Bureau of Mineral Resources, Geology & Geophysics

R E C O R D

RECORD 1991/62

Distributed Image Processing Using the RTI-CAD PC Software Package

Prame N. Chopra
Information Systems Branch

1991/62
c.4

RECORD 1991/62

Distributed Image Processing Using the RTI-CAD PC Software Package

Prame N. Chopra
Information Systems Branch

Table of Contents

ABSTRACT

There is a strong case for the development of a PC-based low-end image processing capability in BMR such as could be provided by RTI-CAD. This distributed image processing capability would be **an adjunct** to the existing state of the art image processing capabilities of the Image Processing Centre (IPC). Such a system would help in particular to cater for low-level and novice users of image processing in BMR. In this way these users would be saved from undertaking more training than they need. The system would also provide a logical path for such users to progress in time to the power (and complexity) of the I²S system. The use of RTI-CAD by low level users would also mean that specialist users could be guaranteed more access to the four available I²S workstations.

RTI-CAD is a powerful PC image processing package which offers an easy to use mouse-driven interface and a very comprehensive vector overlay capability. It provides excellent support for a range of colour and monochrome PC printers and includes an extremely powerful false sun angle capability which is much better than that provided by the I²S system 600 software. On the negative side, RTI-CAD is hampered by the generally poor performance of PC graphics cards and as a result it is only capable of displaying 8-bits of information rather than the 24-bits used by expensive dedicated image processing systems such as I²S. Some degradation in spatial resolution results when RTI-CAD displays two or more bands of an image simultaneously, but in many instances this doesn't limit RTI-CAD's utility.

The cost of the RTI-CAD image processing package when used on an existing 80286, 80386 or 80486 PC which already includes a suitable monitor, maths co-processor and ethernet connections as most BMR PCs will soon do, would be $2630. This figure compares very favourably with the cost of purchasing an additional IVAS workstation for the IPC of around $36000. Thus the use of distributed image processing on existing BMR PCs is very cost effective when compared with further investment in proprietary I²S workstations.

An easy to use routine, Getimage.bat, has been developed to allow users to transparently down-load images from the I²S system to a PC. The routine handles all the necessary image manipulation and file conversions for the user which makes it possible for novice users to gain access to I²S images without lengthy training.

As usage of the IPC by BMR staff becomes more extensive and access to the existing I²S workstations becomes more keenly sought, there will be a considerable cost advantage in BMR developing a low-end image processing system such as could be provided by RTI-CAD.

INTRODUCTION

BMR has acquired a state of the art image processing system for a total outlay in the vicinity of $1,000,000. This system is based on I^2S hardware and software and comprises 2 SUN 4/280 mini-computers with a total of 5 Gbyte of disc storage, 3 IVAS image workstations, 1 Model 75 workstation, a BITE array processor, and associated peripherals including tape drives, a digitiser, video camera and film recorder. The vendor has provided a complex image processing software system implemented from more than of 500,000 lines of source code. Much of this source code is available on the system and expert users are able to modify modules and recompile and link them for specialist applications.

Because of the necessary complexity of a state of the art image processing system, users must undertake an extensive training course before they are able to fully exploit the capabilities of the equipment. The vendor, I^2S, normally allots 2 weeks for general training of users of the system and a further 2 weeks for those users who need to program non- standard image processing procedures. Thus the very extensive facilities offered by the image processing system are not without their cost in terms of training.

Present and Future Usage of Image Processing in BMR

Many BMR scientists will ultimately require image processing facilities in the normal course of their work. This will become increasingly so in the next decade when very sophisticated satellite systems such as the American Earth Observing Satellite (EOS) system are deployed. Notwithstanding this expected long-term growth in the usage of the Image Processing Centre (IPC), there is already very considerable use of the IPC by a number of groups in BMR support of the National Geoscience Mapping Accord (NGMA).

Given the expected growth in usage of the IPC, problems with access to the 4 image processing workstations are likely to develop in the near future. It is therefore prudent to consider now how these workstations in particular, and image processing in general, should be managed.

In broad terms there are always likely to be two types of image processing carried out in BMR. Firstly, there will be a core group of specialist users who will require many of the advanced features offered by the I^2S equipment. Secondly, there will be a much larger group of scientists who need "look and see" image processing capabilities. The latter group will not need much of the functionality provided by the I^2S system and will often be sporadic in their use of image processing. As a result many of these users will inevitably stay at or near the bottom of the learning curve and will remain image processing novices.

Considerable advantages in terms of the efficient use of the image processing facilities can be gained if both the types of users outlined above are specifically catered for. As will be detailed below, the image processing requirements of many

low-level users could be adequately met with PC-based image processing on an efficient Local Area Network (LAN) connected to the IPC.

Low-level users would benefit from the development of such an integrated PC-based system in a number of ways: Firstly, simple image processing procedures such as pseudo-colouring, scaling and sun-angle manipulations would be almost immediately accessible with very little training. Secondly, time constraints on the use of the equipment, such as could be expected to apply in the IPC when usage becomes heavy, will not be a problem for PC users who may, at least initially, be slow and relatively unproductive. Thirdly, learning of image processing would be carried out on a computer with which many/most of the users were already familiar. Thus for example, there would be no need for users to also have to master a new operating system (UNIX).

Specialist users of the IPC would also benefit from the development of an integrated PC-based image processing system. Firstly, such a system would allow these users to store images produced by complex processing in the IPC on PC hard discs and streaming tapes. Thus these users would be able to readily store their work outside the IPC. These storage capabilities could reduce or at least delay the purchase of additional disc storage in the IPC. Secondly, the 4 workstations in the IPC would be more readily available for use by specialist users. This factor would probably reduce the need for additional expensive I^2S workstations to be purchased.

Finally, it must be emphasised that the development of a PC-based image processing system would only be tenable **as an adjunct to the existing equipment in the IPC.** There are a number of essential functions which only the IPC can provide to corporate image processing. Firstly, the sheer size of remotely sensed datasets requires very large disc storage capacities and tapedrives for data entry and archival. These features are rare in the PC environment. Secondly, remotely sensed data is only really useful if it can be warped into map projections to allow overplotting with ground-based and other data such as airborne geophysics. This warping capability is not provided by RTI-CAD and must be done in the IPC.

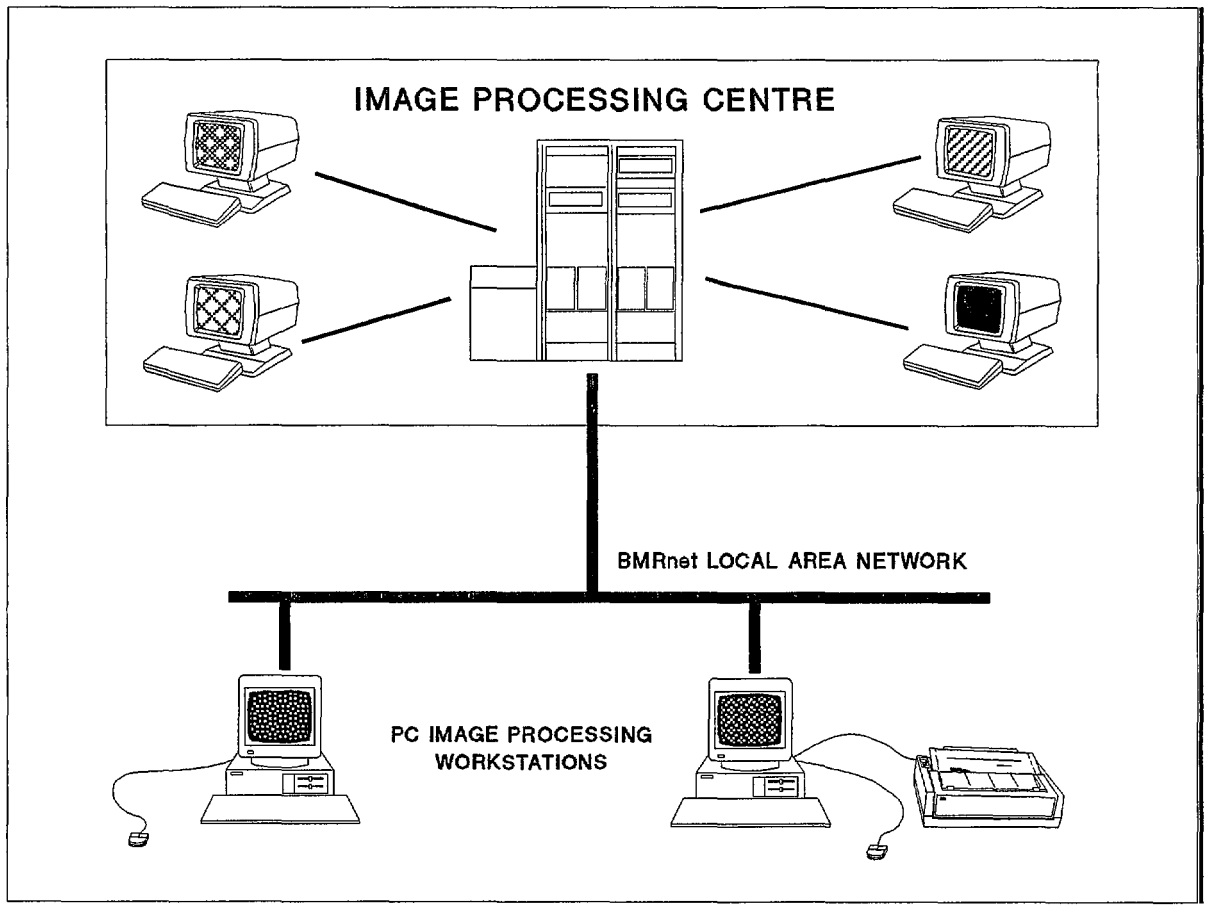
A Proposed Operating Scheme for BMR Image Processing

There is therefore a strong case to be made for the development of so-called low-end image processing workstations in BMR. These workstations would necessarily need to be physically connected to the SUN 4/280 computers in the IPC so that imagery could be down-loaded, but could be remotely located in user areas within the building. Physical connection would be provided by the Ethernet LAN. These low-end workstations would have local processing capabilities so that they would not require the processing capabilities of the SUN's for simple image manipulations. In this way the performance of the IPC is likely to be little affected and image refresh rates on the low-end workstations would not be limited by data transfer rates on the LAN. More complex processing could be carried out by remotely accessing the CPU functions of the I^2S System 600 software running on the SUN's and then downloading the processed imagery to the low-end workstations.

A schematic diagram of such an integrated image processing system is illustrated in Figure 1. For simplicity, the graphics workstations in the IPC and the peripheral devices (e.g. the digitiser) have not been shown in this diagram. Also not shown explicitly in Figure 1 are the essential network bridges to allow connection of the IPC's SUN computers to the Ethernet LAN and the connections between this LAN and the DG MV20000 and the VAX computers. The sort of PC connectivity illustrated obviously also requires Ethernet cards and TCP/IP software for each PC on the LAN.

Figure 1

A schematic diagram of distributed image processing



Required Hardware

Some limitations on the type of PC hardware that can be used are imposed by the need to effectively interface the PC with the I^2S equipment in the IPC. For example, the majority of image processing carried out in the IPC is likely to involve displays of 3 bands of 8-bit imagery (=24-bits) and this will typically involve image display windows with a minimum size of 512 x 512 pixels. This size represents the maximum display capability of the Model 75 workstation while that of the IVAS workstations is 1024 x 1024 pixels. (Both types of workstation are of course capable of manipulating much larger images than these).

It is possible to buy 24-bit colour PC display adaptors at the present time, but these are quite expensive (several thousand dollars each). Image processing systems on PCs therefore currently concentrate on the use of 8-bit displays. Various schemes are employed by PC software designers in order to try to cram 24-bits of information into the 8-bits supported by these displays and these strategies have varying degrees of success. In many cases the compromises inherent in these strategies still allow very useful image processing and display work to be done on the PC. Thus a PC display adaptor and monitor with a 256 colour capability at a resolution of at least 512 x 512 can, in many cases, be used to effectively down-load imagery from an image processing facility such as the IPC. The 800 x 600 extended VGA PC graphics mode with 256 colours is particularly suitable for this purpose. This display mode is available from a number of manufacturers but, as there is no agreed industry standard as yet for the super VGA modes, each manufacturer employs their own proprietary standard. One such manufacturer is the Canadian company ATI who sell a 16-bit data path super VGA card known as a VGA Wonder card.

The ATI VGA Wonder card with 512 Kbyte of memory is required for the RTI-CAD software discussed below, as is a maths co-processor, a minimum of 640 Kbyte of RAM and a multisync analog colour monitor. Approximately 1 MByte of disc storage is required for each 800x600 single band image.

Required Software

One IBM PC-based image processing software package which supports the extended VGA mode with 256 colours is RTI-CAD. This package is produced by GEOPAK Systems in Canada and is marketed in Australia by ENCOM Technologies in Sydney. The package includes the necessary software and, optionally, a VGA Wonder card which has to be installed in the PC to be used for the image processing.

In addition to the image processing software, some software is required to link the PC to the IPC over the BMR backbone ethernet. The preferred software package for this link is SUN Microsystems' PC-NFS, though any TCP/IP (Transmission Control Protocol/ Internet Protocol) compliant package which provides rcp and rsh services would be usable. The way in which these "r services" are used in the downloading of imagery from the IPC is illustrated in getimage.bat - a MS-DOS batch file used to control the link between the IPC and the PC (see Appendix A for a listing of this routine).

Relative Costs

The IPC includes 4 image processing workstations. Three of these workstations are IVAS (Image Viewing and Analysis Systems) workstations and the other, a more expensive extensively optioned Model 75 workstation. Additional IVAS workstations and their associated dedicated electronics can be installed for circa $36,000 per unit on current prices. In total, 4 more IVAS workstations can be added to the IPC under the existing software licenses (each SUN computer can support upto 4 workstations).

The cost of the PC-based RTI-CAD image processing package when used on an existing 80286, 80386 or 80486 PC which already includes a suitable monitor and a maths co-processor, as most BMR PCs do, is detailed in Table 1.

From Table 1 it can be seen that the cost of developing a low-end image processing capability on an existing PC which already has a suitable monitor, maths co-processor and hard disc would be $3950. This figure includes the cost of ethernet connectivity for the individual PC. In practice however, most BMR PCs will soon be provided with ethernet connections as part of the corporate information system irrespective of whether they will be used for image processing or not. Thus a more realistic figure for the installation cost of a PC image processing system using RTI-CAD would not include ethernet associated costs and would therefore fall to $2630.

As discussed above, the cost of purchasing an additional IVAS workstation for the IPC (with its much greater functionality) in order to cope with increased demand for image processing in BMR would be approximately 7 times this amount (i.e. $36,000).

It is clear from these cost comparisons that the use of distributed image processing on existing BMR PCs is very cost effective when compared with further investment in proprietary I^2S workstations. One proviso on this conclusion is that the level of functionality provided by the PC package must be sufficiently good that the software is useful to BMR geoscientists.

Table 1

Cost of an RTI-CAD PC Image Processing System

Item	Estimated Cost
ATI VGA Wonder card	$590.
RTI-CAD	$2040.*
PC-NFS	$610.
Etherlink II (3C503 ethernet card)	$350 (est.)
TOTAL	$3590.

* this figure is for the 4th and all subsequent licences of RTI-CAD purchased by BMR. BMR currently has 2 licences and the 3rd licence will cost $2720.

SYSTEM INSTALLATION

The installation of the RTI-CAD system on the PC for use as a distributed image processing workstation involves 3 steps: installation of the video card, the image processing software and the communications software. The user manuals for each of these products describe the installation steps to follow so only a *precis* is presented here.

VGA Card

The ATI VGA Wonder card must be installed in the PC that is going to be used to run RTI-CAD and it must be suitably configured using the VDRIVER and VSETUP utility programs provided by ATI. The steps to follow are discussed in the ATI manual. The user runs VDRIVER and is then presented with a menu-driven installation routine. This program's interface is illustrated in Figure 2. VDRIVER installs the VGA Wonder system utilities and the application drivers chosen by the user.

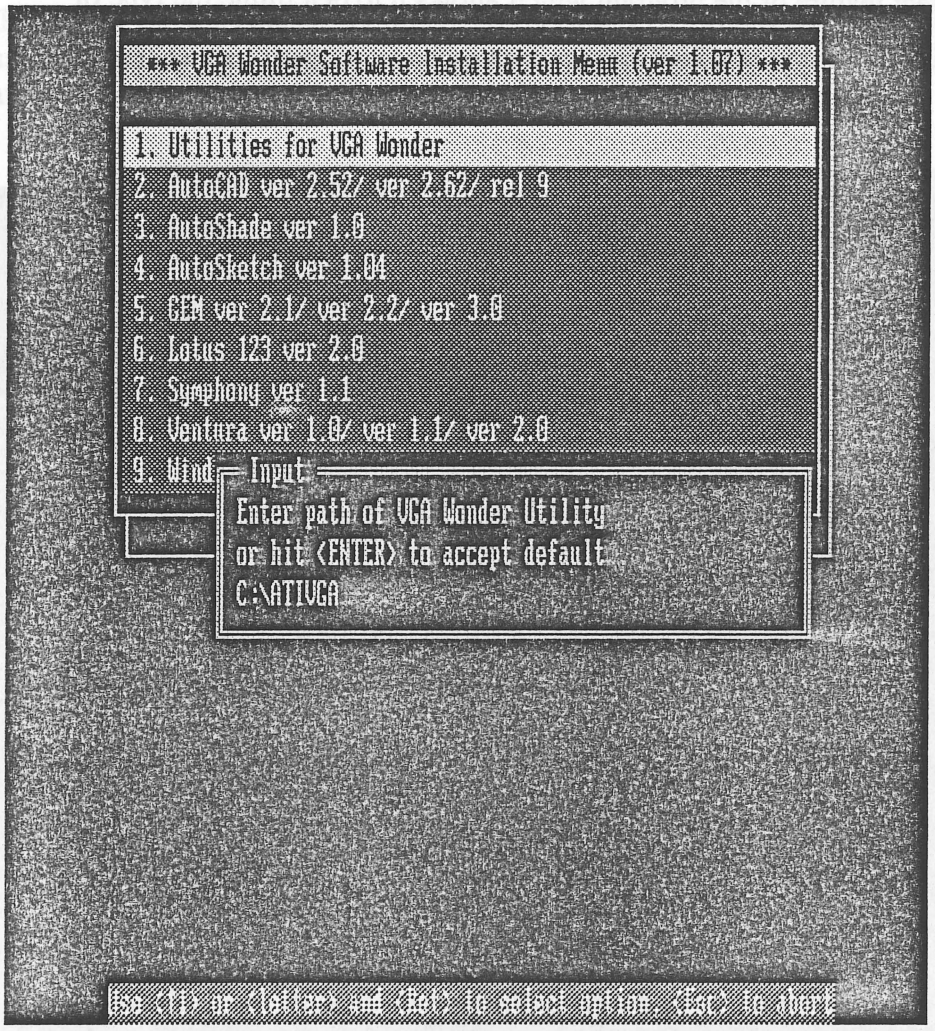


Figure 2
The VDRIVER screen.

The VDRIVER program installs the ATI utilities including VSETUP in a directory on the hard disc; usually \ATIVGA. It also installs the chosen applications drivers in this directory.

A user may modify the configuration settings of the ATI VGA Wonder card at any time by using the VSETUP program. This program is also menu driven and is very easy to use. Examples of the use of VSETUP are given in Figures 3 and 4.

Figure 3 illustrates the monitor selection procedure and, at the bottom of the figure, the auto-detect feature of the program which allows it to determine the type of monitor that is connected to the video card.

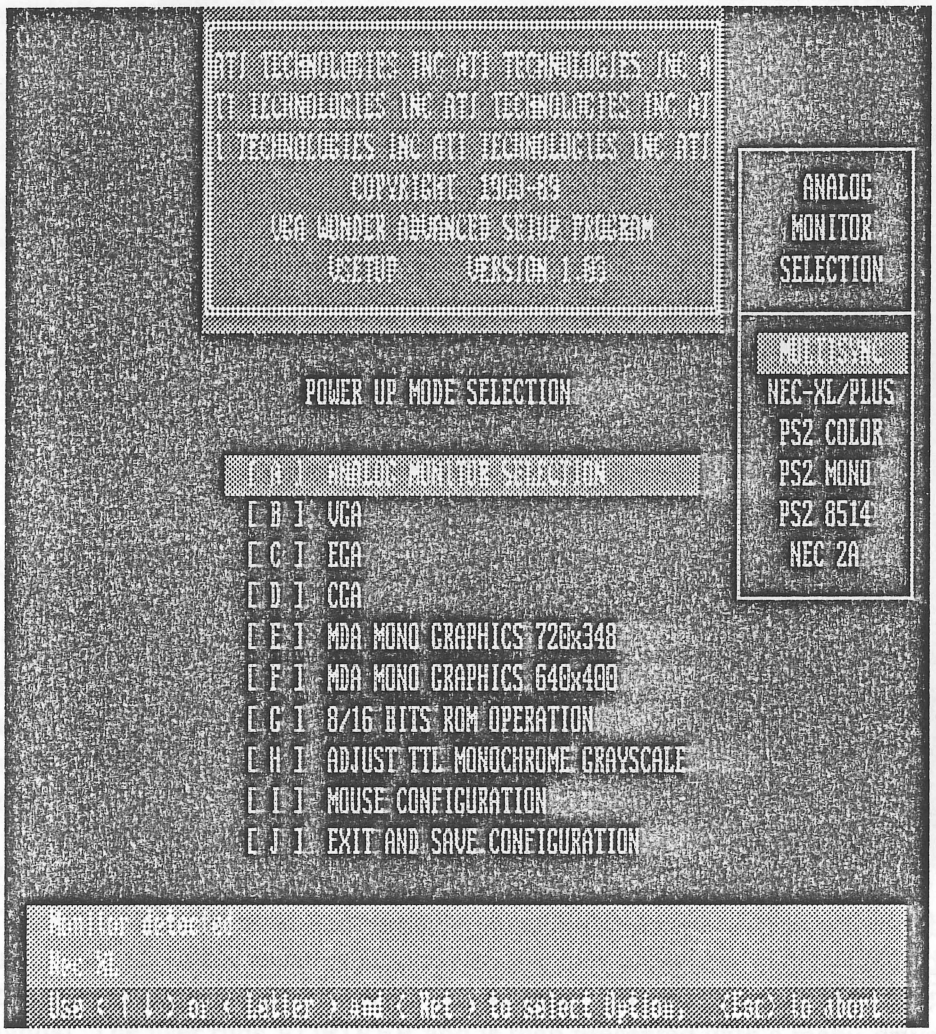


Figure 3
The VSETUP screen for monitor selection

Figure 4 illustrates the procedure to be followed to configure the bus mouse installed on the video card. In this case care must be taken in order to avoid an interrupt conflict with other devices already part of the PC system.

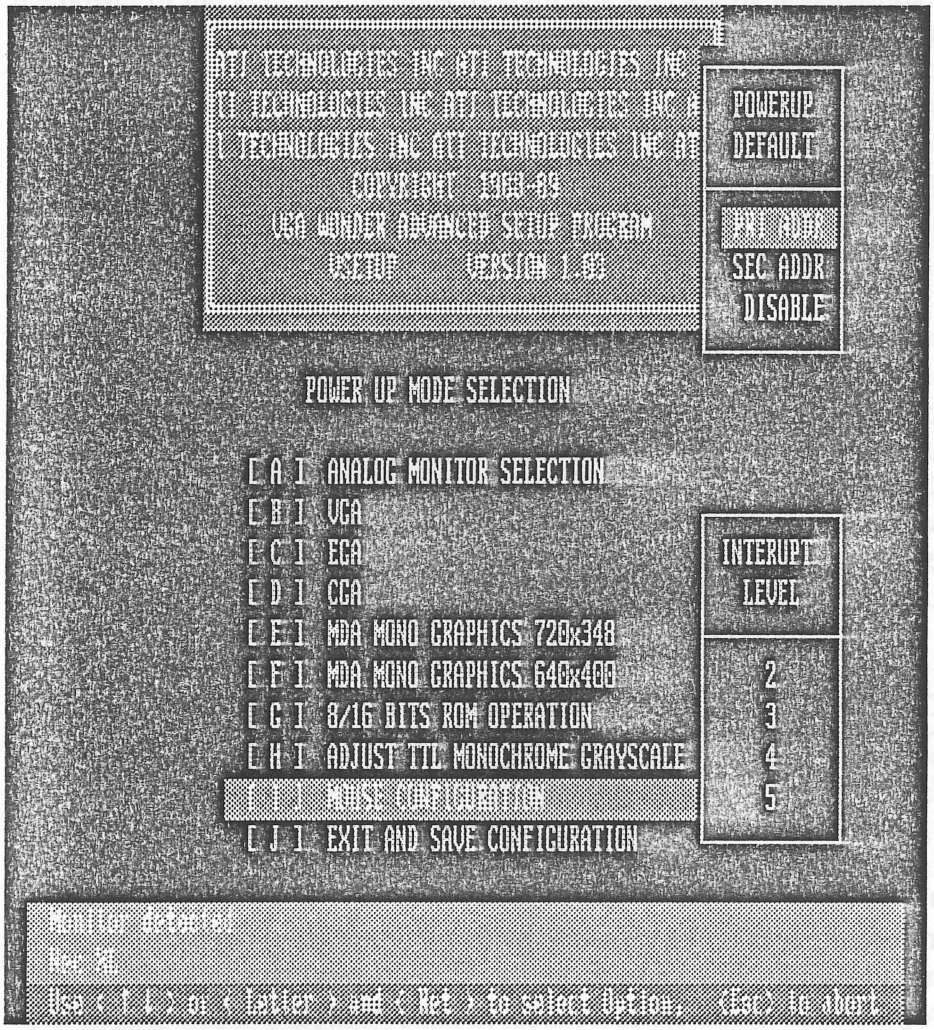


Figure 4
VSETUP screen to configure the integral bus mouse

RTI-CAD Software

The RTI-CAD software is installed on the designated hard disc of the PC by running a program on disc 1 of the set supplied by GEOPAK. The manual refers to an installation program called RSETUP, but this is an error - the actual name of the program is SETUP. This program creates the necessary directories on the hard disc for RTI-CAD, these are illustrated in Figure 5.

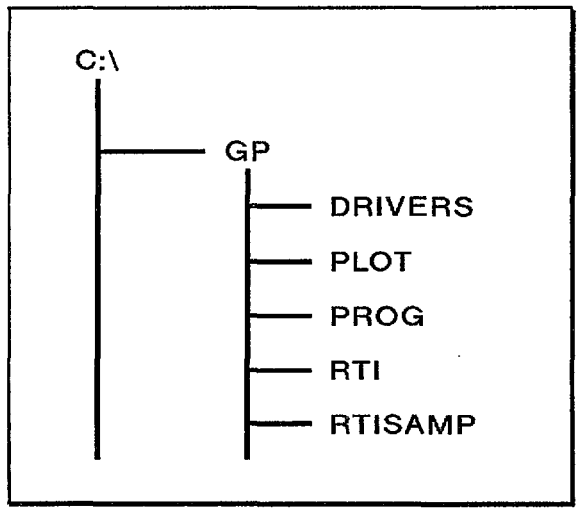


Figure 5
The directory structure for RTI-CAD

Once the RTI-CAD files have been copied to the appropriate directories on the hard disc, the AUTOEXEC.BAT and CONFIG.SYS files must be modified as detailed in the RTI-CAD manual. The software is then ready to run.

PC-NFS Software

In order to transfer images to the PC from the Image Processing Centre it is necessary to install the PC-NFS package on the PC. SUN Microsystems provide an installation program called INSTALL. In order to run this program, it is necessary to know the answers to a number of technical questions that it poses. These questions need to be answered by a system administrator and include such things as the type of ethernet card being used, the internet address assigned to the PC, the internet addresses of the 2 SUN 4/280 computers in the IPC as well as those of any other hosts that are to be connected. Details are also needed of the directory permissions on the hosts, the identity of the PC-NFS server, and a number of even more technical aspects.

Users are recommended to contact Information Systems Branch, or their computer network manager for help with the installation of PC-NFS.

ACCESSING IMAGES

Images resident on the SUN 4/280 computers in the BMR image processing centre can be accessed by two methods. In the first method, images may be processed using the power of the I²S cpu functions by remotely logging on to one of the IPC computers. This remote logon is accomplished with the TELNET facility of TCP/IP supported by the PC-NFS package. Though images may be processed through this remote session capability, the resulting images cannot be viewed using the I²S system without the user subsequently going to the BMR IPC and opening a new session on one of the workstations there. In the second method of remotely accessing images in the IPC, a user can down-load an I²S image to the PC for viewing and further image processing.

Telnet

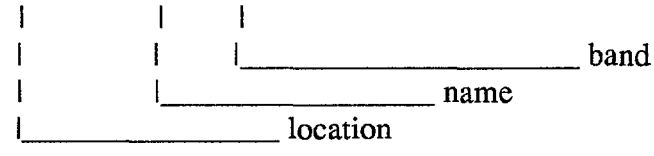
A user may initiate an I²S session on one of the two IPC SUN 4/280 computers by running a TELNET session from the MS-DOS command prompt. To do this the user must have firstly set up PC-NFS to recognize the IPC's computers as valid remote hosts. Once this has been done, all the user needs to do is to change to the PC-NFS directory (normally :\NFS) and type "telnet RETURN". Once the user chooses the appropriate IPC host computer, a normal UNIX logon screen is presented and, provided the user has logon privileges, an I²S session can be started in the normal way.

Down-loading an Image

The procedure for down-loading an image to a PC is automated for the user by an MS-DOS batch routine I have written. This routine, Getimage.bat, is listed in Appendix A. In order to use this routine to download an image to a PC, all the user needs to do is to issue the command:

getimage %1 %2 %3 %4 %5 %6 %7

where %1 is the name, band & location of the image to be down-loaded
(e.g. /scratch/my_image(1)



%2 is the number of samples to be down-loaded (i.e. x dimension)
%3 " " " " lines " " " (i.e. y dimension)
%4 is the location in the x dimension of the first pixel to down-load
%5 " " " " " y " " " " " " "
%6 is the type of image to be created [image or shadow -- see below]
%7 is the name of the PC image file to be created

The steps that Getimage.bat uses to down-load an image using are:

1) calculate the start and end coordinates of the image from the supplied start address of the first pixel and the specified ranges in x and y
2) build a control file which contains the input parameters for the I^2S commands to be executed on the SUN 4/280 computer in the IPC
3) convert the control file from MS-DOS format (lines terminated with CR/LF) to UNIX format (lines terminated only with LF).
4) send the control file to the SUN 4/280 using rcp
5) initiate a remote shell on the SUN 4/280 using rsh and the extract C Shell (see Appendix D). This procedure firstly inverts the image to convert it from the top-left origin used by I^2S to the lower-left origin used by RTI-CAD. The inverted image is then converted from the I^2S proprietary image format to a simple raster file.
6) copy the raster file to the PC using rcp
7) convert the raster file to a GEOPAK grid file using program IIS (see Appendix C)
8) convert the GEOPAK grid file to an image using the RTI-CAD GRIDPREP utility
9) run RTI-CAD for the user
10) delete intermediary files

The details of how Getimage.bat operates can be seen in the listing in Appendix A which includes comprehensive commenting of the routine.

IMAGE PROCESSING FUNCTIONALITY

RTI-CAD comes with a detailed 134 page reference manual and a 71 page tutorial so there is little point in trying to describe the full functionality of the package here. The reader is referred to the manual and tutorial for a complete description, I will only concentrate on what I see as the strengths and weaknesses of the package as they apply to RTI-CAD's use as a distributed image processing platform for BMR.

At the outset it needs to be pointed out that RTI-CAD is more than just an image processing package. RTI-CAD is an acronym which stands for **R**eal **T**ime **I**maging and **C**omputer **A**ided **D**rafting. The package has a very powerful vector drawing and overlay capability which makes it an ideal PC platform for the display, editing and plotting of scale maps of spatial data. RTI-CAD includes the ability to display data in layers and to use different data colours, line widths and font types. In these respects, RTI-CAD includes many of the capabilities of a geographical information system (GIS), though it lacks the all-important topological data structures that characterise a true GIS. RTI-CAD would make an excellent PC front-end to a GIS and could in this way provide distributed access to BMR GIS datasets in much the same way as has been suggested here for imagery.

RTI-CAD's use as an image processing platform is restricted by the limitations of the 8-bit PC graphics display card that it works with. RTI-CAD works best with single-band images where the image can be displayed truthfully on the 800x600 screen. Images may be stored at any size up to that limited by the hard disc's size and the user can zoom the 800x600 window in on any chosen subscene at any desired

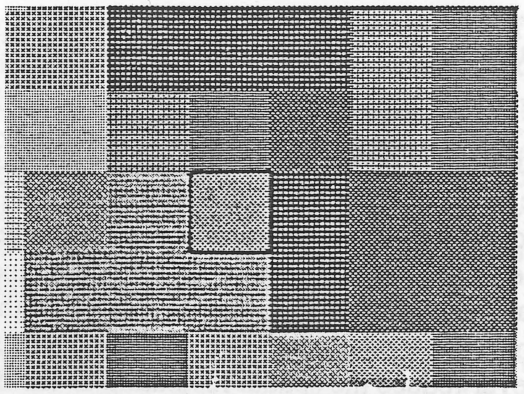
magnification. RTI-CAD's functionality in this respect is essentially similar to that of the I^2S IVAS workstations.

Multiple band images can also be displayed on the PC screen with RTI-CAD but in this case, some compromise must be made in order to display more than 8-bits of information on an 8-bit graphics card. RTI-CAD can display up to 4 separate images and up to 4 files of vector data on the screen simultaneously. It does so by increasing the pixel size of each band through an averaging process (see Figure 6). In this way the total number of pixels remains the same even though two or more bands are displayed. This averaging does however reduce the spatial resolution of the displayed image. The influence of this degradation on overall image quality and usefulness will depend upon the specifics of the data being displayed. Its effect is illustrated in Figure 7 which consists of prints from RTI-CAD of part of a LANDSAT TM scene over Canberra. The fuzziness displayed by the 3-band image is particularly apparent when compared with the print of band 4 alone. The effect is less marked when comparison is made between the 3-band image and band 7.

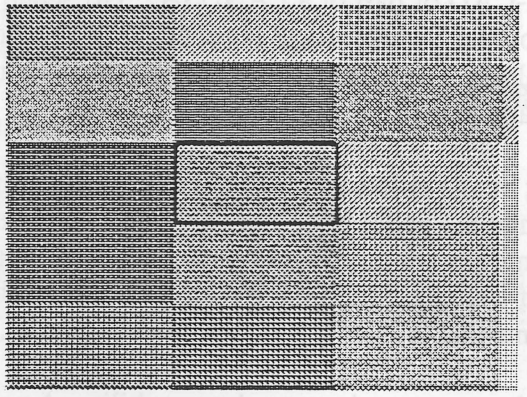
The principal means of modifying image colours in RTI-CAD is to use either a colour wheel or a colour bar to change the pallet of colours and the assignment of colour to digital number. Both techniques are thoroughly described in the manual and are easy to use. Pseudo-colouring of a single-band image with RTI-CAD is both easier to apply and more readily controllable than it is on the I^2S equipment.

RTI-CAD includes a powerful and easy to use image print capability. A print box can be placed anywhere on a screen image using the mouse and the enclosed scene can then be sent to the PC's printer. Output quality depends upon the type of printer attached to the PC, but high quality outputs can be obtained on thermal transfer printers and to a lesser extent on paintjet printers. The ability to readily produce good quality hard-copy is a very useful feature when doing image processing, yet this capability is not available in the IPC.

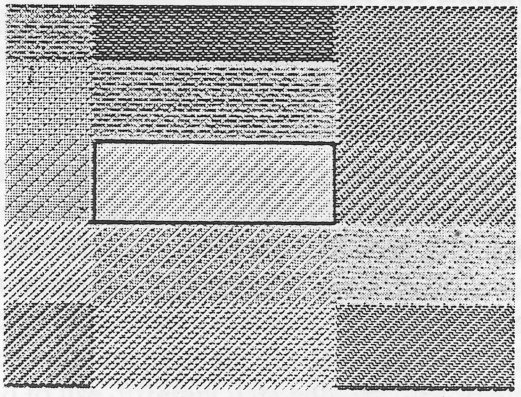
Perhaps the single most powerful image processing capability provided by RTI-CAD is the real time false sun angle processing that it provides. When images are initially created from grid files by the RTI-CAD GRIDPREP utility, the user may elect to produce either an image file or a shadow file. The latter is an image



1 band



2 bands



3 bands

Figure 6

The change in pixel size with number of bands (see text).

In each case the pixels are imaged at a zoom factor of 30. Representative pixels are outlined in each image.

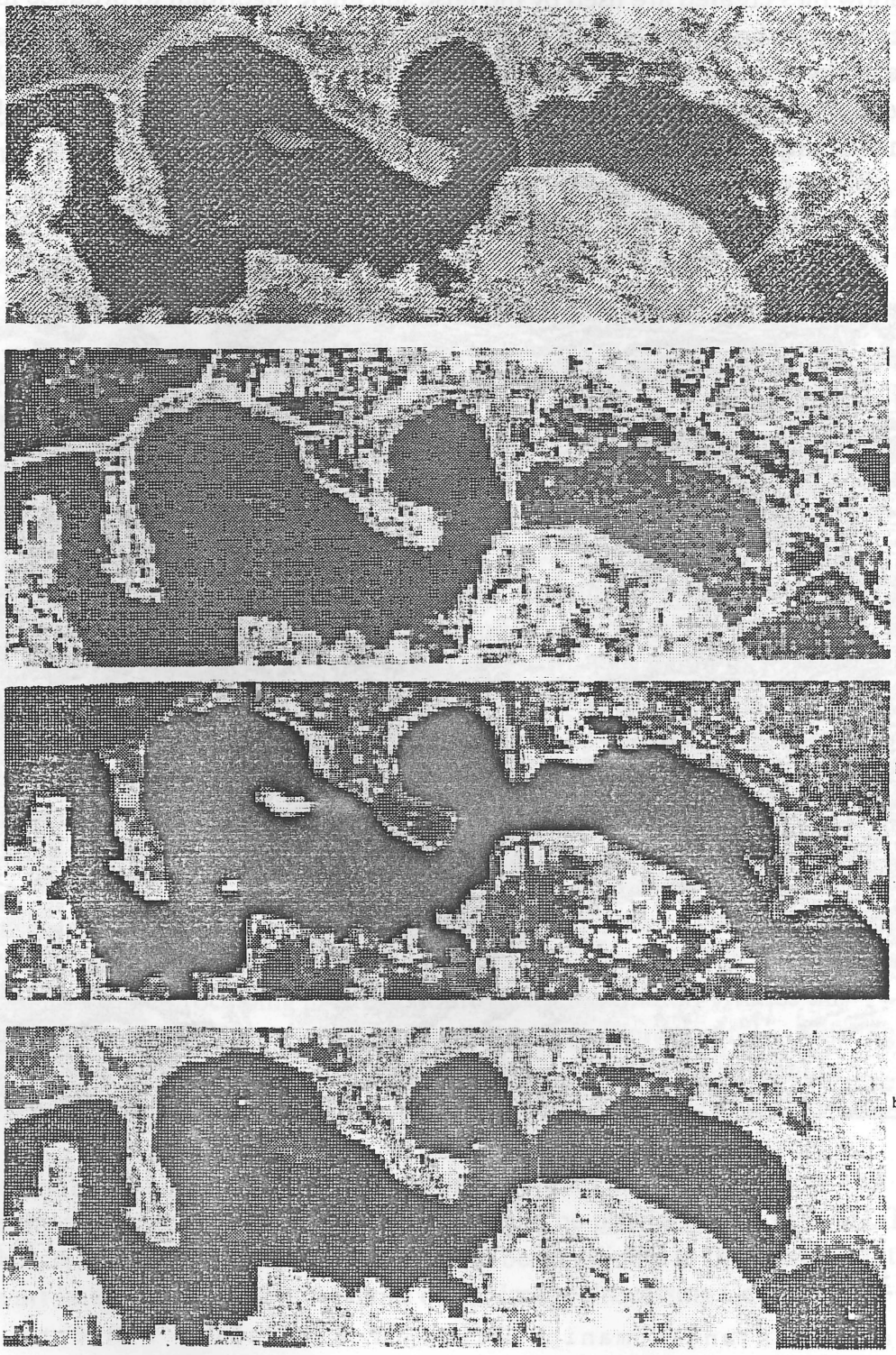


Figure 7

The effect of changing pixel size.
Four views of central Canberra from LANDSAT TM data (see text).

constructed from the gradients in the input grid rather than the actual values. The shadow files created are used by the false sun angle algorithm to produce images such as that shown in Figure 8. The inclination and declination of the false sun are altered by the user by simply moving the mouse and the whole process occurs interactively and in real time. This particular capability is far in advance of the functionality provided by the I²S equipment in the IPC.

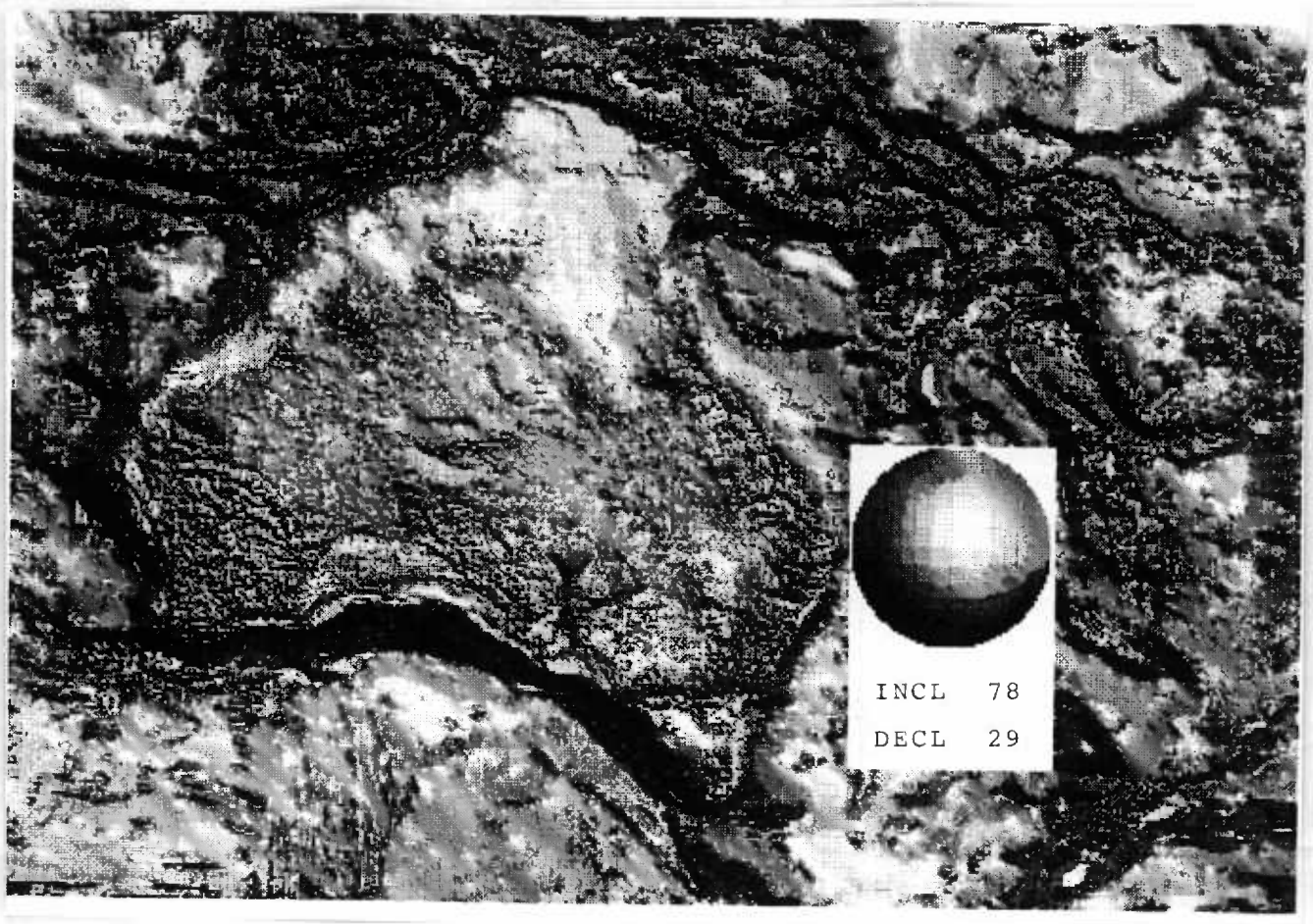


Figure 8

False sun angle manipulation of Australian DTM data. RTI-CAD provides an easy to use interactive interface for doing this type of work.

CONCLUSIONS

RTI-CAD is a powerful PC image processing package which offers an easy to use mouse-driven interface and a very comprehensive vector overlay capability. It provides excellent support for a range of colour and monochrome PC printers and includes an extremely powerful false sun angle capability which is much better than that provided by the I²S system 600 software. On the negative side, RTI-CAD is hampered by the generally poor performance of PC graphics cards and as a result it is only capable of displaying 8-bits of information rather than the 24-bits used by dedicated image processing systems such as I²S.

An easy to use routine, Getimage.bat, has been developed to allow users to transparently down-load images from the I²S system to a PC. The routine handles all the necessary image manipulation and file conversions for the user and this makes it possible for novice users to gain access to I²S images without lengthy training.

The cost of the RTI-CAD image processing package when used on an existing 80286, 80386 or 80486 PC which already includes a suitable monitor and a maths co-processor, as most BMR PCs do, would be $3950. This figure includes the cost of ethernet connectivity for the individual PC. In practice however, most BMR PCs will soon be provided with ethernet connections as part of the corporate information system irrespective of whether they will be used for image processing or not. Thus a more realistic figure for the installation cost of RTI-CAD, not including ethernet associated costs, would be $2630.

As discussed above, the cost of purchasing an additional IVAS workstation for the IPC (with its much greater functionality) in order to cope with increased demand for image processing in BMR would be approximately 7 times this amount (i.e. $36,000). Thus the use of distributed image processing on existing BMR PCs is very cost effective when compared with further investment in proprietary I²S workstations.

There is therefore a strong case for the development of a PC-based low-end image processing capability in BMR such as could be provided by RTI-CAD. This distributed image processing capability would be an adjunct to the existing state of the art image processing capabilities of the Image Processing Centre.

As usage of the IPC by BMR staff becomes more extensive, there will be a considerable cost advantage in developing such a low-end system. This system will help to cater for both specialist users of the IPC who require access to all the functionality of the I²S equipment, and low-level and novice users. In this way the latter users will be saved from undertaking more training than they need and the specialist users can be guaranteed more access to the four available I²S workstations.

APPENDIX A

Listing of Getimage.bat

```
@echo off
rem                 Getimage.bat
rem                 version of 28 September 1990
rem                 Prame Chopra
rem                 Information Systems Branch, BMR
cls
cd \gp\prog
type getimage.dat
        :Usage: GETIMAGE %1 %2 %3 %4 %5 %6 %7
        : %1 is the name of the image to be downloaded from the IIS system
        : %2 is the number of pixels in the image to be downloaded (max 800)
        : %3 is the number of lines  in the image to be downloaded (max 600)
        : %4,%5 are the upper left corner bounds of the selected image:
        : %6 is the type of image to be produced (S = shadow, I = image)
        : %7 is the DOS filename of the final image for use with RTI
rem     (%4, %5)
rem       +---------------------------------------+
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       a                                       a
rem       +---------------------------------------+
rem

rem ------------------------------------------------------------
rem     check that the correct syntax has been used
        if ?%1 == ? goto bad_syntax
        if ?%2 == ? goto bad_syntax
        if ?%3 == ? goto bad_syntax
        if ?%4 == ? goto bad_syntax
        if ?%5 == ? goto bad_syntax
        if ?%6 == ? goto bad_syntax


rem ------------------------------------------------------------
```

APPENDIX A (continued)

Listing of Getimage.bat

```
rem    Syntax is OK
rem    Build control file
     echo 10;1H
       echo ***** Building control file *******
       echo _
       echo usr'pixin > getimage.ctl
       echo %1 >> getimage.ctl
       echo hold.flip >> getimage.ctl
       echo cpu'disktransfer >> getimage.ctl
       echo hold.flip >> getimage.ctl
       echo hold.lin >> getimage.ctl
       echo %2 >> getimage.ctl
       echo quit >> getimage.ctl

rem    calculate the dimensions of the selected window in the image
       echo getimage.ctl > hold
       echo %4 >> hold
       echo %5 >> hold
       echo %2 >> hold
       echo %3 >> hold

       modify < hold

       if ERRORLEVEL 4 goto no_band
       if ERRORLEVEL 3 goto cant_write_file
       if ERRORLEVEL 2 goto bad_file
       if ERRORLEVEL 1 goto no_file

rem ------------------------------------------------------------
rem    converting control file to UNIX format

       cd \nfs
     echo 10;1H
       echo ***** Converting control file to UNIX format *****
       echo _

       dos2unix \gp\prog\getimage.ctl \gp\prog\getimage.unx

rem ------------------------------------------------------------
rem    send control file to IPCONE
```

APPENDIX A (continued)
Listing of Getimage.bat

```
    echo 10;1H
      echo ***** Sending control file to IIS System *****
      echo _

      rcp \gp\prog\getimage.unx ipcone:/mnt/pramec/flip_d/control

rem ------------------------------------------------------------
rem   start shell script on IPCONE to flip the image and convert
rem   it to a line-oriented format. This uses the control file.
rem   previously copied up to IPCONE.

    echo 10;1H
      echo ***** Converting IIS image on IPCONE *****
      echo _

      rsh ipcone /mnt/pramec/flip_d/extract

rem ------------------------------------------------------------
rem   recover the extracted image from IPCONE
cls
type  \gp\prog\getimage.dat
    echo 10;1H
      echo ***** Downloading the image from IPCONE *****
      echo _
      rcp ipcone:/mnt/pramec/flip_d/hold.lin \gp\prog\rtimage

rem ------------------------------------------------------------
rem   build DOS control files for the conversion routines

    echo 10;1H
      echo ***** Building DOS control files *****
      echo _
      cd \gp\prog
      echo rtimage > iis_in
      echo %2 >> iis_in
      echo %3 >> iis_in


      echo iis.out > grid_in
      echo 3 >> grid_in
      echo %6 >> grid_in
```

APPENDIX A (continued)

Listing of Getimage.bat

```
        if .%7 == .     echo %1 >> grid_in
        if NOT .%7 == .  echo %7 >> grid_in
        echo n >> grid_in
        echo e >> grid_in
rem     check to see if a shadow file is requested, if so then insert an
rem     additional line in the control file for scale E-W and N-S independently?
        if %6 == S echo y >> grid_in
        if %6 == s echo y >> grid_in

rem ---------------------------------------------------------
rem     convert the line oriented image to GEOPAK format
cls
    echo 10;1H
        echo ***** Converting file to GEOPAK format *****
        echo _
        iis < iis_in

rem ---------------------------------------------------------
rem     convert the GEOPAK file to an image or shadow file
cls
type getimage.dat
    echo 10;1H
        echo ***** Making image/shadow file *****
        echo _
        gridprep < grid_in

rem ---------------------------------------------------------
rem     move image files to /image directory
    copy %7.* \gp\images > nul
    del  %7.* > nul
    cd \gp\images

rem ---------------------------------------------------------
rem     start GEOPAK/RTI after printing a message
cls
type getimage.dat
    echo 9;1H
        echo The following files have been created for display with RTI.

        if NOT exist %7.* goto no7
        if NOT ?%7 == ? if %6 == i dir %7.i??
```

APPENDIX A (continued)

Listing of Getimage.bat

```
        if NOT ?%7 == ? if %6 == I dir %7.i??
        if NOT ?%7 == ? if %6 == s dir %7.s??
        if NOT ?%7 == ? if %6 == S dir %7.s??
        goto next
:no7
        if %6 == i dir %1.i??
        if %6 == I dir %1.i??
        if %6 == s dir %1.s??
        if %6 == S dir %1.s??
        goto next

:next
        echo +-------------------------------------------------------------------------+
        echo ª                                                                         ª
        echo ª Use this filename (with no extension in the RTI files menu).           ª
        echo ª                                                                         ª
        echo +-------------------------------------------------------------------------+
        pause

        \gp\prog\rti
    cls


rem -----------------------------------------------------
rem     tidy up

        del \gp\prog\hold
        del \gp\prog\rtimage
        del \gp\prog\getimage.ctl
        del \gp\prog\getimage.unx
        del \gp\prog\iis_in
        del \gp\prog\grid_in


    goto end

rem -----------------------------------------------------
:no_file
        cls
echo _
echo  +-------------------------------------------------+
```

APPENDIX A (continued)

Listing of Getimage.bat

```
echo ª                                                    ª
echo ª   Could not open the control file HOLD             ª
echo ª                                                    ª
echo _ +----------------------------------------------------+
       goto screen

:bad_file
       cls
echo _
echo  +----------------------------------------------------+
echo ª                                                    ª
echo ª      Control file HOLD was empty                   ª
echo ª                                                    ª
echo _ +----------------------------------------------------+
       goto screen
:cant_write_file
       cls
echo _
echo  +----------------------------------------------------+
echo ª                                                    ª
echo ª     Cant write to control file HOLD                ª
echo ª                                                    ª
echo _ +----------------------------------------------------+
       goto screen

:no_band
       cls
echo _
echo  +----------------------------------------------------+
echo ª                                                    ª
echo ª  You must specify the band in the image            ª
echo ª  to be used (eg. image(2) for band 2 )             ª
echo ª                                                    ª
echo _ +----------------------------------------------------+
       goto screen
```

APPENDIX A (continued)

Listing of Getimage.bat

```
:bad_syntax
        cls
echo _
echo  +-------------------------------------------------------+
echo  a                                                       a
echo  a  That is incorrect usage of GETIMAGE.BAT              a
echo  a                                                       a
echo _ +-------------------------------------------------------+
        goto screen

:screen
echo _
echo   Correct Usage is: GETIMAGE %%1 %%2 %%3 %%4 %%5 %%6
%%7  Where:
echo    %%1 is the image to be downloaded from the IIS system ( eg /mnt/test(1)
)
echo    %%2 is the number of pixels in the image to be downloaded (max 800)
echo    %%3 is the number of lines  in the image to be downloaded (max 600)
echo    %%4 & %%5 are the upper left corner bounds of the selected image:
echo _
echo          a----------------%%2--------------a
echo  (%%4,+-------------------------------------+  -
echo   %%5)a                                     a  a
echo          a                                 a  a
echo          a                                 a  %%3
echo          a                                 a  a
echo          a                                 a  a
echo          +-------------------------------------+  -
echo    %%6 is the image type to be created (S = shadow, I = image)
echo    %%7 is the [optional] DOS filename of the final image for use with RTI


:end
```

APPENDIX A (continued)

Listing of Getimage.dat

```
+-------------------------------------------------+
a                                                 a
a    GETIMAGE.BAT                                 a
a    version of 5 April 1990                      a
a    Prame Chopra                                 a
a    Information Systems Branch, BMR              a
a                                                 a
+-------------------------------------------------+
```

APPENDIX B

Listing of Modify.for

```
c*****************************************************************
      program modify

c This program is part of the GETIMAGE.BAT suite for transfer of IIS
c images between the SUN 4/280's and an IBM PC running GEOPAK/RTI


c                                   Prame Chopra
c                                   2 April 1990

      integer*2 xstart, ystart, xinc, yinc, xfin, yfin, iband, end
      integer*2 usrindex
      character*20 file_name, band*1, command
      character*4 xst4, yst4, xfn4, yfn4      ! for dimensions > 999
      character*3 xst3, yst3, xfn3, yfn3      ! "    "    < 999
      character lines(8)*50, image*50
      logical debug

      debug = .false.
      luin  = 20
      luout = 21
c     when changing the number of lines in the input file, also change
c     the array subscript for "lines" above.
      num_lines = 8


c---- get file to be modified and window dimensions ----------------
      if ( debug )
     & write(*,*) 'What is the name of the file to be modified? ---> '
      read(*,'(a)') file_name
      if ( debug ) write(*,*) 'Name is ... ',file_name

      if ( debug )
     & write(*,*) 'Enter the 4 window parameters ---> '
      read(*,*) xstart, ystart, xinc, yinc
      if ( debug ) write(*,*) 'Parameters are ... ',xstart, ystart,
     &                  xinc, yinc
```

APPENDIX B (Continued)
Listing of Modify.for

```
c---- open the file to be modified ---------------------------------

    open(unit   = luin,
   &    file   = file_name,
   &    status = 'old',
   &    err    = 500)

99   format(a)
     do 100 k = 1, num_lines
100  read(luin,99, end = 520)lines(k)

     if ( debug ) write(*,*) (lines(ii),ii=1,num_lines)

     command = 'del '//file_name

     close(luin)
     call system(command)

c---- open the new output file of the same name ---------------------
     open(unit   = luout,
    &    file   = file_name,
    &    status = 'new')


c---- write line 1 --------------------------------------------------
     write(UNIT=luout,FMT='(a)',ERR =510) lines(1)
     if ( debug ) write(*,*) lines(1)

c    get the image name; add a trailing blank for later indexing
     image = lines(2)//' '
     if ( debug ) write(*,*) image


c---- calculate the bottom right corner's coords --------------------
     xfin = xstart + xinc - 1
     yfin = ystart + yinc - 1

c convert the coords to type character
c eliminate unwanted spaces for dimensions < 1000
     if (xstart .gt. 999) write(xst4,'(i4)') xstart
     if (xstart .le. 999) write(xst3,'(i3)') xstart
```

APPENDIX B (Continued)

Listing of Modify.for

```
   if (ystart .gt. 999) write(yst4,'(i4)') ystart
    if (ystart .le. 999) write(yst3,'(i3)') ystart
    if (xfin  .gt. 999) write(xfn4,'(i4)') xfin
    if (xfin  .le. 999) write(xfn3,'(i3)') xfin
    if (yfin  .gt. 999) write(yfn4,'(i4)') yfin
    if (yfin  .le. 999) write(yfn3,'(i3)') yfin

c find last alpha character in image name and image band number required
c Images are designated as for eg: canberra(;;3) or $iisimages/kili
    end   = usrindex(image,'(' ) - 1
    if ( end .eq. -1 ) end = index(image,' ') - 1
    iband = usrindex(image,')' ) - 1


c get the band value as a character
    if ( iband .ne. -1 ) then
       band = image(iband:iband+1)
    else
       band = ' '
    endif

    if ( debug ) write(*,*) 'end = ',end,' iband = ',iband
    if ( debug ) write(*,*) 'last char in image name = ',
   &    image(end:end+1),' band = ',band

c---- export the appropriate modified line ------------------------------
    if ( xstart .gt. 999 .and. ystart .gt. 999 .and. xfin .gt. 999
   & .and. yfin .gt. 999) then
    write (luout,fmt='(a)')
   & image(1:end)//'('//xst4//':'//xfn4//';'//yst4//':'//yfn4//';'//ba
   &nd//')'

    else if( xstart .le. 999 .and. ystart .gt. 999 .and. xfin .gt. 999
   & .and. yfin .gt. 999) then
    write (luout,fmt='(a)')
   & image(1:end)//'('//xst3//':'//xfn4//';'//yst4//':'//yfn4//';'//ba
   &nd//')'


    else if( xstart .gt. 999 .and. ystart .le. 999 .and. xfin .gt. 999
   & .and. yfin .gt. 999) then
```

APPENDIX B (Continued)
Listing of Modify.for

```
      write (luout,fmt='(a)')
     & image(1:end)//'('//xst4//':'//xfn4//';'//yst3//':'//yfn4//';'//ba
     &nd//')'

      else if( xstart .le. 999 .and. ystart .le. 999 .and. xfin .gt. 999
     & .and. yfin .gt. 999) then
      write (luout,fmt='(a)')
     & image(1:end)//'('//xst3//':'//xfn4//';'//yst3//':'//yfn4//';'//ba
     &nd//')'

      else if( xstart .le. 999 .and. ystart .le. 999 .and. xfin .le. 999
     & .and. yfin .gt. 999) then
      write (luout,fmt='(a)')
     & image(1:end)//'('//xst3//':'//xfn3//';'//yst3//':'//yfn4//';'//ba
     &nd//')'

      else if( xstart .le. 999 .and. ystart .le. 999 .and. xfin .gt. 999
     & .and. yfin .le. 999) then
      write (luout,fmt='(a)')
     & image(1:end)//'('//xst3//':'//xfn4//';'//yst3//':'//yfn3//';'//ba
     &nd//')'

      else if( xstart .le. 999 .and. ystart .le. 999 .and. xfin .le. 999
     & .and. yfin .le. 999) then
      write (luout,fmt='(a)')
     & image(1:end)//'('//xst3//':'//xfn3//';'//yst3//':'//yfn3//';'//ba
     &nd//')'

      endif

      if ( debug )
     & write (*,*)
     & image(1:end)//'('//xst4//':'//xfn4//';'//yst4//':'//yfn4//';'//ba
     &nd//')'

c---- write the remaining lines ----------------------------------------
      do 400 l = 3,num_lines
400   write(luout,*) lines(l)

      goto 5000
```

APPENDIX B (Continued)

Listing of Modify.for

```
c*****************************************************************
500  write(*,'(A)')'** error opening file '//file_name(1:end-1)//' **'
     goto 5000

510  write(*,'(A)')'** error reading file '//file_name(1:end-1)//' **'
     goto 5000

520  write(*,'(A)')'** data missing in '//file_name(1:end-1)//' **'


5000 stop
     end


c=================================================
     integer function usrindex(c_array,look)

     character c_array*(*), look*1
     logical debug

     debug = .false.

     if ( debug ) write(*,*) 'Inputs are --> ',c_array,' and --> ',look

c    append a tilde to the end of the array to limit the search
     c_array = c_array//'~'

     i = 0

100  i = i + 1
     if ( debug ) write(*,*) 'char in column ',i,' is ',c_array(i:i)
     if ( c_array(i:i) .eq. look ) then
          usrindex = i
        if ( debug ) write(*,*) look,' found in column ',usrindex
        return
     elseif ( c_array(i:i) .eq. '~' ) then
          usrindex = 0
        write(*,*) 'substring '//look//' not found'
        return
     endif
```

```
    if ( i .gt. 100 ) then
       write(*,*) 'substring '//look//' not found in first 100 chara
    &cters'
       usrindex = 0
       return
    endif
    goto 100
    end
```

APPENDIX C
Listing of IIS.for

```
      program iis

c*****************************************************************
c     purpose: This program will read a large line-oriented image file
c        written by the IIS cpu'disktransfer utility
c             and write the contents to an unformatted sequential file
c        called IIS.OUT
c        NB. the input file is expected to be INTEGER
c        NB. IIS MUST be compiled with Microsoft FORTRAN for the
c          output file to be usable with GEOPAK/RTI's GRIDPREP
c          utility.
c                                        Prame Chopra
c                                        Information Systems Branch
c                                        BMR
c                          30 March 1990
c*****************************************************************

      real*4 header(32), xmult, csub, x_origin, y_origin, map_scale,
     &     grnd_units_in_x, grnd_units_in_y, ins_per_grnd_unit,
     &     old_max, old_min
      integer*2 ibuf2(7000), num_rows, num_cols, default
      integer*1 ibuf1(7000)
      character*30 in_filename, out_filename1, out_filename2
      character*4 size
      logical debug


      debug = .TRUE.
      out_filename1 = 'hold'
      out_filename2 = 'iis.out'
      size = '  '

c---------------------------------------------------------------------
c  get the input filename and image dimensions

      write(*,'(a\)')
     &   ' Which IIS line-file do you want to read?  ----> '
      read(5,'(a)') in_filename

      write(*,'(a\)')
     &   ' What is the line-length of this file? [800] ---> '
```

```
      read(5,'(a)' ) size
      if ( size .eq. '   ') then
c       use default
            num_cols = 800
      else
c           use supplied value
            read(size,'(i4)')num_cols
      endif

      write(*,'(a\)')
     &   ' How many lines are there in this file? [600] --> '
      read(5,'(a)' ) size
      if ( size .eq. '   ') then
c       use default
            num_rows = 600
      else
c           use supplied value
            read(size,'(i4)')num_rows
      endif


c-----------------------------------------------------------------------
      open ( unit   = 21,
     &     file   = in_filename,
     &     status = 'OLD',
     &     form   = 'BINARY')

      open ( unit   = 22,
     &     file   = out_filename1,
     &     status = 'UNKNOWN')

      open ( unit   = 23,
     &     file   = out_filename2,
     &     status = 'UNKNOWN',
     &     form   = 'UNFORMATTED',
     &     access = 'SEQUENTIAL')

c-----------------------------------------------------------------------
c  Assemble a header of 32 real*4 words as follows:

c  header(3) and header(21) = xmult : the number by which each grid value
```

```
c                       has been multiplied so as to scale
c                       into integer*2 space
c  header(4) and header(22) = csub  : the number that was subtracted from
c                       each grid value before the value was
c                       scaled with xmult
c  header(5)  : x origin
c  header(6)  : y origin
c  header(9)  : map scale (5000 for 1:5000)
c  header(10) : number of ground units per cell in the x direction
c  header(11) : number of ground units per cell in the y direction
c  header(12) : number of inches per ground unit (ie 12 if ground
c          units are feet, 39.37 if they are metres)
c  header(17) : number of rows in the grid
c  header(18) : number of columns in the grid
c  header(23) : the prescaled maximum of the grid
c  header(24) : the prescaled minimum of the grid
c  header(26) : the default value for no data at a grid point,
c          normally = -32768



      xmult    = 1.
      csub     = 0.
      x_origin  = 0.0
      y_origin  = 0.0
      map_scale = 5000.
    grnd_units_in_x = 25.
    grnd_units_in_y = 25.
    ins_per_grnd_unit = 39.37
    old_max  = 255
    old_min  = 0
    default     = -32768

    header(3) = xmult
    header(4) = csub
    header(5) = x_origin
    header(6) = y_origin
    header(9) = map_scale
    header(10)= grnd_units_in_x
    header(11)= grnd_units_in_y
    header(12)= ins_per_grnd_unit
```

```
      header(17)= num_rows
      header(18)= num_cols
      header(23)= old_max
      header(24)= old_min
      header(26)= default


c----- write header info to screen ----------------------------------------

         write(*,*) 'Data from (line format) IIS image file ',in_filename
         write(*,*) ' '
         write(*,*) 'Each grid value was first reduced by ',csub
         write(*,*) 'Each grid value was then multiplied by ',xmult
         write(*,*) 'X origin of grid is ',x_origin
         write(*,*) 'Y origin of grid is ',y_origin
         write(*,*) 'Map scale is 1:',map_scale

         if ( ins_per_grnd_unit .eq. 12. ) then
         write(*,*)'Ground units are feet'
         write(*,*) 'There are ',grnd_units_in_x,' feet per cell'
     &          ,' in the X direction'
         write(*,*) 'There are ',grnd_units_in_y,' feet per cell'
     &          ,' in the Y direction'
       endif

         if ( ins_per_grnd_unit .eq. 39.37 ) then
         write(*,*)'Ground units are metres'
         write(*,*) 'There are ',grnd_units_in_x,' metres per cell'
     &          ,' in the X direction'
         write(*,*) 'There are ',grnd_units_in_y,' metres per cell'
     &          ,' in the Y direction'
       endif

         write(*,*) 'There are ',num_rows,' rows of data in the grid'
         write(*,*) 'There are ',num_cols,' cols of data in the grid'
         write(*,*) 'The pre-scaling maximum in the grid was ',old_max
         write(*,*) 'The pre-scaling minimum in the grid was ',old_min
         write(*,*) 'The default value for no data at a grid point is ',
     &          default
```

APPENDIX C (Continued)
Listing of IIS.for

```
c----- write header info to file ----------------------------------------

      write(22,*)'Data from (line format) IIS image file ',in_filename
      write(22,*) ' '
      write(22,*) 'Each pixel value was first reduced by ',csub
      write(22,*) 'Each pixel value was then multiplied by ',xmult
      write(22,*) 'X origin of grid is ',x_origin
      write(22,*) 'Y origin of grid is ',y_origin
      write(22,*) 'Map scale is 1:',map_scale

      if ( ins_per_grnd_unit .eq. 12. ) then
      write(22,*)'Ground units are feet'
      write(22,*) 'There are ',grnd_units_in_x,' feet per pixel'
&           ,' in the X direction'
      write(22,*) 'There are ',grnd_units_in_y,' feet per pixel'
&           ,' in the Y direction'
  endif

      if ( ins_per_grnd_unit .eq. 39.37 ) then
      write(22,*)'Ground units are metres'
      write(22,*) 'There are ',grnd_units_in_x,' metre per pixel'
&           ,' in the X direction'
      write(22,*) 'There are ',grnd_units_in_y,' metre per pixel'
&           ,' in the Y direction'
  endif

      write(22,*) 'There are ',num_rows,' rows of data in the image'
      write(22,*) 'There are ',num_cols,' cols of data in the image'
      write(22,*) 'The pre-scaling maximum in the image was ',old_max
      write(22,*) 'The pre-scaling minimum in the image was ',old_min
      write(22,*) 'The default value for no data at a pixel is ',
&         default

c---- write header info to the new GEOPAK/RTI format file
        write(23) header


c----- read iis data and write to file ------------------------------------
c----- convert input data range of -128 -> +127 to 0 -> 255
    write(*,*) ' '
      do 100 i = 1, num_rows
```

```
        read (21)  (ibuf1(j),j=1,num_cols)
      write(*,'(1h+,a,i4)') 'Processing line ',i
        do 99 k = 1,num_cols
 99     ibuf2(k) = ibuf1(k) + 128
        write(23) (ibuf2(j),j=1,num_cols)
100     continue

5000    close(21)
        close(22)
        close(23)
        stop
        end
```

APPENDIX D

Listing of extract C-Shell

```
#       C shell script extract
#               version of 4 April 1990
#               Prame Chopra
#               Information Systems Branch, BMR

cd /mnt/pramec/flip_d
rm hold.*
berti < control
rm hold.flip
rm core
rm control
```

Notes: 1) The file berti is a local be (I^2S batch executive) which includes a routine called usr'pixin which inverts an I^2S image (i.e. turns it upside down and back the front).
2) control is the control file uploaded from the PC by getimage.bat
3) hold.flip is an I^2S format file produced by usr'pixin
4) the output file returned to the PC is called hold.lin