead, the data is kept in a special area of memory, and the program branches to a machine-time reduction and pattern-recognition section.

Defining Speech

Figure 1 shows that spectrum number 2 contains a single peak at channel 13.8, and has occurred 6 times in succession. Number 26 indicates that 18 average spectra have passed in which none of the sixteen filters have averaged at least an energy level of 1. Such blank spectra have been shown to occur repeatedly before and after speech, and their occurrence can be used to define the bounds of the spoken word. For example, speech is considered to have ended in this system when 50 blank spectra occur successively.

Treatment After Speech

Reduction

After the word is voiced and the reduced data stored in memory, further reduction of the data takes place. Blank spectra and spectra which have not repeated significantly are eliminated, and the remaining spectra are weighted to a normal on the basis of their repeated successive occurrence.

Figure 2 shows the reduced and normalized data at this point for the word "six". The "M" of spectrum number 7 indicates that some of the eleven spectra with peaks at channel 13.4 have interspersed in time with some of the twenty-two spectra with peaks at 13.7. Eleven and twenty-two are normalized indications of the recurrence of the spectra.

Vocabulary Construction

If the system is being used in the vocabulary-formation mode, the operator types an ASCII identification at the keyboard before the word is voiced. After the voicing, the data is reduced as described above and moved along with its ASCII to a special section of memory.

Recognition

If the system is being used in the recognize mode, the pattern of reduced and normalized spectra of the voiced word is compared to the spectral pattern of each vocabulary word. Correlation is done on a spectrum-by-spectrum basis, and a score is kept of the number of matches the test word makes with each vocabulary word. Recognition occurs basically as a function of the highest score. The ASCII, associated with the vocabulary word that is selected as the best match, is printed at the teletypewriter.

Software

Organization of Memory

Figure 3 shows how the computer memory has been divided into sections for the various functions of the recognition system.

It should be pointed out again that the programs were written quickly to obtain a feel for the system capabilities, and that emphasis was placed on research rather than on engineering achievement. Therefore, the program construction allows for easy changes in values of many key parameters. Effort was not given to achieving minimum instruction space conditions. Nevertheless, a sizable portion of memory is not used by the recognition system.

New Language

The recognition program has been written as an operating system with its own interpretive language. The interpretive commands can be introduced into the PAL III permanent symbol table, and programs written in the new language may be compiled and loaded as binary tapes over the operating system. As shown in Figure 3, these programs may be written anywhere from location 2ØØ to location 777.

For example, a program which types out a list of the current vocabulary words and spectral patterns is

```
*2ØØ
REVIEW
HLT CLA
JMP  -2
$
```

A program which allows the operator to read in a vocabulary and then to voice a series of words for recognition, is

```
*2ØØ
VOCAB
RECOG
SELECT
JMP .-2
$
```

Other interpretive commands give the operator the facility to examine patterns or move them from one section of memory to another, to compare the scores used in selecting the best match, or to perform a number of other operations essential to the research.

Results

The accuracy of word recognition was a function of the nature and size of the vocabulary.

When the digits zero through nine were entered three times as a vocabulary, recognition accuracy averaged 93% for the original speaker, and 80% for other speakers. Sex and dialect had little bearing upon these figures. By eliminating digits which were sources of confusion, eg. four which was confused with one, and nine which was confused with five, accuracy improved to 98% for the original speaker.

By reading the digits five times into the vocabulary, accuracy for the original speaker increased to about 96%, and after eliminating confusing digits, to about 99%.

A vocabulary of up to 55 single-syllable words could be stored in the 4K memory machine. Recognition time was less than one second from the end of speech.

Expectations

Considering the results obtained as coming from an unoptimized software system that uses only a few characteristics of speech invariance, it is felt that the potential for an extremely accurate speech recognition system is quite high.

Should a more sophisticated software system not suffice, use could be made of a pitch frequency detector that is interfaced to the PDP-8. In addition, the computer is equipped with an analogue-to-digital converter which could prove useful in detecting zero-crossings in the amplitude of the voice input. Finally, the spectral analyzer can be wired to the Program Interrupt System for greater recognition speeds.

References

1. M. Traum and E. Della Torre, Characteristics of Speech Displayed by PDP-8, Proceedings of the Decus Spring Symposium, 1967.


```
WORD SPOKEN: SIX
SPEC=3
DELT=1
MAXN=250
ENDN=50
FIRS=5
LAST=15
TAPE=Ø
1. Ø
2. 138  : 6
3. 136  : 3
4. 138
5. 136
6. 139
7. 137
8. 135
9. 137  : 2
1Ø. 139
11. 137  : 2
12. 139  : 13
13. 137  : 2
14. 113
15. 1ØØ-14Ø
16. 1ØØ
17. 1Ø3
18. 1Ø5
19. 1Ø3
2Ø. 68-1Ø3
21. 68-1Ø1
22. 71-1Ø3  : 7
23. 7Ø-1Ø1
24. 7Ø-1Ø4  : 1
25. 7Ø  : 7
26. Ø  : 18
27. 1Ø2
28. 1ØØ  : 3
29. 14Ø
3Ø. 136
31. 134
32. 137  : 3
33. 135
34. 137  : 4
35. 139  : 1
36. 137  : 2
37. 135
38. 138  : 3
39. 135  : 4
4Ø. 137  : 7
41. 135  : 1
42. 138  : 3
43. 14Ø  : 1
44. Ø  : 51

REPT=
```

Figure 1 QPS No. 8 Display of the Word "Six"

SIX

1. 137 : 24
2. 140 :M: : 14
3. 75-103 : 3
4. 72-103 : 5
5. 70 : 2
6. 137 : 22
7. 134 :M: : 11

Figure 2

Reduced and Normalized Data for the Word "Six"

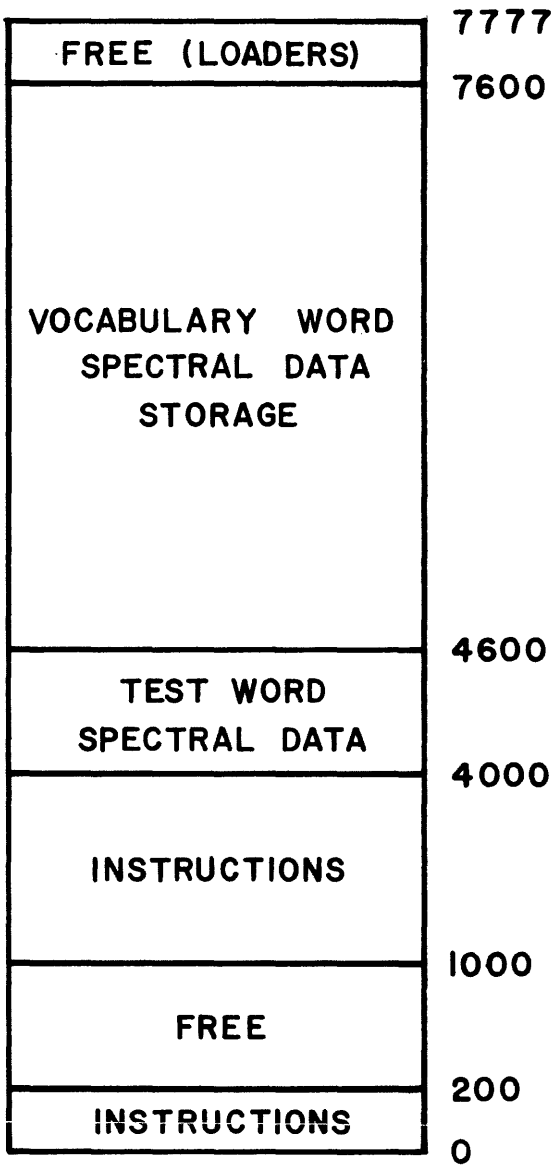


Figure 3 Organization of Memory

APPENDIX 1

DECUS SPRING SYMPOSIUM PROGRAM

Rutgers University
New Brunswick, New Jersey

April 14, 15, 1967

FRIDAY - APRIL 14, 1967

8:30 - 9:50
REGISTRATION

10:00
OPENING
Professor Donald A. Molony, Chairman

WELCOME
Professor J. L. Potter
Chairman, Department of Electrical Engineering, Rutgers University

10:15
Review of Display Activity - Representative from Digital Equipment Corporation Display Marketing Department

10:30
Software Support for a PDP-4/340 Display Configuration
H. Quintin Foster, Jr.,
Department of Defense

11:00
Displays for Studying Signal Detection and Pattern Recognition
Taylor L. Booth, Robert Glorioso, Robert Levy, James Walter, and Herbert Kaufman
University of Connecticut

11:30
COFFEE

11:45
A Status Report on the Application of Processor Controlled Color Displays in Signal Analysis - 1957 to 1967
Charlton M. Walter, Air Force Cambridge Research Laboratories

12:30
Systems Analysis of DEC 338 Programmed Buffered Display
Stephen F. Lundstrom,
University of Michigan

1:15
LUNCH
University Commons Faculty Dining Room

2:30
Graphic Part Programming for Numerical Control Machine Tools
James A. Snow, The Boeing Company

3:10
Enhancements to a Time-Shared Operating System
R. N. Freedman, Massachusetts Institute of Technology

3:40
The Computer Display in a Time-Sharing Environment
Thomas P. Skinner, Massachusetts Institute of Technology, Project MAC

4:10
X-Y Graph Production and Manipulation
J. W. Brackett and R. Kaplow,
Massachusetts Institute of Technology

4:45
DISCUSSION SESSIONS

Computer-Aided Design

General PDP-6 Discussion Session

6:30
COCKTAIL HOUR
Brunswick Inn - Winchester Room

7:30
DINNER
Brunswick Inn - Winchester Room

SATURDAY - APRIL 15, 1967

9:00 - 9:30
REGISTRATION
(For those who did not register on Friday.)

9:30
OPENING - SATURDAY SESSION

9:35
KEYNOTE SPEAKER
Ivan E. Sutherland, Harvard University

10:00
Image Processing of Biological Specimens
Stephen Lorch, Mass. General Hospital

10:30
High Precision CRT Scanning System
C. A. Bordner, Jr., A. E. Brenner, P. de Bruyne, B. J. Reuter, and D. Rudnick, Harvard University

11:00
COFFEE

11:15
General Procedures for Accomplishing Efficient Display Software
J. Richard Wright, Wolf Research and Development Corporation

11:45
TRACD - An Experimental Display Programming Language
Barry Wessler, Digital Equipment Corporation

12:15
LUNCH
University Commons Faculty Dining Room

1:30
HELP - An Integrated Display System for Program Development
D. Friesen and J. Taylor, Massachusetts Institute of Technology

2:00
Displaying the Characteristics of Speech with a PDP-8
Morton M. Traum and Edward Della Torre, American Radiator and Standard Sanitary Corporation

2:30
An Electronic Speech Recognition System
Morton M. Traum and Edward Della Torre, American Radiator and Standard Sanitary Corporation.

3:00
COFFEE

3:15
DISCUSSION SESSIONS

TOURS AND DEMONSTRATIONS

Physics Department, Rutgers
PDP-6

Applied Data Research Corporation
Princeton, New Jersey
PDP-7, PDP-8, and 338 Display

Applied Logic Corporation
Princeton, New Jersey
PDP-6 and 340 Display

APPENDIX 2

AUTHOR AND SPEAKER INDEX

APPENDIX 3
ATTENDANCE

Mr. Donald Allen
Cornell Aeronautical Laboratory
Buffalo, New York

Mr. Donald C. Amoss
Drexel Institute of Technology
Philadelphia, Pennsylvania

Mr. E. Anastasio
Educational Testing Service
Princeton, New Jersey

Mr. Sypko W. Andreae
UCLRL
Berkeley, California

Mr. Dennis G. Austad
Holloman AFB
New Mexico

Dr. Jack Ball
Picker X-Ray Corporation
Cleveland, Ohio

Mr. N. Addison Ball
NSA Department of Defense
Ft. George G. Meade, Maryland

Mr. Wade S. Bartlett
Bell Telephone Laboratories, Inc.
Parsippany, New Jersey

Mr. David C. Barton
Eastman Kodak Company
Rochester, New York

Mr. Ron Bassin
Digital Equipment Corporation
Parsippany, New Jersey

Mr. S. Basendale
Rutgers University
Department of Computer Sciences
New Brunswick, New Jersey

Mr. J. M. Bennett
University of Western Ontario
London, Ontario, Canada

Mr. Joseph P. Benson
University of Pittsburgh
Pittsburgh, Pennsylvania

Mr. Robert P. Bigliano
E I DuPont
Wilmington, Delaware

Mr. W. Wayne Black
Idaho Nuclear Corporation
Idaho Falls, Idaho

Mr. Theodore R. Blakeslee
Trinity College
Harford, Connecticut

Jean Claude Bleuze'
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Robert H. Bliss
Bell Telephone Laboratories
Holmdel, New Jersey

Mr. Taylor L. Booth
University of Connecticut
Storrs, Connecticut

Mr. John W. Brackett
Massachusetts Institute of Technology
Cambridge, Massachusetts

Mr. A. E. Brenner
Harvard University
Cambridge, Massachusetts

Mr. Howard Briscoe
M. I. T. Lincoln Laboratory
Lexington, Massachusetts

Mr. David L. Brown
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Theodore Brown
New York University
Bronx, New York

Mr. Thomas Brown
New York University
Bronx, New York

Mr. E. Bruckner
Siemens Aktiengsellschaft
Wernerwerk fur Med. Technik
8520 Erlangen, Henkestrabe 127
Germany

Mr. Robert Bryan
Lockheed Electronic
Metuchen, New Jersey

Mr. Sam D. Bryan
NIH
Bethesda, Maryland

Mr. Willard A. Bryant
Rensselaer Polytechnic Institute
Troy, New York

Mr. H. Cilke
Sandia Corporation
Albuquerque, New Mexico

Mr. Robert K. Clark
Argonne National Laboratory
Argonne, Illinois

Mr. Peter Clark
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Howard Cohodas
Picker X-Ray Corporation
Cleveland, Ohio

Mr. Paul D. Coleman
University of Maryland
School of Medicine
Baltimore, Maryland

Mr. John D. Collins
Raytheon Company
Bedford, Massachusetts

Mr. Paul Collard
Control Data Corporation
Burlington, Massachusetts

Mr. Richard W. Conn
UCLRL
Livermore, California

Mrs. Angela J. Cossette
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Dave Cotton
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Debruyne
Harvard University
Cambridge, Massachusetts

Mr. Wayne Dengel
Digital Equipment Corporation
Parsippany, New York

Mr. David B. Denniston
Digital Equipment Corporation
Parsippany, New Jersey

Dr. G. L. d'Ombrain
McGill University
Cambridge, Massachusetts

Mrs. Evelyn Dow
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Richard E. Duncan
UCLRL
Livermore, California

Mr. Robert Barry Dydyk
University of Windsor
Windsor, Ontario
Canada

Mr. William B. Easton
Applied Logic
Princeton, New Jersey

Miss Jeanne Eastwood
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Bruice A. Eisenstein
Drexel Institute of Technology
Philadelphia, Pennsylvania

Mr. John Elder
University of Cambridge
Cambridge, England

Mr. J. W. Entwistle
Northern Electric Co. Ltd.
Ottawa, Ontario
Canada

Mr. Marvin D. Erickson
Battelle-Northwest
Richland, Washington

Dr. Donald C. Eteson
Worcester Polytechnic Institute
Worcester, Massachusetts

Mr. Thomas G. Evans
Air Force Cambridge Research Laboratories
L. G. Hanscom Field
Bedford, Massachusetts

Mr. Garry J. Fischer
Northern Electric Advanced Devices Center
Ottawa, Canada

Mrs. Valerie Fischer
Northern Electric
Ottawa, Ontario, Canada

Mr. P. Fleck
M. I. T. Lincoln Laboratory
Lexington, Massachusetts

Mr. Alfred D. Ford
Department of Defense
Ft. George G. Meade, Maryland

Mr. J. Garrett Forsythe
E I du Pont de Nemours
Wilmington, Delaware

Mr. H. Quintin Foster, Jr.
NSA Department of Defense
Ft. George G. Meade, Maryland

Dr. Herbert Freeman
New York University
Bronx, New York

Mr. Richard N. Freedman
M. I. T.
Cambridge, Massachusetts

Mr. Dave Friesen
MIT/LNS
Cambridge, Massachusetts

Mr. Lorin G. Gale
University of Windsor
Windsor, Ontario, Canada

Mr. C. W. Gear
University of Illinois
Urbana, Illinois

Mr. Larry T. Gell
University of Rochester
Center for Visual Science
Rochester, New York

Dr. Genenz
Siemens Aktiengesellschaft
Wernerwerk
fur Med. Technik
8520 Erlangen, Henkestrabe 127
Germany

Mr. Lawrence H. Gerhardstein
Battelle-Northwest
Richland, Washington

Mr. Thomas Giannetti
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Ronald E. Gocht
United Aircraft
East Harford, Connecticut

Mr. W. J. Godfrey, Jr.
E I Du Pont Company
Wilmington, Delaware

Mr. W. Godwin
Educational Testing Service
Princeton, New Jersey

Mr. John J. Golden, Jr.
Brookhaven National Laboratory
Upton, Long Island, New York

Mr. John B. Goodenough
USAF
Bedford, Massachusetts

Mr. Patrick H. Gray
UCLRL
Livermore, California

Mr. Richard Gruen
Digital Equipment Corporation
Palo Alto, California

Mr. Russell B. Ham
U. S. Public Health Service
Winchester, Massachusetts

Mr. Orin C. Hansen, Jr.
Yale University
Department of Physics
New Haven, Connecticut

Mr. Leonard Hantman
Adams Associates, Inc.
Bedford, Massachusetts

Mr. Perry Harris
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Jack Harvey
Communications System, Inc.
Paramus, New Jersey

Mr Michael S. Helfer
New York University
New Rochelle, New York

Mr. John C. Henry
Bolt Beranek & Newman
Cambridge, Massachusetts

Mr. Walter A. Henning
University of Erlangen
5 Egerlandstr, 852 Erlangen
Germany

Mr. Earl D. Hergert, Jr.
Lockheed Electronics
Metuchen, New Jersey

Miss Helen Hill
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. George M. Holmes
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Dale Hurliman
Princeton-Pennsylvania Accelerator
Princeton, New Jersey

Mr. David K. Jefferson
U. S. Naval Weapons Laboratory
Dahlgren, Virginia

Mr. Mike Jennings
Oregon State University
Corvallis, Oregon

Mr. Edward S. Johnson
Psychometric Laboratory
University of North Carolina
Chapel Hill, North Carolina

Mr. Howard C. Johnson
Bell Telephone Laboratories
Holmdel, New Jersey

Mr. Thomas H. Johnson
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. John S. Jorgensen
Digital Equipment Corporation
Philadelphia, Pennsylvania

Mr. John Allen Jones
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Neal Kelley
E I du Pont de Nemours & Co.
Wilmington, Delaware

Mr. Douglas A. Kent
UCLRL
Livermore, California

Mr. Bill Kiesewietter
Digital Equipment Corporation
Philadelphia, Pennsylvania

Mr. Henry P. Kilroy
Potter Instrument Co., Inc.
Plainview, New York

Mr. John Kleboe
R. R. Donnelley & Sons
Mount Vernon, New York

Mr. David Kristol
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. C. Kuehne
Educational Testing Service
Princeton, New Jersey

Mr. Jeffery H. Kulick
University of Pennsylvania
Philadelphia, Pennsylvania

Miss Rita Lalli
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Donald Langbein
Massachusetts Eye & Ear Infirmary
Boston, Massachusetts

Mr. John T. Late
United Aircraft Corporation Systems Center
Framingham, Massachusetts

Mr. Neal Laurance
Ford Motor Company
Dearborn, Michigan

Mr. Martin D. Levine
McGill University
Montreal Quebec, Canada

Mr. D. C. Lincicome
Control Data Corporation
Burlington, Massachusetts

Mr. Peter Lindes
Electronics Associates, Inc.
Princeton, New Jersey

Mr. Philip Loe
University of Maryland
Baltimore, Maryland

Mr. W. H. Long
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Stephen Lorch
Massachusetts General Hospital
Fruit Street
Boston, Massachusetts

Mr. Stephen F. Lundstrom
University of Michigan
Ann Arbor, Michigan

Mr. William T. MacKenzie
University of Windsor
Windsor, Ontario, Canada

Mr. C. Ron Martell
Sandia Corporation
Albuquerque, New Mexico

Mr. Lewis Maylath
Western Electric Engineering Research Center
Princeton, New Jersey

Mr. Les McLean
The Ontario Institute for Studies in Education
Toronto 5, Ontario, Canada

Mr. Richard J. McQuillin
Inforonics, Inc.
Cambridge, Massachusetts

Mr. Ronald E. Medei
Western Electric
Allentown, Pennsylvania

Mr. W. Meir
Applied Logic
Princeton, New Jersey

Mr. John Metzger
Princeton-Pennsylvania Accelerator
Princeton, New Jersey

Mr. Arthur R. Miller
Woods Hole Oceanographic Institute
Woods Hole, Massachusetts

Mr. Walter A. Miller
Chase Brass & Copper Co., Inc.
Montpiller, Ohio

Dr. William P. Minicozzi
NIH DCRT
Bethesda, Maryland

Mr. Donald Molony
Rutgers University
New Brunswick, New Jersey

Mr. John Moran
UAC Research Lab.
East Harford, Connecticut

Mr. Garry P. Morgan
UCLRL
Berkeley, California

Mr. Ronald A. Morrison
General Electric Company
Cincinnatti, Ohio

Miss Linda Sue Nathanson
University of Maryland
College Park, Maryland

Mr. Christopher Nelson
U. S. Public Health Service
Winchester, Massachusetts

Mr. Gerald Nonken
United Aircraft Corporate System Center
Framingham, Massachusetts

Mr. Stewart Ogden
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Stan Pacholski
Control Data Corporation
Burlington, Massachusetts

Mr. Marshall Parker
Denver Research Institute
University of Denver
Denver, Colorado

Mr. D. Payne
Educational Testing Service
Princeton, New Jersey

Mr. L. J. Peek
Western Electric Co., Inc.
Princeton, New Jersey

Mr. S. P. Penstone
Queens University
Kingston, Ontario, Canada

Dr. H. J. Perlis
Rutgers University
New Brunswick, New Jersey

Mr. Richard J. Plano
Rutgers University
New Brunswick, New Jersey

Mr. David Plumer
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Lawrence Portner
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Vern Poulter
Digital Equipment Corporation
Bellevue, Washington

Mr. Bishnu D. Pradhan
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. T. M. Randal
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Fredericka Rapp
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Charles W. Rhynard
NSA Department of Defense
Ft. George G. Meade, Maryland

Mr. Fontaine K. Richardson
University of Illinois
Urbana, Illinois

Mr. Richard H. Robichaud
Control Data Corporation
Burlington, Massachusetts

Mr. Allan C. Roochvarg
University of Wisconsin
Computer Center
Madison, Wisconsin

Mr. Robert Rose
Computer Center
University of Iowa
Iowa City, Iowa

Mr. Peter Rosenfeld
Bell Telephone Laboratories
Murray Hill, New Jersey

Mr. R. Ruderman
Educational Testing Service
Princeton, New Jersey

Mr. Robb N. Russell
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Stephen Russell
Computer Science Department
Stanford University
Stanford, California

Mr. Gerald A. Sabin
Naval Research Laboratory (USRD)
Orlando, Florida

Mr. August E. Sapega
Trinity College
Hartford, Connecticut

Dr. Dan W. Scott
University Computing Co.
Dallas, Texas

Mr. P. M. Sherman
Bell Telephone Laboratories
Murry Hill, New Jersey

Mr. Harold W. Shipton
University of Iowa
Iowa City, Iowa

Mr. D. S. Siegel
Bolt Beranek & Newman, Inc.
Van Nuys, California

Mr. Thomas P. Skinner
M. I. T. Project MAC
Cambridge, Massachusetts

Mr. F. A. Skove
Rutgers University
New Brunswick, New Jersey

Mr. James A. Snow
The Boeing Co.
Seattle, Washington

Mr. David H. Sorenson
UAC Research Laboratory
East Hartford, Connecticut

Mr. John K. Storey
DRB/DRTE
Shirley Bay, Ottawa
Ontario, Canada

Mr. Robert H. Storz
M. I. T.
Cambridge, Massachusetts

Mr. Anthony J. Stracciolini
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Ivan E. Sutherland
Harvard University
Cambridge, Massachusetts

Mr. Melvin S. Swanson
U. S. Coast Guard Oceanographic Unit
Washington, D. C.

Mr. L. Taylor
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. B. Tiefa
Sandia Corporation
Alburquerque, New Mexico

Mr. Robert H. Tedford
Union Carbide Corporation
White Plains, Connecticut

Mr. Morton M. Traum
American Standard
New Brunswick, New Jersey

Mr. Robin B. Wadleigh
Johns Hopkins University
Baltimore, Maryland

Mr. Geoffrey F. Walker
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Donald Wallace
Bolt Beranek & Newman, Inc.
Van Nuys, California

Mr. Charleton M. Walter
AFCRL
Bedford, Massachusetts

Mr. James R. Walter, Jr.
University of Connecticut
Storrs, Connecticut

Mr. Fred Weeks
The Ontario Institute for Studies in Education
Toronto 5, Ontario, Canada

Mr. William F. Weiher
Stanford University
Stanford, California

Mr. Paul Weinburg
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. N. S. Wells
Atomic Energy of Canada Ltd.
Chalk River, Ontario, Canada

Mr. Barry Wessler
Digital Equipment Corporation
Maynard, Massachusetts

Mr. Michael Wieder
Digital Equipment Corporation
Princeton, New Jersey

Mr. Richard Wilkinson
Digital Equipment Corporation
Seattle, Washington

Mr. John Willis
First Felicia Corporation
New York City, New York

Mr. Don A. Witcraft
Digital Equipment Corporation
Maynard, Massachusetts

Mr. J. Richard Wright
Wolf R & D Corporation
West Concord, Massachusetts

Mr. Richard V. Wolf
University of Pittsburgh
Pittsburgh, Pennsylvania

Mr. Michael S. Wolfberg
Moore School of Electrical Engineering
University of Pennsylvania
Philadelphia, Pennsylvania

Mr. Ronald Zane
UCLRL
Berkeley, California

Mr. Robert Zlotnick
Digital Equipment Corporation
Parsippany, New Jersey

Mr. Walter Zydlewski
Worcester Foundation for Experimental Biology
Shrewsbury, Massachusetts

PRINTED IN U.S.A.

20-6/67