1984

Digital Image Processing Using NTEC Facilities

James F. Roesch
University of Central Florida

DIGITAL IMAGE PROCESSING
USING NTEC FACILITIES

BY

JAMES FREDERICK ROESCH, JR.
B.S.E., University of Central Florida, 1982

RESEARCH REPORT

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in the Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Summer Term
1984

ABSTRACT

Digital image enhancement refers to the improvement of a given image for human interpretation. Digital image processing facilities are those in which hardware and software computing elements are combined in such a way as to enable the processing of digital images. This report describes the use of the Naval Training Equipment Center (NTEC) Computer Systems Laboratory computing facilities to enhance digital images. Described are two major hardware systems, the IKONAS RDS-3000 raster display graphics system and the VAX-11/780, and the digital image processing program (DIMPRP) written by the author. Digital image enhancement theory and practice are addressed through a discussion of the DIMPRP software. Finally, enhancements to the NTEC digital image processing facility such as improvements in hardware reliability, documentation, and increased speed of program execution are discussed.

ACKNOWLEDGEMENTS

The author would like to express his appreciation to the Naval Training Equipment Center (NTEC) Computer Systems Laboratory for the use of their equipment. The author also wishes to express his gratitude to John L. Booker and Code N-74 for funding this research endeavor under Navy contract N61339-80-D-0014.

Special thanks go to Dr. Fred O. Simons who has been a great professor, advisor, and principal investigator under the Navy contract. He has provided encouragement and has been a good friend. Dr. Madjid Belkerdid has also been more than a professor and committee member; he has been a good friend as well.

The author would like to thank Mercedes Wechsler Jones for her assistance in preparing this report. Mercedes was kind enough to use her expertise on the Applicon CAD/CAM and the Apple Lisa personal computer systems to prepare all the drawn figures that appear in this report. Mark Layton has made many helpful suggestions, along the course of the writing of this report, which have been appreciated by the author.

The author would especially like to thank his wife, Helene, for her editorial assistance and her continual love throughout graduate school.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Digital Images

Throughout this report, the term monochrome image or simply image, refers to a two-dimensional light intensity function i(x,y), where x and y denote spatial coordinates and the value of i at any point (x,y) is proportional to the brightness (or gray level intensity) of the image at that point. It is sometimes useful to view an image function in perspective with the third axis being brightness. An image viewed in this way appears as a series of active peaks in regions with numerous changes in brightness levels and smoother regions or plateaus where the brightness levels are constant or vary little. If assigning proportionally higher values to brighter areas, the height of the components in the plot would be proportional to the corresponding brightness in the image.

A digital image is an image i(x,y) which has been discretized and quantized in both spatial coordinates and in brightness. A digital image may be considered as a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The members of such an array are called "image elements", "picture elements", or "pixels", where pixels is the commonly used abbreviation of "picture elements".

The array size and number of gray levels of a digital image may vary with the application. However, to facilitate processing of the image by a digital computer, there are obvious advantages in selecting square arrays with sizes and number of gray levels which are integer powers of 2. Throughout the remaining chapters of this report the image array size will be 512x512 pixels, with each pixel value having 256 gray levels.

With the exception of a discussion in Chapter 3 of pseudocolor techniques for image enhancement, all the images considered in this report are digital monochrome (black and white) images of the form described above.

Digital Image Processing

Techniques for image processing may be divided into four principal categories: (1) image digitization; (2) image enhancement and restoration; (3) image encoding; and (4) image segmentation and representation (Gonzalez and Wintz 1977). With the exception of a short discussion in Chapter 2 of the IKONAS RDS-3000 video input module (image digitization), this report considers only the first topic of category 2; i.e. image enhancement.

Digital image enhancement techniques deal with the improvement of a given image for human interpretation. One of the first applications of image processing techniques in this category was to improve digitized newspaper pictures sent by submarine cable between London and New York in 1929. Some of the initial problems in improving the visual quality of these digital pictures were related to the selection of printing procedures and distribution of brightness levels.

Although improvements on processing methods for transmitted digital pictures continued to be made over the next thirty-five years, it took the combined events of the space program and the development of large-scale digital computers to bring into focus the potentials of image processing concepts. Work on using computer techniques for improving images from a space probe began at the Jet Propulsion Laboratory (Pasadena, California) in 1964, when pictures of the Moon transmitted by Ranger 7 were processed by a computer to correct various types of image distortion inherent in the spacecraft's on-board television camera. These techniques served as the basis for improved methods used in the enhancement and restoration of images from such familiar programs as the Surveyor missions to the Moon, the Mariner series of flyby missions to Mars, and the Apollo manned flights to the Moon (Gonzalez and Wintz 1977).

From 1964 until the present, the field of image processing has experienced vigorous growth. In addition to applications in the space program, digital image processing techniques are used today in a variety of problems which, although often unrelated, share a common need for methods capable of enhancing pictorial information for human interpretation and analysis. In medicine, for instance, physicians are assisted by computer procedures that enhance the contrast or code the intensity levels into color for easier interpretation of x-rays and other biomedical images. Similar techniques are used by geographers in studying pollution patterns from aerial and satellite imagery. Image enhancement and restoration procedures have been used to process degraded images depicting unrecoverable objects or experimental results too expensive to duplicate. There have been instances in archeology, for example, where blurred pictures were the only available records of rare artifacts lost or damaged after being photographed, have been successfully restored by image processing methods. In physics and related fields, images of experiments in such areas as high-energy plasmas and electron microscopy are routinely enhanced by computer techniques. Similar successful applications of image processing concepts can be found in astronomy, biology, nuclear medicine, law enforcement, defense, and industrial applications.

Digital Image Processing Facilities

The components of a basic, general-purpose digital image processing system are shown in Figure 1 (Gonzalez and Wintz). The operation of such a system may be divided into three principal categories: digitization, processing, and display.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. The most commonly used input devices are microdensitometers, flying spot scanners, image disectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency or photograph.

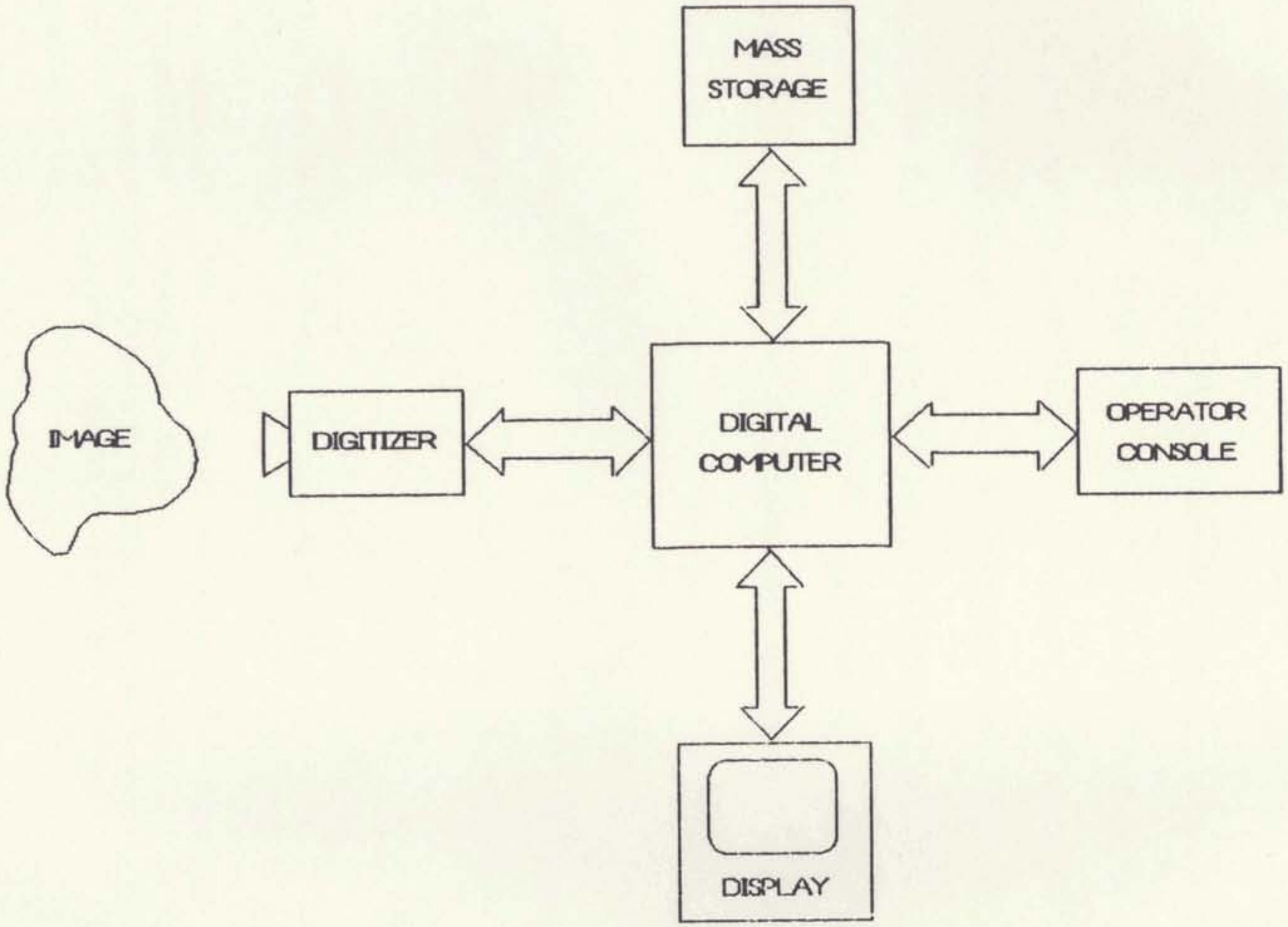

Figure 1. Elements of a Digital Image Processing System

(Gonzalez and Wintz, 1977; redrawn by Mercedes)

Image disectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity to excite the detector.

Systems used for image processing range from microprocessor devices for special-purpose applications to large computer systems capable of performing a variety of functions on high-resolution image arrays. The principal parameter influencing the structure of a computer system for image processing is the required data throughput. For general-purpose laboratory applications where fast data throughput is not essential, a moderately equipped minicomputer is often adequate. Other important parameters in determining a suitable image processing system include the number of users of the system and the intended use of the system. In structuring an image processing facility for the University of Southern California, Andrews (1977) states:

> The general design philosophy behind the USC Image Processing Institute is predicated on the need to service many users at one time. It must provide rapid visual access to processed pictorial results and handle large data arrays while simultaneously providing mass storage for easily accessible intermediate processed images. In addition, because of the hectic pace of the faculty, staff, and student life, it must provide high-quality results for a minimal expenditure of personal time.

In terms of the above requirements, a minimum, yet flexible, computer system for image processing can be structured using a minicomputer with 64,000 words or more of core memory, two disk drives, a magnetic tape unit, and assorted peripherals such as scope editing terminals and a lineprinter or some other hard-copy device.

Image processing programs are often coded in assembly language to gain speed, but for sake of flexibility and readability, programs coded in a high-level language may be a wise alternative.

The function of the display unit in an image processing system is to convert the numerical arrays stored in the computer into a form suitable for human interpretation. The principal display media are Cathode Ray Tubes (CRTs), TV systems, and printing devices.

Television display systems convert an image stored in the computer into a video frame which can be displayed on a TV monitor. The advantage of these systems is that displays created on a video monitor have a tonality which closely resembles that of photographs, thus producing an output which is easily assimilated by the visual system. The disadvantage of TV displays is that they must refresh the monitor at a rate of about 30 frames (e.g. images) per second in order to avoid flicker. Since most general-purpose computers are not capable of transferring data at this rate, the principal problem in designing a TV display system is to provide some buffer storage medium for transferring data to a monitor at video rates. The most common solution to this problem, and the solution used at the NTEC facility, is one of using a fast solid-state memory to store the entire image array; the screen is then refreshed at 30 frames per second by cycling through the memory and combining the stored binary information into an analog signal by means of conditioning circuits and fast digital-to-analog converters.

CHAPTER 2

A DIGITAL IMAGE PROCESSING FACILITY

Hardware

In any computing system the two most primary components of the system are the system hardware and the system software. This section of Chapter 2 will be devoted to describing the two major hardware systems at the Naval Training Equipment Center (NTEC) Computer Systems Laboratory, which are used in structuring a system suitable for digital image processing.

The IKONAS RDS-3000 Raster Display Graphics System

The IKONAS Raster Display System 3000 (RDS-3000) is, as the title indicates, a raster display computer graphics system capable of displaying images at rates up to 60 times a second. The IKONAS is a very powerful system which has features and capabilities that out perform many other raster display computer graphics systems. Only those features and capabilities that are directly related to the use of the IKONAS in a digital image processing facility will be detailed in this report. These features, which may be described as IKONAS subsystems, are the IKONAS VI8 video input module, the IKONAS DR64 image memory, the IKONAS LUVO module, and the IKONAS IF/IK host interface. A block diagram of the entire IKONAS system architecture showing each subsystem and the host computer is given in Figure 2.

NTEC CODE N-74 COMPUTER SYSTEM LABORATORY

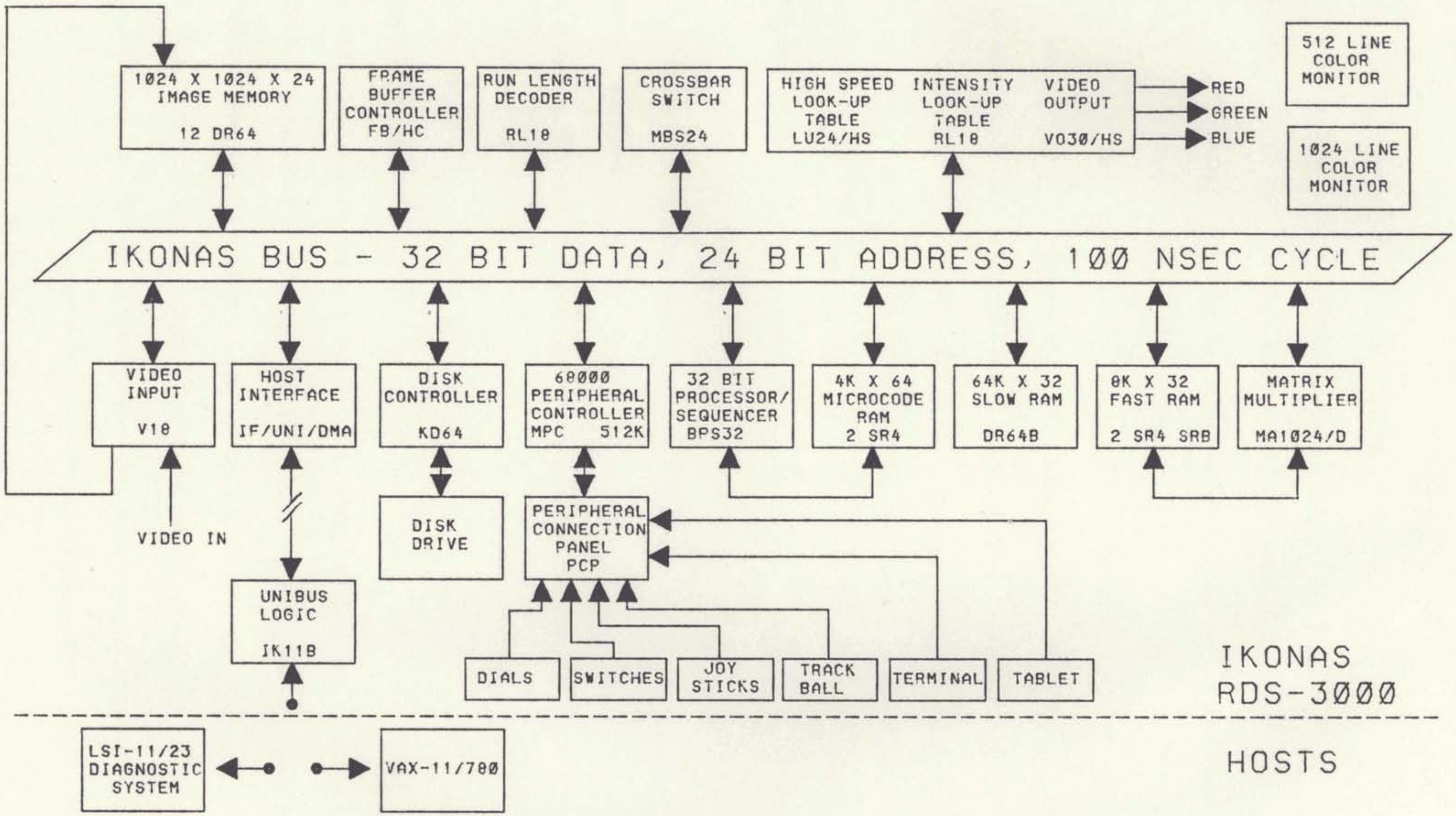

Figure 2. Digital Image Processing Facility Architecture

IKONAS RDS-3000 VI8 Video Input Module

The IKONAS VI8 video input module serves in the digital image processing facility as the digitizer. As stated previously in this report, a digitizer converts an image into a numerical representation suitable for input into a digital computer.

The VI8 video input module accepts information in the form of a continuous voltage from an RS-170A video camera. The RS-170A is an imposed standard by the National Television Standards Committee (NTSC) on the voltage waveform output, specifically the synchronization timing, of a vidicon tube TV camera. The operation of a vidicon tube is based on the principle of photoconductivity. An image focused on the tube surface produces a pattern of varying conductivity which matches the distribution of brightness in the optical image. An independent, finely focused electron beam scans the rear surface of the photoconductive target, and by charge neutralization this beam creates a potential difference and produces on a collector a signal proportional to the input brightness pattern. A digital image is obtained by quantizing this signal, as well as the corresponding position of the scanning beam. The VI8 video input module can sample at a rate from 9 to 11 MHz, and writes the data to the DR64 framebuffer memory (Adage 1982). The sampling frequency, the size of the sampled frame, the location of the sampled frame within the input scene, and the location of the frame within the output framebuffer may be programmed. Sampling may be continuous or may be set to stop after one frame has been digitized.

Also the input scene may be compressed (minified) by an integer divisor to allow storing a scene into a small section of the framebuffer.

IKONAS RDS-3000 DR64 Image Memory

As Adage (1982) states, "The heart of the IKONAS system is the framebuffer memory." The IKONAS framebuffer is a large core of solid state memory capable of storing more than one image. In pixel mode, all DR64's can be viewed as one continuous memory that is addressable in x and y. Each x,y memory location corresponds to a pixel. The bit depth of each memory location (the number of bits which contribute to each pixel) depends on the system configuration.

For the purposes of digital image processing the low resolution (LORES) mode is the most appropriate for two reasons: (1) It is impossible to digitize an image in high resolution (HIRES) mode using the VI8 video input module; (2) The computational complexity of the processing operations in HIRES mode would be greater then that of LORES mode by a factor of four. In LORES mode, each DR64 can provide 262,144 pixels of eight bits each, corresponding to a 512x512 display. Depending on the number of DR64's in the system configuration, the bit depth may be 8, 16, 24, or 32 bits. The x addresses range from 0 to 511. The y address range depends on the number of DR64's and the bit depth. In the digital image processing facility described in this report, only image zero is used. For monochrome images, only the first eight bits of image zero are used; i.e., image zero, red. Given in Table 1 are the standard DR64 configurations for the NTEC IKONAS.

Table 1. Standard DR64 Configurations

CARD ADDRESS	X range	LORES Y range	Bit range	Image #
10	0-511	0-511	0-7	0
11	0-511	0-511	8-15	0
12	0-511	0-511	16-23	0
13	0-511	0-511	24-31	0
14	0-511	512-1023	0-7	1
15	0-511	512-1023	8-15	1
16	0-511	512-1023	16-23	1
17	0-511	512-1023	24-31	1
01	0-511	1024-1535	0-7	2
02	0-511	1024-1535	8-15	2
03	0-511	1024-1535	16-23	2
04	0-511	1024-1535	24-31	2

IKONAS RDS-3000 LUVO Module

The IKONAS LUVO (lookup and video output) functions as the conditioning circuits and fast digital-to-analog converters that transform the binary information to an analog signal. The LUVO accepts data on the 72-pin ribbon cable at the top. Video data comes from whatever is the previous element in the video pipeline, usually the frame buffer controller (FBC) or the 34-bit crossbar switch. The cable allows for 32 bits of video data and two extra "page" bits. In systems with an crossbar switch (XBS), the assignment of memory bits to video data bits is governed by the crossbar. In any case, the 32 video data bits are divided into four eight-bit channels: bits 0-7 (red), 8-15 (green), 16-23 (blue), and 24-31 (overlay). The red, green, and blue channels are referred to as background channels.

The three eight-bit background channels are the inputs to a channel crossbar switch. This crossbar switch has three output channels; each output channel can be taken from any of the three input channels under software control. For full-color operation the output red channel is taken from the input red channel, the output green channel is taken from the input green channel, and the output blue channel is taken from the input blue channel. Thus the channel crossbar switch is transparent and the output data equals the input data. For pseudocolor operation all of the output channels are taken from one of the input channels. Thus, all three lookup tables receive the same data and the resultant image is monochrome. Since the least significant byte, bits 0-7, comprise the red channel, it is convenient to take all the output channels from the red input channel to form a black and white image. Using this scheme, monochrome image processing is accomplished by digitally processing only bits 0-7 of an image.

There are three lookup tables corresponding to the three output channels of the channel crossbar. The lookup memory must have been initialized from the IKONAS bus for meaningful operation. The table lookup memory is normally loaded such that a value of 0 corresponds to black, and a value of 255 corresponds to white, with values between 0 and 255 corresponding to the appropriate gray levels. The data present at the output of a lookup table will be sent to the D/A (digital-to-analog) converters. The D/A converter converts the data to an analog voltage between -1 volt and 0 volts with proper video sync and blanking information added.

VAX/IK Interface

The VAX/IK interface serves as the data communication link between the IKONAS and the Digital Equipment Corporation VAX-11/780. The IKONAS interface to DEC computers consists of two cards connected by two ribbon cables. One card, the IK11B, is plugged into the VAX UNIBUS; the other card, the IKONAS IF/DMA, plugs into the IKONAS frame.

The IK11B interface requires one quad size board slot in an SPC-compatible backplane. This card contains direct memory address (DMA) logic and control for data transfers to and from the IKONAS graphics system. In addition, logic is included for programmed control of IKONAS bus addressing and operation.

The IKONAS IF/DMA consists of one card which plugs into the IKONAS bus to provide interfacing and control of the IKONAS graphics system. A microprogrammed sequencer insures proper timing and protocol for both the UNIBUS and IKONAS systems (Adage 1982).

The VAX-11/780 Minicomputer

The VAX-11/780 is a superminicomputer with a 32-bit architecture that supports over 300 instructions. The VAX-11/780 is designed to meet the needs of many users with large databases and extensive processing needs. Central to its I/O system is a 32-bit wide data and control path that can move up to 13.3 Megabytes of data per second among the systems major hardware components. Up to four UNIBUS and four MASSBUS adaptors may be used for connection to mass storage devices and other peripherals. The support of high-performance disks and tapes by the VAX-11/780, combined with their ability to network with the other VAX family members provides varied configuration possibilities. These features ideally suit the VAX-11/780 for use as the primary computing element in a digital image processing facility. The hardware architecture of the VAX-11/780 showing the central processing unit and various subsystems is given in Figure 3.

VAX Virtual Memory Archtecture

The letters VAX suggest the primary feature of the VAX computers; "Virtual Address eXtension". In a VAX computer, bytes of information are located with a 32-bit address. This means that the computer has an address space of over 4 billion bytes. This large address space is "virtual" in that the main (physical) memory of the computer need not be anywhere near as large as four billion bytes for the machine to process data whose addresses are scattered throughout the address space. The virtual addressing process is accomplished through a scheme called "memory management" which allows programmers to operate as if a big part of the virtual address space were really available to them. Memory management handles all the details of storing programs and subsequently bringing them into the main memory when they are processed. Thus, the bottom two billion bytes of virtual address space can be used for programs without concern about complicated techniques of overlaying or segmenting to squeeze the program into a smaller address range.

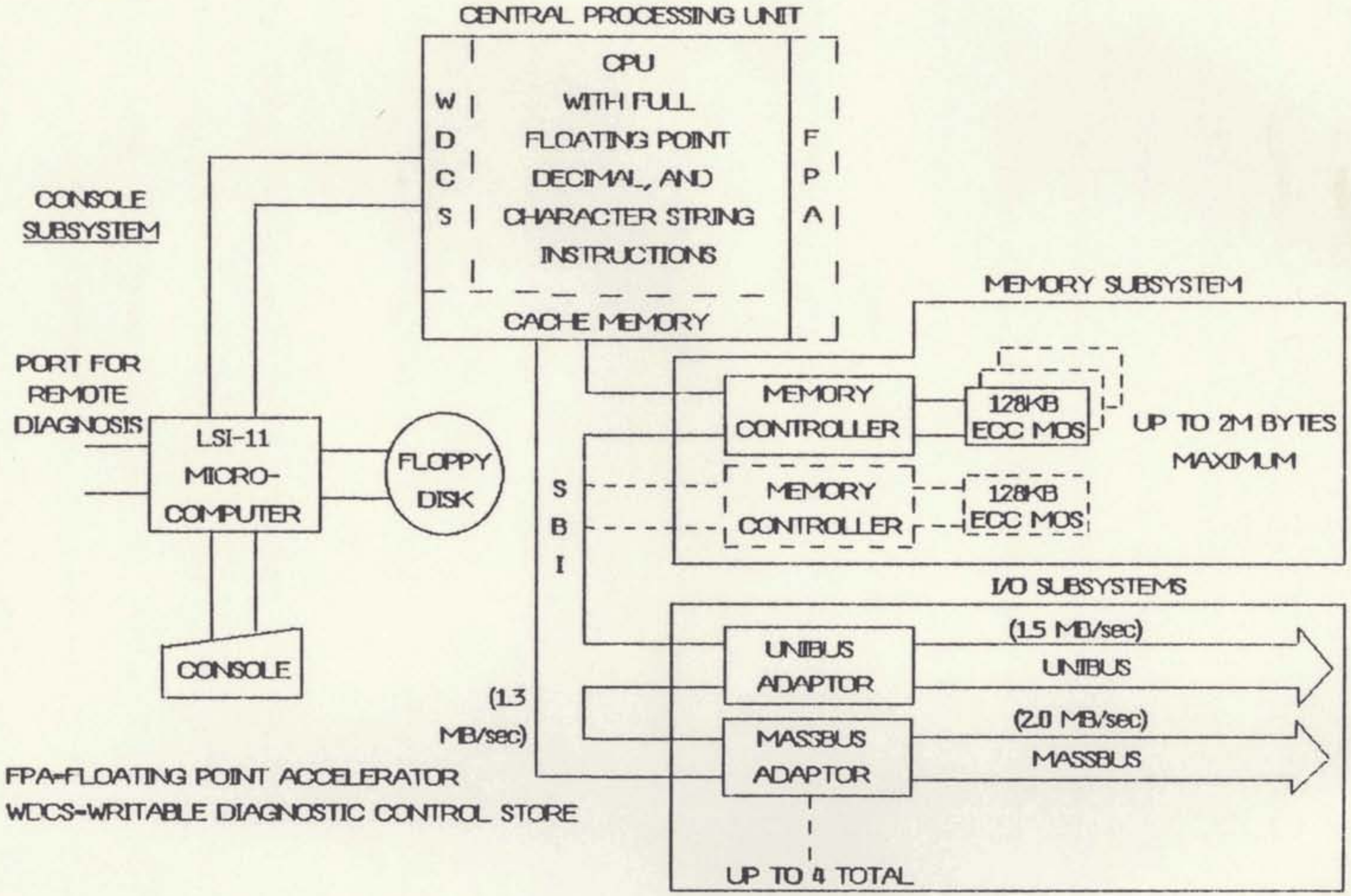

Figure 3. VAX-11/780 Hardware Architecture

(DEC, 1981; redrawn by Mercedes)

The memory requirement for processing digital images is very large. To process a monochrome 8-bit image with an array size of 512x512 pixels requires a memory size of 256 Kilobytes. To pseudocolor enhance an image of the same characteristics as described above requires a memory size of over 4 million bytes of memory. It is obvious that the large memory requirement for processing digital images dictates either a very large and expensive core memory or some compromise between the amount of core memory and secondary memory, with secondary memory usually taking the form of a hard medium such as a disk drive. From an economical point of view, it is less expensive to structure a system with a large secondary memory and a small main memory. From a performance point of view, it is more optimal with respect to speed of execution to choose a very large main memory and a small secondary memory. The advantage of the VAX virtual memory architecture is that through the use of highly efficient memory management algoritms, an economical blend of main memory and disk memory may be chosen such that performance is not compromised.

The VAX hardware architecture design enables further performance enhancements such as engineering into hardware some recurrent operations from high-level languages so that a single instruction can handle them. The FORTRAN DO loop and the three-operand addition (A=B+C) are examples of operations that can be handled by a single VAX instruction (DEC 1981). Also, there is no forced alignment on longword boundaries as required by many languages. Data items bigger than a byte can still reside on any byte boundary.

Mass Memory Storage Devices

In a digital image processing facility that utilizes a computer with a virtual memory architecture such as the VAX-11/780, high performance mass memory storage devices are essential. The main memory of most computers is constructed from high speed solid state MOS technology. The secondary or mass memory of the computer system is usually made up of hard medium devices such as disk drives. The VAX computer system must page between the main memory and the secondary memory, with a page fault generated when the address of a data item requested by a program does not reside in the main memory. When a page fault is generated the memory management unit must "fetch" the requested data item from the secondary memory. Thus, to obtain optimum performance from the computer system, page faulting must be minimized, and the rate at which data is transferred from the mass memory devices to the main memory must be very fast.

With the VAX computer system high-performance mass storage devices, such as the RP series and RM moving head disks, are connected to the VAX-11/780 system using a MASSBUS adaptor. The MASSBUS adaptor is the interface between the MASSBUS and the SBI (see Figure 3) and performs all control, arbitration, and buffering functions. There may be a total of four MASSBUS adaptors on each VAX-11/780 system. The VAX/VMS operating system supports transfers of 65KBytes to and from noncontiguous pages in physical memory. Each MASSBUS adaptor uses a 32-byte silo data buffer, which permits transfers at rates up to 2 million bytes/second to and from physical memory. The combination of UNIBUS and MASSBUS transfer rates gives a maximum throughput of 9.5 bytes/second to and from the SBI (DEC 1981).

Since one monochrome digital image of the type detailed in this report requires 256Kbytes of memory for storage, another desired attribute of disk drives used in an image processing facility is a large data capacity. It is often necessary to store an image once it has been digitized so that it is easily accesible at a later date. Since a user may have the need to digitize and save several images, the capability of storing multiple images on a single disk drive is advantageous. The VAX-11/780 system at NTEC is equipped with two RM05 moveable head disk drives, with the capacity of each disk drive being 300MBytes of memory. Thus, there is ample data storage capacity on a single RM05 disk drive to store multiple images.

Software

As stated previously, the two most primary components of a computing system are the system hardware and the system software. This section of Chapter 2 describes the major software components used to structure the NTEC digital image processing facility.

VAX/VMS Operating System

The VAX/VMS operating system is the operating system chosen by NTEC to run on the VAX-11/780. The VAX/VMS operating system is designed to make all the hardware work together as one unit to provide the VAX-11/780 with its multi-user, multiprogramming, virtual memory capabilities.

Because of the VAX-11/780 virtual memory architecture programs do not have to reside in main memory at one time. This means that portions of the program can be on the system disk and other portions in main memory. Programs are divided into small pieces called "pages", which are 512 bytes long. A "process" is a collection of pages which runs a program. A process consists of an address space plus both hardware and software context. That part of a process which is resident in main memory is called the process's "working set". At any given point in time, there are many processes running on the system. The assemblage of processes which are resident in main memory is called the "balance set". The action of bringing pages into and out of main memory is called "paging". The action of bringing complete working sets into and out of main memory is called "swapping".

In order to control the simultaneous processing of many large programs, the VAX-11/780 incorporates sophisticated virtual memory management capabilities. The VAX-11 memory management system is a tightly coupled hardware/software function. The hardware performs the task of translating from virtual addresses into physical addresses. The VAX/VMS operating system provides the capabilities for paging, swapping, overlaying, protection and sharing.

<u>FORTRAN Subroutines IKPRD,IKPWR,IKBWR</u>

For easy communication between the IKONAS and the VAX-11/780 several input/ouput routines have been developed by ADAGE in the high level language FORTRAN. Within a FORTRAN program, the IKPRD and

IKPWR routines allow the programmer to easily read data from, or write data to the IKONAS framebuffer memory by merely calling the proper FORTRAN subroutine. The IKBWR subroutine allows the programmer to send data values to the IKONAS bus within a FORTRAN program. The standard protocol for each of these I/O routines is given below (ADAGE 1982).

Pixel I/O:

1. IKPRD (code, x addr, y addr, length, data, IKONAS status,
   system status)

   code = IKONAS function code.
   Default = 2;
   pixel-mode bit (bit 1) will be set;
   write bit (bit 4) will be cleared.

   x addr = 10-bit x-value

   y addr = 14-bit y-value

   length = number of 32-bit IKONAS words to read.
   One word of whatever size is the default on
   the user's machine (to allow this argument
   to be specified as a constant)

   data = IKONAS data, sequential 32-bit variables.

This entry point starts to read, then returns to caller.
The status arguments are optional. Errors detected by the IKONAS system during a transfer are reported in the IKONAS status argument. Errors detected by the Host operating system are reported in the system status argument. IKONAS status of 0 indicates normal completion; system status is defined by the operating system. If an error occurs, and the corresponding status argument is not given, the user's program is terminated.

2. IKPWR (code, x addr, y addr, length, data, IKONAS status,
   system status)

   code = IKONAS function code.
   Default = 22 octal;
   the pixel-mode bit (bit 1) and
   the write bit (bit 4) will be set.

   (address, length, data, status, as for IKPRD)

Word I/O:

Functions to perform word I/O differ from pixel I/O only in the method of addressing. The address is specified as a 24-bit number instead of X and Y addresses.

1. IKBWR (code, address, length, data, IKONAS status, system status)

code = IKONAS function code.
Default = 0;
the write bit (bit 4) will be cleared.

address = 24-bit address

Introduction to DIMPRP

The software elements of the NTEC facility previously described in this report were present at that facility before the writing of this report. What was needed to complete the structuring of a digital image processing facility was software that would enable a user to implement many common digital image processing functions. This final step in constructing a digital image processing facility at NTEC has been completed. A general Digital Image Processing Program (DIMPRP) has been coded, debugged, and tested by the author and currently resides at the NTEC facilty.

The DIMPRP software is a FORTRAN coded program that performs many digital image enhancement functions. In addition, it allows the user to read or write image data, to or from, either the IKONAS framebuffer memory, or a VAX-11/780 disk drive.

CHAPTER 3

A GENERAL DIGITAL IMAGE PROCESSING PROGRAM

The DIMPRP Menu

The DIMPRP software is menu driven which means that upon running the program, a list of "command options" appears on the terminal screen, prompting the user to select and execute one of the commands. The commands are the various image processing functions that may be performed by the user. If, after selecting a command option, the program requires additional input data, the user will be prompted for the data as necessary. The advantage of coding the DIMPRP software such that it is menu driven is that, while the image processing functions that it performs may be very powerful and complex, the program itself is very easy to use; even for a user with a relatively small amount of exeperience in digital image processing.

An exact representation of the command option menu that appears on the terminal screen upon running DIMPRP is given on page 24. Figure 4 shows the author at a terminal which displays the command option menu. Figure 5 shows the Tektronix 4014-1 graphics terminal which is used to display the histogram of an image. Figure 6 shows the IKONAS RDS-3000 computer graphics system with an enhanced image displayed on the TV monitor.

COMMAND OPTIONS

```
 0) Display Command Options
 1) Read Image Data for Processing from Disk
 2) Read Image Data for Processing from IKONAS Framebuffer memory
 3) Store Image on Disk
 4) Display Image on IKONAS
 5) Histogram of B/W Image  (displayed on TEK Graphics Terminal)
 6) Lowpass Filter  (Smoothing Operation)
 7) Highpass Filter  (Image Sharpening)
 8) Edge Enhancement using a Gradient Operator
 9) Contrast Enhancement  (Linear Stretch of Histogram)
10) Gray Level Reduction
11) Invert Grayscale
12) Set Threshold Intensity Levels
13) Pseudocolor Enhance
-1) EXIT Program
```

Figure 4. Author Executing the DIMPRP Software

Figure 5. The TEK 4014-1 Graphics Terminal

Figure 6. The IKONAS Hardware

Obtaining an Image

There are two possible sources of images. Once the image has been digitized, it may be read from the IKONAS framebuffer. The image may also be obtained from the VAX-11/780 disk if the image was stored there previously. The command options that select the image source are command options one and two.

The bit depth and array size of the image may vary. The image may have an array size of either 512x512 pixels or 1024x1024 pixels, with the 512x512 image array size being the most common. The bit depth may be either 24 bits, 8 bits, or 4 bits corresponding to full color, monochrome, or data compressed images, respectively.

The 24-bit full color image may be read directly from the IKONAS frame buffer memory or from an unformatted integer disk file. As detailed previously in this report the basic memory element of the IKONAS framebuffer memory is a 32-bit integer with bits 0-7 containing the red intensity level information, bits 8-15 containing the green intensity level information, and bits 16-23 containing the blue intensity level information. Therefore, there are only 24 bits of intensity information within the 32 bit integer. The most significant byte of the integer, bits 24-31, contain no information. It is possible to transfer this 32 bit full color image from the framebuffer to a disk file using command option 3 of DIMPRP. Thus the inverse operation of reading an image from a disk file depends on the method used by command option 3 to store the image in the file originally. Full color images are stored in unformatted two-dimensional integer files. These files are structured so that there are 512 rows by 512 columns of 32-bit integers. Once the image is read from a disk file or the IKONAS framebuffer memory, the intensity information of each pixel is stored in the two-dimensional array IMG. An image is read from a disk file using the FORTRAN READ statement. Images are read from the framebuffer memory using the FORTRAN IKPRD subroutine described previously in this report. The FORTRAN code that implements these functions is given in the source listing of DIMPRP in the appendix.

Monochrome images require only 8 bits of intensity level information as opposed to the 24 bits reqired for a full color image. Thus for a monochrome image, there are 256 (2**8) possible gray levels and, all the image intensity information is contained in the red region of the 32-bit integer (bits 0-7). Since bits 0-7 contain all the necessary image intensity data, when a monochrome image is read from the IKONAS framebuffer, bits 8-31 are "masked off". Similarly, when a monochrome image is stored in a disk file, only bits 0-7 of the integer variable IMG are written to the file. Thus monochrome images are stored in unformatted byte files. These files contain 512 rows by 512 columns of 8-bit bytes. This results in a disk file that reqires 4 times less space on the disk than the full color image files.

By reducing the number of gray levels from 256 to 16, it is possible to further reduce the amount of space on a disk that an image file requires. Images with only 16 (2**4) gray levels may be represented by 4-bit intensity level values. An image may be coded such that the even numbered array elements are contained in the 4

LSB's (least significant bits) of a byte and the odd numbered array elements are contained in the 4 MSB's (most significant bits) of the same byte. An image coded in this manner may be stored in an unformatted byte file of 256 rows by 256 columns. This results in a disk file that requires 2 times less disk space than conventional 256 gray level monochrome image files, and 8 times less disk space than full color image files. When read from disk, the coded image data must be decoded properly to form a monochrome image. The FORTRAN code that implements the coding of an image for writing to a disk file and the decoding of an image after reading from a disk file is contained in the source listing of DIMPRP in the appendix.

Displaying an Image

Both color and monochrome images may be displayed by using the FORTRAN subroutines IKPWR and IKBWR. The IKPWR subroutine is used to write image data from the two-dimensional array, IMG, to the IKONAS framebuffer. As described previously in this report, the image intensity data is represented by 8-bit pixel values in the case of a monochrome image, and 24-bit pixel values for a full color image. The IKBWR subroutine is used to write data to the IKONAS bus for controling the operation of the channel crossbar switch. If the data value passed to the IKBWR subroutine is the octal number zero, all of the output channels are taken from the red (bits 0-7) input channel. Thus, all three lookup tables receive the same image intensity data and the resultant image is monochrome. If the data value passed to the IKBWR subroutine is the octal number 44, the channel crossbar switch becomes transparrent. That is, the output red channel is taken from the input red channel, the output green channel is taken from the input green channel, and the output blue channel is taken from the input blue channel. In this way, a full color image may be displayed.

Image zero of the IKONAS framebuffer memory is cycled through 30 times a second, with the binary data passing through the LUVO module. The LUVO module converts the binary data to an analog signal suitable for "driving" RGB (red, green, and blue) TV monitors. Thus, the TV monitor is refreshed at a rate of 30 times a second with the image data that is located in image zero of the IKONAS framebuffer memory. Therefore data that is written to image zero of the IKONAS framebuffer is simultaneously displayed on the TV monitor.

Image Enhancement

Most image processing techniques fall into one of two broad categories: subjective and quantitative. "Subjective image processing, often called image enhancement, stems from the use of the computer as a means of manipulating an image into a form more amenable to human use" (Green 1983). As a rule, enhancement broadly refers to the manipulation of imagery to present to the viewer additional information or insight into some factor concerning the preenhanced image. Broad categories of enhancement techniques might be listed as follows: (1) intensity mappings or point processes; (2) spatial processes.

The following sections of this paper address these categories. The first of these, intensity or point processes, refers to the mapping of individual pixels to new values independent of their neighboring pixel values. Image enhancement by histogram modification, and pseudocolor image enhancement are both examples of point processes. The second topic covered in the following sections of this report is that of spatial processing. In spatial processing, a given pixel is mapped into a new value as a function of its surrounding pixel values. Image smoothing, edge enhancement, and image sharpening are all examples of spatial processes.

The Image Histogram

A basic tool utilized in performing subjective enhancement, especially contrast enhancement, is the image histogram. The image hitogram reveals the distribution of digitized intensity within an image; it is represented graphically as a plot of the number of picture elements at a given intensity, N(i), plotted versus intensity, i. Three examples of image histograms are shown in Figure 7.

The top histogram in Figure 7 represents a well-exposed image, with most of the pixel intensities falling around midscale or midgray. The middle histogram in Figure 7 is a bimodal histogram with two distinct peaks. This indicates that the image could be a high-contrast image with many dark and many bright regions. The bottom histogram of Figure 7 corresponds to an overexposed image. The majority of the pixels have been saturated at or near full white.

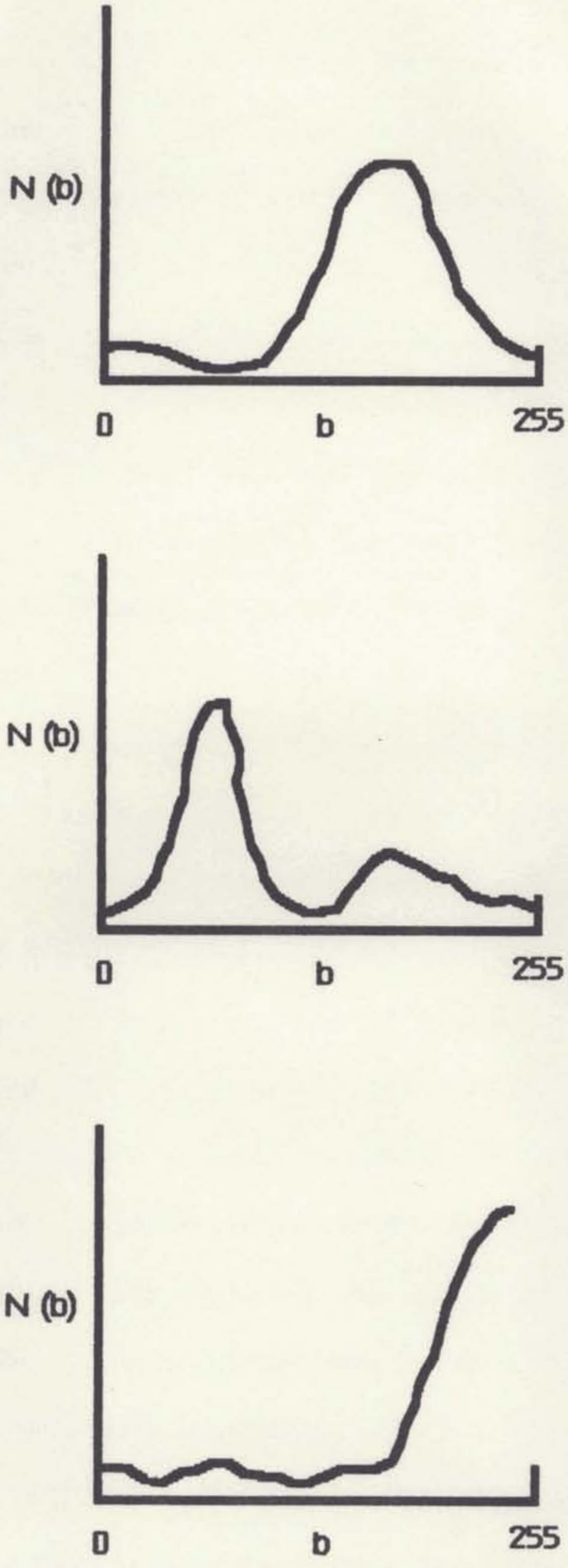

Figure 7. Image Histograms

Image Enhancement by Histogram Modification

A histogram of gray level content provides a global description of the appearance of the image. The methods discussed in this section achieve enhancement by modifying the histogram of a given image in a specified manner. The phrase, image enhancement by histogram modification, may seem somewhat confusing since almost all image enhancement procedures will modify the image histogram to at least some degree. In this section, the original image histogram is actually used as the basis for the modification of an image. The type of image enhancement referred to in this section will be contrast enhancement.

Most digital imaging systems can resolve more levels of intensity than can be presented on either film or volatile display. The NTEC facility produces monochrome images using 8 bits of resolution, and therfore image intensity levels range from 0 to 255. The human eye is able to resolve approximately 32 distinct shades of gray (Green, 1983). The digital imaging system therefore provides more intensity information than the human eye can resolve.

When an image is displayed on a TV monitor system, the full intensity range of the imaging system is displayed so that black corresponds to the minimum quantized intensity and white corresponds to the maximum quantized intensity. For an 8-bit system, black thus represents $i = 0$ and white represents $i = 255$. For an 8-bit image, the 255 shades of gray available within the image data are subsampled at a ratio of approximately 255:32, or around 8:1.

If a digital image has an intensity distribution such that the majority of pixels fall within a subset of the digitized intensity value range, straightforward display of the unprocessed image will yield a very low-contrast result that does not fully display the information actually available in the digitized image. In the histogram shown in Figure 7a, the majority of the pixels in the image are at intensity values clustered about the midgray point of the histogram. If the image is diplayed without modification of the digital data, variations of only one or two gray shades about midgray will occur.

It is necessary to transform the intensity values in the unprocessed image in order to display the subtle intensity variation present within the image. There are several commonly ultilized techniques that perform this transformation: (1) histogram equalization; (2) direct histogram specification; (3) ramp CDF stretch; (4) linear or contrast stretch. Only the linear or contrast stretch technique will be presented in this report as it is the method of image contrast enhancement used by DIMPRP.

The objective of linear contrast enhancement is to utilize the full dynamic range of the output display medium. The technique is particularly well suited to images with Gaussian or near-Gaussian histograms, where all the intensity values fall generally within a narrow subset of the intensity range available within a particular imaging system. The technique is a point process, as described previously, and is based on transforming each pixel intensity level in the input image into a new value in the output image.

In order to perform a linear contrast enhancement, the analyst examines the image histogram and determines the intensity range containing the large majority of the input pixels. The range is defined by limiting intensity values IH (i high) and IL (i low). An output image is constructed in the following manner:

1. For each pixel with intensity at or below IL, the intensity value is reassigned to black (i = 0).

2. For each pixel with intensity at or above IH, the intensity value is reassigned to full white (i = 255).

3. For all pixels with input intensity values between IL and IH, an output intensity is assigned that represents a linear transforamtion between IL and IH.

The algorithm can be represented mathematically by the following equations:

$$(i\ out) = \begin{cases} 0, & (i\ in) \leq IL \\ 255*((i\ in) - IL)/(IH - IL), & IL \leq (i\ in) \leq IH \\ 255, & (i\ in) \geq IH \end{cases} \qquad (3.1)$$

Figure 8b shows the results of applying a linear contrast enhancement to the image in Figure 8a. Figures 9a and 9b show the image histograms of the images given in Figures 8a and 8b, respectively. It can be seen that a linear contrast enhancement preserves the relative brightness relationships within the image but has the effect of redisdlaying the information content by utilizing the full available dynamic range of the output medium.

Figure 8a. Original Image

Figure 8b. Contrast Enhanced Image

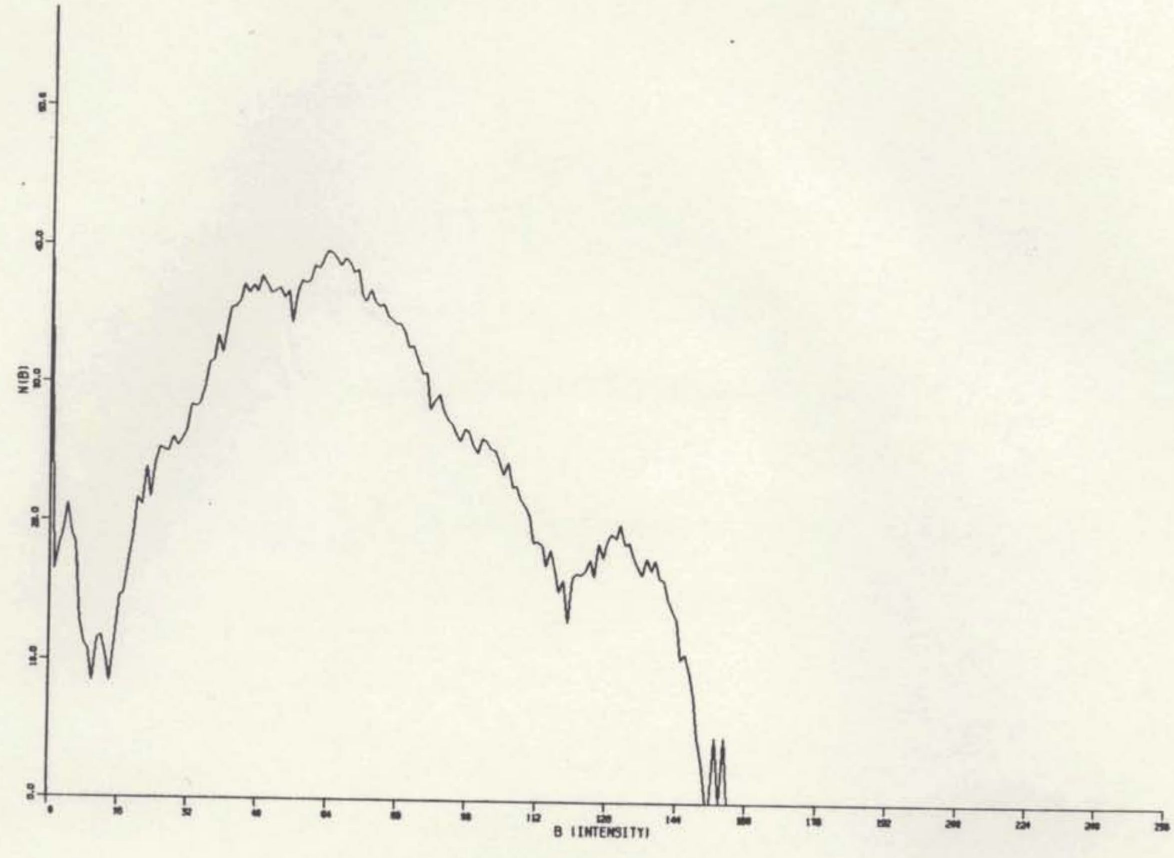

Figure 9a. Original Image Histogram

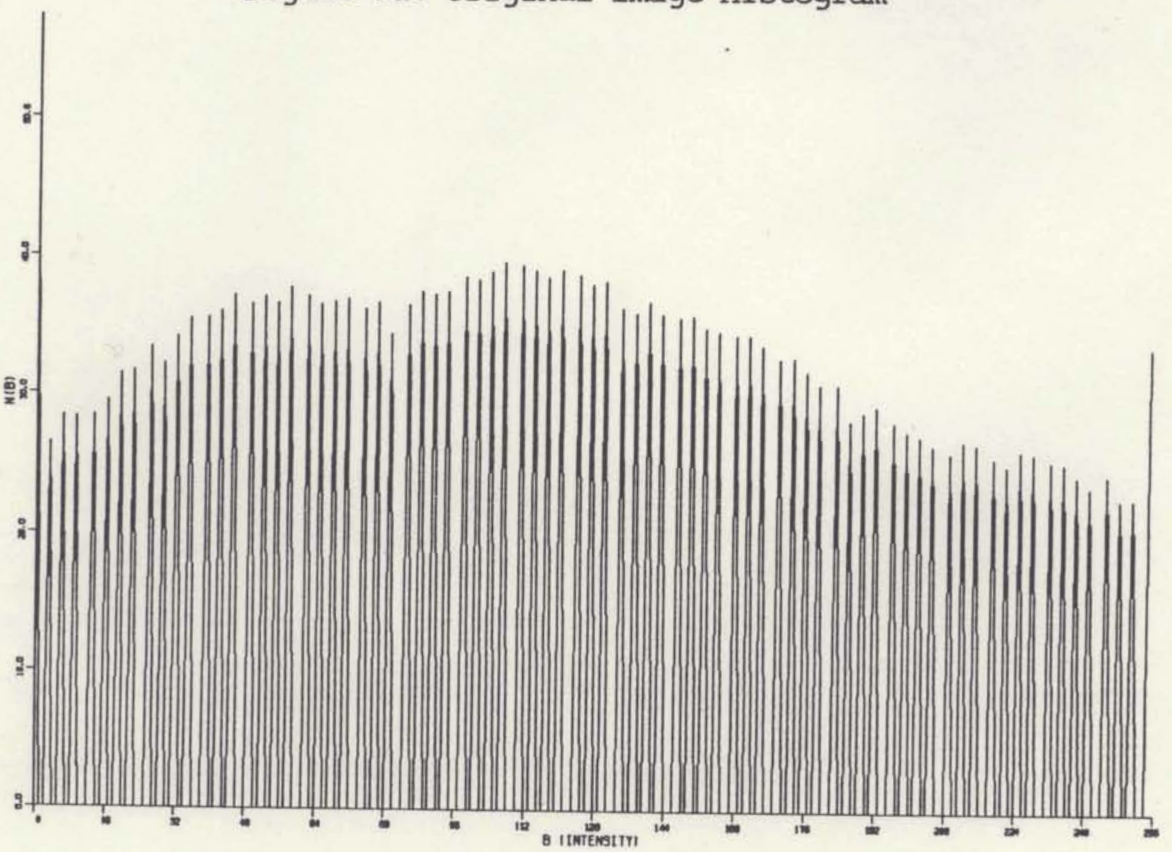

Figure 9b. Enhanced Image Histogram

Image Smoothing

Smoothing operations are used primarily for diminishing the spurious effects that may be present in a digital image as a result of a poor sampling system or transmission channel. Smoothing is equivalent to a lowpass filter operation in the frequency domain. Lowpass filtering in the frequency domain is equivalent to integration (summation in the discrete case) in the time domain. Since digital images are both discretized and quantized, images may be lowpass filtered by means of summation or averaging in the time (spatial) domain.

Neighborhood averaging is a straightforward spatial-domain technique for image smoothing. Given an N x N image f(x,y), the procedure is to generate a smoothed image g(x,y) whose gray level at every point (x,y) is obtained by averaging the gray-level values of the pixels of f contained in a predefined neighborhood of (x,y). In other words, the smoothed image is obtained using the relation

$$g(x,y) = \frac{1}{M} \sum_{(n,m) \in S} f(n,m) \qquad (3.2)$$

for x and y = 0,1,...,N-1. S is the set of coordinates of points in the neighborhood of the point (x,y), and M is the total number of points defined by the coordinates in S. Neighborhoods that are symmetrical are the most commonly used neighborhoods in image smoothing, except on or near the edges of an image. Points in these regions may be replaced by partial neighborhoods.

Another way of representing the spatial neighborhood averaging process is as a convolutional process. In this way an l x l "mask" is convolved with an N x N image. The mask is an array of numbers that represent the inverse Fourier transform (two-dimensional pulse response) of a lowpass filter. The two-dimensional convolution summation for a linear shift-invariant system is given below.

$$y(m,n) = \sum_{k=-\infty}^{\infty} \sum_{r=-\infty}^{\infty} h(k,r)x(m-k,n-r) \qquad (3.3)$$

For a 3 x 3 mask (l = 3), the convolutional summation may be written with new limits, since the pulse function is zero outside these limits. i.e.

$$y(m,n) = \sum_{k=-\frac{l}{2}}^{\frac{l}{2}} \sum_{r=-\frac{l}{2}}^{\frac{l}{2}} h(k,r)x(m-k,n-r) \qquad (3.4)$$

In equation (3.4), h represents the lowpass convolutional mask, x is the original image, and y is the output or processed image.

Below are the three lowpass convolutional arrays that are used in DIMPRP. It should be noted that mask1 is identical to the neighborhood averaging process desribed above with a radius of $\sqrt{2}\delta x$ and $\sqrt{2}\delta y$. The other masks, mask2 and mask3, are also similar to the neighborhood averaging, although these masks weight some pixel values in the neighborhood with a weighting coefficient greater than one.

Mask 1 $$h = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix} \qquad (3.5)$$

Mask 2 $$h = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{vmatrix} \qquad (3.6)$$

Mask 3 $$h = \begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix} \qquad (3.7)$$

The neighborhood averaging implementations of the image smoothing process are usually referred to as mean filters. This is because in the processes described above, the center pixel intensity value of a given mask is replaced with the mathematical mean value, or some type of weighted mean, of that pixel value and the neighboring pixel values within the mask. Thus the straightforward convolutional summation implementation of the above masks yields the desired smoothing effect.

Another implementation of the image smoothing process is the median filter. In this implementation, the center pixel intensity value of an l x l mask is replaced by the median value of the numbers within the mask. Therfore this process is not implemented as a mathematical summation, but as a sorting process. The pixel values within a given mask are sorted using a bubble sort such that the values within the mask are arranged in increasing order of magnitude. The median value of the numbers within the range of the mask is then automatically in the center position of the mask, if l is an odd integer.

Swown in Figures 10 and 11 are mean and median filtered images respectively.

Figure 10a. Original Image

Figure 10b. Mean Filtered Image

Figure 11a. Original Image

Figure 11b.Median Filtered Image

For purposes of noise filtering the two implementations of image smoothing ,the mean filter and the median filter, are compared below.

Mean Filter
- better for additive gaussian noise
- blurs edges and image in general
- seperable operator
- recursive mathematical implementation

Median Filter
- better for impulsive, periodic, occasional noise
- maintains edges
- rounds corners
- erases smaller objects
- quasi seperable
- sort implementation

Edge Enhancement

It was noted in the previous section that averaging pixels over a region tends to blur detail in an image. Since averaging is analogous to integration, it is natural to expect that differentiation will have the opposite effect and thus sharpen a given image.

The most commonly used method of differentiation in image processing is the "gradient". Given a function f(x,y), the gradient of f at coordinates (x,y) is defined as the vector

$$\mathbf{G}(f(x,y)) = \begin{vmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{vmatrix} \qquad (3.8)$$

Two important properties of the gradient are: (1) the vector $\mathbf{G}(f(x,y))$ points in the direction of maximum rate of increase of the function f(x,y); and (2) the magnitude of $\mathbf{G}(f(x,y))$, denoted by

G(f(x,y)), and given by

$$G(f(x,y)) = mag(G) = ((\partial f/\partial x)^2 + (\partial f/\partial y)^2)^{1/2} \tag{3.9}$$

equals the maximum rate of increase of f(x,y) per unit distance in the direction of **G**.

Equation (3.9) is the basis for a number of approaches to image differentiation. It is noted that this expression is in the form of a two-dimensional derivative function and that it is always positive. In practice, the scalar function G(f(x,y)) is commonly referred to as the gradient of f. This terminology will be used throughout the following discussion to avoid having to continually refer to G(f(x,y)) as "the magnitude of the gradient."

For a digital image, the derivatives in equation (3.9) are approximated by differences. One typical approximation is given by the relation

$$G(f(x,y)) = |f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)| \tag{3.10}$$

The above arrangement for approximating the gradient is not unique. There are other useful approximations which can be made. In any case, it should be noted that for all the approximations, the value of the gradient is proportional to the difference in gray levels between adjacent pixels. Thus, as expected, the gradient assumes relatively large values for prominent edges in an image, and small values in regions that are fairly smooth, being zero only in regions that have a constant gray level.

Therefore, the gradient operator is often used for image edge enhancement or edge "detection".

Once a method of approximating the gradient has been selected, there are numerous ways of using the results for generating a gradient image g(x,y). Only those that have been used in the DIMPRP software package will be dicussed in this report. The various ways of using the results of equation (3.10) for generating a gradient image, g(x,y), will be referred to as implementations (implmt).

In the first implementation (implmt1), the edges are set to a specified gray level, LG, if the gradient is above a specified gray level threshold value, T. Otherwise, the gradient image g(x,y) is set equal to the input image, f(x,y). It is sometimes desirable to study the gray-level variation of edges without interference from the background. This can be accomplished by forming the gradient (implmt2) as follows: (1) Let the gradient image g(x,y) equal the gradient if the gradient is above a specified threshold value, T; (2) In all other cases the gradient image g(x,y) is set equal to a specified background level, LB. Finally, if only the location of edges is of interest, a binary gradient picture (implmt3) may be formed such that the edges and the background are displayed in any two specified gray levels. This may be accomplished in the following manner: (1) The edges are set to a specified gray level, LG, if the gradient is above a specified threshold value, T; (2) In all other cases the gadient image g(x,y) is set to a specified background level, LB.

The types of edge enhancement that can be obtained by using equation (3.10) and implmt1, implmt2, and implmt3 are illustrated in Figures 12b, 12c, and 12d. Figure 12a shows an original image of moderate complexity.

Figure 12b is the result of using implmt1 with T = 25 and LG = 255, the latter being full white.

Figure 12c was obtained by using implmt2 with the same threshold as above and a background level of LB = 0, which is full black. The principal use of this particular approach is to examine the relative strength of the points which exceed the specified threshold.

Finally, Figure 12d was obtained by using implmt3 with T = 25, LG = 255, and LB = 0. This implementation is useful for displaying all the gradient points above the specified threshold.

Image Sharpening

An image will contain information at a wide range of spatial frequencies. Overall gradual transitions from light to dark within an image can be interpreted as low-frequency components. Rapid, local variations in contrast represent information at a higher frequency. Local texture is an example of image information that is high frequency information.

Image sharpening usually refers to the enhancement of local texture as described above. Thus image sharpening corresponds to high pass filtering an image. It is possible to perform an analysis of the spatial frequencies present in a digital image utilizing Fourier transform techniques.

Figure 12a. Original Image

Figure 12b. Edge Enhanced Image using Implmt1

Figure 12c. Edge Enhanced Image using Implmt2

Figure 12d. Edge Enhanced Image using Implmt3

One or two-dimensional Fourier transforms of an image will reveal a large low-frequency component, corresponding to the gradual intensity variation within the scene, and a variety of high-frequency "spikes" may be present in the transform, indicating periodic structure or periodic scene detail present within an image.

Simple techniques of implementing high pass filtering operations have been developed that involve convolution in the spatial domain. The mathematical foundations of these routines are identical to those developed previously in this report for the lowpass or smoothing operations. Recall that in that previous section it was explained that an image smoothing operation corresponded to a lowpass filtering of the image. It was also explained that the filtering could be implemented in the spatial domain by convolving a mask (two-dimensional pulse function) with the image as opposed to multiplying an image by a two-dimensional transfer function in the frequency domain. Thus, image sharpening may be performed in the same manner as image smoothing. The only diference is that a high pass filter is used instead of a lowpass filter. Therefore the two-dimensional pulse function (mask) that represents the high pass filter in the spatial domain is different than that for image smoothing but the mathematics remain the same.

Below are the three high-pass convolutional arrays used in the DIMPRP software to sharpen images.

Mask 1 $$h = \begin{vmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{vmatrix} \qquad (3.11)$$

Mask 2 $$h = \begin{vmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{vmatrix} \quad (3.12)$$

Mask 3 $$h = \begin{vmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{vmatrix} \quad (3.13)$$

Figures 13a and 13b show an original image and an image that was sharpened using mask2, respectively.

Gray Level Reduction

In image processing, it is sometimes advantageous to reduce the number of gray levels that are displayed from 256 (for an 8-bit system) to some smaller number of gray levels. There are several motivations for reducing the number of gray levels in an image: (1) As discussed previously, the eye is only able to resolve approximately 32 distinct shades of gray; (2) By reducing the amount of gray levels within an image, it is possible to reduce the amount of memory necessary to store the image; (3) By reducing the number of gray levels in an image, it is possible to transmit the image over a communication channel with some bandwidth, B, at a greater speed, or transmit the image at the same speed over a channel with a bandwidth smaller than B; (4) Reducing the number of gray levels within an image facilitates more readily the determination of suitable intensity threshold levels for a given image.

Reducing the number of gray levels in an image by an integer divisor is a very staightforward process.

Figure 13a. Original Image

Figure 13b. Sharpened Image

Each pixel is divided by an integer divisor using integer division, then multiplied by the same number. The gray level reducing equation is given below.

$$(i\ out) = N*((i\ in)/N) \qquad (3.14)$$

In the above equation, N is the reduction factor and must be a power of two. If N = 8 then the number of gray levels is reduced, from the maximum amount of 256, to 32 (256/8). Recall that eqaution (3.14) is performed using integer arithmetic such that any (i in) that is not a power of two, is lowered, by N, to the nearest power of two. For example, if N were chosen to be 128 such that the number of gray levels was to be reduced to 2, then any intensity below 128 would become 0, and any intensity level greater than or equal to 128 would become 128.

Figure 14a shows an original image displayed with 256 distinct levels of gray. Figure 14b shows the same image represented by 16 levels of gray. Finally, Figure 14c. shows the same image as in Figure 14a, except with only 4 distinct levels of gray.

Gray Scale Inversion

Gray scale inversion merely refers to an intensity level mapping whereby input pixel intensities that are black (0) become white (255), and input pixels that are white (255) become black (0). All pixel values between full black and full white are transformed in a similar fashion.

Figure 14a. Original Image with 256 Gray Levels

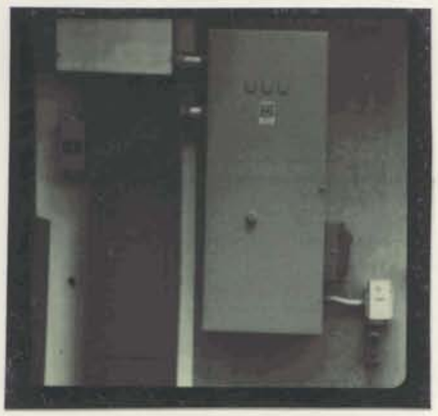

Figure 14b. Image with 16 Gray Levels

Figure 14c. Image with 4 Gray Levels

The following equation represents this intensity transformation for a system with a maximum of 256 gray levels.

$$(i\ out) = 255 - (i\ in) \qquad (3.15)$$

Thresholding

Thresholding, sometimes referred to as bit-slicing, is a technique designed to isolate particular intensity intervals within an image. The technique involves generation of an output image with all pixels black (0), except for those pixels in the input image whose intensities fall within a particular region of interest.

In the thresholding scheme used in the DIMPRP software, the output image will be generated with white (255) pixels wherever the input image intensity levels fall within the lower white (LW) and upper white (UW) intensity level boundaries. The output image will be generated black (0) if the input image is below the upper black (UB) intensity level boundary. The output image will be identical to the input image any time the input image is not within predescribed limits. The following equations, with UB, LW, and UW as given above, represent the DIMPRP thresholding scheme.

$$(i\ out) \left\{ = \begin{array}{ll} 0, & (i\ in)\quad UB \\ 255, & LW\quad (i\ in)\quad UW \\ (i\ in), & \text{elsewhere} \end{array} \right. \qquad (3.16)$$

Pseudocolor Image Enhancement

Attention has been focused thus far in this report on processing techniques for monochrome images. A powerful area of digital image processing is the use of pseudocolor for image display and enhancement. The motivation for using color in image processing is provided by the fact that the human eye can discern thousands of color shades and intensities. This is in sharp contrast with the eye's relatively poor performance with gray levels where, as indicated previously, only approximately 32 shades of gray are detectable at any point in an image by the average observer. The reader needs only to turn off the color next time he views a TV set in order to verify the superior performance of the eye when interpreting color versus monochrome information.

The basic idea of processing in pseudocolor is to start with a monochrome image and then assign a color to each pixel based, for example, on its intensity. The range of techniques for color assignment are limited only by the capabilities of the display system and the ingenuity of the user. Straightforward techniques for color coding can sometimes bring out information which is often difficult to detect and interpret in a monochrome image. Examples of areas in which pseudocolor image processing techniques are used are: (1) TV weather forcasters use pseudocolor techniques to enhance satellite imagery and to produce "digital color radar"; (2) Doctors use pseudocolor enhancement techniques on X-rays to enable clearer interpretations of the X-rays.

The technique of "density (or intensity) slicing" and color coding is one of the simplest examples of pseudocolor image processing, and is also the technique employed by the DIMPRP software. If an image is viewed as a two-dimensional intensity function, the method can be interpreted as one of placing planes parallel to the coordinate plane of the image; each plane then "slices" the function in the area of intersection (Gonzalez and Wintz, 1977). The term "density slicing" arises from calling the gray levels densities, a terminology that is commonly associated with this particular method.

It is evident that if a different color is assigned to each side of the plane, then any pixel whose gray level is above the plane will be coded with one color, while any pixel below the plane will be coded with the other. Levels that lie on the plane itself may be arbitrarily assigned one of the two colors. The result of this scheme would produce a two-color image whose relative appearance can be controlled by moving the slicing plane up and down the gray-level axis.

The technique described above may be generalized to more than two colors by using more than one slicing plane. The pseudocolor technique employed by the DIMPRP software is that of first reducing the number of gray levels from 256 to 8. These gray level intensities are represented by the decimal numbers 0 - 7, or the binary numbers 000 - 111. Color data is represented by 24 bit data values with the first 8 bits, bits 0 - 7, representing the red intensity data, bits 8 - 15 representing the green intensity data, and bits 16 - 23 representing the blue intensity data. Thus it is possible to generate the primary colors, (red, greeen, and blue), the secondary colors (magenta, cyan, and yellow), plus white and black by mapping the binary values 0 - 7, to the hexidecimal values 00/00/00 - FF/FF/FF. This procedure is illustrated in Table 2.

Table 2. Gray level to Pseudocolor coding.

Binary value	Hexidecimal value	Color
000	00/00/00	Black
001	00/00/FF	Red
010	00/FF/00	Green
011	00/FF/FF	Yellow
100	FF/00/00	Blue
101	FF/00/FF	Magenta
110	FF/FF/00	Cyan
111	FF/FF/FF	White

Figures 15a and 15b illustrate an original image, and an image that has been pseudocolor enhanced using the coding scheme above, respectively.

Program Usage

The DIMPRP software, that has been designed and coded to perform the image processing functions that have been detailed in this report, is very user friendly. To obtain this goal, it was coded such that all functions are accessible through a "main menu" as detailed previously in this report. Therefore upon running DIMPRP, it is only necessary to choose a "command option" from the main menu. Selecting an option from the menu is as simple as typing, or entering, the number that appears to the left of the desired command option.

Figure 15a. Original Image

Figure 15b. Pseudocolor Enhanced Image

If additional input data is required to execute a particular command, the user will be "prompted" for this data as necessary. Much of the type of data that is required by DIMPRP has been outlined already in this report.

The DIMPRP software is an executive program that must be linked with several subroutines. These subroutines are listed below.

ALUVMS - Used in interfacing IKONAS routines with VAX-11/780 software

DMAVMS - Same as above

CLEARS - Used to clear the IKONAS display

TEKSUBS - Used to plot the image histogram on the Tektronix 4014-1

VPLOT1 - Same as above

VPLOT2 - Same as above

Given below are the names, along with a short description, of the major variables found within DIMPRP.

ANS	: Numerical answer to user command request
B	: 1-D vector of bytes (intensity information)
CODE	: Used in IKPWR to select mode
COND	: Logical variable used to test bit placement
FILENAME	: Name of disk file on the VAX-11/780
GRAYSHADE	: Used internally by IKPWR and IKPRD
HISTO	: 1-D vector containing histogram information
IKBUFF	: 1-D vector of integers (intensity information)
IMAGE	: Same as above
IM	: Temporary storage vector for median sort
IMG	: 2-D array of image intensity information
IMG1	: Same as above
NPIX	: Used internally by IKPWR and IKPRD
OFFSET	: Same as above
P	: Partial histogram of an image
TYPE	: Length of one raster scan of an image

CHAPTER 4

CONCLUSION

NTEC Facility Enhancement

It can be concluded that the digital image processing program (DIMPRP) that has been generated can be used with the VAX and IKONAS hardware to provide an immediate and convenient image processing capability for NTEC. Further conclusions of this report will be based on a discussion of enhancements that would further the digital image processing capabilities of the NTEC facility.

Hardware

The two major hardware systems that have been presented in this report are the IKONAS RDS-3000 raster display graphics system and the VAX-11/780 minicomputer. Functionally, there are no changes that could be made to either hardware system that would enhance their use in the NTEC digital image processing facility. However, the IKONAS system has existing capabilities that have not been presented in this report and cannot be fully utilized because of two problems.

The IKONAS system present at the NTEC facility is one of the first such systems produced and as such has "bugs" in it. Much of the IKONAS memory and subsystems consist of wire-wrapped boards that have been plagued with reliablity problems. Other IKONAS systems that contain printed circuit memory and subsystem boards are more reliable, having a longer mean time between failure. Thus it is suggested that the IKONAS system be upgraded with "PC" boards to ensure a more reliable operation.

The IKONAS systems lacks the type of documentation necessary for easy use of the system. The documentation of many system features can be confusing to all but the most experienced users. Thus the proper enhancement to be made in this case would be to generate better written documentation of the IKONAS user procedures.

Software

The major software element of the NTEC digital image processing facility is the digital image processing program (DIMPRP). This program is relatively simple to use and contains a variety of image enhancement procedures. The DIMPRP program could be enhanced in two ways. The first enhancement would be to further the image processing capabilities of the program by adding more processing functions. These functions could be in an entirely different category than image enhancement. For example image restoration, image segmentation and image recognition could be added to extend the existing capabilities of the DIMPRP software. Finally, the speed of execution could be increased by coding various image processing functions in either VAX assembly language or AMD 2900 microcode. The AMD 2900 microcode implementation would result in the fastest execution but has the disadvantages associated with the IKONAS system discussed previously such as poor reliability and documentation. Thus the VAX assembly language implementation of the various image processing functions within the DIMPRP software would sufficiently increase the execution speed of the software to warrant its use.

APPENDIX

DIMPRP Source Listing

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                     C
C       PROGRAMMER - JAMES F. ROESCH JR.                              C
C       ........................................................     C
C                                                                     C
C       PROGRAM - DIGITAL IMAGE PROCESSING PROGRAM  (DIMPRP)          C
C       ........................................................     C
C                                                                     C
C       PURPOSE - THIS PROGRAM HAS BEEN PROVIDED TO NTEC, CODE N-74   C
C                 TO ENABLE USERS AT THIS LOCATION TO BE EDUCATED IN  C
C                 THE USE OF, AND PERFORM MANY USEFUL IMAGE           C
C                 PROCESSING FUNCTIONS. FOR EXAMPLE, THIS PROGRAM     C
C                 MAY BE USED TO SMOOTH OR ELIMINATE GAUSSIAN NOISE   C
C                 IN DIGITIZED IMAGES, CONTRAST ENHANCE POORLY        C
C                 DEVELOPED IMAGES, ENHANCE THE EDGES OF AN IMAGE,    C
C                 PSEUDO-COLOR ENHANCE AN IMAGE, AND EXECUTE MANY     C
C                 OTHER USEFUL IMAGE PROCESSING FUNCTIONS.            C
C       ........................................................     C
C                                                                     C
C       USAGE - THIS PROGRAM HAS BEEN WRITTEN SO AS TO BE VERY EASY   C
C               TO USE. TO RUN IT, LOGIN TO THE OHIO STATE ACCOUNT,   C
C               e.g. USERNAME: OHIOST                                 C
C                    PASSWORD: ANTS                                   C
C               THEN TYPE, DIMPRP. UPON RUNNING THE PROGRAM, A MAIN   C
C               MENU, WHICH IS THE COMMAND OPTIONS LIST, WILL APPEAR  C
C               ON THE TERMINAL SCREEN. TO SELECT AN OPTION, TYPE     C
C               THE NUMBER THAT APPEARS TO THE LEFT OF THE DESIRED    C
C               COMMAND OPTION. IF ADDITIONAL IMFORMATION IS REQUIRED C
C               FROM THE USER, THE TERMINAL SCREEN WILL CLEAR, AND    C
C               THE USER WILL BE PROMTED FOR THE ADDITIONAL INPUT     C
C               DATA AS REQUIRED. A REPORT WILL ACCOMPANY THIS        C
C               PROGRAM TO FURTHER CLARIFY WHAT ADDITIONAL INPUT DATA C
C               IS REQUIRED BY EACH COMMAND AND WHY.                  C
C       ........................................................     C
C                                                                     C
C       DESCIPTION OF PARAMETERS                                      C
C                                                                     C
C                 ANS : NUMERICAL ANSWER FROM USER RESPONSES          C
C                   B : 1-D VECTOR OF BYTES (INTENSITY INFO)          C
C                CODE : USED IN IKPWR TO SELECT MODE                  C
C                COND : LOGICAL VARIABLE USED TO TEST BIT PLACEMENT   C
C            FILENAME : NAME OF DISK FILE ON VAX 11/780               C
C           GRAYSHADE : USED INTERNALLY BY IKPWR                      C
C               HISTO : HISTOGRAM OF AN IMAGE                         C
C              IKBUFF : 1-D VECTOR OF INTEGERS (INTENSITY INFO)       C
C               IMAGE : SAME AS ABOVE                                 C
C                  IM : TEMPORARY STORAGE VECTOR FOR MEDIAN SORT      C
C                 IMG : 2-D ARRAY OF IMAGE INTENSITY INFO             C
C                IMG1 : SAME AS ABOVE                                 C
```

```
C            NPIX : USED INTERNALLY BY IKPWR                          C
C          OFFSET : USED INTERNALLY BY IKPWR                          C
C               P : PARTIAL HISOGRAM OF AN IMAGE                      C
C            TYPE : LENGTH OF ONE RASTER SCAN OF AN IMAGE             C
C       .........................................................     C
C                                                                     C
C       METHOD - DESCRIBED IN RESEARCH REPORT BY JAMES F. ROESCH, JR. C
C                ENTITLED, DIGITAL IMAGE PROCESSING USING NTEC        C
C                FACILITIES.                                          C
C       .........................................................     C
C                                                                     C
C       SUBROUTINES AND SUBPROGRAMS REQUIRED - ALUVMS,DMAVMS,CLEARS   C
C                                             TEKSUBS,VPLOT1,VPLOT2   C
C       .........................................................     C
C                                                                     C
C       REMARKS - RESERVED FOR FUTURE COMMENTS                        C
C                                                                     C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

      INTEGER*4 IMG(1024,1024),IKBUFF(1024),OFFSET,GRAYSHADE,CODE,TYPE
      INTEGER*2 IMG1(512,512),IM(81),IMAGE(1024),NPIX
      REAL*4 HISTO(256),P(256)
      LOGICAL COND
      BYTE B(4096)
      CHARACTER*50 FILENAME
      EQUIVALENCE (IKBUFF(1),B(1))
      COMMON IINVAL,IINVBL,IMODE


10    WRITE(6,*) (CHAR(12),I=1,2)
      TYPE*
      TYPE*,'                    ***COMMAND OPTIONS***'
      TYPE*
      TYPE*,' 0) Display Command Options'
      TYPE*,' 1) Read Image Data for Processing from Disk'
      TYPE*,' 2) Read Image Data for Processing from IKONAS
     2 Framebuffer memory'
      TYPE*,' 3) Store Image on Disk'
      TYPE*,' 4) Display Image on IKONAS'
      TYPE*,' 5) Histogram of B/W Image  (displayed on TEK 4014
     5 Graphics Terminal)'
      TYPE*,' 6) Lowpass Filter  (Smoothing Operation)'
      TYPE*,' 7) Highpass Filter  (Image Sharpening)'
      TYPE*,' 8) Edge Enhancement using a Gradient Operator'
      TYPE*,' 9) Contrast Enhancement  (Linear Stretch of Histogram)'
      TYPE*,'10) Gray Level Reduction'
      TYPE*,'11) Invert Grayscale'
      TYPE*,'12) Set Threshold Intensity Levels'
      TYPE*,'13) Pseudocolor Enhance'
      TYPE*,'-1) EXIT Program'
```

```
C       -1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1

        ACCEPT*, ANS
        IF(ANS.EQ.-1) GOTO 2000

C       0000000000000000000000000000000000000000000000000000000000000000

        IF(ANS.EQ.0) GOTO 10

C       1111111111111111111111111111111111111111111111111111111111111111

        IF(ANS.EQ.1) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'1) Full Color Image  (24 bits)'
           TYPE*,'2) B/W Image  (8 bits)'
           TYPE*,'3) Data Compressed B/W Image  (4 bits)'
           ACCEPT*, ANS
           TYPE*
           TYPE*,'ENTER ARRAY SIZE: 1=512X512,  2=1024X1024'
           ACCEPT*, TYPE
           TYPE = TYPE*512
           TYPE*
           TYPE*,'ENTER IMAGE FILENAME'
           ACCEPT 1000, FILENAME
           OPEN (UNIT=1,FILE=FILENAME,STATUS='OLD',FORM='UNFORMATTED')
           IF(ANS.EQ.1) THEN

C       READ (TYPE X TYPE) # OF UNSIGNED INTEGERS FROM AN UNFORMATTED
C       INTEGER DISK FILE. NOTE: THESE INTEGERS ARE "READ IN", IN
C       "COLUMN MAJOR ORDER" TO MINIMIZE PAGE FAULTING!

              DO J=1,TYPE
                READ (1) (IMG(I,J),I=1,TYPE)
              END DO
           ELSE IF(ANS.EQ.2) THEN

C       READ (TYPE X TYPE) # OF SIGNED BYTES FROM AN UNFORMATTED BYTE
C       FILE. CONVERT THE SIGNED BYTES TO UNSIGNED INTEGERS.

              DO J=1,TYPE
                READ (1) (B(I),I=1,TYPE)
                DO I=1,TYPE
                   IMG(I,J) = B(I)
                   IMG(I,J) = JIAND(IMG(I,J),255)
                END DO
              END DO
```

```
         ELSE IF(ANS.EQ.3) THEN

C       READ (TYPE X TYPE) # OF SIGNED BYTES FROM AN UNFORMATTED BYTE
C       FILE. THEN, COVERT THE SIGNED BYTES TO UNSIGNED INTEGERS.
C       THE 4 LSB'S OF EACH INTEGER, MULTIPLIED BY 16 AND ADDED TO 8,
C       FORM THE EVEN NUMBERED IMAGE ARRAY ELEMENTS. THE 4 MSB'S, OF
C       THE LEAST SIGNIFICANT BYTE OF EACH INTEGER, ADDED TO 8, FORM
C       THE ODD NUMBERED IMAGE ARRAY ELEMENTS.

            DO J=1,TYPE
              READ (1) (B(I),I=1,TYPE/2)
              DO I=1,TYPE,2
                 IMG(I,J) = B((I+1)/2)
                 IMG(I,J) = JIAND(IMG(I,J),255)
                 IMG(I+1,J) = IMG(I,J)
                 IMG(I+1,J) = 16*(JIAND(IMG(I+1,J),15))+8
                 IMG(I,J) = JIAND(IMG(I,J),240)+8
              END DO
            END DO
         END IF
         CLOSE (UNIT=1)
         GOTO 10

C       2222222222222222222222222222222222222222222222222222222222222

        ELSE IF(ANS.EQ.2) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'1) Full Color Image  (24 bits)'
           TYPE*,'2) B/W Image  (8 bits)'
           ACCEPT*, ANS
           TYPE*
           TYPE*,'ENTER ARRAY SIZE: 1=512X512,  2=1024X1024'
           ACCEPT*, TYPE
           IF(TYPE.EQ.2) THEN
              CODE=3
              TYPE=1024
           ELSE
              CODE=2
              TYPE=512
           END IF
           CALL IKASSIGN
           IF(ANS.EQ.1) THEN

C       READ (TYPE X TYPE) # OF UNSIGNED INTEGERS FROM THE IKONAS DR64
C       IMAGE ZERO, IMAGE MEMORY.
```

```
            DO J=0,TYPE-1
              CALL IKPRD(CODE,0,J,TYPE,IKBUFF(1))
              DO I=1,TYPE
                 IMG(I,J+1) = IKBUFF(I)
              END DO
            END DO
         ELSE

C       READ (TYPE X TYPE) # OF UNSIGNED BYTES FROM THE IKONAS DR64
C       IMAGE ZERO, RED, IMAGE MEMORY.

            DO J=0,TYPE-1
              CALL IKPRD(CODE,0,J,TYPE,IKBUFF(1))
              DO I=1,TYPE
                 IMG(I,J+1) = B(4*I-3)
                 IMG(I,J+1) = JIAND(IMG(I,J+1),255)
              END DO
            END DO
         END IF
         GOTO 10

C       3333333333333333333333333333333333333333333333333333333333333

        ELSE IF(ANS.EQ.3) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'1) Full Color Image  (24 bits)'
           TYPE*,'2) B/W Image  (8 bits)'
           TYPE*,'3) Data Compressed B/W Image  (4 bits)'
           ACCEPT*, ANS
           TYPE*
           TYPE*,'ENTER ARRAY SIZE: 1=512X512,  2=1024X1024'
           ACCEPT*, TYPE
           TYPE = TYPE*512
           TYPE*
           TYPE*,'ENTER IMAGE FILENAME'
           ACCEPT 1000, FILENAME
           OPEN (UNIT=1,FILE=FILENAME,STATUS='NEW',FORM='UNFORMATTED')
           IF(ANS.EQ.1) THEN

C       STORE (TYPE X TYPE) # OF UNSIGNED INTEGERS IN AN UNFORMATTED
C       INTEGER DISK FILE. NOTE: THESE INTEGERS ARE "READ OUT", IN
C       "COLUMN MAJOR ORDER" TO MINIMIZE PAGE FAULTING!
```

```
            DO J=1,TYPE
              WRITE(1) (IMG(I,J),I=1,TYPE)
            END DO
          ELSE IF(ANS.EQ.2) THEN

C       STORE (TYPE X TYPE) # OF SIGNED BYTES IN AN UNFORMATTED BYTE
C       FILE. NOTE: B(1) IS EQUVALENCED TO IKBUFF(1), SO THAT WHEN
C       B(I), I=1,5,9,... IS WRITTEN TO THE DISK FILE, IT HAS THE SAME
C       EFFECT AS WRITING THE FIRST 8 BITS OF EACH 32 BIT WORD,
C       IKBUFF(I), I=1,2,3,...  .

            DO J=1,TYPE
              DO I=1,TYPE
                 IKBUFF(I) = IMG(I,J)
              END DO
              WRITE(1) (B(4*I-3),I=1,TYPE)
            END DO
          ELSE IF(ANS.EQ.3) THEN

C       STORE (TYPE X TYPE) # OF SIGNED BYTES IN AN UNFORMATTED BYTE
C       FILE. THE EVEN NUMBERED IMAGE ARRAY ELEMENTS FORM THE 4
C       LSB'S OF EACH BYTE. THE ODD NUMBERED IMAGE ARRAY ELEMENTS
C       FORM THE 4 MSB'S OF EACH BYTE. NOTE: THE BYTES ARE "WRITTEN
C       OUT" IN THE SAME MANNER AS DESCRIBED ABOVE.

            DO J=1,TYPE
              DO I=1,TYPE,2
          IKBUFF(I)=JISHFT(((JIAND(IMG(I,J),255))/16),4)
          IKBUFF((I+1)/2)=JIOR(IKBUFF(I),(JIAND(IMG(I+1,J),255))/16)
              END DO
              WRITE(1) (B(4*I-3),I=1,TYPE/2)
            END DO
          END IF
          CLOSE (UNIT=1)
          GOTO 10

C       4444444444444444444444444444444444444444444444444444444444444

        ELSE IF(ANS.EQ.4) THEN

C       CLEAR THE MAIN MENU SCREEN
          WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
          TYPE*,'ENTER MODE: 1=512X512 30Hz,  2=512X512 60Hz,
        1 3=1024X1024 30Hz'
          ACCEPT*, IMODE
```

```
      IF(IMODE.EQ.3) THEN
         CODE=19
         TYPE=1024
      ELSE
         CODE=18
         TYPE=512
      END IF
      TYPE*
      TYPE*,'1) B/W Image  (4 or 8 bits)'
      TYPE*,'2) Full Color Image  (24 bits)'
      ACCEPT*, ANS

      CALL IKASSIGN

      IINVAL = 1
      IINVBL = 13
      CALL CLEAR
      IINVAL = 0
      CALL CLEAR
      IF(ANS.EQ.1) THEN

C     IF THE IMAGE DATA IS B/W, USE THE CHANNEL CROSSBAR SWITCH
C     TO "ROUTE" THE IMAGE ZERO, RED, INTENSITY LEVEL DATA TO THE
C     GREEN AND BLUE CHANNELS AS WELL, THUS PRODUCING A B/W IMAGE.

         CALL IKBWR("20,"40602000,1,"0)
      ELSE

C     IF THE IMAGE IS A FULL COLOR IMAGE, SET THE CHANNEL XBAR
C     SWITCH TO IDENTITY.  i.e. RED DATA = RED CHANNEL, GREEN DATA
C     = GREEN CHANNEL, etc.

         CALL IKBWR("20,"40602000,1,"44)
      END IF

C     WRITE (TYPE X TYPE) # OF UNSIGNED INTEGERS TO THE IKONAS
C     DR64 IMAGE ZERO, IMAGE MEMORY.

      DO J=0,TYPE-1
         DO I=1,TYPE
           IKBUFF(I) = IMG(I,J+1)
         END DO
         CALL IKPWR(CODE,0,J,TYPE,IKBUFF(1))
      END DO
      GOTO 10
```

```
C        5555555555555555555555555555555555555555555555555555555555555555

         ELSE IF(ANS.EQ.5) THEN
            HI = 256
            LO = 0
            LEVEL = 256
            BINWID = (HI - LO)/LEVEL

C        INITIALIZE THE HISTOGRAM ARRAY TO ALL ZERO'S
            DO I=1,256,BINWID
               HISTO(I) = 0.
            END DO

C        CALCULATE HISTOGRAM
            DO J=1,TYPE
               DO I=1,TYPE
                 BIN = ININT(IMG(I,J)/BINWID)+1
                 HISTO(BIN) = HISTO(BIN)+1.
               END DO
            END DO

C        CLEAR THE TEK SCREEN
            CALL PLINT

C        DRAW THE X (INTENSITY LEVEL), AND Y (NUMBER OF PIXELS), AXIS
         CALL AXIS(.5,.5,'b (Intensity)',-13,13.5,0.,.84375,0.,16.,'I3')
         CALL AXIS(.5,.5,'N(b)',4,10.,90.,1.6666667,0.,10.,'F4.1')
         CALL OFFSET(-9.481481481,18.962962962,-3.,6.)

C        PLOT THE HISTOGRAM ON THE TEKTRONIX 4014-1
            DO X=0,255,BINWID
               IF(HISTO(X+1).EQ.0.) THEN
                 Y = 0.
               ELSE
                 Y = 10.*ALOG10(HISTO(X+1))
               END IF
               IF(X.EQ.0.) THEN
                 CALL PLOT(X,Y,13)
               ELSE
                 CALL PLOT(X,Y,12)
               END IF
            END DO
            GOTO 10

C        6666666666666666666666666666666666666666666666666666666666666666

         ELSE IF(ANS.EQ.6) THEN
```

```
C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'1) Mean Filter'
           TYPE*,'2) Median Filter'
           ACCEPT*, ANS
           TYPE*
           TYPE*,'Enter the Spatial Filter Dimension, N ,(e.g. N=3
        1 for 3X3 filter mask)'
           TYPE*,'Note: N must be an odd Integer!'
           ACCEPT*, N
           IN = (N-1)/2
           IF(ANS.EQ.1) THEN
              TYPE*
              TYPE*,'Enter the desired Mean Filter Mask: 1=Mask1,
        1  2=Mask2,  3=Mask3'
              ACCEPT*, ANS

C       CONVOLVE THE IMAGE WITH EITHER LOWPASS MASK1, MASK2, OR
C       MASK3 DEPENDING ON THE RESPONSE OF THE USER TO THE ABOVE
C       REQUEST.
              DO J=1+IN,TYPE-IN
                DO I=1+IN,TYPE-IN
                   ISUM = 0
                   DO K=J-IN,J+IN
                      DO L=I-IN,I+IN
                        ISUM = ISUM + IMG(L,K)
                      END DO
                   END DO
                   IF(ANS.EQ.1) THEN
                      IMG1(I,J)=ISUM/(N*N)
                   ELSE IF(ANS.EQ.2) THEN
                      IMG1(I,J)=(ISUM+IMG(I,J))/(N*N+1)
                   ELSE
                      IMG1(I,J)=(ISUM+IMG(I,J-1)+IMG(I-1,J)+3*IMG(I,J)
        1 +IMG(I+1,J)+IMG(I,J+1))/(N*N+7)
                   END IF
                END DO
              END DO
           ELSE
C       CONVOLVE THE IMAGE WITH AN NXN MEDIAN MASK.
```

```
C       STORE THE TWO DIMENSIONAL NXN ARRAY IN A LINEAR VECTOR OF
C       LENGTH N**2.
              DO J=1,TYPE-IN
                DO I=1,TYPE-IN
                   K=1
                   M=1
15                 DO L=M,M+N-1
                      IM(L)=IMG(L-M+I,K+J-1)
                   END DO
                   M=M+N
                   K=K+1
                   IF(K.LE.N) GOTO 15

C       PERFORM A BUBBLE SORT ON THIS VECTOR, PLACING THE ELEMENTS OF
C       THE VECTOR IN ORDER OF INCREASING MAGNITUDE.
                   DO L=1,N*N/2+1
                      DO M=L+1,N*N
                        IF(IM(L).GT.IM(M)) THEN
                           ITMP = IM(L)
                           IM(L) = IM(M)
                           IM(M) = ITMP
                        END IF
                      END DO
                   END DO

C       THE MEDIAN OF THIS VECTOR, WHICH IS THE MEDIAN OF THE 2-D NXN
C       ARRAY, IS NOW THE N**2/2+1 ELEMENT OF THE VECTOR!
                   IMG1(I+IN,J+IN) = IM((N*N)/2+1)
                END DO
              END DO
           END IF

C       PLACE THE NEW IMAGE DATA, IMG1(I,J), INTO THE ORIGINAL IMAGE
C       DATA ARRAY, IMG(I,J)
           DO J=1+IN,TYPE-IN
              DO I=1+IN,TYPE-IN
                IMG(I,J) = IMG1(I,J)
              END DO
           END DO
           GOTO 10

C       777777777777777777777777777777777777777777777777777777777777777

        ELSE IF(ANS.EQ.7) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)
```

```
C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'Enter the desired High-Pass Filter Mask: 1=Mask1,
        1  2=Mask2,  3=Mask3,  4=Mask4'
           ACCEPT*, ANS
           IH1 = 1
           IW1 = 1
           IF(ANS.EQ.1) THEN
              TYPE*
           TYPE*,'Note: Window height and width must be an odd integer'
              TYPE*
              TYPE*,'Enter height of window, H:'
              ACCEPT*, IH
              TYPE*
              TYPE*,'Enter width of window, W:'
              ACCEPT*, IW
              IH1 = (IH-1)/2
              IW1 = (IW-1)/2
           END IF

C       CONVOLVE THE IMAGE WITH EITHER HIGHPASS MASK1, MASK2, MASK3,
C       OR MASK4 DEPNDING ON THE CHOICE OF THE USER.
           DO J=1+IW1,TYPE-IW1
              DO I=1+IH1,TYPE-IH1
                IF((ANS.EQ.1).OR.(ANS.EQ.2)) THEN
                   ISUM = 0
                   DO K=J-IW1,J+IW1
                      DO L=I-IH1,I+IH1
                        ISUM = ISUM + IMG(L,K)
                      END DO
                   END DO
                   IF(ANS.EQ.1) THEN
                      IMG1(I,J)=128+IMG(I,J)-ISUM/(IH*IW)
                   ELSE
                      IMG1(I,J)=10*IMG(I,J)-ISUM
                   END IF
                ELSE IF(ANS.EQ.3) THEN
                   IMG1(I,J)=-IMG(I,J-1)-IMG(I-1,J)+5*IMG(I,J)
        1 -IMG(I+1,J)-IMG(I,J+1)
                ELSE IF(ANS.EQ.4) THEN
                   IMG1(I,J)=IMG(I-1,J-1)-2*IMG(I,J-1)+IMG(I+1,J-1)
        1 -2*IMG(I-1,J)+5*IMG(I,J)-2*IMG(I+1,J)+IMG(I-1,J+1)
        2 -2*IMG(I,J+1)+IMG(I+1,J+1)
                END IF
              END DO
           END DO
```

```
C       PLACE THE NEW IMAGE DATA, IMG1(I,J), INTO THE ORIGINAL IMAGE
C       DATA ARRAY, IMG(I,J)
           DO J=1+IW1,TYPE-IW1
              DO I=1+IH1,TYPE-IH1
                IF(IMG1(I,J).LE.0) THEN
                   IMG(I,J) = 0
                ELSE IF(IMG1(I,J).GE.255) THEN
                   IMG(I,J) = 255
                ELSE
                   IMG(I,J) = IMG1(I,J)
                END IF
              END DO
           END DO
           GOTO 10

C       8888888888888888888888888888888888888888888888888888888888888888

        ELSE IF(ANS.EQ.8) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'Enter the desired gradient implementation:  1=implmt1,
        1  2=implmt2,  3=implmt3'
           ACCEPT*, ANS
           TYPE*
           TYPE*,'Enter the desired threshold level, T:'
           ACCEPT*, IT
           TYPE*
           TYPE*,'Enter the background and gradient intensity levels,
        1  (ILB,ILG):'
           ACCEPT*, ILB,ILG

C       COMPUTE THE MAGNITUDE OF THE GRADIENT
           DO J=1,TYPE-1
              DO I=1,TYPE-1
        IMG1(I,J)=JIABS(IMG(I,J)-IMG(I+1,J))+JIABS(IMG(I,J)-IMG(I,J+1))
              END DO
           END DO

           DO J=1,TYPE-1
              DO I=1,TYPE-1
                IF(ANS.EQ.1) THEN

C       SET THE EDGES TO A SPECIFIED GRAY LEVEL, ILG, WITHOUT
C       DESTROYING THE SMOOTH BAKGROUND.
                   IF(IMG1(I,J).GE.IT) THEN
                      IMG(I,J) = ILG
                   END IF
                ELSE IF(ANS.EQ.2) THEN
```

```
C       SET THE BACKROUND TO A SPECIFIED GRAY LEVEL, ILB, AND SET THE
C       EDGES TO THE GRADIENT, IF THE GRADIENT IS ABOVE A SPECIFIED
C       THRESHOLD VALUE.
                    IF(IMG1(I,J).GE.IT) THEN
                       IMG(I,J) = IMG1(I,J)
                    ELSE
                       IMG(I,J) = ILB
                    END IF
                 ELSE

C       SET THE EDGES TO A SPECIFIED GRAY LEVEL, ILG, IF THE GRADIENT
C       IS ABOVE A SPECIFIED THRESHOLD LEVEL. SET THE BACKGROUND TO A
C       SPECIFIED GRAY LEVEL, ILB, IF THE GRADIENT IS BELOW THE
C       THRESHOLD LEVEL.
                    IF(IMG1(I,J).GE.IT) THEN
                       IMG(I,J) = ILG
                    ELSE
                       IMG(I,J) = ILB
                    END IF
                 END IF
              END DO
           END DO
           GOTO 10

C       9999999999999999999999999999999999999999999999999999999999999999

        ELSE IF(ANS.EQ.9) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
           TYPE*,'1) Automated Contrast Enhancement'
           TYPE*,'2) Manual Contrast Enhancement'
           ACCEPT*, ANS
           IF(ANS.EQ.1) THEN

C       IF AUTOMATED CONTRAST ENHANCEMENT HAS BEEN CHOSEN, THEN FLAG
C       THE USER THAT THE IMAGE HISTOGRAM MUST HAVE BEEN PREVIOUSLY
C       CALCULATED IN ADDITION, REQUEST ADDITIONAL INPUT DATA FROM THE
C       USER.
              TYPE*
              TYPE*,'Note: The Image Histogram must be calculated
        1 before Automated Contrast'
              TYPE*,'Enhancement can be performed'
              TYPE*
              TYPE*,'If the Image Histogram has not been caculated,
        1 type, NO <RTN>'
              ACCEPT 1000, FILENAME
              IF(FILENAME.EQ.'NO') GOTO 10
```

```
      TYPE*
      TYPE*,'Enter relative percentage of pixels below and
     1 above a certain intensity level'
      TYPE*,' (PL,PH)
      ACCEPT*, PL,PH
      TYPE*

C     CALCULATE THE LOWER INTENSITY LEVEL THRESHOLD VALUE SUCH THAT
C     A SPECIFIED PERCENTAGE, PL, OF PIXEL INTENSITY LEVELS "FALL"
C     BELOW THIS LOWER THRESHOLD VALUE.
      I = 1
      C = 0
20    P(I) = 100*HISTO(I)/(TYPE*TYPE)
      DO K=1,I
        C = C + P(K)
      END DO
      IF(C.LT.PL) THEN
        I = I + 1
        GOTO 20
      END IF
      IBL = I - 1

C     PRINT THE CALCULATED LOWER INTENSITY LEVEL THRESHOLD VALUE
      TYPE*,'BL = ',IBL

C     CALCULATE THE UPPER INTENSITY LEVEL THRESHOLD VALUE SUCH THAT
C     A SPECIFIED PERCENTAGE, PH, OF PIXEL INTENSITY LEVELS "FALL"
C     ABOVE THIS UPPER THRESHOLD VALUE.
      I = 256
      C = 0
30    P(I) = 100*HISTO(I)/(TYPE*TYPE)
      DO K=256,I,-1
        C = C + P(K)
      END DO
      IF(C.LE.PH) THEN
        I = I - 1
        GOTO 30
      END IF
      IBH = I - 1

C     PRINT THE CALCULATED UPPER INTENSITY LEVEL THRESHOLD VALUE
      TYPE*,'BH = ',IBH
    ELSE
```

```
C        IF THE USER HAS CHOSEN MANUAL CONTRAST ENHANCEMENT, REQUEST
C        FROM THE USER BOTH THE LOWER AND UPPER INTENSITY LEVEL
C        THRHESHOLD VALUES.
               TYPE*
      TYPE*,'Enter low,high intensity level threshold values  (BL,BH)'
               ACCEPT*, IBL,IBH
               TYPE*
               TYPE*,'BL = ',IBL
               TYPE*,'BH = ',IBH
            END IF

C        ENHANCE THE CONTRAST OF THE IMAGE BY PERFORMING A LINEAR
C        STRETCH OF THE HISTOGRAM.
            DO J=1,TYPE
               DO I=1,TYPE
                 IF(IMG(I,J).LE.IBL) THEN
                    IMG(I,J) = 0
                 ELSE IF(IMG(I,J).GE.IBH) THEN
                    IMG(I,J) = 255
                 ELSE
                    IMG(I,J) = 255*(IMG(I,J)-IBL)/(IBH-IBL)
                 END IF
               END DO
            END DO
            GOTO 10

C        10101010101010101010101010101010101010101010101010101010101010

         ELSE IF(ANS.EQ.10) THEN

C        CLEAR THE MAIN MENU SCREEN
            WRITE(6,*) (CHAR(12),I=1,2)

C        REQUEST INPUT DATA FROM THE USER
            TYPE*,'Enter the gray level reduction factor, N:'
            TYPE*
            TYPE*,'Note: N must be a power of 2'
            TYPE*
            TYPE*,'(e.g. N=8 implies 32, (256/8), gray levels)'
            ACCEPT*, N

C        REDUCE THE NUMBER OF GRAY LEVELS TO 256/N, WHERE N IS THE
C        GRAY LEVEL REDUCTION FACTOR SPECIFIED ABOVE
            DO J=1,TYPE
               DO I=1,TYPE
                 IMG(I,J) = N*(IMG(I,J)/N) + N/2
               END DO
            END DO
            GOTO 10
```

```
C       1111111111111111111111111111111111111111111111111111111111

        ELSE IF(ANS.EQ.11) THEN

C       INVERT THE IMAGE GRAY SCALE. i.e. WHITE BECOMES BLACK; BLACK
C       BECOMES WHITE, etc.
           DO J=1,TYPE
              DO I=1,TYPE
                IMG(I,J) = 255 - IMG(I,J)
              END DO
           END DO
           GOTO 10

C       12121212121212121212121212121212121212121212121212121212121212

        ELSE IF(ANS.EQ.12) THEN

C       CLEAR THE MAIN MENU SCREEN
           WRITE(6,*) (CHAR(12),I=1,2)

C       REQUEST INPUT DATA FROM THE USER
    TYPE*,'Enter  upper black,lower white,upper white  intensity level
        1 threshold values:'
           TYPE*,'(UB,LW,UW)'
           ACCEPT*, IUB,ILW,IUW

C       REPLACE ANY IMAGE INTENSITY LEVEL BELOW UB BY 0, (BLACK).
C       REPLACE ANY IMAGE INTENSITY LEVEL BETWEEN LW AND UW BY 255.
C       REPLACE ANY IMAGE INTENSITY LEVEL ABOVE UW BY 0, (BLACK).
C       IMAGE INTENSITY LEVELS ARE NOT AFFFECTED IF THE IMAGE INTENSITY
C       LEVELS ARE NOT IN THE ABOVE RANGES. NOTE: GIVEN THIS
C       THRESHOLDING REGIME, IT IS POSSIBLE TO PRODUCE A BINARY IMAGE.
           DO J=1,TYPE
              DO I=1,TYPE
                IF((IMG(I,J).LE.IUB).OR.(IMG(I,J).GT.IUW)) THEN
                   IMG(I,J) = 0
                ELSE IF((IMG(I,J).GE.ILW).OR.(IMG(I,J).LE.IUW)) THEN
                   IMG(I,J) = 255
                END IF
              END DO
           END DO
           GOTO 10

C       13131313131313131313131313131313131313131313131313131313131313

        ELSE IF(ANS.EQ.13) THEN
```

```
C       NOTE: COLOR IMAGES ARE PRODUCED BY 24 BIT DATA WITH THE FIRST 8
C       BITS REPESENTING THE RED INTENSITY DATA, BITS 8-15 REPRESENTING
C       THE GREEN INTENSITY DATA, AND BITS 16-23 REPRESENTING THE BLUE
C       INTENSITY DATA THE PSUEDO-COLOR MAPPING BELOW IS REFERRED TO AS
C       INTENSITY SLICING.
C       e.g.    000 (BINARY)  ----------->  00/00/00  (HEXIDECIMAL)
C               001 (  "   )  ----------->  00/00/FF  (     "     )
C               010 (  "   )  ----------->  00/FF/00  (     "     )
C       etc.
           DO J=1,TYPE
              DO I=1,TYPE
                IMG(I,J) = IMG(I,J)/32
                ITMP = 0
                DO K=2,0,-1
                   COND = BJTEST(IMG(I,J),K)
                   IF(COND) THEN
                      ITMP = ITMP*256 + 255
                   ELSE
                      ITMP = ITMP*256
                   END IF
                END DO
                IMG(I,J) = ITMP
              END DO
           END DO
           GOTO 10

C       END END END END END END END END END END END END END END END

        END IF

1000    FORMAT(A)

2000    CALL EXIT
        END
```

LIST OF REFERENCES

Adage Inc. RDS-3000 Programming Reference Manual. Billerica, MA: Adage Inc., 1982.

Andrews, H. C. "An Educational Digital Image Processing Facility." Computer, August 1977, pp. 48-53.

Andrews, H. C. "Monochrome Digital Image Enhancement." Applied Optics 15 (February 1976): 495-503.

Digital Equipment Corporation. VAX Architecture Handbook. Bedford, MA: Digital Press, 1981.

Digital Equipment Corporation. VAX Hardware Handbook. Bedford, MA: Digital Press, 1982.

Digital Equipment Corporation. VAX Software Handbook. Bedford, MA: Digital Press, 1982.

Gonzalez, Rafael C., and Wintz, Paul. Digital Image Processing. Reading, MA: Addison-Wesley Advanced Book Program/World Science Division, 1977.

Green, William B. Digital Image Processing: A Systems Approach. New York: Van Nostrand Reinhold Electrical/Computer Science and Engineering Series, 1983.

Oppenheim, Alan V., and Schafer, Ronald W. Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.

Pratt, William K. Digital Image Processing. New York: John Wiley and Sons, 1978.

Rabiner, Lawrence R., and Gold, Bernard. Theory and Application of Digital Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1975.