Reports and Technical Reports | All Technical Reports Collection

1987-08

Using the Eikonix digitizer camera with the IRIS graphics work-station

Sando, Jean M.; Wetherald, Thomas S.; Zyda, Michael J.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/26325

NPS52-87-038

NAVAL POSTGRADUATE SCHOOL
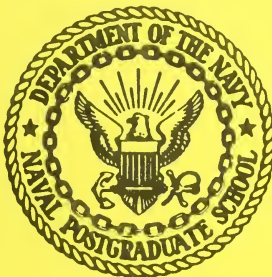
Monterey, California

USING THE EIKONIX DIGITIZER CAMERA WITH
THE IRIS GRAPHICS WORKSTATION

Jean M. Sando
Thomas S. Wetherald
Michael J. Zyda

August 1987

Approved for public release; distribution unlimited

Prepared for:

Naval Ocean Systems Center
San Diego, CA 92152

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. C. Austin
Superintendent

D. A. Schrady
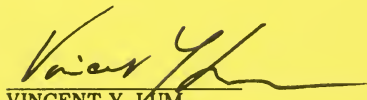Provost

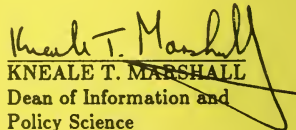Reproduction of all or part of this report is authorized.

This report was prepared by:

MICHAEL J. ZYDA
Associate Professor
of Computer Science

Reviewed by:

VINCENT Y. LUM
Chairman
Department of Computer Science

Released by:

KNEALE T. MARSHALL
Dean of Information and
Policy Science

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION classified		1b RESTRICTIVE MARKINGS			
SECURITY CLASSIFICATION AUTHORITY		3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution unlimited.			
DECLASSIFICATION / DOWNGRADING SCHEDULE					
ERFORMING ORGANIZATION REPORT NUMBER(S) S52-87-038		5 MONITORING ORGANIZATION REPORT NUMBER(S)			
NAME OF PERFORMING ORGANIZATION val Postgraduate School	6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION Naval Ocean Systems Center			
ADDRESS (City, State, and ZIP Code) nterey, CA 93943		7b. ADDRESS (City, State, and ZIP Code) San Diego, CA			
NAME OF FUNDING / SPONSORING ORGANIZATION Naval Ocean Systems Center	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N0001487WX4B419AB			
ADDRESS (City, State, and ZIP Code) San Diego, CA 92152		10. SOURCE OF FUNDING NUMBERS			
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO	WORK UNIT ACCESSION NO.

ITLE (Include Security Classification)

ng the Eikonix Digitizer Camera with the IRIS Graphics Workstation

ERSONAL AUTHOR(S)
n M. Sando, Thomas S. Wetherald and Michael J. Zyda

TYPE OF REPORT hnical	13b TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) August 1987	15 PAGE COUNT 40

UPPLEMENTARY NOTATION

COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	

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

describe in this document the operations of the Eikonix digitizer camera and the display
the digitized images on the IRIS graphics workstation. We also present source code
tings for both the camera and the display software.

ISTRIBUTION / AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified	
NAME OF RESPONSIBLE INDIVIDUAL ael J. Zyda	22b. TELEPHONE (Include Area Code) (408) 646-2305	22c. OFFICE SYMBOL 52Zk

ORM 1473, 84 MAR 83 APR edition may be used until exhausted.   SECURITY CLASSIFICATION OF THIS PAGE
All other editions are obsolete

☆ U.S. Government Printing Office: 1986—806-24.

Using the Eikonix Digitizer Camera with the IRIS Graphics Workstation ‡

Jean M. Sando, Thomas S. Wetherald and Michael J. Zyda *

Naval Postgraduate School,
Code 52, Dept. of Computer Science,
Monterey, California 93943

ABSTRACT

We describe in this document the operation of the Eikonix digitizer camera and the display of the digitized images on the IRIS graphics workstation. We also present source code listings for both the camera and the display software.

TABLE OF CONTENTS

1

LIST OF TABLES

I. INTRODUCTION

A. PURPOSE OF THIS DOCUMENT

This document describes the operation of the EIKONIX digitizer camera and the IRIS graphics workstation in regards to the display of data from the EIKONIX camera. This document also includes software documentation for both the camera and display software as well as the source code for both programs.

B. DESCRIPTION OF DIGITIZING CAMERA

The EIKONIX 785 Digitizer Camera System is an image digitizer capable of storing an image with a resolution of 4096 X 4096 pixels. The camera is able to store both color and black and white image data. The camera is currently fitted with a fixed focal length 55mm SLR lens with an f-stop range of 2.8 to 22. The camera is suspended from a variable height backplane that provides for camera heights between 32 and 140 centimeters. The camera system is equipped with four fixed flood lights attached to the system base board. The camera circuitry is located in a separate box referred to as the camera controller. All switches and indications associated with camera operation are on this panel unless otherwise noted.

The camera head, to which the lens is attached, slides forward and backward to two positions. In the back position, the image can be seen through the view port on the front of the camera head. This position is used to position the target image and focus the lens. The forward position is used by the camera to digitize the image. In this position the image is not visible to the user. NOTE: The movement of the camera head can cause the backplane and camera head to oscillate. Wait for the oscillations to cease before starting the digitizer.

C. DESCRIPTION OF THE DISPLAY SYSTEM

The software presented in this manual is designed specifically to run on a Silicon Graphics IRIS Graphics Workstation. The IRIS workstation is capable of displaying both color and black and white data from the digitizer camera. However, the resolution is limited to 768 X 1024 pixels. Thus for display purposes, digitizing an image larger than this is not possible. If new display hardware capable of displaying larger images is acquired in the future, the display software can be easily modified to

4

display a larger image. The IRIS system can display either twenty-four bit color information as well as color or black and white information contained in twelve bits or less. Software has been developed to utilize both of these modes. Color data can be displayed using the twenty-four bit mode in up to $2**24$ different colors. Color data can also be displayed using eight bits of information in up to 216 different colors.

D. POSSIBLE APPLICATIONS OF THE CAMERA AND DISPLAY SYSTEMS

To date, data from the digitizing camera has been used in the following applications:

- Texture analysis of an aerial photo with the long term goal of contributing to the direction of an autonomous land vehicle.
- A texture editor.
- Enhancements to a command and control workstation that allow a library picture of a particular threat platform to be displayed in a window in the command and control display.

Using the eight bit color mode, it is possible to use the data for texture and animation purposes. Another planned use of this data is a font editor that allows a printed font to be extracted from a digitized image. One planned font extraction is that of a Japanese kana set. It is also possible to use the digitized version of an aerial photograph to provide a data base for a flight simulator.

5

II. USE OF EIKONIX DIGITIZER CAMERA

A.   HARDWARE SETUP

The EIKONIX Digitizer camera is located in Spanagel Hall room 513. It is hardwired to the VAX 11/780 (VAX/VMS) computer. The following is a list of switches and indicators that appear on the front of the camera normalizer board:

- power button
- scan button - sometimes referred to as the run switch
- error light - located on top of scan button
- color filter wheel - usually set in software, however for manual setting the color wheel positions are as follows:
  0.   clear
  1.   red
  2.   green
  3.   blue
  4.   dark shutter
  5.   diffuser
- frame size wheel - usually set in software, however for manual setting the frame size wheel positions are as follows:
  0.
  1.   4096 x 6400 origin 0,0
  2.   2048 x 2048 origin 0,0
  3.   1024 x 1024 origin 0,0
  4.   512 x 512 origin 0,0
  5.   4096 x 4096 origin 0,0
  6.   2048 x 2048 origin 1024,1024
  7.   1024 x 1024 origin 1536,1536
  8.   512 x 512 origin 1792,1792
- exp time - usually set in software, however for manual setting, it is used in conjunction with the FSL/EXL meter, and indicates the highest light intensity by a high needle indication.
- FSL/EXL meter

The flood light switches are located on the front of the camera base board. Once on, the lights can be individually controlled by switches on the back of each light.

The VAX/VMS operating system must be up for the camera to operate. If the camera has been powered off, it must be turned on and recalibrated before attempting to digitize an image. NOTE: The camera is normally left on!

B.   OPERATION OF CAMERA

1. Modes of operation

The software in this package utilizes two of the camera's digitizing modes; black and white and color. The black and white mode uses the clear filter, wheel position 0. The color mode scans the image three times using in succession the red, green and blue filters, wheel positions 1, 2 and 3. Software later combines the three sets of data to provide a single color file.

The camera software provides two methods to determine which portion of the image is to be digitized. The digitized image is either centered on the lens cross hairs or taken offset from the upper right hand corner of the image visible through the lens.

The camera is also run periodically in calibrate mode to set gain and bias corrections. This mode is used only after the camera has been turned off and then turned back on. The calibrate procedure is discussed later.

2. User Instructions

a. Digitizing Images

To digitize an image do the following:

1.   If the camera is not powered on, see calibration procedure.
2.   Remove lens cap and power on lights.
3.   Place camera head in the back position. See introduction for details.
4.   Position picture. See placement section below for details.
5.   Set F-stop on camera lens. See lighting section for details.
6.   Focus lens.
7.   At a VAX/VMS terminal type < camera >. A system path has been established that makes "camera" available from the user's directory.
8.   Answer each question as asked. An understanding of the modes of operation and subject placement sections is necessary.
9.   When directed, push the scan button.
10. Depending on the digitizing mode and the size of the of the scanned image, the digitizing program may take from ten seconds to five minutes to run. Be patient.
11. Program will terminate with a "FORTRAN STOP".

b. Calibration

The program to calibrate the camera is located in directory DRA0:[EIKONIX]. It has the file name calibrate.exe. The procedure to perform calibration follows:

1. login and change directory to DRA0:[EIKONIX] by entering the following: < set def DRA0:[EIKONIX]>.
2. Insure camera is powered on (the power light should be on).
3. Insure camera head is in forward position (pulled out).
4. Type < run calibrate>.
5. The first question to appear on the screen is: "Calibrate in transmission on a Series 785?". Enter < N >.
6. The last question is "what is the name of the camera". Enter < IDA0:>. (note the last character is a zero!)
7. Continue to follow instructions printed on the screen.
8. Program will terminate with a "FORTRAN stop".

NOTE - The error light will be lit during part of the calibration procedure, this is normal.

C. PHOTOGRAPHIC ASPECTS OF CAMERA OPERATION

1. Subject Placement Considerations

The camera software allows the user to select a portion of the image to digitize using two different methods. The user can choose the center of the image as seen through the camera to be the center of the digitized image, giving the number of columns (pixels) wide and the number of lines (pixels) deep. The second method is more complicated, but is useful if the image can not be centered. This involves considering the image as viewed thought the camera to have an origin at the upper right corner. The user then specifies a column and row offset to begin digitizing as well as the number of columns and rows of the image to be digitized.

When using the centered mode, the center of the image is determined by the cross hairs seen through the camera lens. The first set of lines on the horizontal and vertical cross hairs are 1000 pixels apart. The second set of lines are 2000 pixels apart. The user will notice that only a small portion of the image that is visible through the camera can be displayed (1024 X 768 pixels). The centered mode is by far the easiest of the two to use.

There is a possible solution to the image size limitation that has not yet been explored. The IRIS window manager tools library contains a routine called "shrink", that should allow a large image to be digitized and then scaled down to be displayed on an IRIS display screen. This routine is located in the following directory: usr/people/gifts/mextools/imgtools. The file format required by this routine is unknown.

2. Photograph Size Considerations

The actual area of the image digitized depends on the height of the camera head and the scan size given during execution of the camera program. To aid in choosing the correct height and size, Table 1 is provided. Table 1 gives for a specified camera head height (Ht) the number of pixels per centimeter of measured distance on the surface of the target image. Also included are several popular scan sizes and the approximate number of centimeters of the target image that will be digitized for each.

TABLE 1

IMAGE SIZE CONVERSION TABLE

Ht	Pix/cm	100	300	512	750	768	900	1024
32	286	.35	1.05	1.80	2.62	2.68	3.15	3.58
35	267	.37	1.12	1.92	2.81	2.87	3.37	3.83
40	229	.44	1.31	2.24	3.28	3.35	3.93	4.47
45	200	.50	1.50	2.56	3.75	3.83	4.50	5.12
50	178	.56	1.69	2.88	4.21	4.30	5.16	5.75
55	160	.63	1.88	3.20	4.69	4.79	5.63	6.40
60	146	.68	2.06	3.51	5.14	5.25	6.16	7.01
65	133	.75	2.26	3.85	5.64	5.76	6.77	7.70
70	120	.83	2.50	4.27	6.25	6.38	7.50	8.53
75	113	.89	2.66	4.53	6.64	6.78	7.97	9.06
80	109	.92	2.75	4.70	6.88	7.03	8.26	9.40
85	100	1.00	3.00	5.12	7.50	7.66	9.00	10.24
90	93	1.08	3.23	5.51	8.07	8.24	9.68	11.01
95	89	1.12	3.37	5.75	8.43	8.61	10.11	11.51
100	85	1.18	3.53	6.02	8.82	9.01	10.59	12.05
105	80	1.25	3.75	6.40	9.38	9.58	11.25	12.80
110	77	1.30	3.90	6.65	9.74	9.95	11.69	13.30
115	73	1.37	4.11	7.01	10.27	10.49	12.33	14.03
120	70	1.43	4.29	7.31	10.71	10.94	12.86	14.63
125	67	1.49	4.48	7.64	11.19	11.43	13.43	15.28
130	63	1.59	4.76	8.13	11.91	12.16	14.29	16.25
135	60	1.67	5.00	8.53	12.50	12.77	15.00	17.07
140	57	1.75	5.26	8.98	13.16	13.44	15.79	17.97

3. Aspects of Lighting

The camera is equipped with four individually controlled spot lights. Each light can be adjusted to be on/off and/or shining directly on the camera base platform or positioned to produce indirect lighting. Additionally, the camera is equipped with a standard 55mm SLR lens with an f-stop range of 2.8 to 22. The base platform of the camera is bright white, but can be covered with a mat that has a light gray or dark gray side. All of these items together must be adjusted in order to get desired image quality. The FSL/EXL meter is apparently an exposure meter, however it does not appear to work correctly. The documentation from EIKONIX does not adequately cover the use of this meter. Perhaps some future user will figure it out. Until then, the meter does not work. The best advice is to experiment, however for a 'normal' color picture, a good start is f5.6, light gray mat and all lights on, directed towards the image. Take a series of small images (100 to 200 pixels) until the correct lighting effect is achieved and then take the larger images. This will save a great deal of time.

III. USE OF THE IRIS DISPLAY SYSTEM

A.   TRANSFER OF DIGITIZED DATA TO IRIS

The digitized data must be transferred from the VAX/VMS system to the IRIS workstations to be displayed. Other display systems may be available in the future. The data is transferred using the File Transfer Protocol (FTP) on the Ethernet local area network. Data must be transferred in the binary mode. The procedure for using FTP from an IRIS terminal follows:

1.   Log onto an IRIS terminal and change directory into the directory where the data is to be stored.

2.   Type < ftp npscs-vms1 >

3.   Follow logon procedures.

4.   When the ftp prompt is received, type < bin > to put the link in binary transfer mode. Failure to do this can cause the displayed image to be skewed.

5.   The command to retrieve the file is command "get" followed by the VMS filename and then by the filename to be used to store the file on the IRIS system. Command syntax follows:
    Type < get filename filename CR >

6.   When transfer terminates, type < quit > to return to the UNIX shell on the IRIS.

B.   DISPLAY AND STORAGE MODES

All images are displayed by typing < display > followed by one or two file name parameters depending on the storage mode used.

1. **Black and White**

Black and white image data is stored as eight bits of information per pixel. This allows 256 grey levels. To display a black and white image the image must have been digitized by the EIKONIX camera in black and white mode. When an image file is created by the camera software, the display mode is indicated in the file header and can not be changed. Black and white images are displayed in the IRIS RGB mode. However, a color file can not be displayed in black and white and vice versa.

2. **Color**

Color files are stored with twenty-four bits of color information for each pixel. The twenty-four bits are made up of eight bits each of red, green, and blue color

information. Files of this type are very large and take a considerable amount of the available IRIS storage. A 750 X 750 pixel color image takes approximately 1.7 megabytes to store. These color images are displayed in the IRIS RGB mode which provides 2**24 different colors. A large image will take about ten seconds to draw on an IRIS screen.

To display a full color image, "display" is executed with one filename parameter which is the name of the color image file. The user is asked if he desires to display or dither the image, if the file is not already dithered. If display is chosen, the software will determine from the file header that a color file is to be displayed.

3. **Dithering Color Files**

An additional mode for displaying color information is provided in this package that reduces the size of the stored file by two thirds. The process is referred to as dithering. The twenty-four bits of color information are compressed into eight bits. With this information, the software in this package can display a color image in up to 216 colors while the file requires only one third of the storage space required by a full color file. The dithering process is time consuming, as much as five minutes for a large image. The image also loses some degree of sharpness. A dithered image also takes considerably longer to display. However, dithered images can be displayed in the IRIS single buffer mode rather than RGB mode. If a future application requires the use of buffer switching, a dithered file could easily be displayed in double buffer mode where as a full color RGB file can not.

4. **Storage and Display of Dithered Color Files**

To create a dithered file, an image is first digitized by the camera in normal color mode. After the full color file is transferred to the IRIS, the display software is used to create the dithered file.

A dithered file is created by executing the "display" program and using two file name parameters. The first is the name of the full color file and the second is the file to be opened for the dithered image. The user must know prior to running "display" that an image is to be dithered. There is no chance later on to provide the dithered file name. The program terminates after the dithered file is finished. To save space, the full color file must be deleted by the user.

Dithered color images are displayed by executing the "display" program with a file name parameter of a file that has already been dithered. The software will know that the file is ready to be displayed.

12

C. DETAILED USER INSTRUCTIONS

The software in this package must be run from an IRIS side terminal. All displays are directed to the associated IRIS display terminal. If the display terminal is in use, this software will cause the digitized image to over write any work in progress.

1. Black and White

To display a black and white image type < display filename CR >, where filename is the name of a file that contains the black and white image data. The image will be displayed on the large screen with the comment (if applicable) at the bottom of the image. To terminate the program press the right mouse button.

2. Display of Color or Dithered Images

To display a color image (full or dithered) follow these steps:

1. Type < display filename CR >, where filename is the name of the file that contains the image.

2. The question "Do you want to store a dithered file?" is written on the screen, if the file has not already been dithered. Type < n >.

3. The image will be displayed on the large terminal.

4. To terminate the program, press the right mouse button.

3. Dithering an Image

To dither an image complete the following:

1. Type < display filename filenameout CR >, where filename is the name of the file that contains the image and filenameout is the name of the file that the program will create for the dithered image.

2. The question "Do you want to store a dithered file?" is written on the screen. Type < y >.

3. The next question is "enter the number of intensity levels (1-6)". Six levels will give the maximum number of dithered colors (216). Two levels will provide eight colors.

4. The next question is "enter gamma factor". The gamma factor must be a positive floating point number. The normal value is 1.5. The program is actually not very sensitive to changes in gamma factor.

5. The program will execute at this point giving progress reports as it goes.

6. When dithering is completed, terminate the program by pressing the right mouse button. The program terminate with instructions for displaying the dithered image.

4. Error Messages

1. If the filename(s) are omitted from the call to "display", the program will begin to execute but will terminate with the error message "We were unable to open the file".

13

2. If the file that is entered to be displayed or dithered is not a file that was created by the "camera" or "display" programs, the message "Picture type was something other than BW, Color, or Dithered" is given.

IV. SOFTWARE DOCUMENTATION

The primary purpose of this section is to discuss the digitized data file formats and the location of the needed files. Further information about the code and algorithms is included as inline comments in the source code (Appendixes A and B).

A. CAMERA SOFTWARE

The camera software is all written in FORTRAN. The "camera" program utilizes a number of the routines that were originally provided with the EIKONIX camera. These routines are primarily used to set various camera parameters. Any modification of the "camera" program will require an understanding of these routines. A list of all of the routines and a brief explanation of each one is contained in the "Image Digitizer Software Library Programming Reference Manual". See Mike Williams (Computer Science Department) for this manual. Just a warning to anyone so inclined, this manual is terse and difficult to understand.

The files created by the camera software are essentially the same for both color and black and white images. All of the data are stored in standard VAX/VMS unformatted sequential files. Each of the files is made up of an eighty-six byte header followed by the data. The header consists of the following:

- The first two bytes are of an integer type and represent the file type. The least significant byte (LSB) comes first. The file types follow:
  1. Black and white
  2. Color
  3. Dithered
- The next two bytes are of an integer type and represent the number of lines of image data stored in the file. Again, the LSB comes first.
- The next two bytes represent the number of columns of image data stored in the file. The format is the same as for lines.
- The next eighty bytes represent a comment field that is optional. If no comment is entered, the field is filled with blank characters. The field is made up of character type data.

The image data follows the header immediately. The image is composed of byte type data that can be thought of as single integer bytes. Black and white data files are made up of sequential bytes representing each image scan line one after another with no

15

demarkation between lines. A color image is made up of red, green and blue data that result from three separate camera scans. Each line of image data is stored in three parts, a line of red data followed by a line of green and then a line of blue. There is no demarkation between red, green and blue parts or between lines.

B.   DISPLAY SOFTWARE

The only file created by the "display" program is a type three or dithered image file. The header in this file is exactly the same as the black and white and color files described above. However, the data is composed of indices into the IRIS mapcolor function. The indices are stored as single character data. Implicit coercion is used to write integers out to this file and to convert the character data into integers during the display process.

The algorithm for the dithering program came from the University of Utah, Computer Science Department, IRIS graphics package that is released for public use.

C.   SOFTWARE LIBRARY

The "camera" and "calibrate" programs as well as all of the routines provided with the camera are stored on the VAX/VMS system in directory < DRA0:[EIKONIX]>.

The "display" program for the IRIS system is stored on the VAX/UNIX System in directory < work2/zyda/digit>.

APPENDIX A
CAMERA PROGRAM SOURCE CODE

The code on the following pages is the source code for the "camera" program that is used in conjuction with the EIKONIX digitizer camera to digitize black and white and color images.

```
C **********************************************************************
C CAMERA
C        ROUTINE DESCRIPTION
C               This program is used to digitize an image using the EIKONIX
C               Digitizer Camera. It  places the digitize data in a file
C               specified by the user.
C
C        COMMON VARIABLES USED
C               none
C
C        INTERNAL VARIABLES USED
C         Name          Description
C
C         ICHAN         INTEGER*4 contains the channel number assigned to
C                       the digitizer camera, at NPS 'IDAO:'
C         ISTEP         INTEGER*4 number of steps to be taken after each
C                       line is digitized
C         LINE          INTEGER*4 line counter for error reporting
C         NDIODES       INTEGER*4 number of diodes in digitizer array
C         DIGITIZER     CHARACTER*8 for name assigned to digitizer
C         BUFFER        BYTE buffer used for black and white images
C         BUFFERR       BYTE buffer used for color images, to hold red data
C         BUFFERG       BYTE buffer used for color images, to hold green data
C         BUFFERB       BYTE buffer used for color images, to hold blue data
C         OUT_FILE_NAME CHARACTER*25 holds name of output file
C         COMMENT       CHARACTER*80 used to insert comment into output file
C         TYPEPIC       INTEGER*2 indicates type of picture 1 black/white 2 color
C         LINES         INTEGER*2 used in centered image,
C                       the # of lines to digitize, # lines = # pixels
C         COLS          INTEGER*2 used in centered image,
C                       the # of columns to save, # columns = # pixels
C         BWTIME        INTEGER black and white integration time
C         REDTIME       INTEGER red integration time
C         BLUETIME      INTEGER blue integration time
C         GREENTIME     INTEGER green integration time
C         WINBIT        INTEGER see vendors documentation
C         STARTLINE     INTEGER line digitizer will start on
C         LASTLINE      INTEGER last line to be digitized
C         STARTCOL      INTEGER first column of digitized data to be saved
C         LASTCOL       INTEGER last column of digitized data to be saved
C         ORIGIN        INTEGER indicates if image is centered under camer lens
C
C        ROUTINES CALLED
C
C        IDOPEN, IDREADY, IDSCFW, IDSTIM, IDSWIN, IDSPOS, IDSCAB
C
C
C
C        FILES USED
C
C        OUT_FILE_NAME- An output file for the digitiized data
C        COLOR.RED, COLOR.GREEN, COLOR.BLUE- Temporary files, deleted if
C                                            normal termination of program
C
C
C        COMPILATION INSTRUCTIONS
C
C        VAX FORTRAN Version 2.0 or later
C        Link with DIGITIZER.OLB
C
C        PROGRAMMER: LT JEAN SANDO USN
C                    LT TOM WETHERALD USN
```

```
PARAMETER    MAX ARRAY=4096
INTEGER*4    ICHAN,ISTEP,LINE,NDIODES
CHARACTER*8  DIGITIZER
INTEGER*2    ISTAT(4)
BYTE         BUFFER(MAX_ARRAY)   ! 8-bit data
BYTE         BUFFERR(MAX_ARRAY)  ! 8-bit data
BYTE         BUFFERG(MAX_ARRAY)  ! 8-bit data
BYTE         BUFFERB(MAX_ARRAY)  ! 8-bit data
PARAMETER    SS$_NORMAL=1
CHARACTER*25 OUT_FILE_NAME       ! Output data file
CHARACTER*80 COMMENT
INTEGER*2    TYPEPIC,LINES,COLS
INTEGER      BWTIME,REDTIME,BLUETIME,GREENTIME,WINBIT
INTEGER      STARTLINE,LASTLINE,STARTCOL,LASTCOL,ORIGIN

THE FOLLOWING ITEMS ARE STANDARD FOR NPS AND SHOULD NOT BE CHANGED

DIGITIZER = 'IDAO:'
NDIODES = 4096
WINBIT = 8
BWTIME = 10000
REDTIME =10000
GREENTIME = 12000
BLUETIME = 24000
```

```
C  Open the digitizer channel

        Call IDOPEN (ICHAN,DIGITIZER,ISTAT)
        If (ISTAT(1) .ne. SS$_NORMAL)  GO TO 500

C
C  Select name of data output file
C
        Type *, ' WHAT IS THE OUTPUT FILE NAME?  '
        READ (*,10) OUT_FILE_NAME
10      FORMAT (A)


C Open the output file

        OPEN (UNIT= 20, FILE= OUT_FILE_NAME, ACCESS= 'SEQUENTIAL',
     1      FORM= 'UNFORMATTED', STATUS ='NEW', RECORDTYPE= 'VARIABLE' )

C Have user indicate which type of image to produce

        TYPE *, ' DO YOU WISH TO DO BLACK AND WHITE OR COLOR IMAGE?  '
        TYPE *, ' ENTER 1 FOR BLACK AND WHITE OR 2 FOR COLOR '
        READ(*,11) TYPEPIC
11      FORMAT(I)

        TYPE *, ' DO YOU WISH THE CENTER OF YOUR IMAGE TO BE THE '
        TYPE *, ' SAME AS THE CENTER OF THE CROSS HAIRS ON THE CAMERA? '
        TYPE *, ' ENTER EITHER 1 FOR YES AND 2 FOR NO '
        READ (*,12), ORIGIN
12      FORMAT (I)

        IF (ORIGIN .EQ. 2) THEN


C  User must choose what rectangle under the lens to be digitized

        TYPE *,' ORIGIN IS SET TO UPPER RIGHT OF IMAGE,'
        TYPE *,' AS SEEN THROUGHT THE CAMERA LENS'
        TYPE *,' NOTE- IN REAL WORLD THIS IS THE UPPER LEFT OF IMAGE '
        TYPE *,' WHICH LINE DO YOU WISH TO START DIGITIZING '
        TYPE *,' THE IMAGE ON? '
        READ(*,13),STARTLINE
13      FORMAT(I)

        TYPE *, ' WHAT IS THE LAST LINE YOU WISH TO DIGITIZE '
        READ(*,13),LASTLINE

        TYPE *, ' WHAT IS THE FIRST COLUMN YOU WISH TO DIGITIZE '
        READ(*,13),STARTCOL

        TYPE *, ' WHAT IS THE LAST COLUMN YOU WISH TO DIGITIZE '
        READ(*,13),LASTCOL

        LINES = LASTLINE - STARTLINE + 1
        COLS  = LASTCOL  - STARTCOL + 1
        LINES = (LINES/2) * 2        ! LINES MUST BE AN EVEN NUMBER
        COLS  = (COLS/2) *2          ! COLS MUST BE AN EVEN NUMBER
        LASTCOL = LASTCOL + 1
        LASTLINE = LASTLINE + 1
```

```
      ELSE

he image is centered in the lens

      TYPE *, ' ENTER THE NUMBER OF COLUMNS WIDE THE DIGITIZED '
      TYPE *, ' IMAGE WILL BE '
      READ(*,13),COLS

      TYPE *, 'ENTER THE NUMBER OF LINES DEEP THE DIGITIZED IMAGE WILL BE '
      READ(*,13),LINES

      LINES = (LINES/2) *2      ! LINES MUST BE EVEN
      COLS  = (COLS/2)  *2      ! COLS MUST BE EVEN
      STARTLINE = NDIODES/2 - LINES/2
      LASTLINE  = NDIODES/2 + LINES/2
      STARTCOL  = NDIODES/2 - COLS/2
      LASTCOL   = NDIODES/2 + COLS/2
      ENDIF

      type *, ' ENTER THE TITLE OF THE IMAGE OR A COMMENT UP TO 80 CHAR '
      type *, ' OR <CR> FOR NO COMMENT '
      read(*,15),comment
      format(A)

      WRITE(20) typepic, lines, cols, comment

      IF (TYPEPIC .EQ. 2) GOTO 50       ! TO DO A COLOR SCAN
```

```
C *******************************************************************
C THIS SECTION DOES A BLACK AND WHITE SCAN
C *******************************************************************

C Wait for the operator to push the Run switch.

        TYPE *, ' *** BEFORE YOU PUSH THE RUN SWITCH, INSURE THAT: '
        TYPE *, ' THE LIGHTS ARE ON '
        TYPE *, ' THE LENS CAP IS OFF '
        TYPE *, ' THE CAMERA IS IN FOCUS '
        TYPE *, ' THE CAMERA HEAD IS PULLED OUT '
        TYPE *, ' ** THE RUN SWITCH IS LABLED "SCAN" BELOW THE BUTTON '

        ISTAT(1) = IDREADY (ICHAN)                  ! sets digitizer to ready
        IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 510
C
C Do black and white scan.
C
        CALL IDSCFW(ICHAN,1,0,ISTAT)                 ! sets color filter to clear
        IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 560

        call IDSTIM(ICHAN, 1, BWTIME, ISTAT)        ! set integration time
        if (istat(1) .ne. ss$_normal) go to 540

        call IDSWIN(ICHAN, WINBIT, ISTAT)           ! sets winbit
        if (istat(1) .ne. ss$_normal) go to 550

        call IDSPOS(ICHAN, STARTLINE, ISTAT)        ! SET CAMERA POSITION
        IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 520
C
C
C The digitizer is now ready to start digitizing data. It reads 1 line of
C data and writes that line to the output file. It then goes on to the next
C line of data.
C
C
        type *, ' Digitizer is now starting digitizing, WAIT PLEASE '

        ISTEP = 1
        DO 99  I = 1,LINES-1,ISTEP

            call IDSCAB(ICHAN,BUFFER,NDIODES, ISTEP, ISTAT) ! digitize 1 line
            IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 530

C Now write to the output file

        WRITE (20)  (BUFFER(J), J= startcol,lastcol-1)
   99   CONTINUE

C Digitize the last line of data from the digitizer.

        CALL IDSCAB(ICHAN,BUFFER,NDIODES,0,ISTAT)
        IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 530

C Write last line to the output file

        WRITE (20)  (BUFFER(J), J= startcol,lastcol-1)

        go to 400
```

```
!**********************************************************************
    ! START OF COLOR SECTION
!**********************************************************************

!Open tempory color files

50    OPEN (UNIT= 30, FILE= 'COLOR.RED', ACCESS= 'SEQUENTIAL',
   1       FORM= 'UNFORMATTED', STATUS ='NEW', RECORDTYPE= 'VARIABLE' )

      OPEN (UNIT= 40, FILE= 'COLOR.GREEN', ACCESS= 'SEQUENTIAL',
   1       FORM= 'UNFORMATTED', STATUS ='NEW', RECORDTYPE= 'VARIABLE' )

      OPEN (UNIT= 50, FILE= 'COLOR.BLUE', ACCESS= 'SEQUENTIAL',
   1       FORM= 'UNFORMATTED', STATUS ='NEW', RECORDTYPE= 'VARIABLE' )

      TYPE *, ' *** BEFORE YOU PUSH THE RUN SWITCH, INSURE THAT: '
      TYPE *, ' 1. THE LIGHTS ARE ON '
      TYPE *, ' 2. THE LENS CAP IS OFF '
      TYPE *, ' 3. THE CAMERA IS IN FOCUS '
      TYPE *, ' 4. THE CAMERA HEAD IS PULLED OUT '
      TYPE *, ' ** THE RUN SWITCH IS LABLED "SCAN" BELOW THE BUTTON '

!Wait for the operator to push the Run switch.

      ISTAT(1) = IDREADY (ICHAN)                  ! ready digitizer
      IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 510


      type *, ' now scanning red, be patient '

!Do red scan.

      call IDSTIM(ICHAN,1,REDTIME,ISTAT)          ! set integration time
      if (istat(1) .ne. ss$_normaL) go to 540

      call IDSWIN(ICHAN,WINBIT,ISTAT)             ! set winbit
      if (istat(1) .ne. ss$_normaL) go to 550

      call IDSCFW(ICHAN,1,1,ISTAT)                ! set filter to red
      if (istat(1) .ne. ss$_normaL) go to 560

!Send the stage to start of field

      call IDSPOS(ICHAN,STARTLINE,ISTAT)          ! position camera
      IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 520


!The digitizer is now ready to start digitizing data.  It reads 1 line of
!data and writes that line to the output file.  It then goes on to the next
!line of data.


      ISTEP = 1
      DO 100  I = 1,LINES-1,ISTEP

!Read a line of data from the digitizer.

      call IDSCAB(ICHAN,BUFFERR,NDIODES,ISTEP,ISTAT)
      IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 530
```

```
C
C  Now write to the output file
C
          WRITE (30)  (BUFFERR(J), J= startcol,lastcol-1)
 100      CONTINUE
C
C Read the last line of data from the digitizer.
C
          CALL IDSCAB (ICHAN,BUFFERR,NDIODES,0,istat)
          IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 530
C
C Write last line to the output file
          WRITE (30)  (BUFFERR(J), J= startcol,lastcol-1)

C Do green scan.

          type *,' now doing green scan, be patient'

          CALL IDSTIM(ICHAN,1,GREENTIME,ISTAT)        ! set integration time
          if (istat(1) .ne. ss$_normal) go to 540

          CALL IDSWIN(ICHAN,WINBIT,ISTAT)             ! set winbit
          if (istat(1) .ne. ss$_normal) go to 550

          CALL IDSCFW(ICHAN,1,2,ISTAT)                ! set filter to green
          if (istat(1) .ne. ss$_normal) go to 560

C Send the stage to start of field

          CALL IDSPOS(ICHAN,STARTLINE,ISTAT)
          IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 520

C
C The digitizer is now ready to start digitizing data.  It reads 1 line of
C data and writes that line to the output file.  It then goes on to the next
C line of data.
C
C
          ISTEP = 1
          DO 200  I = 1,LINES-1,ISTEP
C
C Read a line of data from the digitizer.
C
          CALL IDSCAB (ICHAN,BUFFERG,NDIODES,ISTEP,ISTAT)
          IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 530
C
C  Now write to the output file
C
          WRITE (40)  (BUFFERG(J), J= startcol,lastcol-1)
 200      CONTINUE
C
C Read the last line of data from the digitizer.
C
          CALL IDSCAB (ICHAN,BUFFERG,NDIODES,0,ISTAT)
          IF (ISTAT(1) .NE. SS$_NORMAL)  GO TO 530
C
C Write last line to the output file
          WRITE (40)  (BUFFERG(J), J= startcol,lastcol-1)

C DO BLUE SCAN

          type *,' now doing blue, be patient '

          CALL IDSTIM(ICHAN,1,BLUETIME,ISTAT)                    ! set integration time
          if (istat(1) .ne. ss$_normal) go to 540
```

```fortran
        CALL IDSWIN(ICHAN,WINBIT,ISTAT)                    ! set winbit
        if (istat(1) .ne. ss$_normal) go to 550

        CALL IDSCFW(ICHAN,1,3,ISTAT)                       ! set filter to blue
        if (istat(1) .ne. ss$_normal) go to 560

!Send the stage to start of field

        CALL IDSPOS(ICHAN,STARTLINE,ISTAT)
        IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 520


!The digitizer is now ready to start digitizing data.  It reads 1 line of
!data and writes that line to the output file.  It then goes on to the next
!line of data.


        ISTEP = 1
        DO 300  I = 1,LINES-1,ISTEP

!Read a line of data from the digitizer.

        CALL IDSCAB (ICHAN,BUFFERB,NDIODES,ISTEP,ISTAT)
        IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 530

! Now write to the output file

        WRITE (50) (BUFFERB(J), J= startcol,lastcol-1)
300     CONTINUE

!Read the last line of data from the digitizer.

        CALL IDSCAB (ICHAN,BUFFERB,NDIODES,0,ISTAT)
        IF (ISTAT(1) .NE. SS$_NORMAL) GO TO 530

!Write last line to the output file

        WRITE (50) (BUFFERB(J), J= startcol,lastcol-1)

!Close the digitizer.

        CALL IDSCFW(ICHAN,1,0,ISTAT)                       ! reset filter to clear

        CALL IDCLOS (ICHAN)                                ! close digitizer channel


!Join color files together

        REWIND(UNIT=30)
        REWIND(UNIT=40)
        REWIND(UNIT=50)


        type *, 'now doing lots and lots of I/O, be patient '

        DO 250  A = 1,lines,1
        READ(30) (BUFFERR(J), J= 1,cols)
        READ(40) (BUFFERG(J), J= 1,cols)
        READ(50) (BUFFERB(J), J= 1,cols)
        WRITE(20) (BUFFERR(J), J= 1,cols)
        WRITE(20) (BUFFERG(J), J= 1,cols)
        WRITE(20) (BUFFERB(J), J= 1,cols)
```

```
250    Continue


C Close and delete tempory data output file

       CLOSE (UNIT=50, STATUS= 'DELETE')
       CLOSE (UNIT=40, STATUS= 'DELETE')
       CLOSE (UNIT=30, STATUS= 'DELETE')


400    CLOSE (UNIT=20)                    ! close output file
       STOP

.bp
C Error exits

500    TYPE 501,ISTAT(1)
501    FORMAT (' Open failure on IDA0:  --  Status =',Z9.4)
       STOP '*****   Program aborted   *****'

510    TYPE 511,ISTAT(1)
511    FORMAT (' Error while waiting for Run switch  --  Status =',Z9.4)
       STOP '*****   Program aborted   *****'

520    type 521,istat(1)
521    FORMAT (' Error while moving to line 0  --  Status =',Z9.4)
       STOP


530    TYPE 531, ISTAT(1)
531    FORMAT (' ERROR WHILE READING LINE',I5,' -- STATUS =',Z9.4)
       STOP '****  PROGRAM ABORTED *****'

540    TYPE 541, ISTAT(1)
541    FORMAT (' ERROR WHILE SETTING TIMING -- STATUS= ',Z9.4)
       STOP '**** PROGRAM ABORTED *****'

550    TYPE 551, ISTAT(1)
551    FORMAT (' ERROR WHILE SETTING WINBIT -- STATUS = ',Z9.4)
       STOP '**** PROGRAM ABORTED ****'

560    TYPE 561, ISTAT(1)
561    FORMAT (' COLOR FILTER FAILURE: -- STATUS = ',Z9.4)
       STOP '**** PROGRAM ABORTED ****'

       END
```

APPENDIX B
DISPLAY PROGRAM SOURCE CODE

The code on the following pages is the source code for the "display" program that displays digitized images on an IRIS 2400 system.

```
his is an IRIS-2400 program */
    /* This is file display.c.  Its purpose is to read
    through a file created by the EIKONIX Digitizer camera
    and produce an image on the IRIS...  */

Programmers

_T TOM WETHERALD USN
_T JEAN SANDO USN


lude "gl.h"
lude "device.h"
lude <stdio.h>
dithering macro */
ine dmap(v,col,row) (modN[v]> magic[col][row] ? divN[v] + 1 : divN[v])

(argc,argv)
argc;
**argv;


tatic RGBvalue buffer1[4096]; /* buffers to hold red          */
tatic RGBvalue buffer2[4096]; /*                    green and */
tatic RGBvalue buffer3[4096]; /*                    blue image data */

nt i,j;    /* temp loop counter */

hort lines, cols;      /* number of lines and cols in digitized file */
hort type;             /* type of file- b/w, color, dithered         */
hort mask = 255;       /* used   to tweak bits                       */
nt levels;             /* input, number of levels for dithering      */
nt row, col;           /* display loop variables                     */
nt divN[256];          /* used in dither function                    */
nt modN[256];          /* used in dither function                    */
nt magic[16][16];      /* just what it says                          */
ouble gamma;           /* gamma factor for dithering                 */

har comment[80];       /* comment stored in image file               */
har byt1[2], byt2[2], byt3[2]; /* used to input from image file      */
hort t;                /* output for dithered file, type of file     */
har ans = 'a';
tatic unsigned char buffer0[4096]; /* buffer for dithered file       */

nt fd,fd1;             /* file descriptor */
```

```c
    /* init the graphics system */
    ginit();

    /* put into rgb mode */
    RGBmode();

    /* configure the iris...*/
    gconfig();

    /* set the color to cyan */
    RGBcolor(0,255,255);

    /* turn cursor off */
    cursoff();

    /* draw a cyan background */
    rectfi(0,0,1023,768);

    /* open the file... */
    fd = open(argv[1],0);

    if(fd < 0)
    {
        printf("We couldnt open the file!\n");
        exit(1);
    }

    /* read the header information off the file */
    read(fd,byt1,2);
    read(fd,byt2,2);
    read(fd,byt3,2);
    read(fd,comment,80);

/* the following is required because VAX/VMS stores a two byte integer
   with the LSB first and MSB last, IRIS reads it as MSB first and
   LSB last
*/

    type = byt1[1] << 8;
    type = type | (mask & byt1[0]);
    lines = byt2[1] << 8;
    lines = lines | (mask & byt2[0]);
    cols = byt3[1] << 8;
    cols = cols | (mask & byt3[0]);
```

```c
case 1:                          /* display black and white images */
    /* read through the rows of the file... */

    for(i=0; i < lines; i=i+1)
    {
        /* read a row from the file */
        read(fd,buffer1,cols);

        /* draw that line */
        cmov2i(((1024/2) - (cols/2)),(((766/2) + (lines/2)) - i));
        writeRGB(cols,buffer1,buffer1,buffer1);

    }
    break;




case 2:                          /* display color image or dither a file */
    gamma = 0.0;
    while (ans != 'y' && ans !='Y' && ans !='n' && ans !='N')
    {
      printf("Do you want to store a dithered file? (y/n) \n");
      ans = getchar();
    }
    if ( ans == 'Y' || ans == 'y')
    {
      levels = 0;
      while (levels < 2 || levels > 6)
      {
       printf("enter the number of intensity levels(2-6)\n");
       scanf("%d", &levels);
      }
      while (gamma <= 0)
      {
        printf(" enter gamma correction factor, floating point number");
        printf(" value must be positive \n");
        scanf("%lf", &gamma);
      }
      fd1 = creat(argv[2],0666);
        if(fd1 < 0)
        {
          printf("We couldnt open the file!\n");
          exit(1);
        }
```

```c
      /* set buffer mode to single buffer */
      singlebuffer();

      /* do a graphic init */
      ginit();

   /* write out header information to new dithered file */

   t = 768;    /* this is the file type, will be a 3, 768 used to
                  be consistend with VAX/VMS files ie:
                  00000011 00000000 is 768 on IRIS but 3 on VAX/VMS */

   write(fd1,&t,2);            /* write type to file */
   write(fd1,byt2,2);          /* write number of lines to file */
   write(fd1,byt3,2);          /* write number of columns to file */
   write(fd1,comment,80);      /* write comment to file */
   write(fd1,&gamma,4);        /* write gamma factor to file */
   write(fd1,&levels,4);       /* write number of levels to file */


      /* create the dithered color map and magic matrix */
      dithermap(&levels,&gamma,&divN,modN,magic);
      for(i=0; i < lines; i=i+1)
      {
        /* read a row from the file */
        read(fd,buffer1,cols);
        read(fd,buffer2,cols);
        read(fd,buffer3,cols);
        for (j = 0; j < cols; ++j)
        {

        col = (i % 16);
        row = (j % 16);
        /* dither each pixel and store an index into the dithered
           color map for each pixel    */
          buffer0[j ]= (dmap(buffer1[j],col,row) +
                        (dmap(buffer2[j],col,row) * levels) +
                        (dmap(buffer3[j],col,row) * levels * levels ));
        }

       /* write each line buffer out to the file */
            write(fd1,buffer0,cols);
            if ( (i+1) %50 == 0)
            printf("printed line %d to file \n",i+1);
      }
      printf(" to display the file, run display again with the");
      printf(" file name of the dithered file \n");
    }
    else                    /* display a full color RGB image */
    {

      for(i=0; i < lines; i=i+1)
      {
        /* read a row from the file */
        read(fd,buffer1,cols);
        read(fd,buffer2,cols);
        read(fd,buffer3,cols);

         /* draw that line */
         cmov2i(((1023/2) -(cols/2)),(((765/2) + (lines/2)) -i));
         writeRGB(cols,buffer1,buffer2,buffer3);
      }
    }
    break;
```

```c
    case 3:                       /* display a dithered image */
        singlebuffer();
        ginit();
        cursoff();
        color(4);                 /* change background color, dithermap changes
                                     system defined colors */
        clear();
        read(fd,&gamma,4);
        read(fd,&levels,4);
          /* create dithered color map and magic matrix */
          dithermap(&levels,&gamma,divN,modN,magic);

          for (i = 0; i < lines; ++i)
          {
            /* read each line from the file into a buffer */
            read(fd,buffer0,cols);
            for (j = 0; j < cols; ++j)
            {
              /* set system color to pixel index value */
              color(buffer0[j]);
              /* draw each pixel */
              pnt2s(((1024/2)-(cols/2) + j),((766/2) + (lines/2))-i);
            }
          }
        break;

    default:
        printf("Picture type was something other than BW,");
        printf("Color or Colormapped.\n");
    }

close(fd);   /* close the file */

/* print comment at base of picture on screen */
RGBcolor(0,0,0);
cmov2i(((1024/2) - 80), 2);
/* ((766/2) - (lines/2))); */
charstr(comment);

/* wait until the mouse is hit */
while(getbutton(MOUSE1) == 0);

/* reset the iris for the next guy */
color(0);
clear();
ginit();

/* exit graphics */
gexit();


include <math.h>

magic4x4[4][4] = {
     0, 14,  3, 13,
    11,  5,  8,  6,
    12,  2, 15,  1,
     7,  9,  4, 10
```

```c
/***************************************************************
 * TAG( dithermap )
 *
 * Create a color dithering map with a specified number of intensity levels.
 * Inputs:
 *      levels:         Intensity levels per primary.
 *      gamma:          Display gamma value.
 * Outputs:
 *      rgbmap:         Generated color map.
 *      divN:           "div" function for dithering.
 *      modN:           "mod" function for dithering.
 * Assumptions:
 *      rgbmap will hold levels^3 entries.
 * Algorithm:
 *      Compute gamma compensation map.
 *      N = 255.0 / (levels - 1) is number of pixel values per level.
 *      Compute rgbmap with red ramping fastest, green slower, and blue
 *      slowest (treat it as if it were rgbmap[levels][levels][levels][3]).
 *      Call make_square to get divN, modN, and magic
 *
 * Note:
 *      Call dithergb( x, y, r, g, b, levels, divN, modN, magic ) to get index
 *      into rgbmap for a given color/location pair, or use
 *          row = y % 16; col = x % 16;
 *          DMAP(v,col,row) =def (divN[v] + (modN[v]>magic[col][row] ? 1 : 0))
 *          DMAP(r,col,row) + DMAP(g,col,row)*levels + DMAP(b,col,row)*levels^2
 *      if you don't want function call overhead.
 */
dithermap( levels, gamma, divN, modN, magic )
int     *levels;
double *gamma;
int divN[256];
int modN[256];
int magic[16][16];
{
    double N;
    register int i;
    int levelsq, levelsc;
    int gammamap[256];

    for ( i = 0; i < 256; i++ )
{
        gammamap[i] = (int)(0.5 + 255 * pow( i / 255.0, 1.0/ *gamma));
}
    levelsq = (*levels) * (*levels);    /* squared */
    levelsc = (*levels) * levelsq; /* and cubed */

    N = 255.0 / (*levels - 1);    /* Get size of each step */

    /* Set up the  IRIS color map entries */
    for(i = 1; i < levelsc; i++)

        mapcolor(i,gammamap[(int)(0.5 + (i% *levels) * N)],
                    gammamap[(int)(0.5 + ((i/ *levels)% *levels) * N)],
                    gammamap[(int)(0.5 + ((i/ levelsq)% *levels) * N)]);


    make_square( &N, divN, modN, magic );

}
```

```
/*********************************************************************
AG( make_square )

uild the magic square for a given number of levels.
nputs:
    N:              Pixel values per level (255.0 / levels).
utputs:
    divN:           Integer value of pixval / N
    modN:           Integer remainder between pixval and divN[pixval]*N
    magic:          Magic square for dithering to N sublevels.
ssumptions:

lgorithm:
    divN[pixval] = (int)(pixval / N) maps pixval to its appropriate level.
    modN[pixval] = pixval - (int)(N * divN[pixval]) maps pixval to
    its sublevel, and is used in the dithering computation.
    The magic square is computed as the (modified) outer product of
    a 4x4 magic square with itself.
    magic[4*k + i][4*l + j] = (magic4x4[i][j] + magic4x4[k][l]/16.0)
    multiplied by an appropriate factor to get the correct dithering
    range.
```

```c
square( N, divN, modN, magic )
e *N;
ivN[256];
odN[256];
agic[16][16] ;

egister int i, j, k, l;
ouble magicfact;

or ( i = 0; i < 256; i++ )

    divN[i] = (int)(i / *N);
    modN[i] = i - (int)(*N * divN[i]);


/*
 * Expand 4x4 dither pattern to 16x16.  4x4 leaves obvious patterning,
 * and doesn't give us full intensity range (only 17 sublevels,
 * we want at least 51).  Note that 8x8 would be sufficient, but
 * 16x16 is easier to build.
 *
 * magicfact is (N - 2)/16 so that we get numbers in the matrix from 0 to
 * N - 2: mod N gives numbers in 0 to N - 1, we want some chance that the
 * number is greater than a matrix entry, so the highest matrix
 * entry must be N - 2.
 */
agicfact = (*N - 2) / 16.;
or ( i = 0; i < 4; i++ )
    for ( j = 0; j < 4; j++ )
        for ( k = 0; k < 4; k++ )
            for ( l = 0; l < 4; l++ )
                magic[4*k+i][4*l+j] =
                    (int)(0.5 + magic4x4[i][j] * magicfact +
                          (magic4x4[k][l] / 16.) * magicfact);
```

APPENDIX C
MAKEFILE FOR DISPLAY PROGRAM

The following is the Makefile used to compile the display.c program and generate the executable version of the display program.

```
cc -c display display.c -Zg -Zf
```

Distribution List for Dr. Michael J. Zyda

Defense Technical Information Center
Cameron Station
Alexandria, VA 22314   2 copies

Library, Code 0142
Naval Postgraduate School
Monterey, CA 93943   2 copies

Center for Naval Analyses
2000 N. Beauregard Street
Alexandria, VA 22311   1 copy

Director of Research Administration
Code 012
Naval Postgraduate School
Monterey, CA 93943   1 copy

Dr. Michael J. Zyda
Naval Postgraduate School
Code 52, Dept. of Computer Science
Monterey, California 93943-5100   150 copies

Mr. Russell Davis
HQ, USACDEC
Attention: ATEC-IM
Fort Ord, California 93941   1 copy

John Maynard
Naval Ocean Systems Center
Code 402
San Diego, California 92152   1 copy

El Wells
Naval Ocean Systems Center
Code 443
San Diego, California 92152   1 copy

Roger Casey
Naval Ocean Systems Center
Code 84
San Diego, California 92152   1 copy

Dr. Al Zied
Naval Ocean Systems Center
Code 433
San Diego, California 92152   1 copy

Dr. Egbert D. Maynard
OUSDR&E VHSIC Program Office
Room 3D-139, 400 A/N
The Pentagon
Washington, DC 20301-3060   **1 copy**

U236874