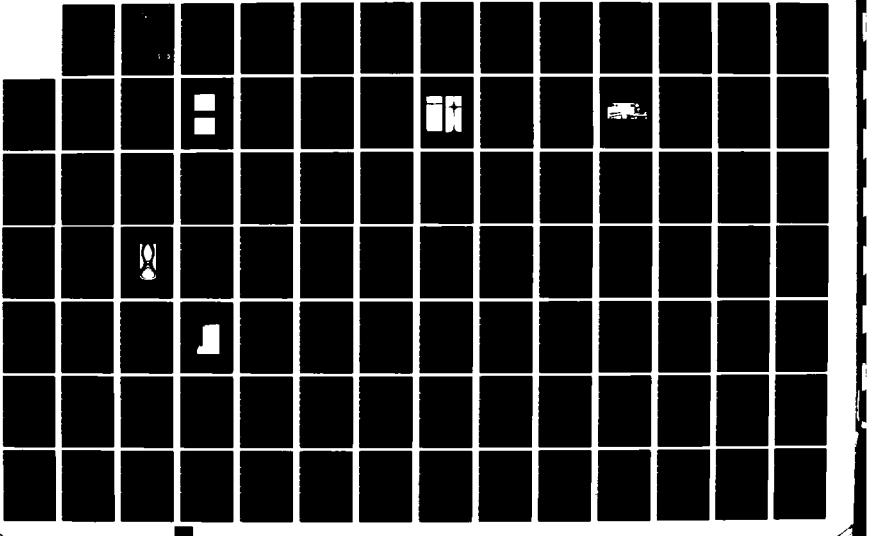
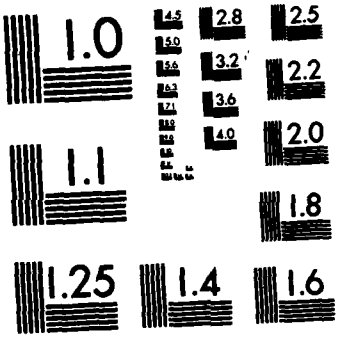
AD-A124 751 VIDEO DIGITIZATION OF DISPLACEMENT/GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... 1/2

UNCLASSIFIED D R CHAFFEE DEC 82 AFIT/GAE/AA/82D-5 F/G 9/2 NL





DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

VIDEO DIGITIZATION OF DISPLACEMENT/ GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS

THESIS

AFIT/GAE/AA/82D-5 David R. Chaffee
Captain USAF

Approved for public release; distribution unlimited.

VIDEO DIGITIZATION OF DISPLACEMENT/GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
David R. Chaffee, B.S.
Captain USAF

Graduate Aeronautical Engineering
December 1982

Approved for public release; distribution unlimited.

PREFACE

In today's technical Air Force, it is not uncommon to find instances where personnel must work with slow and tedious equipment in accomplishing their tasks. The area of digitizing data from optical visualization systems is one such example. This thesis is an investigation into the feasibility, accuracy, and applicability of a video digitizer which holds promise for reducing the tedium and greatly reducing the time factor of obtaining usable engineering results from optical instrumentation systems.

The study was sponsored by the Air Force Wright Aeronautical Laboratories, Aero Propulsion Laboratory, under the auspices of Dr R. Rivir. The end result of this effort will considerably reduce the workload of obtaining displacement measurements from holograms as was its purpose, and open new doors to more efficient data reduction techniques through video digitization.

I would like to thank Lieutenants N. Grosek and R. Koop for helping to define the accuracy of this system. Thanks also to Captain H. Briggs for his time and helpful assistance in developing my understanding of microcomputers.

A very special thanks to my advisor, Dr H.E. Wright, for his continued friendly encouragement, and most especially to Ms Sandra Lovely, whose typing ability is second to none.

David R. Chaffee

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

SYMBOL	DESCRIPTION	UNITS
b	Temperature gradient at the test cell wall	R/inch
C_1	Constant for interferogram as function of pressure	1/R
D2	Distance between undisturbed fringes	inches
E_n	Ratio of Y data displacement to the original distance between fringes	dimensionless
K	Thermal conductivity	Btu/Hr-ft-R
q	Local heat flux/unit depth	Btu/hr-in
t	Temperature	R
T_1	Temperature at wall of test cell	R
T_2	Temperature at first fringe from wall	R
T_3	Temperature at second fringe from wall	R
W	Out of plane displacement	inches

ABSTRACT

A general purpose video digitizer is developed to be used for a broad range of optical data acquisition techniques. It allows the engineer to digitize optical data using a computer image and cursor control to build an array of coordinates or displacements for subsequent conversion to engineering results. An assessment of this technique is made qualititatively and quantitatively and discussion is presented on its potential applications and accuracy.

Free convective heat transfer interferograms are digitized and the data reduced on the same microcomputer system. Results from this data compare favorably to those using a microcomparator digitizer. Time averaged holograms from a vibration rig are reduced with the video processor. These results are also favorable when compared to similar data taken with an optical comparator. Displacement and bending strain plots are overlaid using both data sets and show excellent correlation.

Recommendations are made for expanding and improving the digitizer so that it may achieve its full potential.

VIDEO DIGITIZATION OF DISPLACEMENT/GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS

I. INTRODUCTION

Optical data acquisition systems involve such techniques as schlieren, interferometry, shadographs, and a form of laser interferometry known as time averaged holography. Through the use of these tools, data can be obtained from fluid flow fields, structural tests of materials, or other items without any physical inteference of the flow stream or test article. Unfortunately, the reduction of raw data from these systems can be extremely tedious, time-consuming, and often times inaccurate due to readability problems. Video digitization offers an efficient and reasonably accurate method for converting optical information into digital data and allowing immediate reduction to yield useful engineering results.

Previous Work. Prior to this effort, Lieutenants K. Jones and F. Slane began developing software to interface a set of TecMar real time video digitizer boards on a Cromemco System III microcomputer. Their effort at the Air Force Institute of Technology was a first step in developing a video digitizer. Using their software, pictures could be frozen in computer memory, displayed on a monitor, and a cursor moved to any location showing gray values for a 128 by 128 pixel picture. In an automatic track mode, the cursor would attempt to find and track a given fringe line.

Before this effort, digitization of optical data was accomplished using film readers, such as maintained by 4950TW, Wright-Patterson AFB, Ohio, or a microcomparator used at the Air Force Institute of Technology in the case of heat transfer test results.

Quotes from users of these digitizers indicate that one and one-half hours in the case of the microcomparator, and up to two hours on an optical

comparator are required to gather sufficient data from one interferogram. The data must then be fed into a computer.

Current Work. At the Air Force Institute of Technology, Lieutenant N. Grosek is conducting a thesis using interferometry. In that effort, more than 45 hours may be spent digitizing 30 interferograms to investigate the relative heat transfer rates of various free convective test cell configurations. A microcomparator is used in the effort to obtain fringe coordinates to be used in approximating the temperature gradient and hence the heat transfer rates at the temperature controlled test cell surfaces.

Engineers from the Aero Propulsion Laboratory currently use an optical comparator to digitize data from time average holograms. Lieutenant Robert Koop, assigned to the Laboratory, is conducting a thesis in which he must extract holograph fringe displacement data for five vibration modes, on three different aspect ratio plates, each plate having five variations of twist. Potential exists for more than 75 manhours of effort simply for digitizing the fringe displacement information.

In other efforts, a series of manometers provide pressure data for airfoil test specimens and flow field properties. Thirty-five millimeter photographs of these boards are placed in a film reader at the 4950 Test Wing at Wright-Patterson AFB, and fluid column displacements are measured for data reduction. This represents another extremely time-consuming phase in the efforts of engineers to extract useful coordinate/displacement data from optical systems.

Present Study: It has been suggested that the use of a microcomputer video digitizer might be effective in streamlining the data acquisition phase of optical systems while maintaining, at a minimum, current standards of precision and accuracy. The scope of this effort is to investigate the

feasibility of such a video digitizer and its applicability for particular engineering tests conducted at Wright-Patterson AFB. The objectives of the effort are to:

1. Develop software necessary to exercise a set of TecMar Real Time Video Digitizer Gray Level/Graphics Monitor Interface boards. The software is to allow accurate determination of coordinates from a digitized picture on the video screen using a cursor arrangement and an input set of reference dimensions.

2. Apply the digitizer to interferograms of a free convective test cell comparing results to those obtained using existing data procedures.

3. Apply the digitizer to time average holographic interferograms, input the data to a stress analysis program HOLOCURVE, and compare final results to those obtained through conventional procedures.

4. Qualitatively assess:

   a) video restrictions of the system

   b) image resolution and readability to an observer

   c) the usability of the final system for support of technical research

General observations concerning the video digitizer are addressed and an appendix provides potential users with necessary operating instructions.

II. PRINCIPLES OF OPTICAL DATA REDUCTION

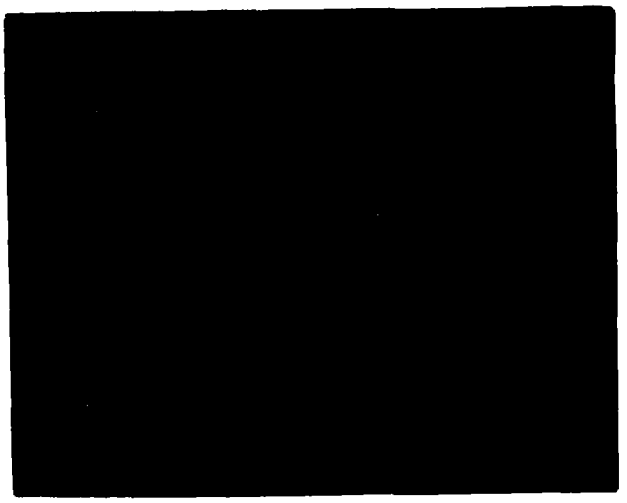
Heat Transfer Interferometry

An experimental test device is used at the Air Force Institute of Technology in which the relative heat transfer rates of different free convection configurations can be compared. The optical data acquisition technique applied is a Mach Zehnder interferometer (Ref 5).

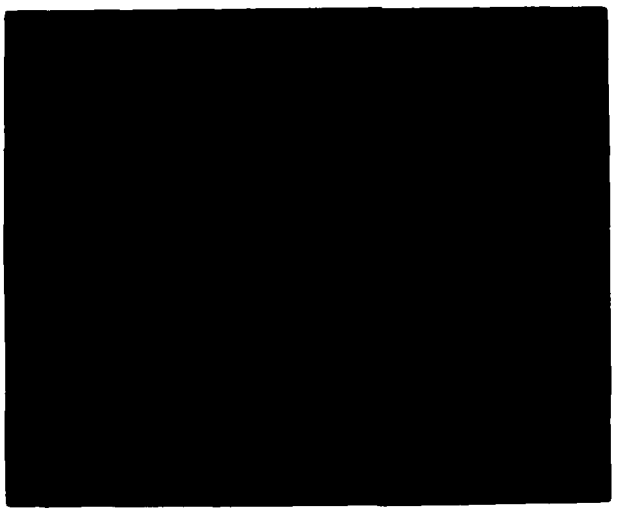
A collimated beam of light is divided by a splitter plate. Each of the split beams follow different optical paths but are eventually rejoined after deflection with a series of mirrors. Beam 1 is left undisturbed while the other beam is passed through a 1 ft. by 3 inch by 2.35 inch test section. Since the optical length of beam 2 is altered due to refractive properties within the active test section, the two beams interfere with one another once they are rejoined. This interference produces a series of fringes which represent density variations in the test section of the device. A more detailed description of this device is available in Khan's thesis on free convection (Ref 6).

Pictures taken with low speed film produce images of the fringes such as those shown in Figure 1. For this particular experiment, Lieutenant N. Grosek investigated the relative heat transfer rates at a hot and cold wall when a variety of separation plates were inserted between the walls. Figure 1a shows the fringe pattern in the test area cross section without a divider and 1b presents the pattern with a single vertical divider.

In order to produce usable results, it is necessary to approximate the temperature gradient at the walls of the rig. Since the displacements of the fringe lines represent density changes, an approximation is developed with which temperatues at certain points on a fringe can be determined. From these



(a)



(b)

Figure 1. Mach-Zehnder Interferograms of Free Convection Between Plates of Different Temperature (a) No Divider (b) Single Vertical Divider

temperatures, a second order curve can be derived which is then differentiated to calculate the temperature gradient of the air at the wall, $dT/dx)_{x=0}$. With this gradient, Fourier's law of heat transfer can be applied to determine the local heat flux. This law states

$$q = -k(dT/dx)_{x=0} \tag{1}$$

where k is the thermal conductivity of the air in the test section.

T_1 is a known temperature and it can be shown that T_2 and T_3 (see Figure 2) are approximated by

$$T_2 = T_1/(1 + C_1E_1T_1) \tag{2}$$

$$T_3 = T_1/(1 + C_1E_2T_2) \tag{3}$$

where $E_n = Y_n/D2$ and C_1 is a constant for a given picture based on atmospheric pressure.

By curve fitting these temperatures and taking the derivative at the wall,

$$(\partial T/\partial x)_{x=0} = b \tag{4}$$

$$\text{where } b = \frac{T_2 - T_1 - cx_1}{x_1} \tag{5}$$

$$\text{and } c = \frac{x_2 (T_2 - T_1) - x_1 (T_3 - T_1)}{x_1 x_2 (x_1 - x_2)} \tag{6}$$

In short, the entire process is reduced to determining the displacements X1, X2, Y1, Y2, and the distance between the fringes at the wall. These required measurements are shown in Figure 2. By integrating several of these sets of points along a wall, the total heat flux is determined.

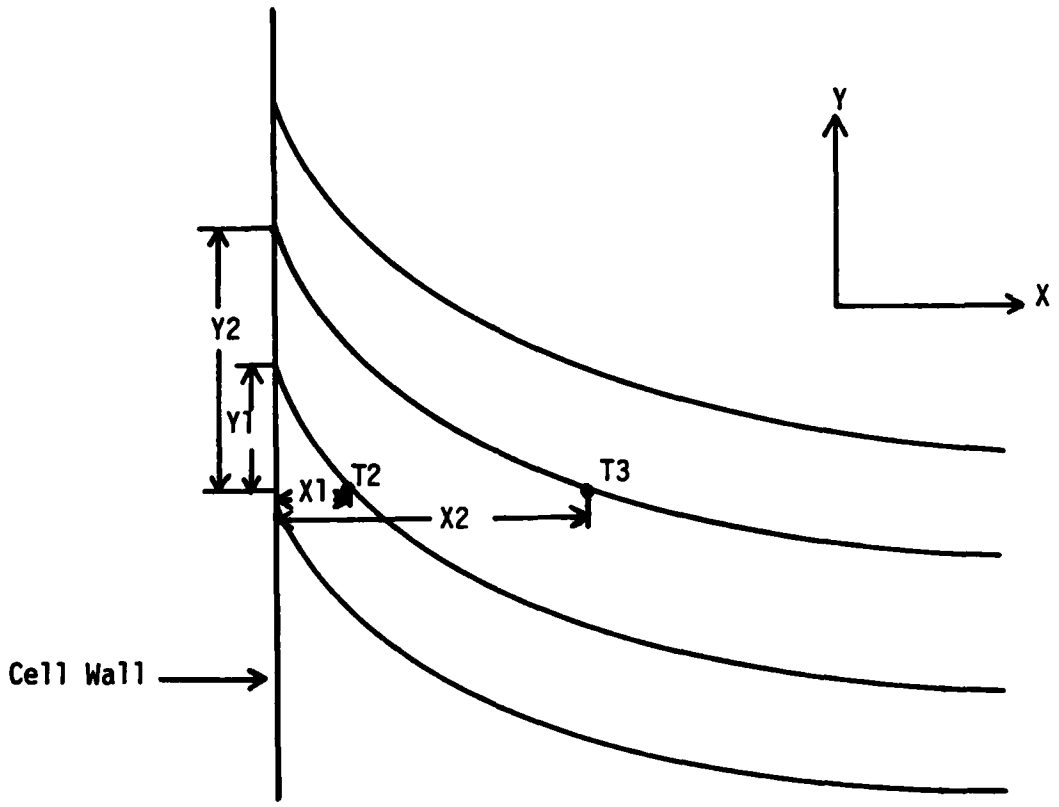


Figure 2. Fringe Displacement Data Required to Calculate Local Heat Flux

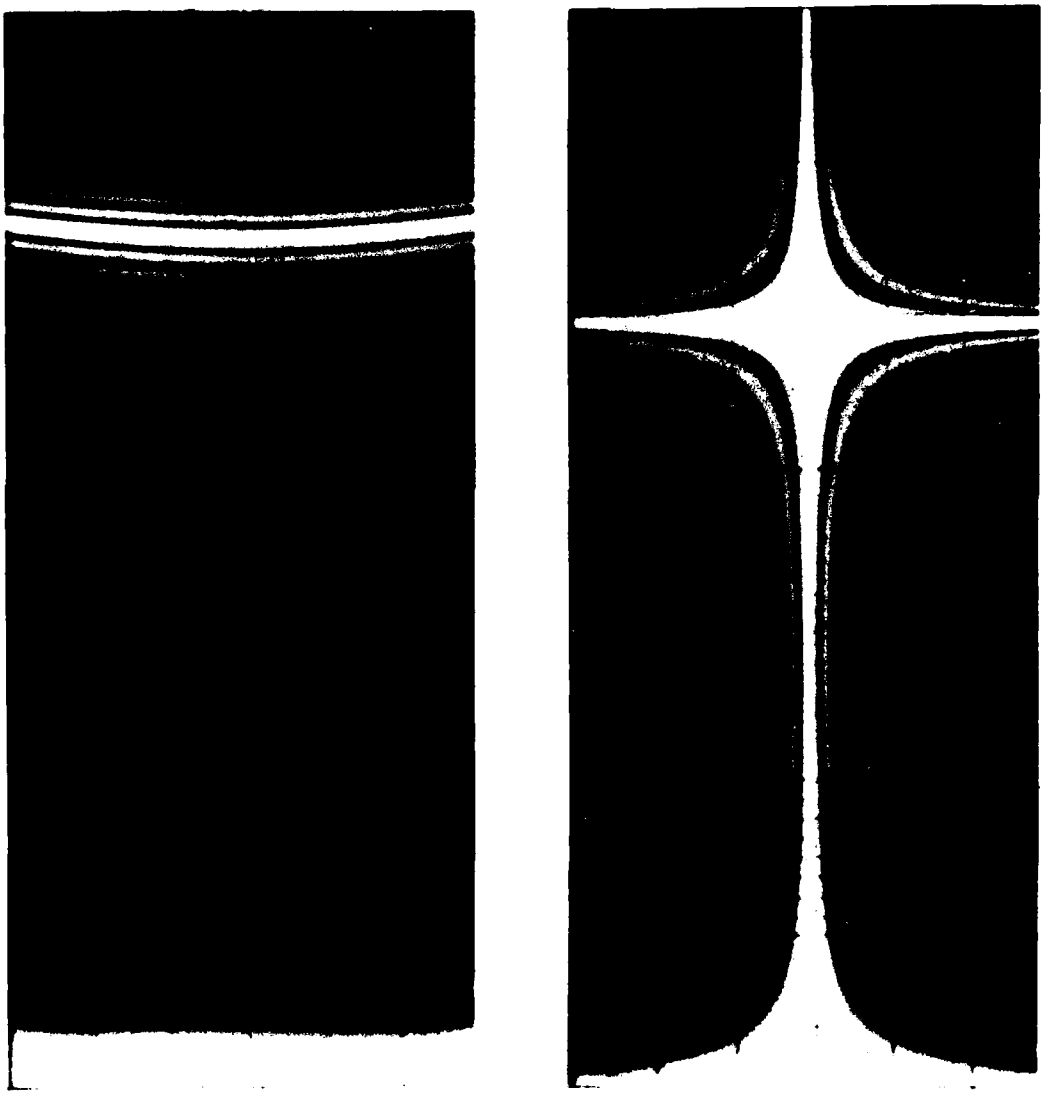
Time Averaged Holographic Interferometry

Using time averaged holographic interferometry, McBain (Ref 7) has shown that normal displacements and bending strain can be appoximated for a plate-like structure in vibration using computer software called HOLOCURVE. Photographs of such holograms are shown in Figure 3. With this technique, the modal deformation of a structure is determined by measuring fringe displacements from a known edge condition. The fringes create a contour map of the normal displacement of the blade out of the plane of the hologram. Bright fringes represent nodal lines or regions of zero displacement. Travelling away from these nodal lines, the dark fringes show the varying magnitude of displacement out of plane.

The curvature of the plate-like structure can be approximated by a curve fit of measured spanwise or chordwise displacements of the fringes. Then, using clamped-free or free-free boundary conditions set up in HOLOCURVE, derivatives of these functions are taken and modal weighting coefficients determined such that normal displacement and bending strain can be calculated and output.

Once again, the final results of normal displacement and bending strain are based on input data of the displacement of the fringes from a boundary condition reference point. The distance of every fringe from the reference is not necessary since the curvature of the blade is determined using a curve fit routine. More data points, however, should lead to a better approximation.

HOLOCURVE, used in this effort, is a specialized, interactive computer routine which is currently executable on the Control Data Cyber computer at Wright-Patterson AFB. A complete discussion of its use, including required data inputs and user interactions, is presented in McBain's work (Ref 7). The program listing also appears in that technical report.



(a) (b)

Figure 3. Sample Time Averaged Laser Holograms of a 3 inch x 7 inch Plate in Vibration (a) Second Bending Mode (b) Second Torsion Mode

Other Applications

There are quite a number of applications where coordinates or displacements must be extracted from optical or visual sources. Large manometry boards are currently used to record pressures on and around test articles in the five foot wind tunnel at the Air Force Institute of Technology. Photographs of these manometers are taken and various optical comparators are used to measure the column heights. This proves to be extremely time consuming.

Strip charts on a few experimental rigs must be measured to determine voltage variations for data reduction. Another convenient use, at some time in the future, may be to pick data off of existing graphs to input to computer routines for data comparison or correlation. Both of these applications could become commonplace if an accurate, rapid optical digitizer were available.

III. HARDWARE AND SOFTWARE DESCRIPTION

The video digitizer developed in this effort has some dependence on microcomputer hardware. These restrictions are now specified and a discussion follows to describe how subroutine modules may be added to the current software.

TecVI Microcomputer System

A TecVI microcomputer system, assembled by TecMar, Incorporated, Cleveland, Ohio, was purchased specifically for this effort. Figure 4 is a photograph of the system.

The computer itself is a Cromemco Single Card Computer (SCC) which includes three 8-bit parallel ports and one RS232 serial port (Ref 3). 64K bytes of static RAM are available. It uses CP/M™ version 2.2 as a disk operating system.

A Televideo™ model 910 terminal provides user interface. Using appropriate switches, it may be modified to emulate other terminals, including ADDS 25, Hazeltine 1410, or a Lear Siegler ADM-3A/5. Its interface with the SCC is set at 9600 baud. No particular restrictions result from the terminal. The port address is 40 hexidecimal (40H).

The printer, which was not an integral part of the TecMar system, is a C.ITOH & Company, Model 1550, Dot Matrix Serial Impact Printer. It is connected in parallel to the SCC at port address 04H. Although it has graphics capability, the SCC does not accommodate the additional bit of information necessary for this feature. A modification of the operating system to simulate an eighth data bit for the printer port would allow the printed output of any image appearing on the accompanying nine inch monitor.

Figure 4. TecMar TecVI Microcomputer System

Information necessary to accomplish these changes is found in the appropriate hardware manuals.

Two cameras were used in the effort. The first was a Phillips video camera with variable aperture and a lens focal length of 12.5 mm. It was used early in the effort but provided poor video quality compared to the RCA camera which accompanied the TevVI. The RCA camera provided a very clear view of the test pictures and was used for final data analysis. Its focal length is 16mm.

The video digitizer is essentially independent of the aforementioned hardware, although changes to the software would be required to reflect any port address differences of the terminal or possibly the printer. The critical hardware requirement is a set of TecMar Real Time Video Digitizer and Monitor Interface (RT+MI) boards (Ref 8). Another requirement is that the computer be capable of accommodating direct memory access.

The RT+MI, S-100 boards are the hardware which allow direct memory access (DMA). They are responsible for digitizing and displaying a video image from the camera to the television monitor. The video digitizer, DIGITIZE, developed in this effort, is dependent on these boards. Other computer systems may be used but the RT+MI set must be included in the system. As an example, they have been successfully used in a Cromemco System III microcomputer.

Digitizer Program. The computer software developed as the digitizer is called DIGITIZE. It is based on a TecMar routine, VIDISOFT, which accompanied the TecVI. Appendix A is a user's guide to the program and provides specific information on its use and how to make alterations when necessary.

In general, the program provides the user with a series of menu items. One may:

1) Continually display a digitized camera image.

2) Freeze a particular picture on the monitor.

3) Save a picture on an 8 inch floppy disk.

4) Retrieve a picture from a disk.

5) Move a cursor to any location on the digitized image.

6) Print gray values and coordinates of the two pixels comprising the one byte cursor.

7) Establish reference dimensions for any new picture so that the computer knows how many pixels there are per inch, centimeter, etc.

8) Reset the origin anywhere on the picture.

9) Save coordinate data in an array called SAV( ).

10) Write the contents of SAV( ) to disk.

11) Retrieve SAV( ) from disk.

12) Write the contents of SAV( ) to the printer.

13) Reduce heat transfer data using an example of a user implanted subroutine module.

14) Exit the program and return to CP/M.

FORT//80™ is the compiler/computer language used in DIGITIZE. It is advertised as a Fortran IV compiler but in fact does not acknowledge some of the standard Fortran IV functions recognized by most programmers. It has been extended to allow the programmer to specifically locate variables and arrays in computer memory. This is necessary to prevent information from overwriting the memory stored picture. A future transition to another compiler may be desirable, keeping in mind that memory space must be allocatable by the user.

DIGITIZE places the 32K picture image in the upper half of the 64K byte capacity computer. Because of this arrangement, the operating system is established as a 32K system. It will be convenient to use a 64K operating

system when programs other than DIGITIZE are being executed to afford full use of the micro's memory capacity.

A complete listing of DIGITIZE appears in Appendix A, Part II. It is printed in twelve pitch letter gothic which is recognizable on a Compuscan Alpha Word III optical data reader.

Future Software Integration

DIGITIZE is modularized in such a manner that a user may build a subroutine to manipulate coordinate data as desired. As exemplified by subroutine HEAT@TRANSFER, this capability can provide the user an immediate tool for reducing coordinate/displacement information. Any additional subroutines are restricted to the limitations of the FORT//80 compiler.

If FORT//80 is deemed too restrictive, it is possible for the user to write a routine to manipulate and store data on disk, exit DIGITIZE, and then recall the data in a program written in a more versatile language.

IV. CHARACTERISTICS OF VIDEO DIGITIZATION

General

General characteristics of video digitization were investigated. The readability of several possible resolutions had to be determined. Effects of using a finitely dimensioned cursor were considered. Various methods of "taking" pictures were investigated to provide a user with consistently good video images. These characteristics are discussed in the following sections.

Image Resolution.

Digitization in this effort involved two resolutions of monitor image. Noting that one pixel is the smallest possible sample of an image, the two resolutions were 256 x 256 pixels and 512 x 256 pixels. Actual image sizes are reduced because; (1) picture synchronization for the monitor screen uses some pixel space, and (2) the memory size for the picture is restricted to 32K bytes so only one-half of the higher resolution can be displayed. Picture sizes are therefore 256 x 240 and 512 x 120 pixels. The higher resolution consistently provides more accurate results and better readability.

Pictures used to verify the video digitizer in the effort ranged from a 3 inch x 2 inch specimen to a 3 inch by 7 inch specimen. The smallest measurements were on the order of .02 inches. A procedure was therefore established to determine the ability of the digitizer to read these small displacements at both resolutions.

A picture of a grid was taken with one inch squares so that known distances could be measured. Figure 5 presents the 256 x 240 resolution results and Figure 6 shows those of the higher resolution. Both of these images were processed with input reference dimensions of three inches horizontally and one inch vertically.

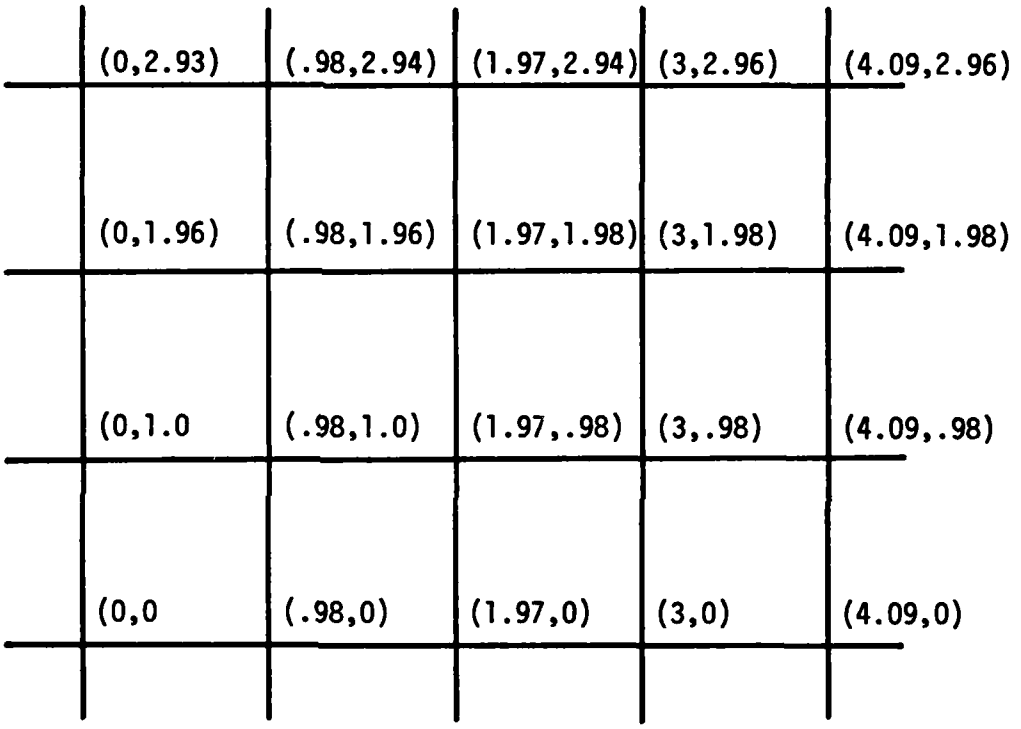


Figure 5. Measurements of Known One Inch Grids at a Resolution of 256 x 240

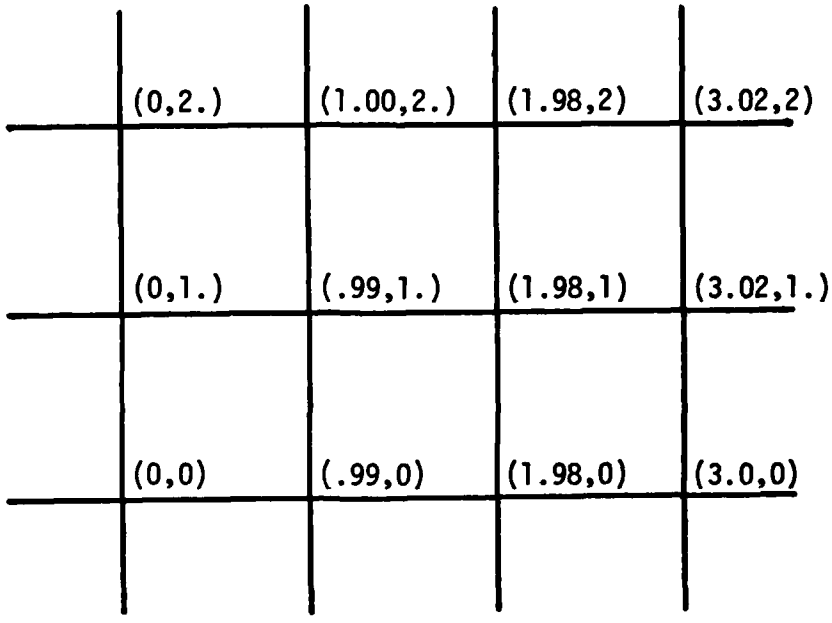


Figure 6. Measurements of Known One Inch Grid at a Resolution of 512 x 120

One can observe some error in these measurements. Some of the error is a result of parallax at the camera. This may eventually be reduced by improving optics of the system. In the short term however, observations were made regarding the choice of the user input reference dimensions. Repeatedly, the most consistent and accurate measurements for any given picture were those in which the user established reference dimensions were approximately two-thirds the length of the largest displacement to be measured. For example, the largest measurement required during the heat transfer data reduction was, at most, one inch. Best results were obtained at reference dimensions of .6 to .7 inches.

Cursor Traits.

One cursor on the monitor contains two pixels and is the object used to establish the dimensions of a picture and to read coordinates or displacements of any point on that picture. As shown in Figure A-2, three different coordinate sets may be selected by the operation of the cursor. Each of the two pixels within the cursor contains a gray value ranging from 0 to 15 where 0 is black and 15 white.

It is important to understand that the cursor and each of its pixels are of finite dimension. There are instances where the cursor cannot be placed to record the actual coordinate information of a desired data point. The best a user can do in this case is to get within one-half pixel width from the desired point. Figure 7 exhibits this restriction. Later discussions of data utilization will show that this does not disqualify the video digitizer from being extremely useful.

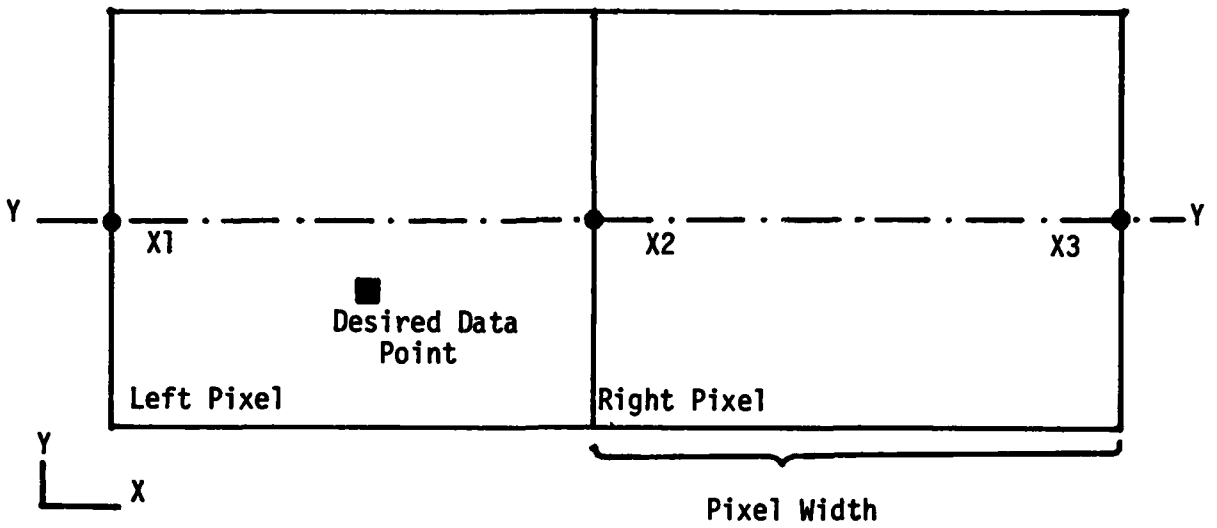


Figure 7. Example of Cursor "Finiteness" Error

"Freezing" the Picture.

"Freezing" the picture means to cease the continual display from the camera source and to display only the current computer memory image on the monitor screen. It is important for the monitor image to be clear to the observer and correctly positioned on the screen. Pictures of various quality were frozen on the monitor to determine possible induced error due to misalignment and poor picture quality.

Significant error can result if the original picture is not taken or frozen perfectly square on the screen. Optimum images resulted when the picture was attached to a stable, vertical platform, with the camera mounted on

a tripod. This is not a necessary step although it can simplify the picture taking process. After freezing a picture, the cursor can be used in horizontal and vertical movements to determine how squarely the image has been placed on the monitor.

Several experiments were undertaken to establish a "best" procedure for freezing pictures in memory. The "best" approach seems to be for any user to try several techniques himself. The quality and type of picture to be frozen definitely determines the best additional lighting to use, if any, the optimum proximity of the camera to the test object or picture, etc. The cursor is more accurately placed if the image fills the monitor, but too much image can create accuracy problems because of parallax error.

Automation of Cursor Movement.

An original proposal behind this effort was to determine to what extent the gray value information could be used to track fringe lines, or obtain data automatically without user interface. Two problems arise with regard to this approach.

First of all, when a picture is frozen on the monitor, a certain amount of noise in the system and reflected light result in inconsistent grays to the point that the center of a fringe may not actually be the blackest region in the fringe. The logic to try and automatically track such a line could easily be modularized and input to DIGITIZE. However, there is no certainty that the center of the fringe could actually be located. It is possible to improve the noise error in the picture by storing the same picture several times, then recalling and averaging these images to create a single picture for data reduction. Lieutenant J.H. Cromer has developed this logic for thesis work at

the Air Force Institute of Technology. Reference is made to his efforts (AFIT/GE/82D).

The averaging technique was not used because of a second problem with automatic data collection. The data necessary for both the heat transfer work and the laser holography effort did not, by any means, lend itself to such automation. This is described in more detail in the specific characteristics section for those reduction schemes.

A closer look at specific examples will help to explain some of the aforementioned characteristics.

Heat Transfer Reduction

As discussed in Section II, four displacements, X1, X2, Y1, and Y2 are required to approximate the local heat flux at any point on the wall of the test section. Very briefly the procedure for recording data was to set an origin at each location for which q was to be calculated. The cursor was then moved to the four displacement points in a prescribed sequence and coordinates were stored in a working array called SAV. Sample SAV contents for this application are presented in Table C-I.

The pictures were most accurately reduced at the higher resolution 512 x 120 because the fringe slopes at the wall were too steep to accurately use the larger size cursor of the 256 resolution. The data comparisons in the next chapter were based on reference dimensions of three inches and one inch but it has been proven that better accuracy for this case is possible with a smaller reference scale.

Samples of heat transfer interferograms were presented in Figure 1. One picture appears lighter than the other. This presented no particular problem to the digitizer. Lighting on the picture was simply varied using a desk lamp or other light source until a usable image appeared in the monitor.

One of the definite advantages of the video digitizer in the heat transfer effort was the global view of the plate being analyzed. On the micro comparator, traditionally used for such data extraction, only a single fringe may be seen at one time. The accuracy of reading any one data point is excellent but it is difficult to reference that point to a known or fixed point elsewhere on the picture. The global view afforded by the video digitizer is very good in this regard. The entire plate to be analyzed is in view and fringes still appear clearly enough for adequate data reduction.

The pixel width for these interferograms was consistently .0105 to .0107 inches. Pixel height was normally .015 inches. As previously indicated, these represent the finite dimensions of the cursor. Because of this finiteness, the cursor could not always be placed directly on the point for which coordinates were desired.

As a practice, the cursor was placed at a data point by eyeballing the cursor on the monitor. On occasion, it was helpful to use printed gray values to locate the center of a fringe or the edge of the wall. That approach was an exception rather than the rule. A more appropriate approach for thick fringes was to count the number of cursor widths or heights of a fringe and simply place the finite cursor in the center of the fringe. Such an approach worked quite well and was very consistent.

Automatic fringe "seeking" was not deemed appropriate for this phase of the effort since no more than two consecutive data points were taken in a given direction.

As a note, the acquisition of data from these pictures could have been improved if the 512 x 120 picture were expanded to 512 x 240. This would allow the picture to be reoriented 90° on the screen and the finer cursor dimension

of .0105 inches could have been used to extract the smaller displacements of X1 and X2 more accurately. This expansion, of course, will require 32K bytes more memory in the microcomputer and a different central processing unit in the case of the TecVI.

An attempt was made to take video pictures directly from the interferogram image at the test device. Resulting pictures were extremely poor. The light beam concentration at the center of the frosted glass plate on which the image appeared was so intense that fringes were virtually worthless for data extraction. If optics can be obtained to alleviate the brightness of the beam at the center of the interferogram, real time use of the video digitizer could be greatly expanded.

Holographic Reduction

Three pictures of time averaged holograms were digitized for data input to HOLOCURVE. All three of the pictures appeared in McBain's work and computer images for this effort were taken directly from the text of that report (Ref 7).

Spanwise and chordwise traverses were made across the blades as indicated in Figure 8. The coordinates of the center of each (but not necessarily every) fringe along the path of these traverses were recorded for data input to HOLOCURVE. Original pictures were reduced with the 256 resolution image but final data, being much more accurate, was taken at the 512 x 120 resolution. To take data, an origin was set at a known boundary condition and fringe displacements from that point were measured and recorded. A sample raw data set is presented for this application in Table C-II.

Using gray values to locate the center of a fringe was definitely more applicable for this data extraction scheme since the fringes were wider than those of the heat transfer holograms. Once again, however, due to the finite

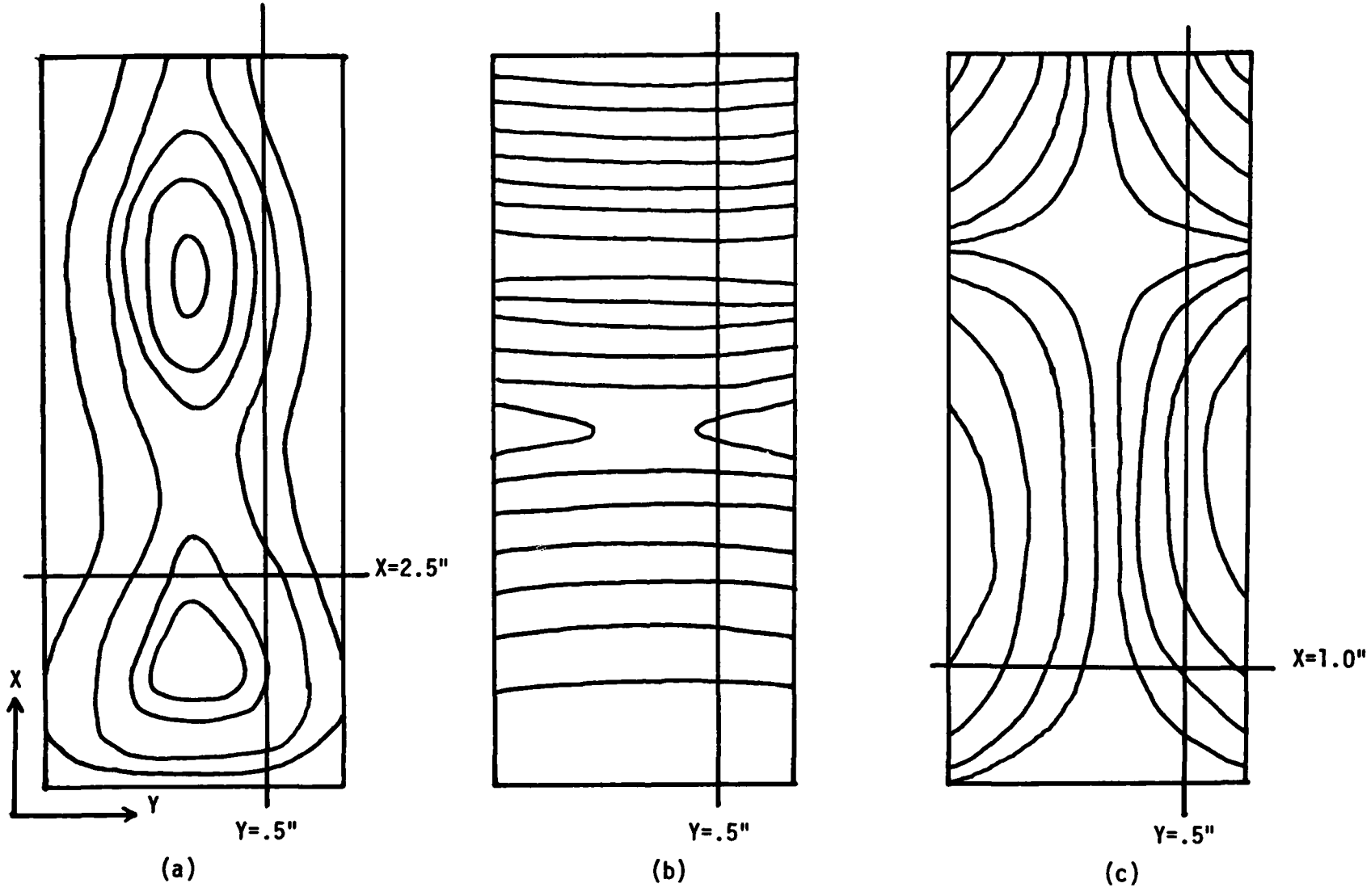


Figure 8. Schematic of Hologram Data Tracks Input to HOLOCURVE (a) Lyre Mode (b) Second Bending Mode (c) Second Torsion Mode

size of the cursor, it was considerably easier to simply move the cursor across a fringe, count the number of cursor movements over that distance and then backtrack to put the cursor in the fringe center.

Automation of data extraction was considered. If only dark fringe data was required for input to HOLOCURVE, automation would have definitely been appropriate. Unfortunately, nodal data, that is, coordinates of the white fringes indicating zero out of plane displacement, was also necessary. It was this requirement that precluded the use of cursor tracking automation for holography data extraction.

V. DISCUSSION OF RESULTS

Heat Transfer Comparisons

Microcomparator fringe displacement data and video digitizer data from the same interferograms was reduced with common computer software. Accuracy of the video digitizer proved to be acceptable and the time savings involved was remarkable.

Local heat flux was calculated at points along a given wall of the test cell. Numerical integration of these local values gave the total approximated heat flux at the wall. Figure 9 and Figures B-1, B-2, and B-3 in Appendix B compare plots of local heat flux for four interferograms. Correlation is reasonably consistent although the video digitizer results show much more randomness than those of the microcomparator results. A careful study of particular data sets in each figure reveals that a consistent trend is apparent. Although some of the microcomparator data seems to neatly fit a "smooth" curve, it is not necessarily correct. The microcomparator method shows less scatter but error is still very much apparent. The greater scatter of the video approach is believed to be due to the finite cursor size problem mentioned earlier. See Section IV for this discussion. Accuracy could have been improved if a full 512 x 240 picture were available. The finer dimension of the x cursor coordinates could have been exploited.

It is important to note that the microcomparator is not proven to be completely accurate. It only represents the acceptable method for extracting data from heat transfer interferograms at the Air Force Institute of Technology. Using that device, a great deal of time, approximately e and one-half hours, is spent in taking data at every fringe location for one wall on one picture. Then the data must be input to a computer by hand which takes

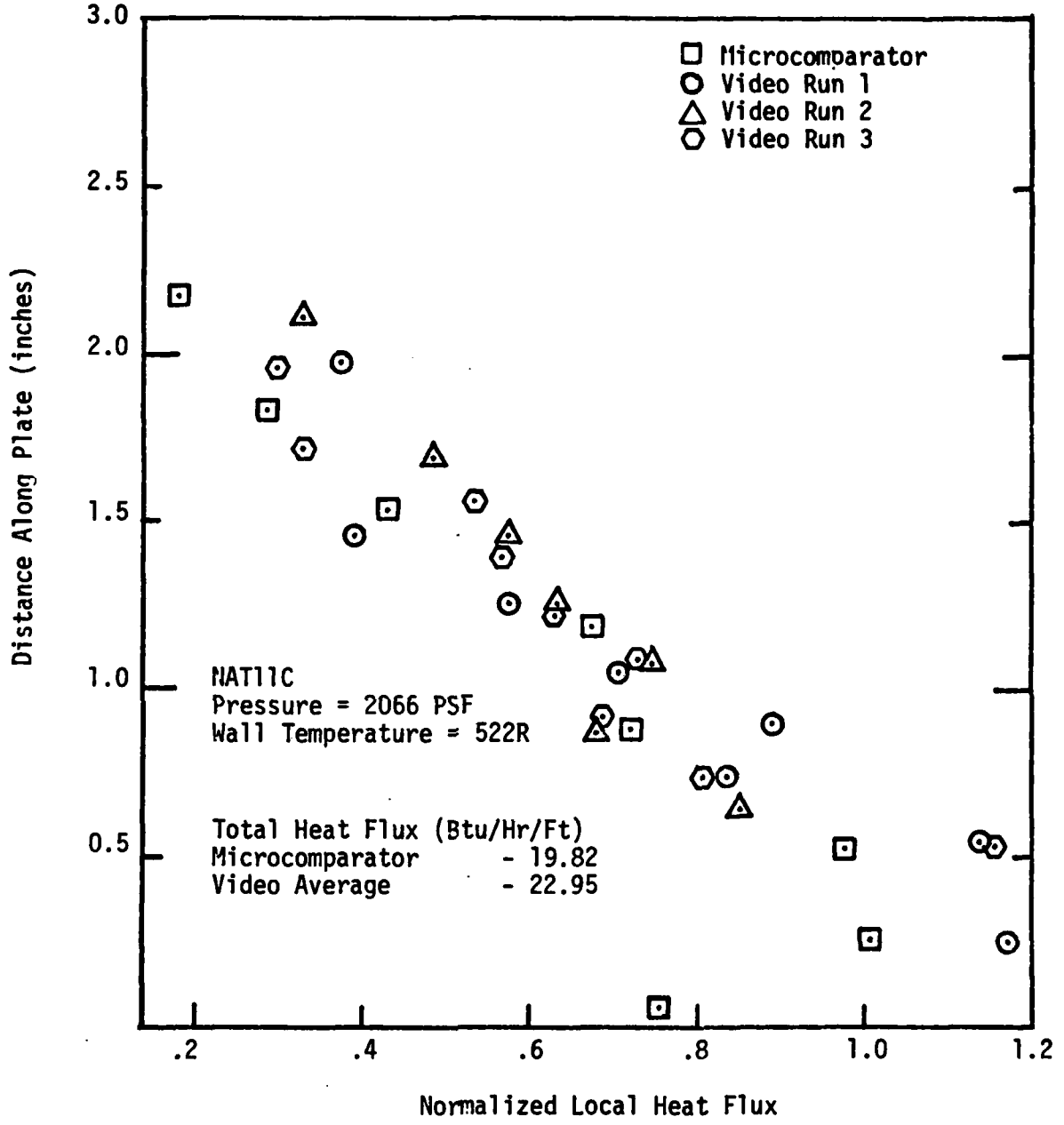


Figure 9. Local Heat Flux of a Cold Plate with Vertical Divider in the Test Section. (Normalized to Maximum Microcomparator Heat Flux)

more time. Because of this time factor, data is not normally taken at every fringe and each picture is only processed once.

With the video digitizer, data at every fringe on one wall can be taken in absolutely no more than fifteen minutes for this application. The coordinate readings of displacements are measured less accurately than the microcomparator but more data can be taken, more than once. This affords an added feature of data quality control. Several data sets can be taken for any picture, the data reduced, and results averaged for a more acceptable result all within half the time it takes to reduce data with a microcomparator.

A comparison of calculated total heat flux for the several interferograms is presented in Table I. The video digitizer results represent the average of three data sets and shows the consistency of the video approach. Three of the four cases are definitely within user specified acceptable limits. The third picture, NAT11C, shows more error than would normally be accepted. Theoretically, for this picture, the heat transfer of the cold (NAT11C) and hot (NAT11H) sides should be approximately equal. Assuming only small heat losses in the test cell itself, the video data seems much more consistent than the microcomparitor result.

Based on the above comparisons, users of the heat transfer facility selected the video digitizer as an acceptably accurate, more convenient, and more rapid approach to data reduction than the microcomparator. DIGITIZE and its associated hardware is currently being used for remaining heat transfer calculations necessary in that effort. To date, no problems have been identified by the user.

Holographic Comparisons

The video digitizer was used to collect data from three pictures of time averaged holograms. See Section IV of this report. Data was input to

TABLE I

Calculated Total Heat Flux Comparisons of Four Interferograms

Interferogram	Data Source	Total Heat Flux (Btu/Hr/Ft)
NAT11C (Figure 9)	Microcomparator	-19.82
	Video Run 1	-23.07
	Video Run 2	-23.04
	Video Run 3	-22.74
	Video Average	-22.95
NAT11H (Figure B-1)	Microcomparator	25.96
	Video Run 1	26.41
	Video Run 2	26.30
	Video Run 3	27.05
	Video Average	26.59
NAT13C (Figure B-2)	Microcomparator	-22.86
	Video Run 1	-24.08
	Video Run 2	-22.28
	Video Run 3	-23.36
	Video Average	-23.24
NAT13H (Figure B-3)	Microcomparator	26.18
	Video Run 1	24.48
	Video Run 2	29.49
	Video Run 3	28.70
	Video Average	27.56

HOLOCURVE and compared to results obtained from data gathered using an overhead projected image of the test blade. Normal displacements and bending strains compare quite satisfactorily for the two approaches.

The original HOLOCURVE data obtained by McBain was the result of projecting the hologram picture on a screen and picking coordinates off the exploded view. Parallax error with this approach can be significant. It is not an absolutely accurate system. Therefore, comparisons to that method using the video digitizer only show the relative merits of the approach to an accepted standard.

Two of the holograms reduced are shown previously in Figure 3 and the third in Figure 10. Figures 11 and 12 present graphic comparisons of McBain's results and those of the video digitizer. Additional results are presented in Appendix B. In Figure 11, lyre mode displacements at constant chord location show that the video results tend to be somewhat less than McBain's although the bending strains correlate very well in Figure 12. The displacement differences are possibly the result of a combination of questionable picture quality, and less than optimum reference dimensions used for reduction. The span length was used as the reference dimension. Note, however, that fourteen data points were input to HOLOCURVE versus seven used for McBain's results. This affords more data for a better curve fit. It is difficult to conclude which is more accurate but the correlation is reasonable. The data from these figures represents the worst correlation of any comparisons made to McBain's results. Remaining lyre mode results in Appendix B also indicate error while both the torsion and bending results are excellent. This supports the possibility of poor picture quality of the lyre mode of vibration.

Comparisons made of the second bending mode are more representative of the video digitizer's capability. Figure 13 shows excellent correlation of

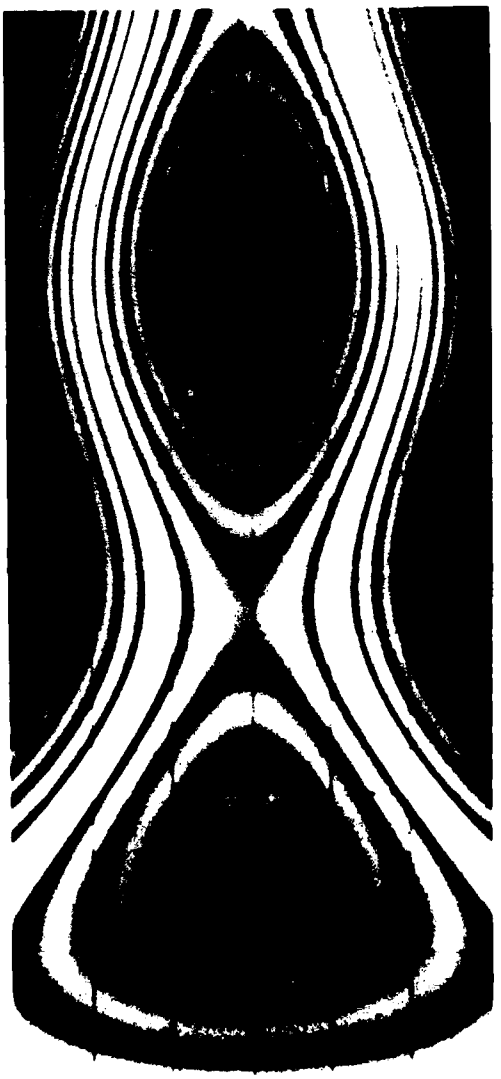


Figure 10. Lyre Mode of Vibration for a 3 inch x 7 inch Plate

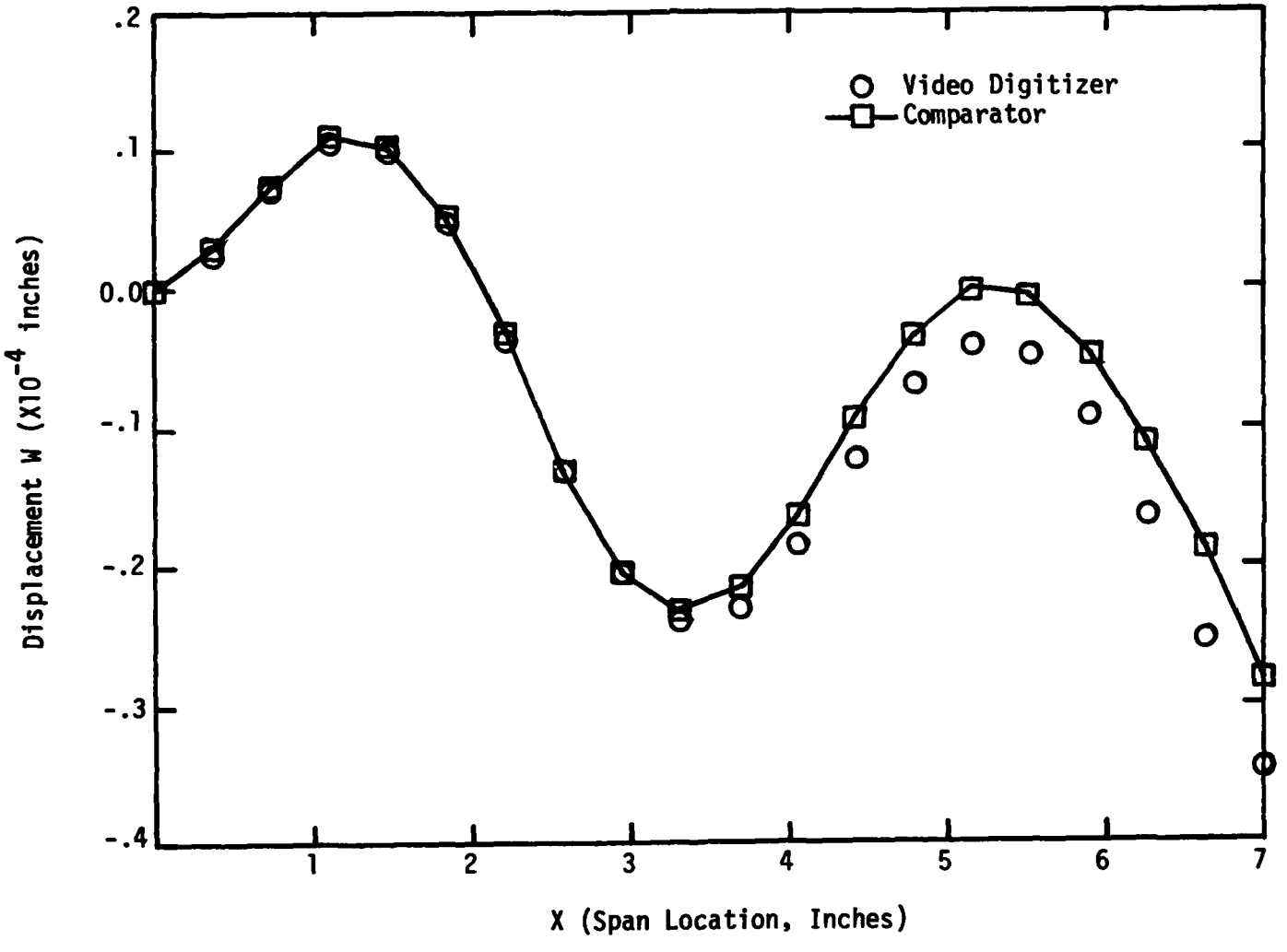


Figure 11. Lyre Mode Out of Plane Displacement W at Constant Chord Location, Y=.5 inches

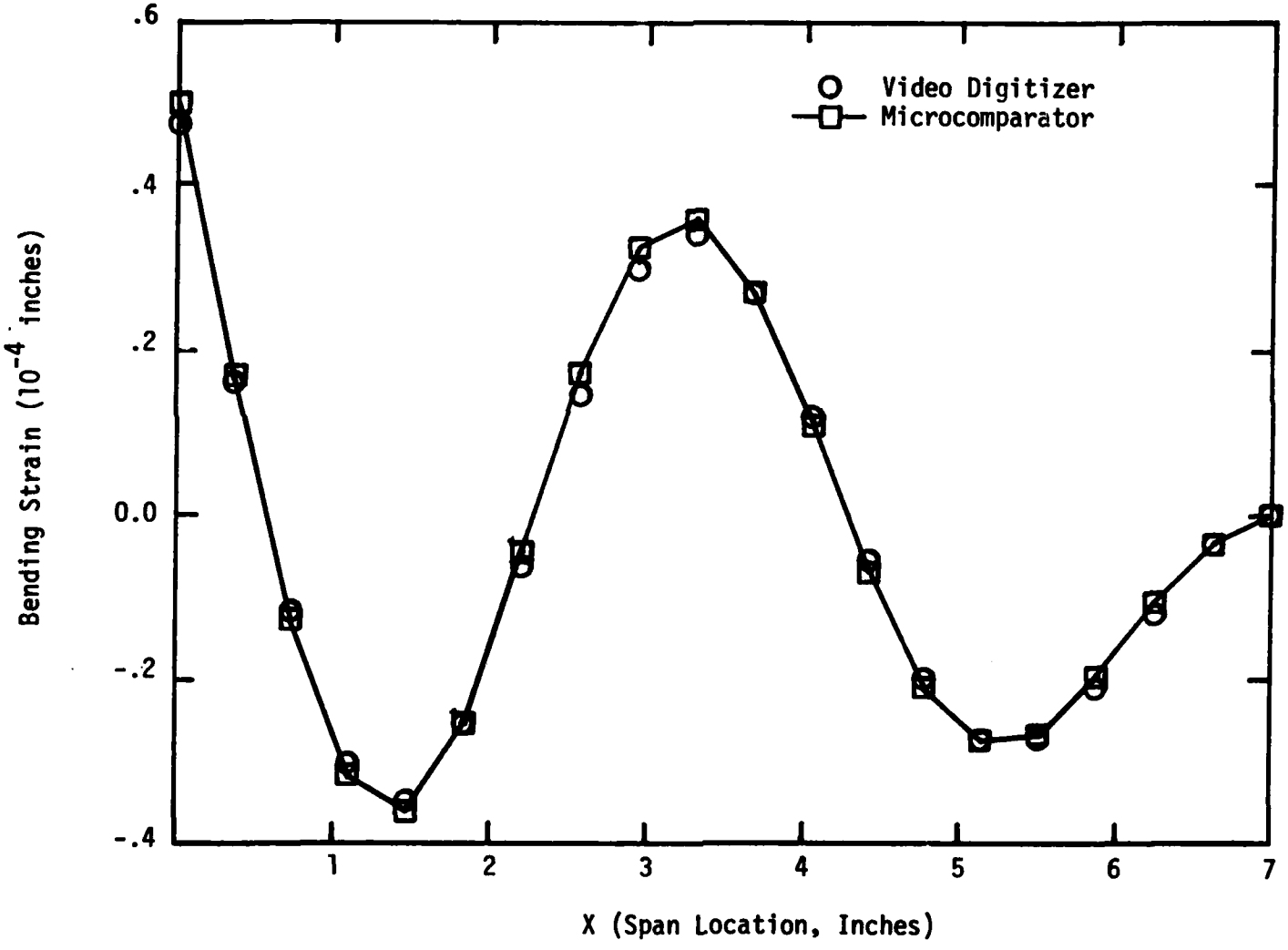


Figure 12. Lyre Mode Bending Strain at Constant Chord Location, Y=.5 inches

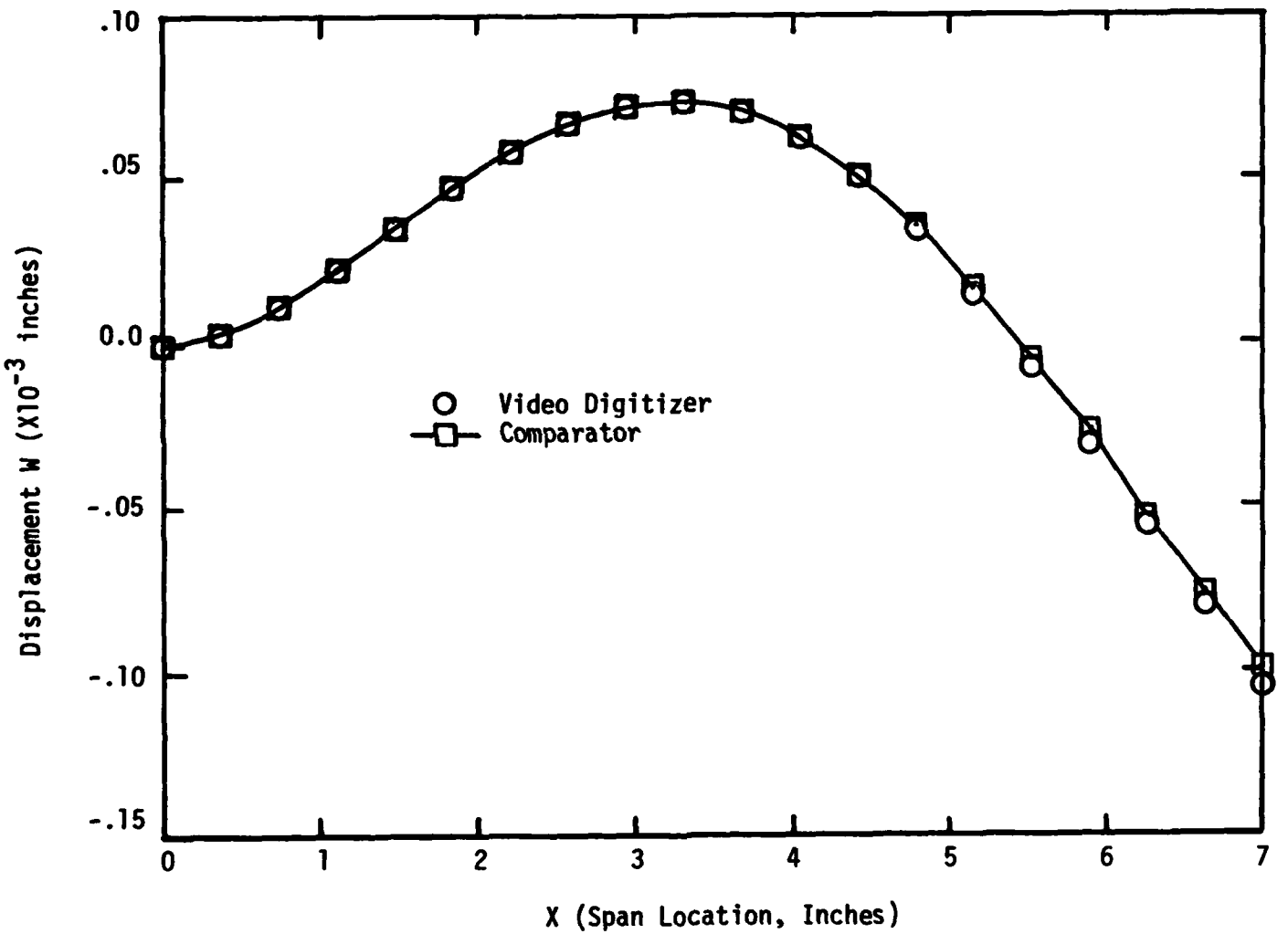


Figure 13. Second Bending Mode Out of Plane Displacement W at Constant Chord Location, Y=.5 Inches

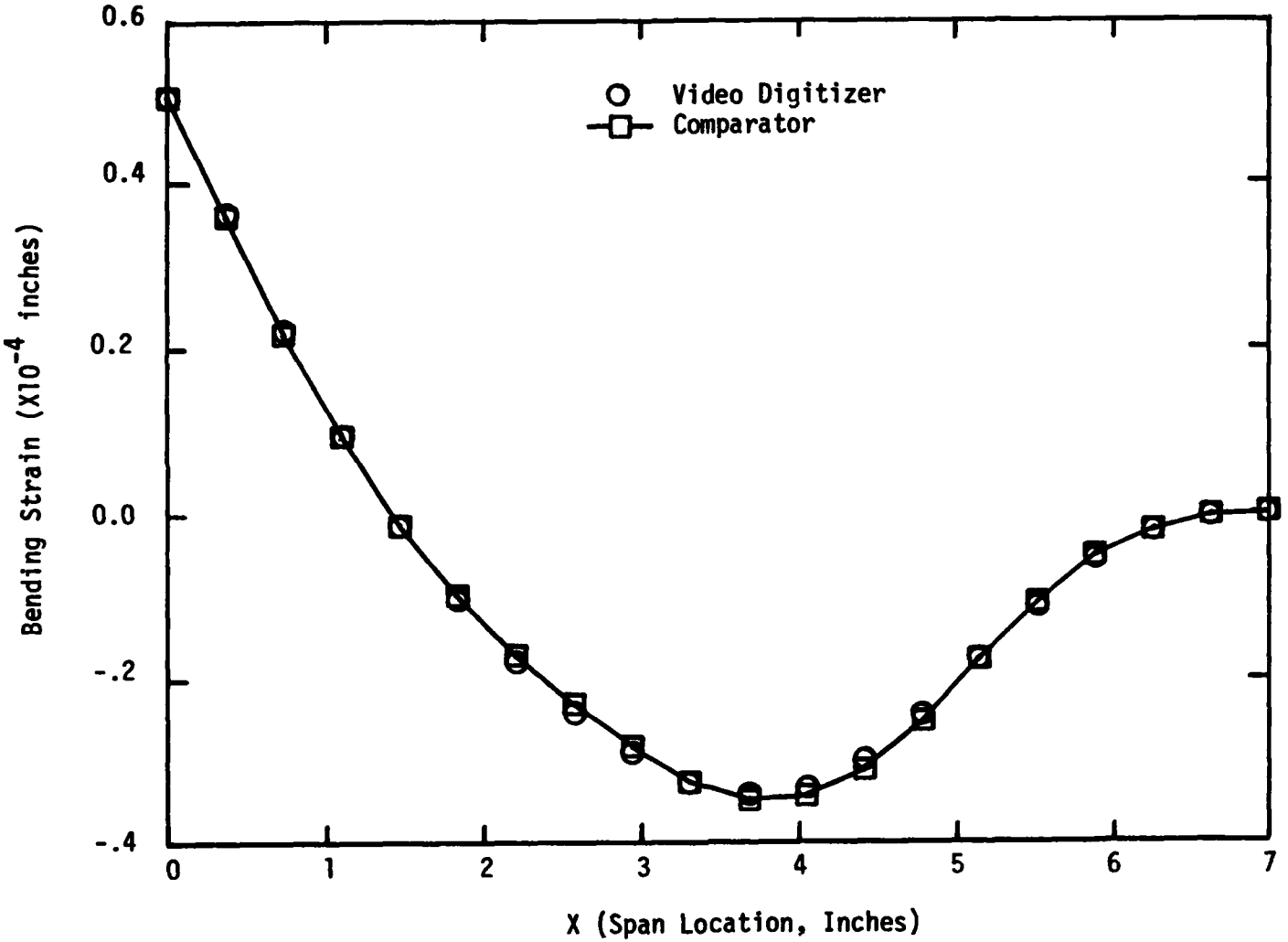


Figure 14. Second Bending Mode Bending Strain at Constant Chord Location, Y=.5 Inches

blade displacement calculations. The video approach produced essentially equivalent results to McBain's method. Likewise, the curvature results in Figure 14 are very much in agreement. Torsion comparisons in Appendix B support the video accuracy.

Pictures of holograms from further Aero Propulsion Laboratory testing may soon be available for additional verification of the video approach. Since the new photographs will be the same as those reduced on an optical comparator, instead of technical report reprints of pictures, comparisons of the two data acquisition systems should prove even better correlation.

Once again, the video approach proved to be much more efficient than the current methods employed to extract data from optical instrumentation images. McBain has estimated that approximately 45 minutes to an hour were required to read data for one picture. Using an optical comparator, Lieutenant R. Koop spends nearly one hour per picture. The video digitizer consistently took no more than 15 minutes and frequently as little as five minutes per picture.

The future possibility of freezing the hologram directly on the computer system instead of using a photograph is an extremely positive step towards rapid, on-location, data reduction of an optical instrumentation source. It would eliminate error induced by photography methods and allow immediate feedback of engineering results.

VI. SUMMARY AND CONCLUSIONS

Using a video digitizer to extract coordinate data from pictures or images is not only feasible, it is a proven method. The digitizer developed for this effort is a user oriented, interactive software package that can be used within the confines of the hardware restrictions specified in Section III and Appendix A. It has been accepted and is being used by researchers requiring a digitizer for heat transfer interferograms.

A comparison of heat transfer rates using a microcomparator and the video digitizer show correlation to within four percent. The video digitizer involves 80 percent less time than the microcomparator and maintains acceptable accuracy.

Bending strain and out of plane displacements calculated with video data agree quite well with the results of data taken from an exploded view of vibration test holograms. Once again, time involved using the video digitizer was substantially less than the time spent in extracting data using a comparator.

The video digitizer can digitize photographs of holograms, interferograms, etc. allowing displacement data extraction using a cursor device. Currently, it does not have optics capability to digitize direct images from visual data systems. Its verification has been proven acceptable for pictures on the order of a few inches in size. Its use on larger images from smoke tunnels, manometer boards, and the like will depend on improved optics capability.

Two resolutions of pictures are available although the 512 x 120 resolution is considerably more dependable and accurate than the 256 x 240

setting. Memory space only allows half of a monitor image at the higher resolution. This can be inconvenient for a user, but it does not mean that the 512 setting is not useful.

The total hardware and software package is already being used for further heat transfer data reduction. Its time savings will free the researcher and engineer so that more effort can be spent analyzing results instead of obtaining them. Some further advancements will improve the utility of the video digitizer such that data acquisition from optical systems will become extremely convenient and accurate for producing consistent, reliable engineering results.

VII. RECOMMENDATIONS

As it currently exists, the video digitizer developed in this effort, is usable for engineering data reduction. Several areas of improvement should be further investigated to develop this system to its full potential.

(1) Much of the perceived error in the digitizer stems from picture distortion due to the parallax of the camera. A detailed study should be conducted to develop the digitizer optics in two ways. First and foremost, investigate methods of reducing the distortion when attempting to digitize a photograph. Second, piece together an optics system whereby one can directly process the actual optical image (schlieren, hologram, etc,) without having to rely on photographs of the image.

(2) Since the digitizer has proven its worthiness and acceptability as an experimental data reduction device, effort should be undertaken to expand its memory capability. The TecVI central processing unit (CPU) will not allow this. Perhaps a more capable CPU in the TecVI or another microcomputer equipped with the TecMar video boards would accomplish this. A full 512 x 240 video image should then be possible.

(3) The compiler used to build DIGITIZE is not as versatile as other computer languages tend to be. Although FORT//80 can be easily manipulated by a user to build additional subroutine modules into DIGITIZE, the more versatile option will be preferable for future expansion. DIGITIZE should therefore be rewritten for a new compiler such as Microsoft fortran F80. This means, however, that the picture residing in memory must be protected while optimizing remaining memory usage.

(4) Regardless of a picture's clarity, system noise will invariably cause irregular gray patterns on the image. It is recommended that Cromer's logic for averaging a picture be inserted as an option to DIGITIZE.

(5) Software was not developed in this effort for the cursor to seek out fringe lines or to follow fringes. The data reduction did not require it. Logic for this capability is straightforward and can be easily added to DIGITIZE as an additional user menu item. This advancement is dependent on the accomplishment of item (4).

(6) Documentation from TecMar, Inc. has revealed that hardware restrictions of the RT + MI boards are responsible for random vertical lines in the monitor display. Modifications to the boards can resolve this problem. These modifications should be made to the RT + MI boards. Documentation for these changes will remain with the TecVI system.

Bibliography

1. Bohl, Marilyn. Information Processing (Second edition). Chicago: Science Research Associates, Inc., 1976.

2. CP/M™ Operating System Manual. Pacific Grove, California: Digital Research, 1982.

3. Cromemco Single Card Computer Instruction Manual. Mountain View, California: Cromemco, Inc., 1980.

4. Hogan, Thom. Osborne CP/M™ User's Guide, Berkeley, California: McGraw-Hill, Inc., 1981.

5. Holman, J.P. Experimental Methods for Engineers. New York: McGraw-Hill Inc., 1971.

6. Khan, M. Khalid. A Study of Free Convection in an Inclined Rectangular Cavity. MS Thesis, Wright-Patterson AFB, Ohio: Air Force Institute of Technology, December 1980.

7. McBain, James C. Quantitative Displacement and Strain Distribution of Vibrating Plate-Like Structures Based on Time Average Holographic Interferometry. AFAPL-TR-77-44. Wright-Patterson AFB, Ohio: Aero Propulsion Laboratory, July, 1977.

8. Real Time Video Digitizer Gray Level/Graphics Monitor Interface. Cleveland, Ohio: TecMar, Inc., 1978.

9. TM-2001A. FORT//80, Fortran IV Compiler, Fortran IV Language Manual. Cleveland, Ohio: Realistic Controls Corporation, March, 1977.

APPENDIX A

DIGITIZE USER'S MANUAL

DIGITIZE is an interactive video digitizer program designed to expedite the process of extracting coordinate data from optical information systems. It has been proven effective for several data acquisition processes but it is suggested that any potential user compare results of DIGITIZE to those obtained by traditional means. The accuracy of coordinate data is a function of the size of the picture or object to be digitized and the resolution of the computer image. DIGITIZE works very well for object sizes on the order of several inches. Images of larger size exhibit much more distortion strictly due to video limitations.

Part I is intended to provide the user with a functional introduction to DIGITIZE. It includes a discussion of menu items and commands necessary to execute the program for digitizing video images and extracting coordinate or displacement data. This section also lists the equipment requirements on which the software currently functions. With some modification to the software, DIGITIZE can be used on other microcomputer systems that contain TecMar Real Time Video Digitizer Gray Level/Graphics Monitor Interface boards with direct memory access (DMA) capability. It is operated under a CP/M™ operating system (Ref 2).

Part II of the manual gives instructions for compiling DIGITIZE with a FORT//80 compiler (Ref 9) and lists and describes the program in detail. The program listing appears in twelve pitch, letter gothic type which is compatible with a Compuscan Alpha Word III, optical character reader (OCR). This OCR device can scan the listing of DIGITIZE from Part II and place it on an eight inch floppy disk.

DIGITIZE is based on a software package called VIDISOFT, Copyright, 1980. It was developed by Mr Kenneth Stern for TecMar, Inc., Beachwood, Ohio.

NOTE: DIGITIZE can be used by a novice but some understanding of CP/M and microcomputers is suggested. References 1 and 4 are excellent beginner guides.

LIST OF FLAGS

FROZEN = 0	Image is not frozen on screen (i.e., DMA is disenabled)
= 1	Image is frozen on screen (DMA is enabled)
IFLAG = 0	Picture reference dimensions have not been established
= 1	Picture reference dimensions have been established
NEWCMENU = 0	The user menu is not to be displayed on the terminal
= 1	The user menu is displayed
ORIGIN@FLAG	This flag keeps a running count of the number of times the user has established an origin for any given picture
WRITE@FLAG = 0	No function associated with WRITE@FLAG=0
= 1	Reference dimension print flag (see CURSOR@MENU)
= 2	Reference dimension print flag (see CURSOR@MENU)
= 3	Reference dimension print flag (see CURSOR@MENU)
= 4	Reference dimension print flag (see CURSOR@MENU)
= 5	Irdicates to subroutines that an origin has been established on the picture
= 6	Indicates that SAVE or GET operate on a file type .DAT instead of .PIC
= 8	When miscellaneous data points are saved in STORE@DATA, this flag indicates two additional elements of array SAV be skipped for easier user identification of the extra data

PART I

Introduction to DIGITIZE

DIGITIZE is a modular computer program designed to allow the extraction of coordinate or displacement data from computer images of camera pictures. Two different resolutions of picture are presently available. They display either a full screen, 256 x 240 pixel, image with switch B of the DMA board set to 1.-ON, 2. OFF, 3.OFF, 4.OFF, or a 512 x 120 pixel image with the above switches all turned off (Ref 8). Figure A-1 shows these switch locations. Both resolutions create a computer image of 30,720 bytes which is stored in the upper 32K of computer memory.

Because of this reserved 32K picture area, DIGITIZE requires a 32K CP/M based operating system when exercised on a 64K microcomputer. Using a feature of the FORT//80 compiler, DIGITIZE conveniently isolates the picture memory image and precludes overwrites.

The heart of DIGITIZE is a 432 element array SAV( ). Any time coordiantes of a data point are saved, they take their place in the I + 1 and I + 2 locations of SAV. Therefore, the user must establish a procedure to identify the location of each coordinate set. Starting at the end of SAV, the array is back filled with a record of the number of times the origin is set for a picture, followed by the x and y displacements of each subsequent origin from an established reference (the first origin).

No working knowledge of the software is required unless the user wishes to implant a subroutine module to manipulate or reduce the data. In this likely event, the user should reference Part II of this manual.

Equipment and Software Requirement for Current DIGITIZE Version

TecMar TecVI Video Digitization System

TV Camera

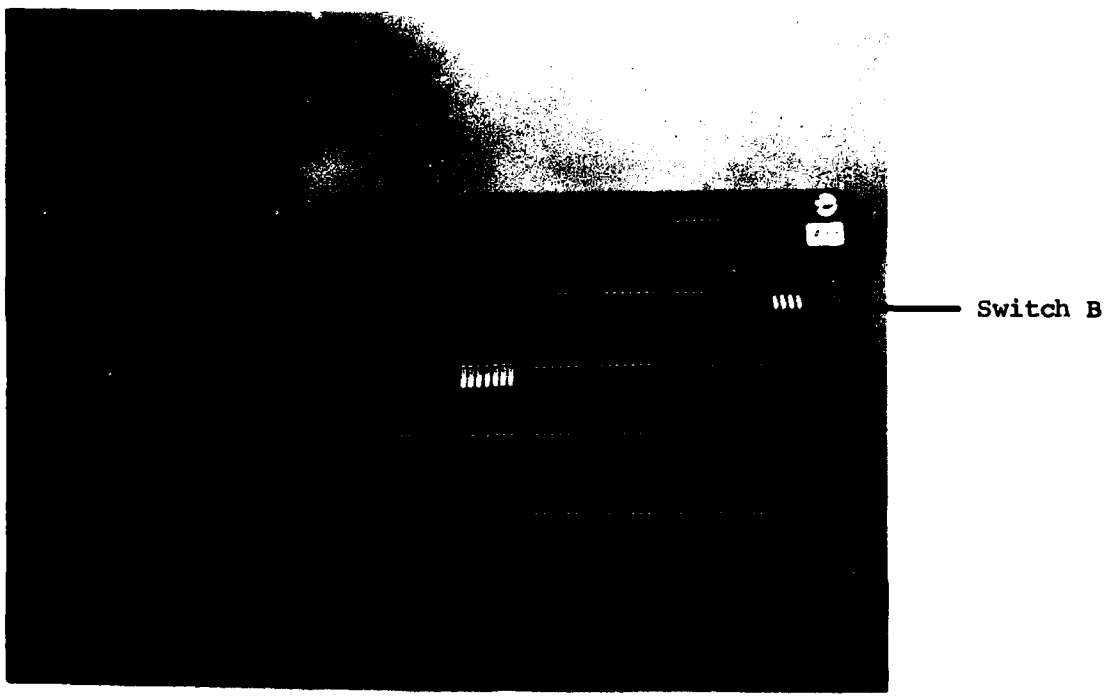


Figure A-1. Location of Resolution Switches on the DMA Board

Black and White Display Monitor

Printer

32K CP/M V2.2

Utility Program FRUN (FRUN.COM)

FORT//80 Compiler (FOR.COM)

Dynamic Debugging Tool (DDT.COM)

DIGITIZE.COM, DIGITIZE.FOR

How to Exercise DIGITIZE

DIGITIZE accesses stored pictures and data using disk drive B. It is ordinarily convenient to use drive A for the main program and drive B solely for data storage. After inserting the disks and booting the system, type the command:

FRUN <CR>

When the prompt returns, type:

DIGITIZE <CR>

The user is first prompted to input the horizontal resolution of the picture. This number (512 or 256) must correspond to the setting of switch B on the DMA board in the computer as discussed previously. Input the resolution and <CR>.

The following menu will now appear:

SELECT FUNCTION:

<D>ISPLAY PICTURE

<F>REEZE PICTURE

<S>AVE PICTURE ON DISKETTE

<G>ET PICTURE FROM DISKETTE

<C>URSOR CONTROL

<P>RINT GREY CODES AND COORDINATES

<R>EFERENCE PICTURE DIMENSIONS

<O>RIGIN RESET OPTION

S<A>VE DATA IN ARRAY

S<T>ORE DATA ARRAY ON DISK

G<E>T DATA ARRAY FROM DISK

<W>RITE SAV( ) ARRAY ON PRINTER

<H>EAT@TRANSFER DATA REDUCTION

E<X>IT TO OPERATING SYSTEM

By typing a single key (< shown in brackets >) for any menu item, the user may execute the options of DIGITIZE.

1) DISPLAY PICTURE; Type: D

Display enables direct memory access allowing simultaneous digitization and display of any image seen by the camera.

2) FREEZE PICTURE; Type: F

Freeze stores the current video picture in the upper 32K memory block of the 64K computer capacity. The image is frozen on the monitor.

Note: When using the 512 pixel/line resolution, only the EXIT TO OPERATING SYSTEM menu item will function after FREEZE. Use EXIT to return to the operating system. Re-enter by typing FRUN<CR>and DIGITIZE<CR>. The frozen image will appear once again. Now any menu item may be selected. It is believed that this problem is a result of hardware restrictions in the TecVI system. The problem does not exist at the 256 pixel/line resolution.

3) SAVE PICTURE ON DISKETTE; Type: S

This option will prompt the user to input a filename up to eight characters in length, followed by <CR> . The currently frozen image will be stored on disk B in the name format: filename.PIC.

4) GET PICTURE FROM DISKETTE; Type: G

GET prompts the user to input the filename of a picture on disk B to be displayed on the monitor. Only the filename need be input followed by <CR>. The extension .PIC need not be included. GET always resets the SAV array to element SAV(1).

5) CURSOR CONTROL; Type: C

Cursor displays a blinking cursor the size of two horizontal pixels. Each pixel on the screen contains a gray value of 0 to 15 where 0 indicates black and 15 white. A selection of cursor movement options appears: Left, Right, Down, Up. Use the appropriate key to move to any desired location. Hit the space bar to return to the main menu selection. Cursor can be used any time without destroying memory content.

6) PRINT GREY CODES AND COORDINATES; Type: P

This option can be used anytime without destroying memory content. The output is as follows:

PIXEL	ROW	COL	GRAY	X	Y
5159.	110	39	0	0.000	0.000
	110			0.000	0.000
5160.	110	40	15	0.000	0.000

The cursor, made up of two pixels, is sketched in Figure A-2.
"Pixel" represents the pixel number ranging sequentially from zero to 61,440. Pixel 1 lives in the upper left corner of the picture and pixel 61440 is in the lower right corner. Row and column provide a two-dimensional location of the cursor with row 1, column 1 in the lower left picture corner. If reference dimensions have not been established for the current picture,

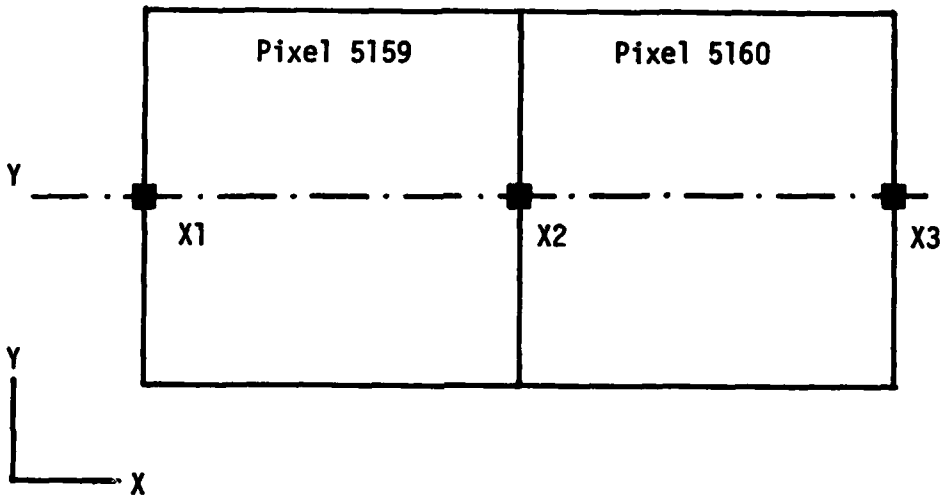


Figure A-2. Cursor Schematic

x and y coordinates will always read 0.0, 0.0. Once references have been set using menu item <R>, the coordinates will be displayed as units in the

X and Y directions from the current origin as set by the user in menu items <R> or <0> . In this example schematic, pixel 5159 lies in row 110, column 39. The coordinates (X1,Y) at the left edge of the cursor are in the top row of the print information. The center of the cursor (X2, Y) shows only row and coordinate information. The right pixel, 5160, column 110, row 40, shows the gray value of that pixel and displays coordinates (X3, Y). This arrangement allows the user more accuracy when measuring the X coordinate of any cursor location.

7) REFERENCE PICTURE DIMENSIONS; Type: R

In order to determine coordinates of a point, the computer must know the size of the picture. Selection of <R> prompts the user to move the cursor across a horizontal distance of know dimension from left to right and then a known vertical distance from top to bottom. Then the user moves the cursor to the location of the desired origin (X,Y) = 0.,0. A choice of X location, X1, X2, or X3 is prompted. These are the X locations of the left, center, and right data points of the cursor as explained in menu item <P> .

8) ORIGIN RESET OPTION; Type: 0

The user can reset the origin (X,Y) = 0.,0. anytime, anywhere after the <R> menu item has been established for a given picture. The user prompts are self explanatory. All coordinates displayed after the <0> option will show (X, Y) displacements from the most recent origin.

9) SAVE@DATA IN ARRAY; Type: A

SAVE DATA stores an X and Y coordinate value in a one dimensional array named SAV, of dimension 432. The user must again select X1, X2, or X3. SAV( ) is filled according to the format SAV(n) = X, SAV(n+1) = Y. The dummy variable "n" is incremented automatically.

The bottom portion of SAV is backfilled in the following manner:

SAV(432) = Number of times the origin has been set by the user.

SAV(431) = X value of the original origin showing its distance from the lower left corner of the computer image.

SAV(430) = Y value of the original origin showing its distance from the lower left corner of the computer image.

SAV(430-n) = X displacement of a particular origin from the first origin etablished (n=1,3,5,...)

SAV(430-(n+1)) = Y displacement of a particular origin from the first origin established.

For example, SAV(427) houses the X displacement of the third origin from the original origin. This information proves quite useful in tracking coordinate information of a picture.

10) STORE DATA ARRAY ON DISK; Type: T

This option allows the user to store array SAV on disk B. If additional data points are to be saved, they may be input after <T> is selected. Any data stored at this point, however, is referenced as coordinates based on the very first origin established by the user.

11) GET DATA ARRAY FROM DISK; Type: E

The user is prompted to input the filename of a data file living on disk B. An extension name .DAT is automatic and is not required as input. The SAV array is filled with data from this file exactly as it was stored originally by the user with the <T> option.

12) WRITE SAV( ) ARRAY ON PRINTER; Type: W

All 432 elements of the SAV array will be output to the printer.

13) HEAT TRANSFER DATA REDUCTION; Type: H

This subroutine is independent of the main program DIGITIZE. It is presented in this appendix to demonstrate the use of a user designed data reduction module. This particular routine was written to calculate heat transfer rates using X and Y fringe displacements of interferograms.

14) EXIT TO OPERATING SYSTEM; Type: X

This option terminates DIGITIZE and returns to the CP/M operating system.

PART II

This portion explains program alteration procedures and presents a listing and explanation of the program.

A. It will be desirable to write a data reduction routine as a module of DIGITIZE for various applications. Subroutine HEAT@TRANSFER is an example of such a routine.

DIGITIZE is written in FORT//80 which is a subset of Fortran IV. With some exceptions, a program written in standard Fortran IV or F80 will be executable with the FORT//80 compiler (Ref 9). It should be noted that FORT//80 is very sensitive to mixed mode arithmetic. It allows larger variable names and admits multiple replacements (e.g., A=B=C=0.0). In particular, this compiler provides the capability to designate computer memory allocation through the use of the COMPILER( ) function. When constructing a data reduction module, it is reasonably safe to start with a Fortran IV routine and alter it as necessary to accommodate FORT//80. Note that any program compiled under FORT//80 must be preceded by execution of the run time routine, FRUN.COM.

Any data reduction module(s) should be located immediately prior to DIGITIZE@MAIN. Besides implanting the reduction module into DIGITIZE, appropriate subroutine calls/labels should be added to subroutines DIGITIZE@MAIN and MAIN@MENU. No other alterations to DIGITIZE are required.

B. Compiling the Expanded DIGITIZE Program

Step 1. Type the following at the terminal:

FOR filename = filename/N <CR>

where filename is up to eight characters in length. Do not

include the extension .FOR. Compile time is somewhat lengthy. Be patient.

Step 2. After the COMPILATION COMPLETE message is received, it is necessary to load the .HEX version of filename into computer memory. To do this, type:

DDT filename.HEX <CR>

The computer will respond:

DDT VERS 2.2

NEXT PC

2E94 0100

- _

At the prompt (- _) type:

GO <CR>

Now save the executable .COM file on disk. Type:

SAVE XX filename.COM <CR>

xx should be the number of pages required. See a CP/M manual for instructions on how to determine this number (Ref 2).

NOTE: If there were any errors detected during compilation, look at the filename .PRN file to locate the errors. Correct them in editor then recompile. Errors are numbered and their codes are defined in Ref 9.

A typical compile session is shown below.

A> FOR DIGITIZE = DIGITIZE/N

UTI FORT//80 COMPILER 3.2G

PHASE 2

00 ERRORS

COMPILATION COMPLETED

```
A> DDT DIGITIZE .HEX
DDT VERS 2.2
NEXT PC
2E94 0100
-GO
A> SAVE 46 DIGITIZE.COM
```

C. Subroutines are briefly described. Reference the full program listing at the back of this section for further detail.

Program Digitize

The opening segment of the program designates specific memory locations for variables using the FORT//80 COMPILER( ) function. It is COMPILER( ) that allows the user to conveniently locate variables or arrays within memory so that overwrites do not occur. For example, SCREEN is the array that contains picture information. COMPILER(3) sets the end location of SCREEN at Memory address F7FF Hexidecimal (i.e., 64K). The 30K bytes of picture will always live at that address in memory until the user so specifies with another COMPILER( ) statement. Variables are defined in the discussion of subsequent functions and subroutines.

CURSOR@KEY

KEY = Variable which contains the hexidecimal value of alphanumeric terminal input.

Function CURSOR@KEY takes an integer value corresponding to Up(U), Down(D), Right(R), Left(L), or space(''). Cursor movements are assigned here.

CHAR@IN

KEY = Alphanumeric hexidecimal value of terminal input.

CHAR@IN is a continual loop called from several routines. When called, the function remains in the do-nothing loop until a terminal input key is recognized. The information is returned to the calling routine.

LEFT@HALF/RIGHT@HALF

POINTER = The location of a particular byte of picture data ranging from 1 to 30,720.

When called, these routines sort out the left or right half of the cursor and return the appropriate gray information to the calling subroutine.

DISPLAY

N = counter which sets current location in SAV data array.

IFLAG = Indicates whether or not reference dimensions have been established. For a new picture, IFLAG=0.

SAV( ) = Data storage array. It is zeroed with each new picture.

This subroutine enables the direct memory access such that a continually updated computer digitized image appears on the monitor. FROZEN=0 insures that the image is not to be frozen on the screen or in memory while in DISPLAY mode.

FREEZE

FROZEN = variable used to flag the status of the monitor image.

FROZEN = 1: Picture frozen

= 0: Display mode

Outputting the hexidecimal value 0A0H to port 0FFH freezes an image on the monitor. Later segments output 00H to port 0FFH to turn the image off.

SWAP

TEMP = Temporary storage variable

SAVE@BYTE = Temporary storage location for picture gray data.

Swap is called by other routines to exchange picture gray value data with the black and white cursor. Therefore, when the cursor is moved, swap must be called once again to replace the old picture data to memory at the proper position in SCREEN( ). Swap can be used to leave a cursor mark if desired. SET@ORIGIN does this so the user knows where the origins are set on a picture.

FLIP

TEMP = Temporary Storage

CURSOR = Byte location of the cursor ranging from 1 to 30,720.

Flip alternates the cursor shade from black to white so it can be seen easily on the monitor. The short do loop merely slows the alternation from black to white for viewing convenience.

MOVE@UP/MOVE@DOWN/MOVE@LEFT/MOVE@RIGHT

HOR@RES = The horizontal resolution of the picture which can be 256 pixels or 512 pixels

HOR@RES@HALF = horizontal resolution in bytes.

DUMB = Dummy variable (function of HOR@RES)

These four routines move the cursor when called, in any of four directions. The IF checks are used to insure that the cursor stays within the picture boundaries.

DISK I/O PRIMITIVES

INLINE is a function of FORT//80 which allows direct communication with the operating system. A CP/M manual will explain the function of these primitives quite well. The subroutine names are self explanatory except SET@DMA and SET@DISK@BUFFER@DMA.

A shortfall of FORT//80 is that there is no function such as WRITE to directly store information on disks. Instead, record blocks of 128 bytes must be written directly to disk, one record block at a time. SET@DMA sets the

origin of any general 128 byte record block so that when READ@FILE or WRITE@FILE are called, the correct record is transferred from or to the disk. SET@DISK@BUFFER@DMA sets the origin of the record block back to the operating system default address which is the location of the first element of the 128 byte array DISK@BUFFER. This record block length variable is used for directly storing and retrieving picture data from disk.

GET@FILE@NAME

FILE@NAME = An eight element, single byte array which houses an input filename.

FCB = File Control Block: This 33 element array is recognized by CP/M as an information array containing status and information of a given file.

GET@FILE@NAME gets the file name of a picture or data file to be read from or stored onto disk B. Note that OUTPUT(0FFH)=00H turns off the picture of the monitor (disenables DMA). This must always be done when accessing another device port such as a monitor, printer, etc. OUTPUT(0FFH) = 0A0H turns the picture back on in later routines.

READ(40H) will read data from the terminal at port 40 hexidecimal. If another computer is used, this address must be changed to the appropriate port address.

SAVE

SAV@BUFFER = Transition array which contains one record block of data

START = The starting address of the 128 byte data block to be saved.

INDEX = Dummy computed to flag the location of the next byte of data to be saved.

STATUS = Error status flag returned from operating system interactions.

NEW@MENU = calls for new console menu display

SAVE replaces the normal WRITE function of standard Fortran. It stores data from pictures or data files to disks in increments of 128 bytes, or one record block. SAVE must always be called to write to a disk.

GET

SAV@BUFFER,START,INDEX,STATUS = Same as in SAVE routine.

N = counter used to increment the SAV data array. For a new picture, SAV is set to SAV(1)

ORIGIN@FLAG = Counter which tracks the number of times the origin has been set.

GET acquires a picture or data file from disk B. Note that if the user is getting a data file (WRITE@FLAG=6), reference dimensions of the current monitor image remain unchanged (IFLAG=0) The SAV array and origin counter are always reset to zero for both a picture and a data file.

INITIALIZE@PROGRAM

Global variables are initialized here. Definitions are given in their respective subroutines.

MAIN@MENU

MAIN@MENU prints the user menu selection on the terminal when called. If data reduction modules are added to DIGITIZE, the name of the subroutines should be added to this list of selections.

CURSOR@MENU

WRITE@FLAG = See the list of parameters for possible values. This is a flag which, in this routine, controls terminal write statements to the user.

CURSOR@MENU prints a user menu on the terminal, offering a selection of cursor control directions. The various additional writes, controlled by values of

WRITE@FLAG, give further instructions when this subroutine is called by REF@DIMENSION and SET@ORIGIN.

CURSOR@CONTROL

KEY = Alphanumeric hexidecimal value of terminal input.

CURSOR@CONTROL is the controlling subroutine for cursor movement. After calling CURSOR@MENU to display the choices of cursor movement, this routine loops through a series of IF checks to determine which appropriate movement subroutine to call. Flip alternates cursor from black to white to black and NEW@MENU = 1 returns the original user menu display after CURSOR@CONTROL is complete.

GET@DATA

WRITE@FLAG = 6 indicates to GET that the filename extension .DAT is requested.

GET@DATA is the control subroutine for retrieving a data file from disk.

ROWCOL(Z1, Z2)

DUM = Dummy variable which is a function of HOR@RES.

Z1 = pixel location of left half of cursor

Z2 = pixel location of right half of cursor

IROW, ROW = Integer and real values, respectively, of the row in which the cursor resides.

COL = column location of the cursor as a whole byte.

COLZ1, COLZ2 = column locations of the left and right cursor pixels, respectively.

ROWCOL calculates the row and column locations of the two cursor pixels with column 1, row 1 residing in the lower left corner of the monitor image.

Note that the height of any picture is not 256 or 128 for the 256 x 256 and 512 x 128 pictures, respectively. The heights are actually 240 or 120.

This difference is because of a margin used to synchronize the picture on the monitor.

COORDINATES (X1, X2, X3, Y)

Xn = X coordinate of the left (1), center (2), or right (3) of the present cursor location.

Y = Y coordinate of the horizontal center of the cursor.

X@ORIGIN,Y@ORIGIN = X and Y coordinates of the reference origin with respect to the lower left corner of the picture.

COORDINATES calculates the cursor coordinates with respect to the most recent origin as established by the user in SET@ORIGIN or REF@DIMENSION. RCOL1, RCOL2, etc. are examples of changing variable types to accommodate the mixed mode sensitivity of FORT//80.

CHOICE (X1, X2, X3, Y1, X, Y)

When selecting an origin or saving a coordinate pair, a choice is given to the user of the left, center, or right of the present cursor position. CHOICE displays those choices to the user at the terminal and remains in a loop until an acceptable value of 1, 2, or 3 is input corresponding to the left, center, or right cursor location. It reads in the X1, X2, X3, and YX coordinates of the cursor and returns only the desired X, Y coordinate pair.

SET@ORIGIN

DUMMY = Dummy counter used to increment the location in the save array.

WRITE@FLAG = 5 A flag used in CURSOR@MENU

ORIGIN@FLAG = counter which keeps record of the number of times the origin has been set. Its most recent value always resides in SAV(432).

SET@ORIGIN enables the user to relocate the reference origin anytime it is desired while still maintaining an overall reference to the lower left corner

of the monitor image. The IFLAG check kicks out of SET@ORIGIN and informs the user if reference dimensions have not been established for the current picture. As mentioned earlier, SAV( ) is filled from the beginning of the array with desired coordinate values at any specified point, and is backfilled with origin coordinates of each origin established. A check is made in this routine which will inform the user if the SAV array is filled beyond its 432 element capacity.

When completed, SET@ORIGIN has established a new origin (0,0), stored its location with respect to the original origin for later reference, returns a new user menu flag, and calls SWAP which leaves either a white or black reference mark on the image at the origin location.

REFDIMENSION

IFLAG = 1 Identifies to other subroutines that reference dimensions have been established for the current picture.

WRITE@FLAG = Identifies comments to the user in CURSOR@MENU.

Before coordinates can be identified, the computer must know how "big" the picture is. REFDIMENSION is one of the first menu items the user should select. It translates known dimensions of the picture into inches/pixel (IN@PIX). COORDINATES then uses cursor displacement and IN@PIXX (X dimension) and IN@PIXY(Y dimension) to calculate coordinates of a point. The left column of the cursor COLZ1 is used to determine cursor displacement in the X and Y directions.

PRINT

If reference dimensions have been set, PRINT outputs cursor data to the terminal. If no references exist, it outputs cursor data with all coordinates equal zero. Part I of this manual gives examples of PRINT output and cursor

data. After data is printed to the terminal, the user must hit the space bar to return to DIGITIZE with the menu display.

SAVE@DATA

WRITE@FLAG = 8 Indicates that the user has elected to store data on disk and any miscellaneous data points will be incremented in SAV( ) by an additional n + 2 location in the array.

SAVE@DATA provides a user the ability to save the x, y coordinates of one of the cursor data points in the SAV( ) array. The coordinates of the cursor location are calculated and the user is given a choice of the three x locations on the cursor. Checks are made to insure reference dimensions have been established and that SAV( ) is not full. The saved data is displayed on the terminal and a space bar returns to the main menu.

STORE@DATA

WRITE@FLAG = 6 indicates that the filename extension in SAVE should be .DAT.

STORE@DATA is called by the user to write SAV( ) onto disk B. At its initiation, the routine asks at the terminal if additional coordinates are to be saved. If so, these coordinates will be referenced to the first origin that was established. If further data is to be saved, SAVE@DATA is called with its normal user interface and WRITE@FLAG is flagged to eight.

After any additional points are saved in SAV( ), SAVE@DATA calls SAVE to write the data file to disk B.

PRINT@SAVE

PRINT@SAVE prints the current contents of SAV at the printer. Selection of this routine will not alter the global value of N, so any further data

saved will be done so in the N+1, location of SAV( ). The location of each data point is shown to the left of the printout for user convenience.

HEAT@TRANSFER

This subroutine is not an integral part of DIGITIZE. It is included here as an example for the user to see how to modularize a data reduction routine. HEAT@TRANSFER is specifically designed to manipulate data stored in SAV( ) for approximating temperature gradients at the wall of a plate of known temperature. Heat flux along the plate is then calculated.

Note once again that the READ command is dependent on the terminal port address 40H. Sample output is presented in Table C-III.

RESOLUTION

HOR@RES = user input horizontal resolution of picture in pixels/line

HOR@RES@HALF = calculated horizontal resolution in bytes

RESOLUTION reads the horizontal resolution of 256 or 512 as input by the user. These values are necessary in other subroutines to calculate coordinates, row, column, etc. The value input must correspond to the switch setting of the DMA TecMar board.

DIGITIZE@MAIN

DIGITIZE@MAIN is the controlling routine for DIGITIZE. A loop is set up which remains inactive until one of the user selected menu items is input to the terminal. At the conclusion of DIGITIZE, the DMA is disenabled (the picture turned off) which must occur to return to the operating system.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C            PROGRAM DIGITIZE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
             COMPILER(1) = 0100H
             COMPILER(2) = 0106H
C
C DECLARE THE VIDEO SCREEN MEMORY
C
             COMPILER(3) = 0F7FFH
             INTEGER*1 SCREEN(30720)
C
C DECLARE THE DISK BUFFER AND FILE CONTROL BLOCK
C
             COMPILER(3) = 7CH
             INTEGER*1 FCB(33)
C
             COMPILER(3) = 0FFH
             INTEGER*1 DISK@BUFFER(128)
C
C DECLARE THE GLOBAL VARIABLES
C
             COMPILER(3) = 5800H
C
             REAL*4 SAV(432),SAV@BUFFER(32),IN@PIXX,IN@PIXY
             REAL*4 X@ORIGIN,Y@ORIGIN
             INTEGER*1 TASK,NEW@MENU, FROZEN,FLIP@BYTE,SAVE@BYTE
             INTEGER*1 IFLAG,ORIGIN@FLAG
             INTEGER*2 CURSOR,ROW,COL,COLZ1,COLZ2,IROW,WRITE@FLAG,N
             INTEGER*2 HOR@RES,HOR@RES@HALF

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  NOTE THAT ROUTINES CURSOR@KEY AND CHAR@IN USE NON-CP/M
C  STANDARD I/O.  THEY DO NOT USE THE CP/M BIOS FOR KEY-
C  BOARD INPUT.  RATHER, THEY INTERFACE DIRECTLY WITH THE
C  HARDWARE.  THESE ROUTINES MAY HAVE TO BE ALTERED IF OTHER
C  HARDWARE IS USED.
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:         CURSOR@KEY
C
             INTEGER*1 FUNCTION CURSOR@KEY
             INTEGER*1 KEY

             CURSOR@KEY = -1

             KEY = INPUT (00H)
             IF ((KEY .AND. 40H) .EQ. 0) GOTO 10

                 KEY = INPUT(01H) .AND. 7FH
                 IF (KEY .EQ. ' ') CURSOR@KEY = 0
```

```
                IF (KEY .EQ. 'U') CURSOR@KEY = 1
                IF (KEY .EQ. 'D') CURSOR@KEY = 2
                IF (KEY .EQ. 'R') CURSOR@KEY = 3
                IF (KEY .EQ. 'L') CURSOR@KEY = 4

 10             RETURN
                END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
                   CHAR@IN
C
                INTEGER*1 FUNCTION CHAR@IN
                INTEGER*1 KEY

 10             KEY = INPUT(00H)
                IF ((KEY .AND. 40H) .EQ. 0) GO TO 10

                KEY = INPUT(01H)
                CHAR@IN = KEY .AND. 7FH

                RETURN
                END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:            LEFT@HALF
C
                INTEGER*1 FUNCTION LEFT@HALF(POINTER)
                INTEGER*2 POINTER

                LEFT@HALF = (SCREEN(POINTER) / 16) .AND. OFH

                RETURN
                END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:             RIGHT@HALF
C
                INTEGER*1 FUNCTION RIGHT@HALF(POINTER)
                INTEGER*2 POINTER

                RIGHT@HALF = SCREEN(POINTER) .AND. OFH

                RETURN
                END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:             DISPLAY
C
                SUBROUTINE DISPLAY

                IFLAG = 0
                N = 1
                DO 10 I = 1,432
                 SAV(I) = 0.0
```

```
 10          CONTINUE
             ORIGIN@FLAG = 0

             OUTPUT(OFFH) = OEOH
             FROZEN = 0

             RETURN
             END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:          FREEZE
C
             SUBROUTINE FREEZE

             OUTPUT(OFFH) = OAOH
             FROZEN = 1

             RETURN
             END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:          SWAP
C
             SUBROUTINE SWAP
             INTEGER*1 TEMP

             TEMP = SCREEN(CURSOR)
             SCREEN(CURSOR) = SAVE@BYTE
             SAVE@BYTE = TEMP

             RETURN
             END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:          FLIP
C
             SUBROUTINE FLIP
             INTEGER*1 TEMP

             TEMP = SCREEN(CURSOR)

             IF (TEMP .LT. OOH)  SCREEN(CURSOR) = OOH
             IF (TEMP .GE. OOH) SCREEN(CURSOR) = OFFH

             DO 10 I = 1,200
 10          CONTINUE

             RETURN
             END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:          MOVE@UP
C
             SUBROUTINE MOVE@UP
```

```
            IF (CURSOR .LE. HOR@RES@HALF) GO TO 10
                    CALL SWAP
                    CURSOR = CURSOR - HOR@RES@HALF
                    CALL SWAP

 10         RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:         MOVE@DOWN
C
            SUBROUTINE MOVE@DOWN
            INTEGER*2 DUMB
            IF (HOR@RES .EQ. 512) DUMB = 30464
            IF (HOR@RES .EQ. 256) DUMB = 30592
            IF (CURSOR .GE. DUMB ) GOTO 10
                    CALL SWAP
                    CURSOR = CURSOR + HOR@RES@HALF
                    CALL SWAP

 10         RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:         MOVE@RIGHT
C
            SUBROUTINE MOVE@RIGHT

            IF (CURSOR .GE. 30720) GOTO 10
                    CALL SWAP
                    CURSOR = CURSOR + 1
                    CALL SWAP

 10         RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:         MOVE@LEFT
C
            SUBROUTINE MOVE@LEFT

            IF (CURSOR .LE. 1) GOTO 10
                    CALL SWAP
                    CURSOR = CURSOR - 1
                    CALL SWAP

 10         RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C           THESE ARE THE DISK I/O PRIMITIVES
C           ALL DISK I/O IS THRU THE CP/M BDOS
C
```

```
C INPUT PARAMETERS ARE:
C                       OPERATION TYPE IN REG C
C                       FCB ADDRESS IN DE
C
C RETURNED STATUS IS IN ACC
C
C IN GENERAL ALL INLINE CODE IS OF FORM:
C                       MVI     C,CODE
C                       LXI     D,05CH      -     ADDRESS OF FCB
C                       CALL    0005H       -     ADDRESS OF BDOS
C                                                 ENTRY
C                       STA     STATUS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC C
C                   OPEN@FILE
C
            SUBROUTINE OPEN@FILE(STATUS)
            INTEGER*1 STATUS
            INLINE /0EH,15,11H,5CH,00H,0CDH,05H,00H/
            INLINE /32H,ADDRESS(STATUS)/
            RETURN
            END
C                   CLOSE@FILE
C
            SUBROUTINE CLOSE@FILE(STATUS)
            INTEGER*1 STATUS
            INLINE /0EH,16,11H,5CH,00H,0CDH,05H,00H/
            INLINE /32H,ADDRESS(STATUS)/
            RETURN
            END
C                   DELETE@FILE
C
            SUBROUTINE DELETE@FILE
            INLINE /0EH,19,11H,5CH,00H,0CDH,05H,00H/
            RETURN
            END
C                   READ@FILE
C
            SUBROUTINE READ@FILE(STATUS)
            INTEGER*1 STATUS
            INLINE /0EH,20,11H,5CH,00H,0CDH,05H,00H/
            INLINE /32H,ADDRESS(STATUS)/
            RETURN
            END
C                   WRITE@FILE
C
            SUBROUTINE WRITE@FILE(STATUS)
            INTEGER*1 STATUS
            INLINE /0EH,21,11H,5CH,00H,0CDH,05H,00H/
            INLINE /32H,ADDRESS(STATUS)/
            RETURN
            END
C                   MAKE@FILE
C
```

```
            SUBROUTINE MAKE@FILE(STATUS)
            INTEGER*1 STATUS
            INLINE /0EH,22,11H,5CH,00H,0CDH,05H,00H/
            INLINE /32H,ADDRESS(STATUS)/
            RETURN
            END
C                   SET@DMA
C
            SUBROUTINE SET@DMA(BUF)
            INLINE /0EH,26,11H,ADDRESS(BUF)/
            INLINE /0CDH,05H,00H/
            RETURN
            END
C                   SET@DISK@BUFFER@DMA
C
            SUBROUTINE SET@DISKBUF
            INLINE /0EH,26,11H,80H,00H/
            INLINE /0CDH,05H,00H/
            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                   GET@FILE@NAME
C
C THIS ROUTINE GETS THE FILE NAME OF THE DISK FILE
C TO BE USED TO SAVE OR READ A SCREEN OR DATA FILE.
C INPUT FILE NAME AND FILE TYPE .PIC OR .DATA , AND
C SELECTS DRIVE B.
C
            SUBROUTINE GET@FILE@NAME
            INTEGER*1 FILE@NAME(8)
C
C TURN OFF DMA
C
                    OUTPUT(0FFH)=00H
C
C INITIALIZE THE FCB TO 0'S
C
                    DD  1  I=1,33
                              FCB(I)=0
 1                  CONTINUE
C
C GET USERS FILE NAME
C
                    WRITE(1,2)
 2                  FORMAT(1X,/////////////////////////,
     +   1X,'ENTER FILE NAME;', ////////////////////)
                    READ(40H)STRING(FILE@NAME,8)
C
C PUT FILE NAME IN FCB
C
                    DO 3 I=1,8
                           FCB(I+1)=FILE@NAME(I)
 3                  CONTINUE C
```

```
C SET FILETYPE TO .PIC OR .DAT
C
            IF (WRITE@FLAG .EQ. 6) GOTO 4
                        FCB(10)='P'
                        FCB(11)='1'
                        FCB(12)='C'
                        GOTO 5

 4                      FCB(10)='D'
                        FCB(11)='A'
                        FCB(12)='T'
C
C SELECT DRIVE B
C
 5          FCB(1) = 2

            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                   SAVE
C
C THIS ROUTINE WRITES THE CONTENTS OF THE SCREEN OR DATA
C ARRAY TO A DISK FILE SPECIFIED BY THE USER.
C
            SUBROUTINE SAVE
            INTEGER*2 START, INDEX
            INTEGER*1 STATUS
C
C TURN OFF DMA
                    OUTPUT(OFFH)=00H
C
                    CALL GET@FILE@NAME
                    CALL DELETE@FILE
                    CALL MAKE@FILE(STATUS)
                            IF (STATUS.EQ.OFFH) GO TO 30
                    CALL OPEN@FILE(STATUS)
                            IF (STATUS.EQ.OFFH) GO TO 30
                    IF (WRITE@FLAG .EQ. 6) GOTO 25
 8                  DO 20 I=1,240
                            START=128*(I-1)
                            DO 10 J=1,128
                                    INDEX=START+J
                                    DISK@BUFFER(J)=SCREEN(INDEX)
 10                         CONTINUE
                            CALL WRITE@FILE(STATUS)
                                    IF (STATUS.NE.00H) GO TO 30
 20                 CONTINUE
                    GOTO 28
 25                 CALL SET@DMA( SAV@BUFFER(1))
                     DO 35 I = 1,72
                      IND = (I-1)* 6
                         DO 40 J = 1, 6
                             INDEX = IND + J
```

```
                              SAV@BUFFER(J) = SAV(INDEX)
 40                         CONTINUE
                             CALL WRITE@FILE(STATUS)
                             IF (STATUS .NE. 00H)GOTO 30
 35                    CONTINUE
                     CALL SET@DISKBUF
 28                  CALL CLOSE@FILE(STATUS)
                     GOTO 60
 30                  WRITE(1,50)
 50                  FORMAT(//////////////////////////,
     +                    'ERROR STATUS',//////////////////////)
C
C  TURN ON DMA
C
 60                  OUTPUT(0FFH)=0A0H
                     NEW@MENU = 1

                RETURN
                END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                    GET
C
C THIS ROUTINE READS A FILE FROM THE DISK TO THE
C SCREEN OR SAVE ARRAY DEPENDING ON THE VALUE OF
C WRITE@FLAG.
C
                SUBROUTINE GET
                INTEGER*2 START,INDEX
                INTEGER*1 STATUS
C
C TURN OFF DMA
C
                OUTPUT(0FFH) = 00H
                IF (WRITE@FLAG .NE. 6) IFLAG = 0

                 N = 1
                DO 5 I = 1,432
                 SAV(I) = 0.0
 5              CONTINUE
                ORIGIN@FLAG = 0

                        CALL GET@FILE@NAME
                        CALL OPEN@FILE(STATUS)
                                IF (STATUS.EQ.0FFH) GO TO 30
                        IF (WRITE@FLAG .EQ. 6) GOTO 25
 8                      DO 20 I=1,240
                                CALL READ@FILE(STATUS)
                                        IF (STATUS.NE.00H) TO TO 30
                                START=120*(I-1)
                                DO 10 J=1,128
                                        INDEX=START+J
                                SCREEN(INDEX)=DISK@BUFFER(J)
10                              CONTINUE
```

```
 20                  CONTINUE
                     GOTO 28
 25                   CALL SET@DMA( SAV@BUFFER(1))
                      DO 35 I = 1,72
                       CALL READ@FILE(STATUS)
                         IF (STATUS .NE.00H)GOTO 30
                         IND = (I-1) * 6
                           DO 40 J = 1, 6
                             INDEX = IND + J
                                SAV(INDEX) = SAV@BUFFER(J)
 40                        CONTINUE
 35                    CONTINUE
                      CALL SET@DISKBUF
 28                   CALL CLOSE@FILE(STATUS)
                       GOTO 60
 30                  WRITE(1,50)
 50                  FORMAT(////////////////////////////////,
     +                  'ERROR STATUS',////////////////////////)
C
C  TURN ON DMA
C
 60                  OUTPUT(0FFH)-0A0H
                     NEW@MENU = 1

               RETURN
               END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              INITIALIZE@PROGRAM
C
               SUBROUTINE INITIALIZE@PROGRAM

               FROZEN = 0
               NEW@MENU = 0
               HOR@RES = 0
               HOR@RES@HALF = 0
               FLIP@BYTE = 0FFH
               SAVE@BYTE = 0FFH
               WRITE@FLAG = 0
               ORIGIN@FLAG = 0
               CURSOR = 2580
               IFLAG = 0
               N = 1

               DO 10 I = 1,432
               SAV(I) = 0.0
 10            CONTINUE

               RETURN
               END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              MAIN@MENU
C
```

```
            SUBROUTINE MAIN@MENU

            OUTPUT(OFFH) = 00H

            WRITE (1,10)
10          FORMAT (////////////////////////,
           +    1X,'SELECT FUNCTION:',/,
           +    6X,'(D)ISPLAY PICTURE',/,
           +    6X,'(F)REEZE PICTURE',/,
           +    6X,'(S)AVE PICTURE ON DISKETTE',/,
           +    6X,'(G)ET PICTURE FROM DISKETTE',/,
           +    6X,'(C)URSOR CONTROL',/,
           +    6X,'(P)RINT GREY CODES AND COORDINATES',/,
           +    6X,'(R)EFERENCE PICTURE DIMENSIONS',/,
           +    6X,'(O)RIGIN RESET OPTION',/
           +    6X,'S(A)VE DATA IN ARRAY',/,
           +    6X,'S(T)ORE DATA ARRAY ON DISK',/,
           +    6X,'G(E)T DATA ARRAY FROM DISK',/,
           +    6X,'(W)RITE SAV( ) ARRAY ON PRINTER',/,
           +    6X,'(H)EAT@TRANSFER DATA REDUCTION ',/,
           +    6X,'E(X)IT TO OPERATING SYSTEM',    ///////)

            NEW@MENU = 0

            CALL FREEZE

            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:           CURSOR@MENU
C
            SUBROUTINE CURSOR@MENU

            OUTPUT(OFFH) = 00H
            WRITE(1,1)
1           FORMAT(////////////////////////)

            IF (WRITE@FLAG .NE. 1) GOTO 2
               WRITE(1,15)
2           IF (WRITE@FLAG .NE. 2) GOTO 3
               WRITE(1,20)
3           IF (WRITE@FLAG .NE. 3) GOTO 4
               WRITE(1,25)
4           IF (WRITE@FLAG .NE. 4) GOTO 5
               WRITE(1,30)
5           IF (WRITE@FLAG .NE. 5) GOTO 6
               WRITE(1,35)
6           WRITE(1,10)
10        FORMAT (
         +    1X,'CURSOR CONTROL KEYS:',/,
         +    6X,'U)P',/,
         +    6X,'D)OWN',/,
         +    6X,'R)IGHT',/,
```

```
     +    6X,'L)EFT',//,
     +    6X,'(SPACE) TO EXIT',/////////////)
 15       FORMAT (1X,'MOVE CURSOR TO LEFT ENDPOINT OF HORIZONTAL',/,
     +         1X,'REFERENCE AND SPACE ',//)
 20       FORMAT (1X,'MOVE CURSOR TO RIGHT ENDPOINT OF HORIZONTAL',/,
     +         1X,'REFERENCE AND SPACE ',//)
 25       FORMAT (1X,'MOVE CURSOR TO TOP OF VERTICAL REFERENCE',/,
     +         1X,'AND SPACE ',//)
 30       FORMAT (1X,'MOVE CURSOR TO BOTTOM OF VERTICAL REFERENCE',/,
     +         1X,'AND SPACE ',//)
 35       FORMAT (1X,'MOVE THE CURSOR TO THE DESIRED AXIS ORIGIN',/,
     +         1X,'AND SPACE ',//)

          WRITE@FLAG = 0
          CALL FREEZE
          NEW@MENU = 1

          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:          CURSOR@CONTROL
C
          SUBROUTINE CURSOR@CONTROL
          INTEGER*1 KEY

          CALL CURSOR@MENU

 10       KEY = CURSOR@KEY

          IF (KEY .EQ. 0) GOTO 30

                  IF (KEY .LT. 0) GOTO 20

                       IF (KEY .EQ. 1) CALL MOVE@UP
                       IF (KEY .EQ. 2) CALL MOVE@DOWN
                       IF (KEY .EQ. 3) CALL MOVE@RIGHT
                       IF (KEY .EQ. 4) CALL MOVE@LEFT

 20                CALL FLIP
                   GOTO 10

 30       NEW@MENU = 1

          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:             GET@DATA
C
          SUBROUTINE GET@DATA

          WRITE@FLAG = 6
          CALL GET
```

```
            WRITE@FLAG = 0
            NEW@MENU = 1

            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:             ROWCOL
C
            SUBROUTINE ROWCOL(Z1,Z2)

            INTEGER*2 DUM
            I=CURSOR
            X=I
            IF (HOR@RES .EQ. 512)DUM = 121
            IF (HOR@RES .EQ. 256)DUM = 241
                Z1=2.*(X-1.)+1.
                Z2=Z1+1.
                IROW=(I-1)/HOR@RES@HALF)+1
                ROW=DUM-IROW
                COL=I-(IROW
                COLZ1=2*(COL-1)+1
                COLZ2=COLZ1+1

            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:         COORDINATES
C
            SUBROUTINE COORDINATES(X1,X2,X3,Y)

            RCOL1 = COLZ1
            RCOL2 = COLZ2
            RROW = ROW

            X1 = IN@PIXX*(RCOL1-1.0) - X@ORIGIN
            X2 = IN@PIXX*(RCOL1) - X@ORIGIN
            X3 = IN@PIXX*(RCOL2) - X@ORIGIN
            Y  = IN@PIXY*(RROW-.5)  - Y@ORIGIN

            RETURN
            END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:             CHOICE
C
            SUBROUTINE CHOICE(X1,X2,X3,Y1,X,Y)

            WRITE(1,10)
 10         FORMAT(////////////////////////,
     +          1X,'DO YOU WANT REFERENCE AT X = ?',/,
     +          5X,'X1.....TYPE "1"',/,
     +          5X,'X2.....TYPE "2"',/,
```

```
     +         5X,'X3.....TYPE "3"',/,
     +         ////////////////)

               X = 0.0
 20            TASK = CHAR@IN
                  IF (TASK .EQ. '1') GOTO 22
                  IF (TASK .EQ. '2') GOTO 24
                  IF (TASK .EQ. '3') GOTO 26
                  GOTO 20
 22               X = X1
                  GOTO 30
 24               X = X2
                  GOTO 30
 26               X = X3
 30            Y = Y1
               RETURN
               END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:                SET@ORIGIN
C
               SUBROUTINE SET@ORIGIN

               OUTPUT(OFFH) = OOH
               INTEGER*2 DUMMY
               WRITE@FLAG = 5
               ORIGIN@FLAG = ORIGIN@FLAG + 1
               IF (IFLAG .NE. 1) GOTO 10
               SAV(432) = ORIGIN@FLAG

               CALL CURSOR@CONTROL
                  X@ORIGIN = 0.0
                  Y@ORIGIN = 0.0
               CALL ROWCOL(Z1,Z2)
               CALL COORDINATES(X1,X2,X3,Y)

               CALL CHOICE(X1,X2,X3,Y,X@ORIGIN,Y@ORIGIN)
                IF (ORIGIN@FLAG .NE. 1) GOTO 5
                  SAV(431) = X@ORIGIN
                  SAV(430) = Y@ORIGIN
                  DUMMY = 429
               GOTO 30
 5           IF (SAV(DUMMY-1) .NE. 0.0) GOTO 40
             SAV(DUMMY) = X@ORIGIN - SAV(431)
             SAV(DUMMY-1) = Y@ORIGIN - SAV(430)
             DUMMY = DUMMY - 2
             GOTO 30

 10          WRITE(1,20)
             CALL FREEZE
             NEW@MENU = 0
             RETURN
 40          WRITE(1,50)
 50          FORMAT(//////,1X,'SAV ARRAY IS FULL',///////////////)
```

```
          CALL FREEZE
          NEW@MENU = 0
          RETURN
 20       FORMAT(/////////////////////,1X,'REFDIMENSION FIRST',
     +          //////////////////)
 30       CALL FREEZE
          NEW@MENU = 1
          CALL SWAP
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              REFDIMENSION
C
          SUBROUTINE REFDIMENSION

          OUTPUT(OFFH)=00H

          IFLAG=1
          WRITE(1,10)
          READ(40H)HOR,VER

 10       FORMAT(////////,
     +         1X,'YOU MUST NOW DEFINE A REFERENCE DIMENSION',/,
     +         1X,'BY SETTING THE CURSOR AT A POINT AND',/,
     +         1X,'TRAVERSING HORIZONTALLY OR VERTICALLY',/,
     +         1X,'A KNOWN DISTANCE ON THE IMAGE,',///,
     +         1X,'NOW TYPE IN YOUR REFERENCE DIMENSIONS',/,
     +         1X,'(REAL FORMAT ; SEPARATE WITH COMMA )',//////
     +         /////////)

          WRITE@FLAG = 1
          CALL CURSOR@CONTROL
               CALL ROWCOL(Z1,Z2)
                    KEEP@COLUMN=COLZ1
          WRITE@FLAG = 2
          CALL CURSOR@CONTROL
               CALL ROWCOL(Z1,Z2)
                    HORIZONTALPIX=COLZ1-KEEP@COLUMN
          WRITE@FLAG = 3
          CALL CURSOR@CONTROL
               CALL ROWCOL(Z1,Z2)
                    KEEP@ROW = ROW
          WRITE@FLAG = 4
          CALL CURSOR @ CONTROL
               CALL ROWCOL(Z1,Z2)
                    VERTICALPIX=KEEP@ROW - ROW
          IN@PIXX = HOR/HORIZONTALPIX
          IN@PIXY = VER/VERTICALPIX

          CALL SET@ORIGIN

 70       RETURN
          END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              PRINT
C
           SUBROUTINE PRINT

           OUTPUT(OFFH) = OOH
           CALL SWAP

           CALL ROWCOL(Z1,Z2)
           IF(IFLAG.NE.1)GOTO 10
           CALL COORDINATES(X1,X2,X3,Y)
           GOTO 13
 10         X1=0.0
            X2=0.0
            X3=0.0
            Y =0.0
 13        WRITE(1,15)
           WRITE(1,20)Z1,ROW,COLZ1,LEFT@HALF(CURSOR),X1,Y,ROW,X2 ,Y,
     +     Z2,ROW,COLZ2,RIGHT@HALF(CURSOR),X3,Y

 15        FORMAT(//////////////,
     +          4X,'PIXEL',2X,'ROW',2X,'COL',2X,'GRAY',6X,'X', 9X,'Y',/)
 20        FORMAT(   1X,F8.0,2X,I3,2X,I3,3X,I3,2X,F8.3, 2X,F8.3,/,
     +       11X,I3,13X,F8.3, 2X,F8.3,/,
     +       1X,F8.0,2X,I3,2X,I3,3X,I3,2X,F8.3, 2X,F8.3,//,
     +       1X ,'SPACE TO CONTINUE',//////////)

           CALL SWAP
           CALL FREEZE
 40        TASK = CHAR@IN
             IF (TASK .EQ. ' ')GOTO 50
           GOTO 40
 50        NEW@MENU = 1
           RETURN
           END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              SAVE@DATA
C
           SUBROUTINE SAVE@DATA

           OUTPUT(OFFH) = OOH
           CALL SWAP
           IF (IFLAG .NE. 1) GOTO 10

           CALL ROWCOL(Z1,Z2)
           CALL COORDINATES(X1,X2,X3,Y)
           CALL CHOICE(X1,X2,X3,Y,X,Y)
           IF (SAV(N+1) .NE. 0.0)GOTO 65

           SAV(N) = X
           SAV(N+1) = Y
           N1 = N+1
```

```
          WRITE(1,5)N,SAV(N),N1,SAV(N1)
          IF(WRITE@FLAG .NE.8)N = N+2
          GOTO 30
5         FORMAT
     +      1X,'SAV(',I3,') = ',2X,F12.3,2X,'SAV(',I3,') = ',2X,F12.3,
     +      //,1X,'SPACE TO CONTINUE',
     +      ////////////)
10        WRITE(1,15)
          GOTO 30
65        WRITE(1,70)
70          FORMAT(//////,1X,'SAV ARRAY IS FULL',//////////)
30        CALL SWAP
          CALL FREEZE
50        TASK = CHAR@IN
            IF (TASK .EQ. ' ')GOTO 60
          GOTO 50
15        FORMAT(////////////////////////,
     +      1X,'YOU MUST REFDIMENSION FIRST',//////////)

60        NEW@MENU = 1

          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              STORE@DATA
C
          SUBROUTINE STORE@DATA

5         WRITE(1,50)
10        TASK = CHAR@IN
            IF (TASK .EQ. 'Y') GOTO 20
            IF (TASK .EQ. 'N') GOTO 30
          GOTO 10
20        SAV(N) = 999.
          N = N + 2
          WRITE@FLAG = 8
          CALL CURSOR@CONTROL
            X@ORIGIN = SAV(431)
            Y@ORIGIN = SAV(430)

          CALL SAVE@DATA

          GOTO 5
30        WRITE@FLAG = 6
          CALL SAVE
          WRITE@FLAG = 0
50        FORMAT(//,1X,'ARE THERE ANY MISCELLANEOUS DATA POINTS',/,
     +      1X,'THAT YOU WANT TO SAVE?',//,
     +      1X,'(ANY POINTS SAVED NOW WILL BE WITH RESPECT TO',/,
     +      1X,'THE ORIGINAL ORIGIN!',/,
     +      //,1X,'Y = YES; N = NO',/////////)
          RETURN
          END
```

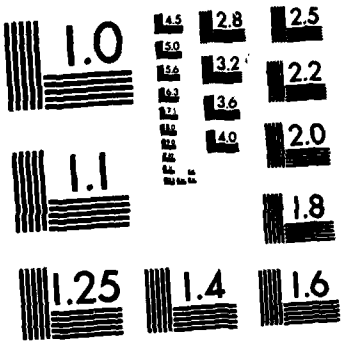
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              PRINT@SAVE
C
          SUBROUTINE PRINT@SAV

          OUTPUT(OFFH) = OOH
          CALL SWAP
          WRITE(2,20)
          DO 10 I = 1,432,4
          WRITE(2,25)I,I+1,I+2,I+3,SAV(I),SAV(I+1),SAV(I+2),SAV(I+3)
 10       CONTINUE
 20       FORMAT(6X,'CONTENTS OF SAV ARRAY',//
     +           6X,'ELEMENTS :',15X,'DATA:',//)
 25       FORMAT(6X,I3,',',I3,',',I3,',',I3,5X,F12.4,1X,
     +           F12,4,1X,F12.4,1X,F12.4,/)
          CALL SWAP
          CALL FREEZE
          NEW@MENU = 1
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              HEAT@TRANSFER
C
          SUBROUTINE HEAT@TRANSFER

          REAL*4 KCONST,DELX(50),QLOC(50),QLOCDELX(50),KEEP431
          INTEGER*2 DUMMY
          M = 9
C
C     DUMMY IS THE SAV ARRAY LOCATION OF THE Y DISPLACEMENT
C     FROM THE FIRST ORIGIN ESTABLISHED IN THE DATA SET.
C
          DUMMY = 429
          QTOT = 0.0
          Y1TOTAL = 0.0
           DO 5 I = 1,50
             DELX(I) = 0.0
             QLOC(I) = 0.0
             QLOCDELX(I) = 0.0
 5          CONTINUE
          WRITE(1,70)
          READ(40H)D1,TEMPERATURE,P,KCONST
          NPTS = SAV(432)
          REALPOINTS = NPTS - 1

          DO 6 I = 2,NPTS
           Y1TOTAL = Y1TOTAL + SAV(M)
           M = M + 8
 6        CONTINUE
          M = 9
          D2 = Y1TOTAL/REALPOINTS
```

```
          C1 = .026195/P
          KEEP431 = SAV(431)
          SAV(431) = 0.0

          DO 31 J = 2,NPTS
             IF (J .NE. 2)GOTO 10
              DELX(2) = (SAV(DUMMY-2)-SAV(DUMMY))/2.+(D1-(D1-SAV(DUMMY)))
              GOTO 30
10            IF (J .EQ. NPTS)GOTO 20

              DELX(J) = (SAV(DUMMY-2)-SAV(DUMMY+2)) /2.0
              DUMMY = DUMMY - 2
              GOTO 30
20        DELX(NPTS) = (SAV(DUMMY)-SAV(DUMMY+2))/2.0
          DELX(NPTS) = DELX(NPTS) + (D1 - SAV(DUMMY))
30       WRITE(2,200)DUMMY,DELX(J)
31        CONTINUE

200       FORMAT(/,1X,I5,F12.4,/)

          DO 60 I = 2,NPTS
             Y1 = SAV(M)
             Y2 = SAV(M+2)
             X1 = -SAV(M+5)
             X2 = -SAV(M+7)
             EPS1 = Y1/D2
             EPS2 = Y2/D2
             IF (TEMPERATURE .GT.535.0) GOTO 40
               T2 = TEMPERATURE /(1.0 - EPS1*C1*TEMPERATURE)
               T3 = TEMPERATURE /(1.0 - EPS2*C1*TEMPERATURE)
               GOTO 50
40             T2 = TEMPERATURE/(1.0 + EPS1*C1*TEMPERATURE)
               T3 = TEMPERATURE/(1.0 + EPS2*C1*TEMPERATURE)
50           C = (X2*(T2-TEMPERATURE)-X1*(T3-TEMPERATURE))/((X1-X2)*X1*X2)
             B = ((T2-TEMPERATURE)/X1) - (C*X1)

               QLOC(I) = (-KCONST)*B
               QLOCDELX(I) = QLOC(I)* DELX(I)
               QTOT = QTOT + QLOCDELX(I)
             WRITE(2,400)T2,T3,C,B
400          FORMAT(   1X,F12.4,F12.4,F12.4,F12.4,/)
                  M = M+8
60             CONTINUE

             WRITE(2,75)D1,D2,TEMPERATURE,P,KCONST
             DO 65 I = 2,NPTS
               WRITE(2, 80)I,QLOC(I)
               WRITE(2, 81)QLOCDELX(I)
65           CONTINUE
               WRITE(2, 85)QTOT

               SAV(431) = KEEP431

70             FORMAT(/,1X,'NOW INPUT:',/,
```

```
     +              3X,'D1 : HEIGHT OF PLATE IN INCHES',/,
     +              3X,'TEMP : TEMPERATURE OF PLATE     (DEG R)',/,
     +              3X,'P  : ATMOSPHERIC PRESSURE (LB/FT**2)',/,
     +              3X,'K  : (KCONST) THERMAL CONDUCTIVITY OF AIR',/,
     +              3X,'                (BTU/HR-FT-DEG R)  ',/,
     +              ///,2X,'ENTER IN REAL FORMAT, SEQUENCED IN ORDER',/,
     +                 2X,'SEPARATED BY COMMAS',//////////)
 75            FORMAT(/,4X,'D1       =',F12.4,/
     +                 4X,'D2       =',F12.4,/,
     +                 4X,'TEMP     =',F12.4,/,
     +                 4X,'PRESSURE =',F12.4,/,
     +                 4X,'K        =',F12.4,///)
 80            FORMAT(/,1X,'QLOC(',I2,') =',F12.4,/)
 81            FORMAT(  12X,F12.4,/)
 85            FORMAT(/,1X,'QTOT      =',F12.4,//)

 100           NEW@MENU = 1
               RETURN
               END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              RESOLUTION
C
          SUBROUTINE RESOLUTION

          WRITE(1,10)
          READ(40H)HOR@RES
          HOR@RES@HALF = HOR@RES/2
 10       FORMAT(//////////,3X,'INPUT HORIZONTAL ',
     +      'RESOLUTION (PIXELS/LINE)',/,
     +      3X,'512 OR 256',//////////////)
          RETURN
          END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C NAME:              DIGITIZE@MAIN
C
          CALL INITIALIZE@PROGRAM
          CALL RESOLUTION
 10       CALL MAIN@MENU

 20       TASK = CHAR@IN

                   IF (TASK .EQ. 'D') CALL DISPLAY
                   IF (TASK .EQ. 'F') CALL FREEZE
                   IF (TASK .EQ. 'S') CALL SAVE
                   IF (TASK .EQ. 'G') CALL GET
                   IF (TASK .EQ. 'C') CALL CURSOR@CONTROL
                   IF (TASK .EQ. 'P') CALL PRINT
                   IF (TASK .EQ. 'R') CALL REFDIMENSION
                   IF (TASK .EQ. 'O') CALL SET@ORIGIN
                   IF (TASK .EQ. 'A') CALL SAVE@DATA
                   IF (TASK .EQ. 'T') CALL STORE@DATA
                   IF (TASK .EQ. 'E') CALL GET@DATA
```

AD-A124 751 VIDEO DIGITIZATION OF DISPLACEMENT/GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI... 2/2

UNCLASSIFIED D R CHAFFEE DEC 82 AFIT/GAE/AA/82D-5 F/G 9/2 NL



```
                    IF (TASK .EQ. 'W') CALL PRINT@SAY
                    IF (TASK .EQ. 'H') CALL HEAT@TRANSFER
                    IF (TASK .EQ. 'X') GOTO 30

                    IF (NEW@MENU .NE.0) GOTO 10
                          GOTO 20

C
C TURN OFF DMA AND EXIT TO CP/M
C
 30         OUTPUT(OFFH) = 00H

            STOP
            END
```

APPENDIX B

SAMPLE DATA OUTPUT

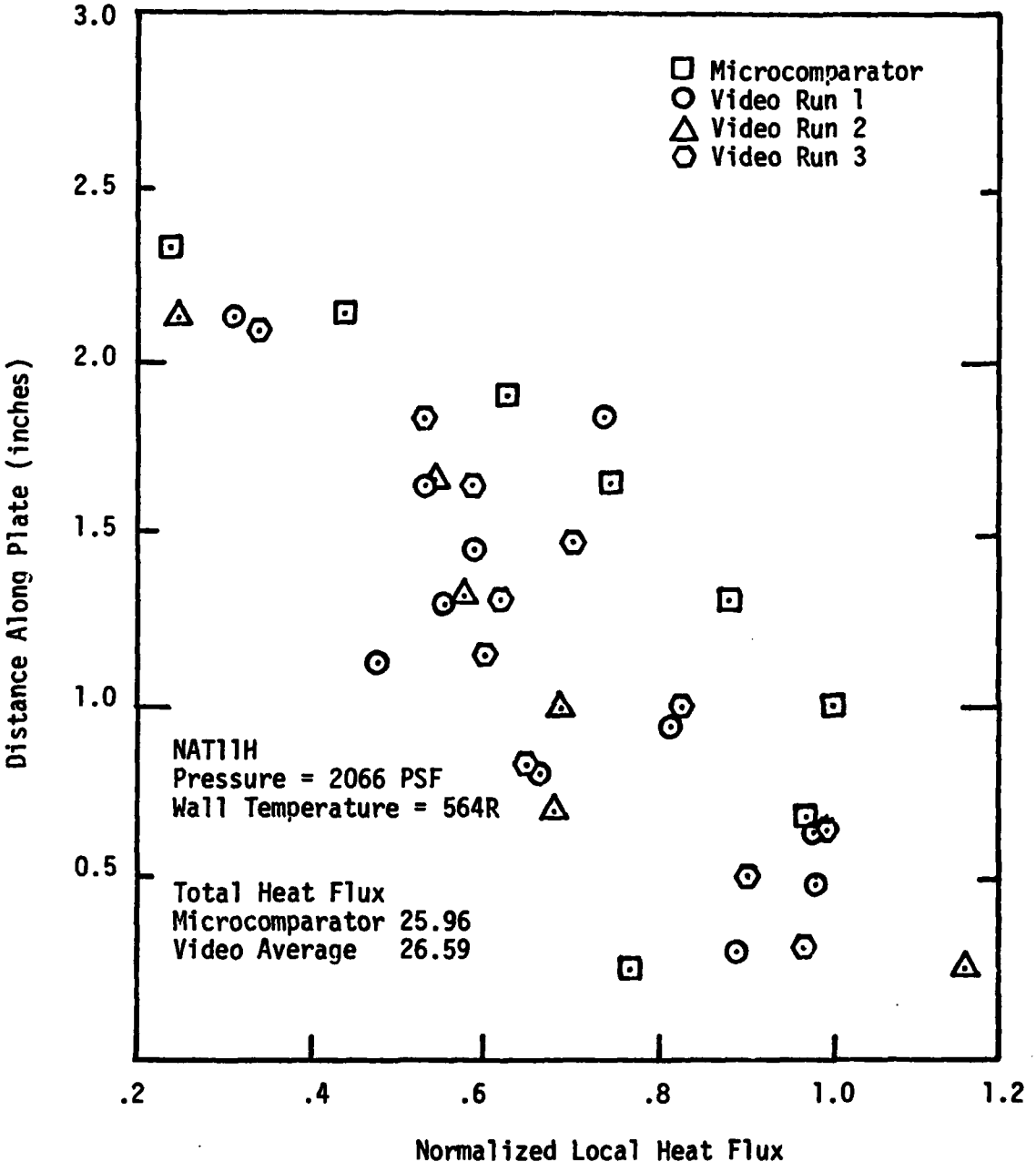


Figure B-1. Local Heat Flux of a Hot Plate with Vertical Divider in the Test Section (Normalized to Maximum Microcomparator Heat Flux)

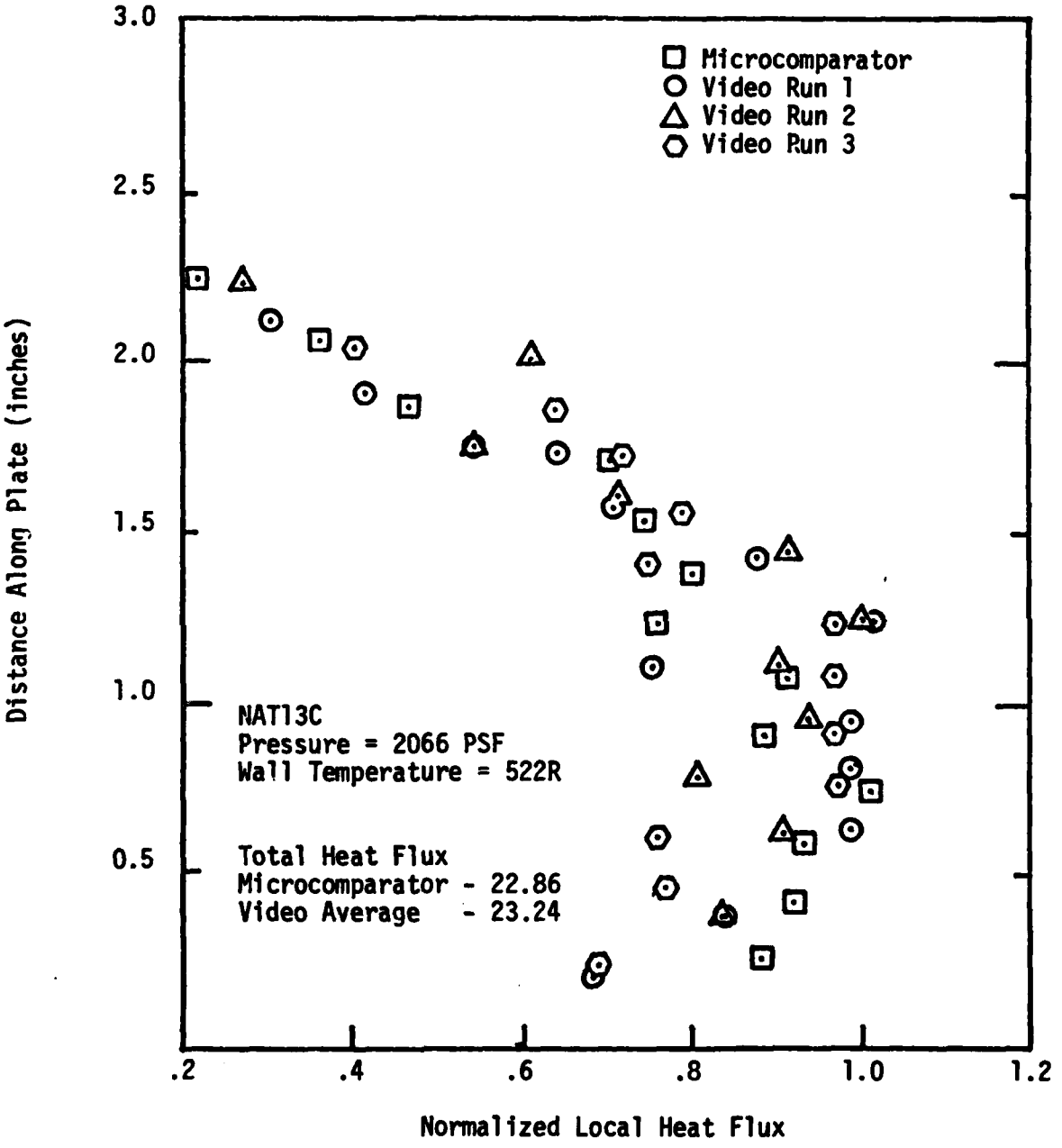


Figure B-2. Local Heat Flux of a Cold Plate with Right Running Diagonal Test Cell Divider (Normalized to maximum Microcomparator Heat Flux)

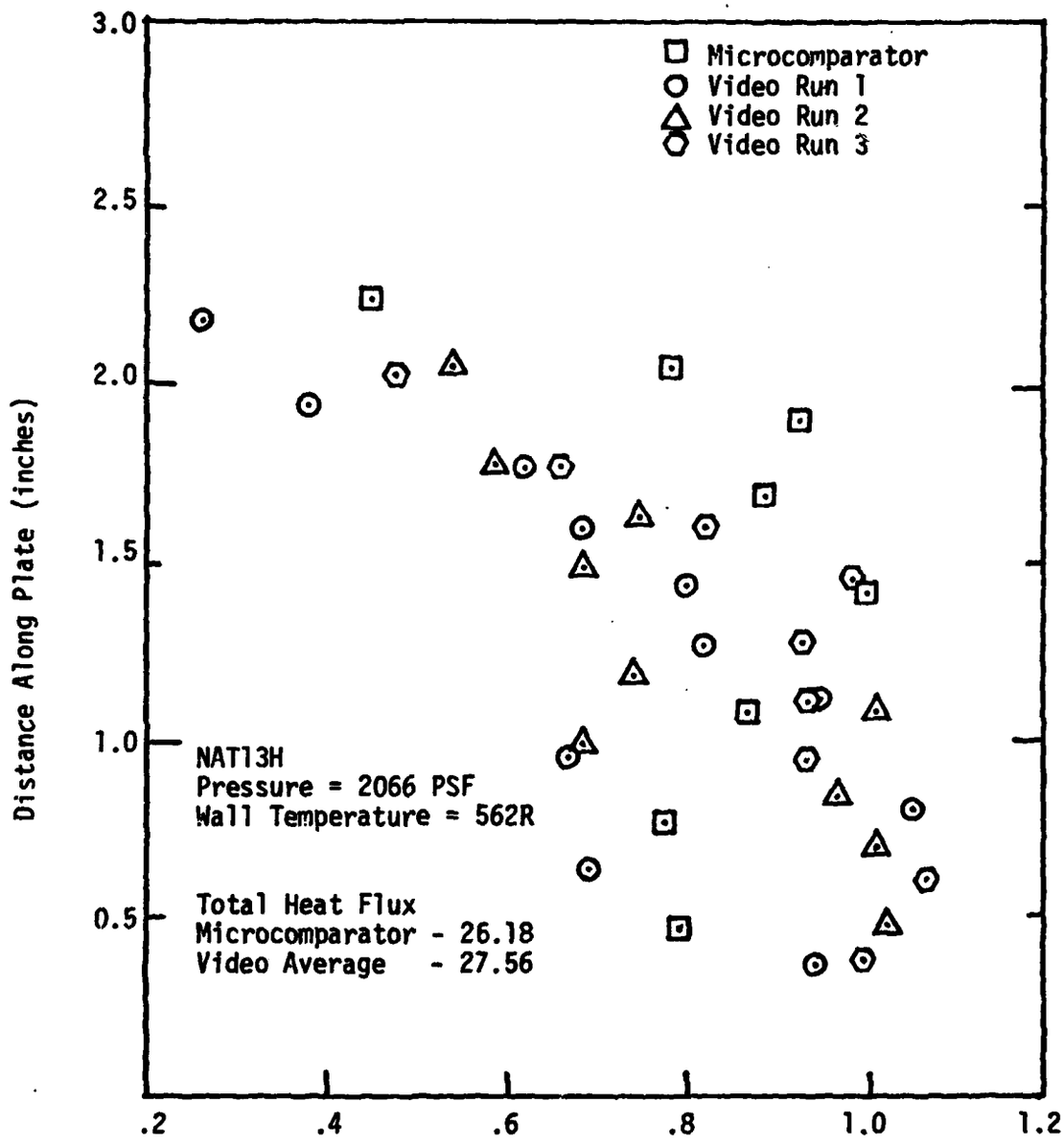


Figure B-3. Local Heat Flux of a Hot Plate with Right Running Diagonal Test Cell Divider (Normalized to Maximum Microcomparator Heat Flux)

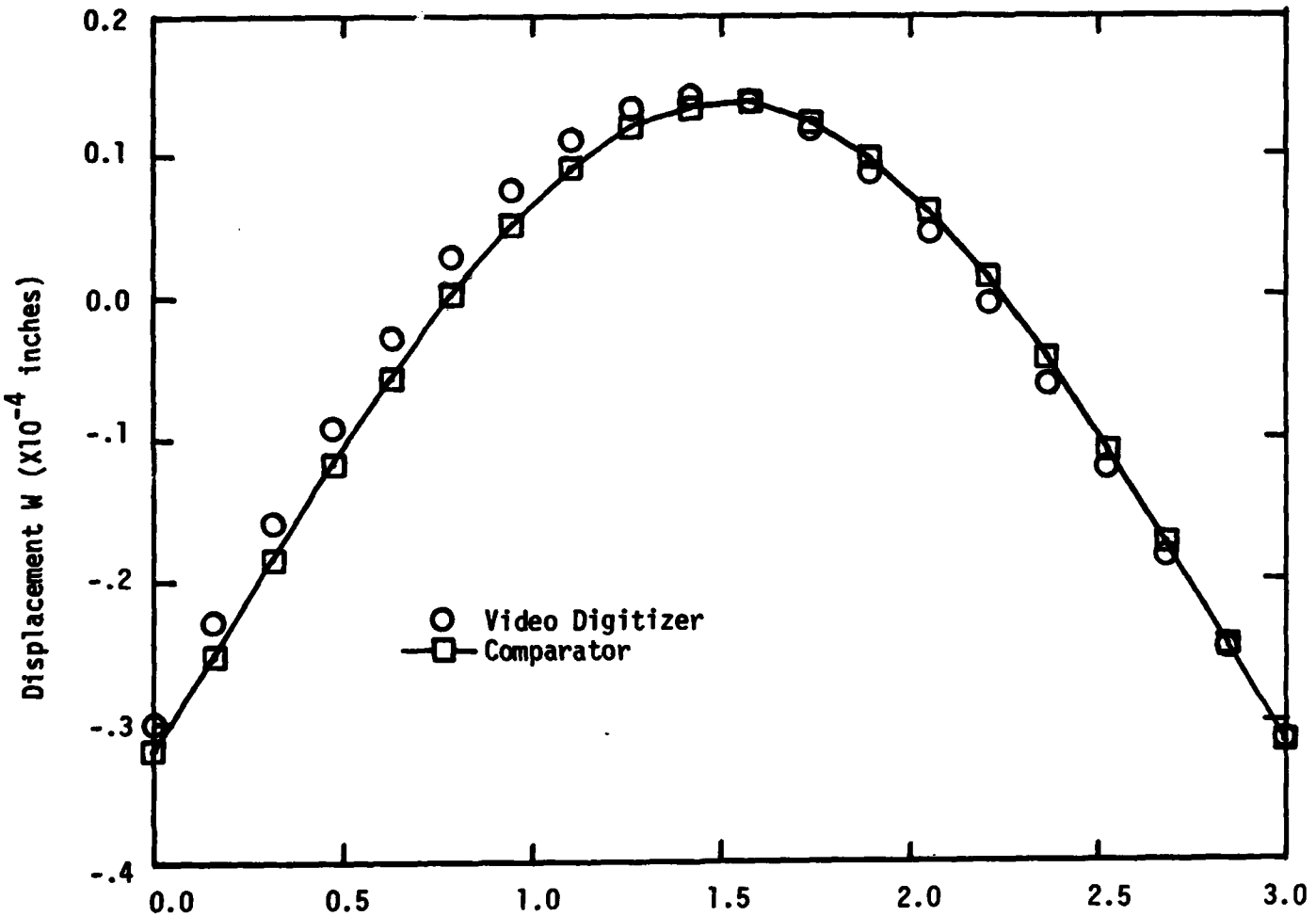


Figure B-4. Lyre Mode Out of Plane Displacement W at Constant Span Location, X=2.5 inches

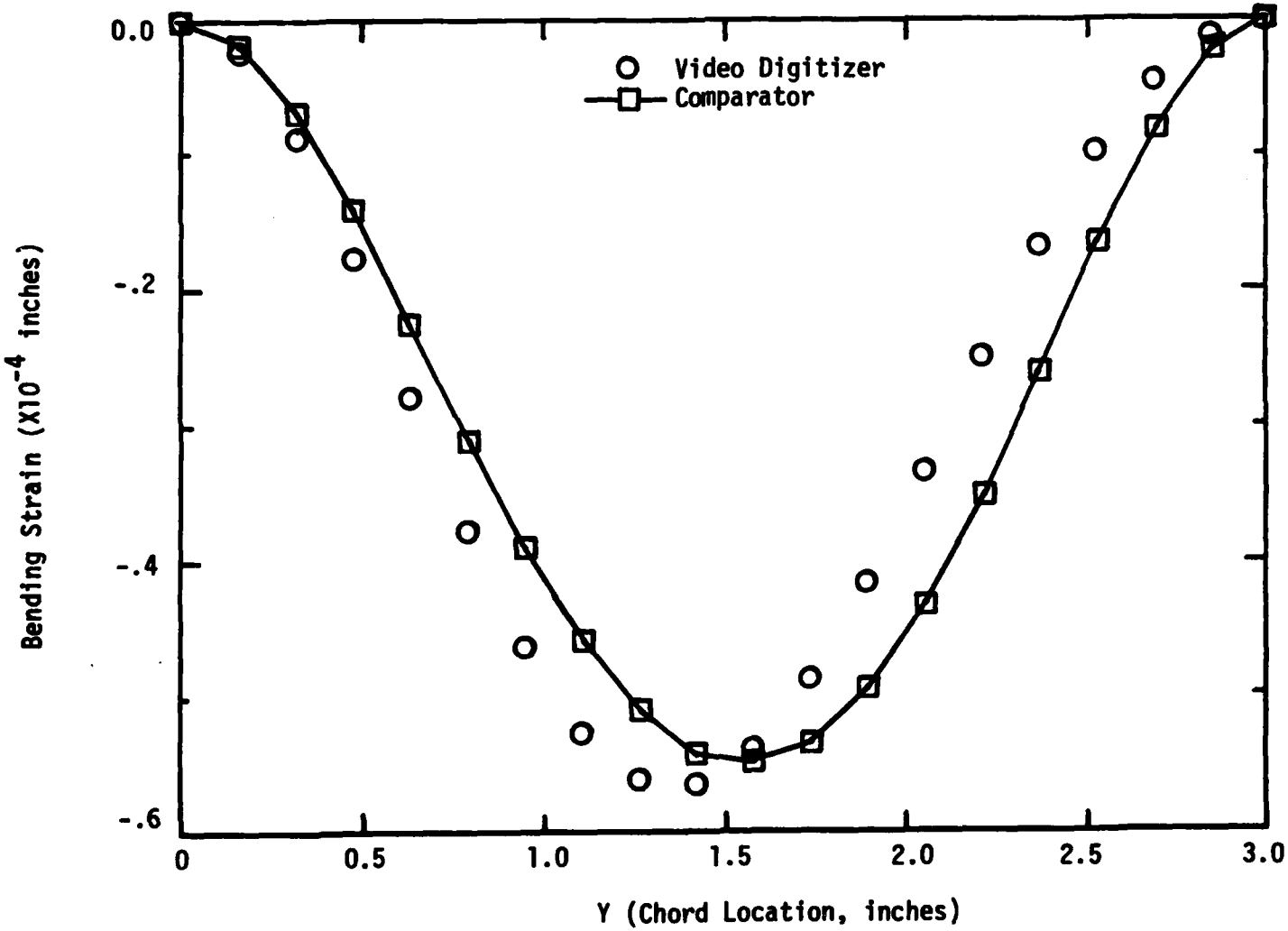


Figure B-5. Lyre Mode Bending Strain at Constant Span Location, X=2.5 inches

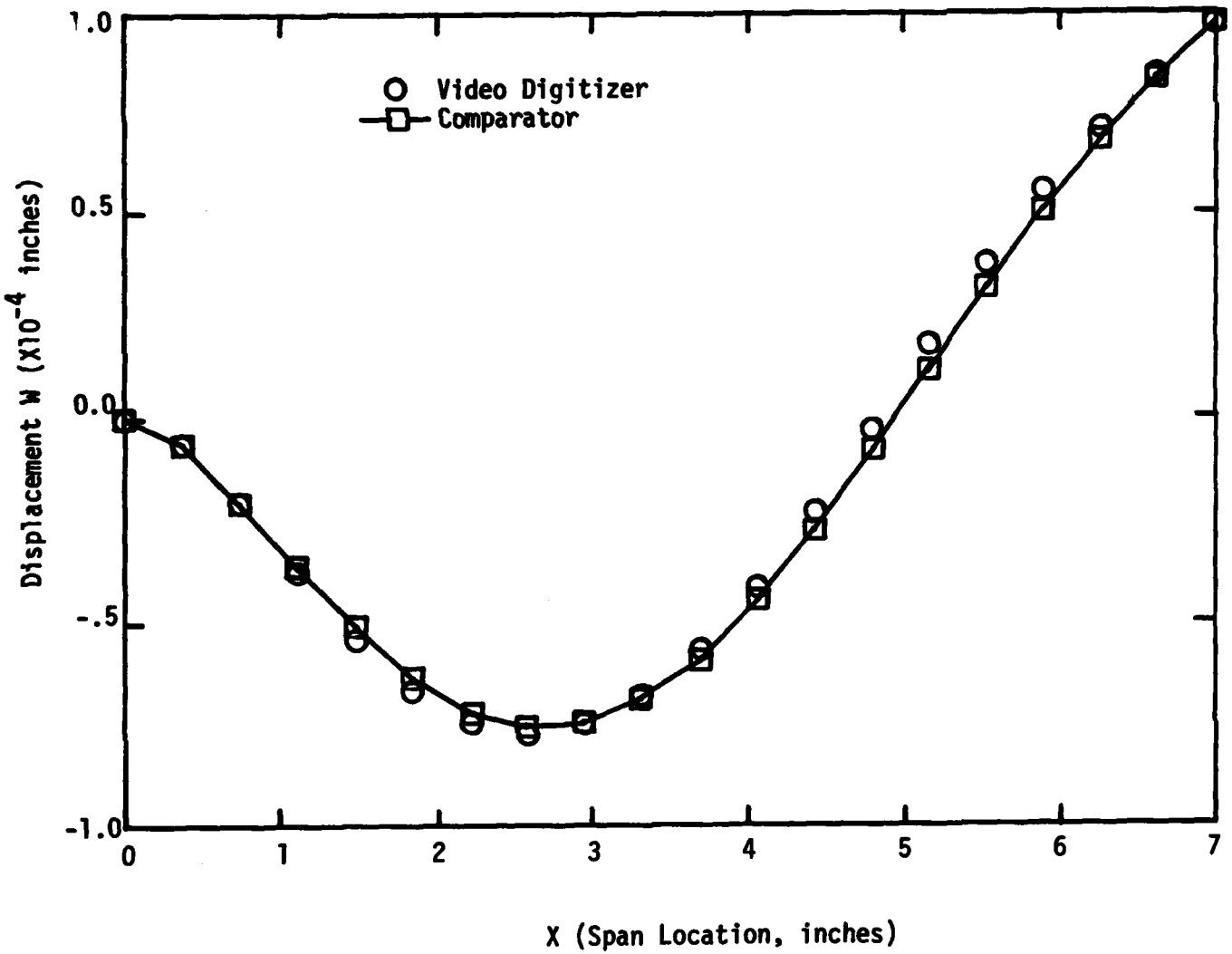


Figure B-6. Second Torsion Mode Out of Plane Displacement W at Constant Chord Location, Y=.5 inches

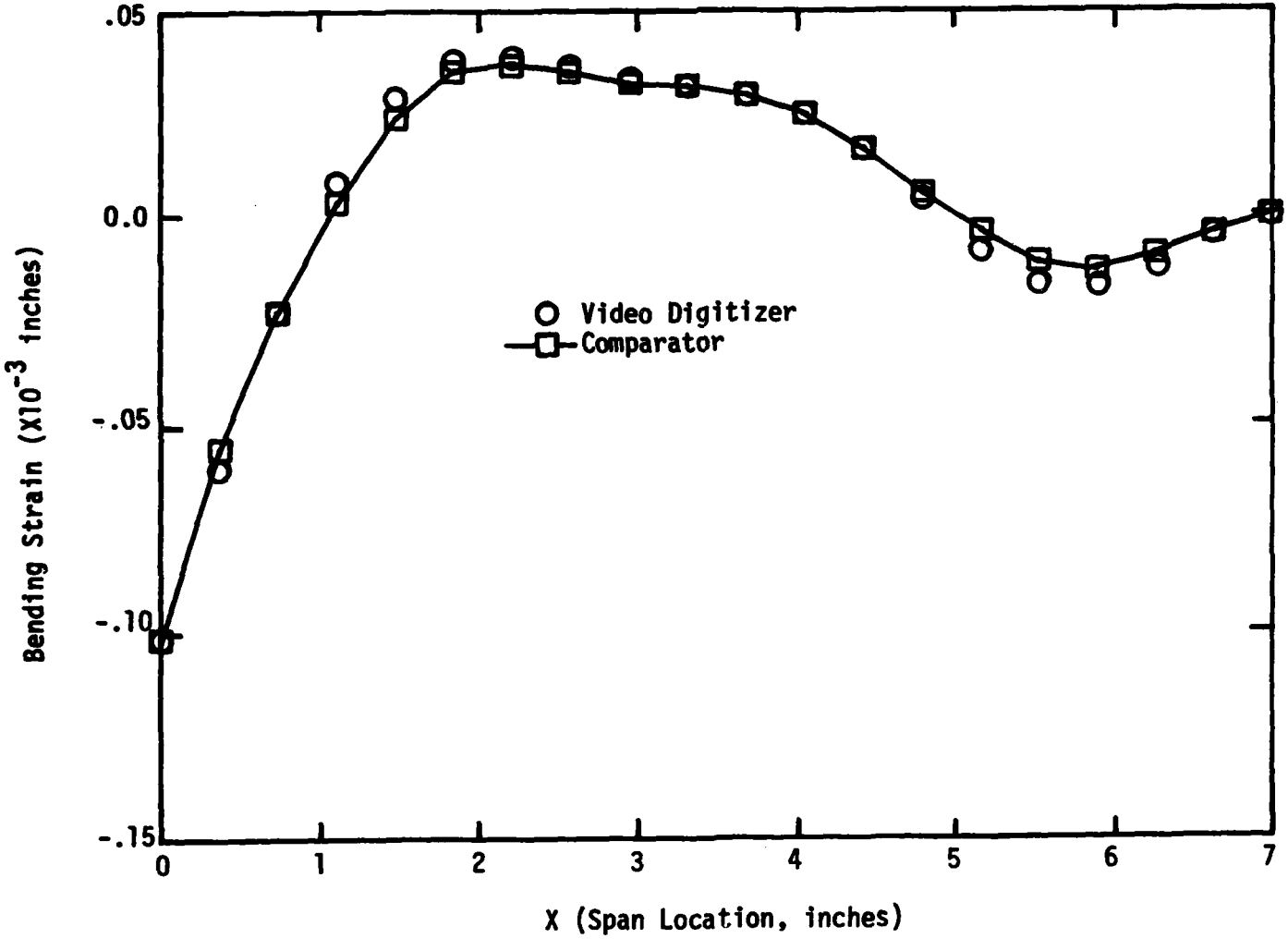


Figure B-7. Second Torsion Mode Bending Strain at Constant Chord Location, Y=.5 inches

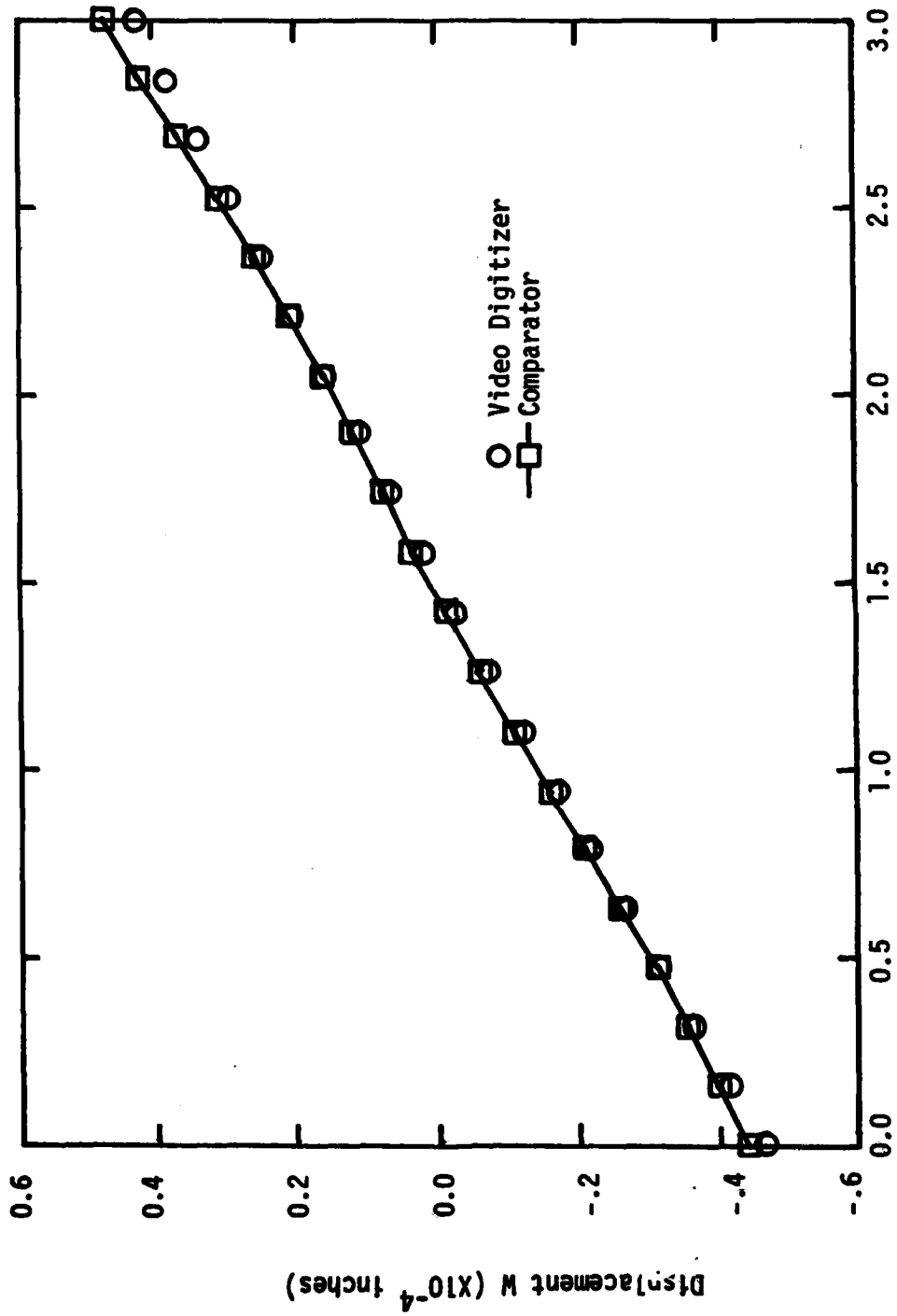


Figure B-8. Second Torsion Mode Out of Plane Displacement W at Constant Span Location, X=1.0 inches

APPENDIX C

SAMPLE RAW DATA

TABLE C-I

Raw Data Array Listing from DIGITIZE
for Heat Transfer Calculations

CONTENTS OF SAV ARRAY

ELEMENTS :	DATA:			
1, 2, 3, 4	-.0100	.0000	.0000	.0000
5, 6, 7, 8	.0100	.0000	.0000	.0000
9, 10, 11, 12	.1000	.0000	.2000	.0000
13, 14, 15, 16	.0000	-.1875	.0000	-.5625
17, 18, 19, 20	.1000	.0000	.2000	.0000
21, 22, 23, 24	.0000	-.1562	.0000	-.3906
25, 26, 27, 28	.1000	.0000	.2000	.0000
29, 30, 31, 32	.0000	-.1250	.0000	-.3125
33, 34, 35, 36	.1000	.0000	.2000	.0000
37, 38, 39, 40	.0000	-.1250	.0000	-.2656
41, 42, 43, 44	.1000	.0000	.2000	.0000
45, 46, 47, 48	.0000	-.1094	.0000	-.2344
49, 50, 51, 52	.1000	.0000	.2000	.0000
53, 54, 55, 56	.0000	-.0937	.0000	-.2187
57, 58, 59, 60	.1000	.0000	.2000	.0000
61, 62, 63, 64	.0000	-.0937	.0000	-.2031
65, 66, 67, 68	.1000	.0000	.2000	.0000
69, 70, 71, 72	.0000	-.0937	.0000	-.1875
73, 74, 75, 76	.1000	.0000	.2000	.0000
77, 78, 79, 80	.0000	-.0781	.0000	-.1719
81, 82, 83, 84	.1000	.0000	.2000	.0000
85, 86, 87, 88	.0000	-.0781	.0000	-.1719
89, 90, 91, 92	.1000	.0000	.2000	.0000
93, 94, 95, 96	.0000	-.0781	.0000	-.1562
97, 98, 99,100	.1000	.0000	.2000	.0000
101,102,103,104	.0000	-.0781	.0000	-.1562
105,106,107,108	.1000	.0000	.2000	.0000
109,110,111,112	.0000	-.0781	.0000	-.1562
113,114,115,116	.1000	.0000	.2200	.0000
117,118,119,120	.0000	-.0781	.0000	-.1406
121,122,123,124	.1000	.0000	.2000	.0000
125,126,127,128	.0000	-.0781	.0000	-.1562
129,130,131,132	.1000	.0000	.2200	.0000
133,134,135,136	.0000	-.0781	.0000	-.1562
⋮	⋮	⋮	⋮	⋮
393,394,395,396	0.0000	0.0000	2.0200	0.0000
397,398,399,400	1.9200	0.0000	1.8200	0.0000
401,402,403,404	1.7200	0.0000	1.6200	0.0000
405,406,407,408	1.5200	0.0000	1.4200	0.0000
409,410,411,412	1.3200	0.0000	1.2200	0.0000
413,414,415,416	1.1200	0.0000	1.0200	0.0000
417,418,419,420	.9200	0.0000	.8200	0.0000
421,422,423,424	.7200	0.0000	.6200	0.0000
425,426,427,428	.5200	0.0000	.4200	0.0000
429,430,431,432	.3200	.8828	1.0900	19.0000

TABLE C-II

Raw Data Array Listing from DIGITIZE for Strain Calculations via HOLOCURVE

CONTENTS OF SAV ARRAY

ELEMENTS :	DATA:			
1, 2, 3, 4	1.1765	.5000	1.4412	.5000
5, 6, 7, 8	1.6765	.5000	1.8824	.5000
9, 10, 11, 12	2.1176	.5000	2.3824	.5000
13, 14, 15, 16	2.6176	.5000	2.8529	.5000
17, 18, 19, 20	3.1471	.5000	3.3824	.5000
21, 22, 23, 24	3.6765	.5000	3.9118	.5000
25, 26, 27, 28	4.2059	.5000	4.4412	.5000
29, 30, 31, 32	4.7353	.5000	4.9706	.5000
33, 34, 35, 36	5.2941	.5000	5.5588	.5000
37, 38, 39, 40	5.8529	.5000	6.1471	.5000
41, 42, 43, 44	6.5000	.5000	6.9118	.5000
45, 46, 47, 48	7.3235	.5000	0.0000	0.0000
49, 50, 51, 52	0.0000	0.0000	0.0000	0.0000
53, 54, 55, 56	0.0000	0.0000	0.0000	0.0000
57, 58, 59, 60	0.0000	0.0000	0.0000	0.0000
61, 62, 63, 64	0.0000	0.0000	0.0000	0.0000
65, 66, 67, 68	0.0000	0.0000	0.0000	0.0000
69, 70, 71, 72	0.0000	0.0000	0.0000	0.0000
73, 74, 75, 76	0.0000	0.0000	0.0000	0.0000
77, 78, 79, 80	0.0000	0.0000	0.0000	0.0000
81, 82, 83, 84	0.0000	0.0000	0.0000	0.0000
85, 86, 87, 88	0.0000	0.0000	0.0000	0.0000
89, 90, 91, 92	0.0000	0.0000	0.0000	0.0000
93, 94, 95, 96	0.0000	0.0000	0.0000	0.0000
97, 98, 99,100	0.0000	0.0000	0.0000	0.0000
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
397,398,399,400	0.0000	0.0000	0.0000	0.0000
401,402,403,404	0.0000	0.0000	0.0000	0.0000
405,406,407,408	0.0000	0.0000	0.0000	0.0000
409,410,411,412	0.0000	0.0000	0.0000	0.0000
413,414,415,416	0.0000	0.0000	0.0000	0.0000
417,418,419,420	0.0000	0.0000	0.0000	0.0000
421,422,423,424	0.0000	0.0000	0.0000	0.0000
425,426,427,428	0.0000	0.0000	0.0000	0.0000
429,430,431,432	0.0000	.0147	2.2941	1.0000

TABLE C-III

Sample DIGITIZE Output for Heat Transfer Calculations

```
D1          =      2.3500
D2          =       .1011
TEMP        =    522.0000
PRESSURE    =   2066.0000
K           =       .2400

QLOC( 2) =     -5.1261
               -1.8967

QLOC( 3) =     -5.9682
                -.5968

QLOC( 4) =     -7.4602
                -.7460

QLOC( 5) =     -6.9117
                -.6912

QLOC( 6) =     -7.9457
                -.7946

QLOC( 7) =     -9.7099
                -.9710

QLOC( 8) =     -9.3384
                -.9338

QLOC( 9) =     -8.7456
                -.8746

QLOC(10) =    -11.3122
               -1.1312

QLOC(11) =    -11.3122
               -1.1312

QLOC(12) =    -10.4948
               -1.0495

QLOC(13) =    -10.4948
               -1.0495

QLOC(14) =    -10.4948
               -1.0495

QLOC(15) =     -7.5010
                -.7501

QLOC(16) =    -10.4948
               -1.0495

QLOC(17) =     -9.4155
                -.9416

QLOC(18) =     -6.5896
                -.6590

QLOC(19) =    -11.2099
               -4.2598

QTOT     =    -20.5753
```

Vita

Captain David R. Chaffee was born 13 January 1955 in Rockwood, Tennessee. He graduated from W.A. Berry High School, Birmingham, Alabama in 1973. In that year, he received an appointment to the United States Air Force Academy and graduated in 1977 with a Bachelor of Science Degree in Aeronautical Engineering. He attended Under Graduate Pilot Training from June 1977 to May 1978 and was then assigned to the Air Force Aero Propulsion Laboratory in June 1978. After this tour of duty, Captain Chaffee entered the Graduate Aeronautical Engineering Program at the Air Force Institute of Technology in June 1981.

Permanent Address: 11702 Cypresswood Drive
Houston, Texas 77070

This thesis was typed by Ms Sandra Lovely.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GAE/AA/82D-5	2. GOVT ACCESSION NO.: AD-A124751
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): VIDEO DIGITIZATION OF DISPLACEMENT/GRADIENT DATA FROM OPTICAL INSTRUMENTATION SYSTEMS	5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): David R. Chaffee, Capt USAF	8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT/EN) Wright-Patterson AFB, Ohio 45433	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:	12. REPORT DATE: December 1982
	13. NUMBER OF PAGES: 110
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

6 JAN 1983

Approved for public release; IAW AFR 190-17

Approved for public release: IAW AFR 190-17.
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Digitizer
Video Digitizer
Holograph data reduction
Interferometry Data Reduction

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A general purpose video digitizer is developed to be used for a broad range of optical data acquisition techniques. It allows the engineer to digitize optical data using a computer image and cursor control to build an array of coordinates or displacements for subsequent conversion to engineering results. An assessment of this technique is made qualitatively and quantitatively and discussion is presented on its potential applications and accuracy.

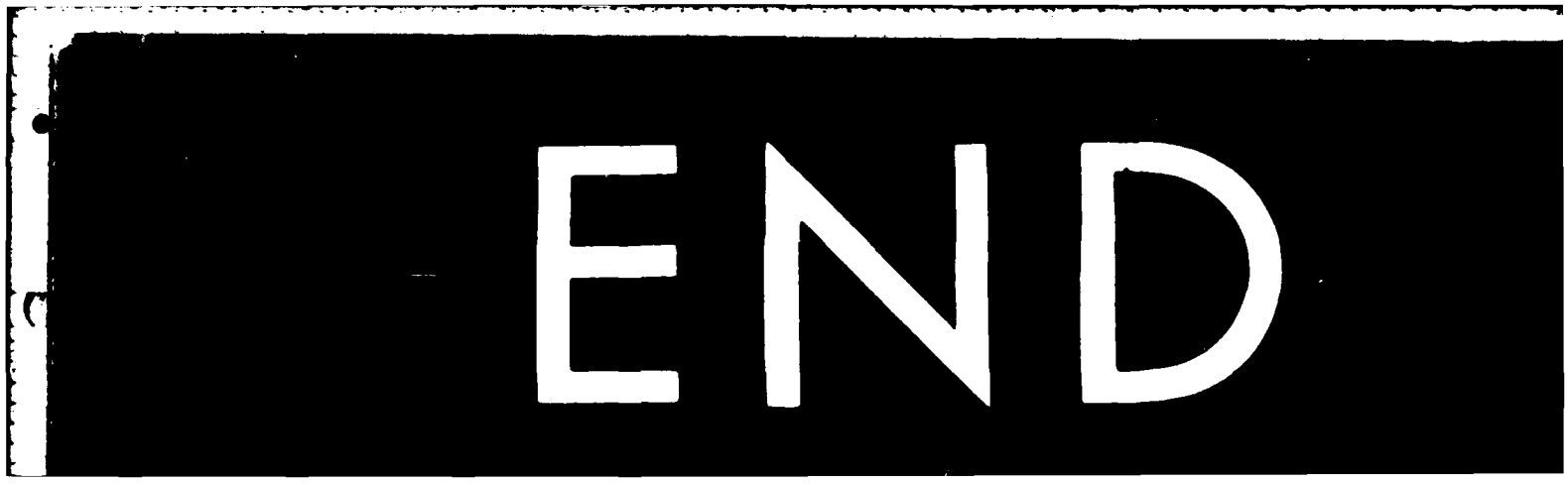
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Free convective heat transfer interferograms are digitized and the data reduced on the same microcomputer system. Results from this data compare favorably to those using a microcomparator digitizer. Time averaged holograms from a vibration rig are reduced with the video processor. These results are also favorable when compared to similar data taken with an optical comparator. Displacement and bending strain plots are overlaid using both data sets and show excellent correlation.

Recommendations are made for expanding and improving the digitizer so that it may achieve its full potential.


END