UNITED STATES DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

Woods Hole Image Processing System Software Implementation:
Using NetCDF as a Software Interface for Image Processing

Open-File Report 92-25
January 1992

UNITED STATES DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

Woods Hole Image Processing System Software Implementation: Using NetCDF as a Software Interface for Image Processing

by

Valerie Paskevich[1]

Open-File Report 92-25

January 1992

[1]Woods Hole, MA. 02543

Abstract

The Branch of Atlantic Marine Geology has been involved in the collection, processing and digital mosaicking of high, medium and low-resolution side-scan sonar data during the past 6 years. In the past, processing and digital mosaicking has been accomplished with a dedicated, shore-based computer system. With the need to process side-scan data in the field with increased power and reduced cost of major workstations, a need to have an image processing package on a UNIX based computer system which could be utilized in the field as well as be more generally available to Branch personnel was identified. This report describes the initial development of that package referred to as the Woods Hole Image Processing System (WHIPS). The software was developed using the Unidata NetCDF software interface to allow data to be more readily portable between different computer operating systems.

Introduction

Since 1985, the Branch of Atlantic Marine Geology (BAMG) has been involved in mapping the Exclusive Economic Zone (EEZ) utilizing the GLORIA (Geological Long-Range Inclined Asdic) side-scan sonar system designed and built by the Institute of Oceanographic Sciences (IOS) in England. Side-scan data were collected, processed and digitally mosaicked from the Exclusive Economic Zone (EEZ) of the Gulf of Mexico and Puerto Rico in 1985 (EEZ scan, 1985). In 1987, the Branch completed a GLORIA survey of the Atlantic coast EEZ and again processed and mosaicked the data (EEZ scan, 1987). To process and digitally mosaic the GLORIA data, the Branch has utilized the Mini-Image Processing System (MIPS) software (Chavez, 1984) first on a Digital Equipment Corporation (DEC) PDP 11/73 computer system with RSX-11/M+ operating system and, more recently, on a DEC MicroVAX-II computer system with VMS operating system. In addition to the GLORIA specific processing programs, the MIPS software provides an excellent general image processing package which can be used for processing a variety of image data including Landsat, TMM and SPOT imagery, and bottom photos.

During recent years, the Branch has expanded it's involvement into collecting and processing mid to high-resolution side-scan sonar data from areas such as Lake Michigan (Folger et al), Boston Harbor (Bothner et al), the Gulf of Farallones (Karl et al) and follow-up studies in the Gulf of Mexico (Twichell et al). The MIPS software was expanded by the Branch by developing new programs to accommodate the varied types of side-scan sonar data.

Recent advances in computer hardware architecture, which has lead to increased computing power with rapidly reducing cost, has also lead to an increased utilization of UNIX workstations within BAMG. With that increased utilization, the need arose to provide image processing support, similar to MIPS, on these workstations. Three major considerations in identifying a UNIX image processing system were defined. The first was the ability to handle multiple bit type data as 8, 16 and 32-bit data. Secondly, that the software run on a variety of UNIX based platforms with X-windows capability. This would ensure software availability regardless of the computer systems utilized by the Branch. The third concern was the desire that any developed software be free of licensing fees and available in the public domain. This issue was to maintain reduced costs and therefore the availability of any developed software to others operating on tightened budgets.

Several public domain and proprietary packages were evaluated. Many of the packages, such as the University of New Mexico's Khoros, the National Center for Supercomputing Application's (NCSA's) Ximage and Stardent's AVS, are data visualization packages. Though data visualization and image processing in the past have been treated as separate specialities, they do contain some overlap. These individual disciplines continue to grow as a complement to each other and emerging trends point to the integration of conventional image processing and data visualization. However, after evaluating these packages, they lacked certain required functionality of a general image processing package. Two key areas to the Branch's image processing activities that were missing from these packages were: geometric and radiometric processing support. Though the visualization software packages provided many interesting capabilities, these capabilities were beyond the current plan to implement a general purpose image and side-scan sonar processing package. Additionally, none of the considered packages provided support for processing side-scan sonar data. For these reasons, the decision to develop an image processing package for Branch specific projects was made.

Software Requirements

In developing a new image processing package, the requirement to have a software package that would operate on a UNIX computer system was a major consideration. The first task then became to develop a software interface to define and access the image data. The software interface needed to provide a C interface and, most importantly, the software must provide direct access to a line of image data. Several data access software packages have been summarized by the "SIGGRAPH '90 Workshop Report: Data Structures and Access Software for Scientific Visualization" (Treinish, 1991). The netCDF software (Unidata, 1991) was chosen to be used in an attempt to develop the initial image processing package.

The netCDF software fulfilled the requirements of UNIX support, public domain availability, a C interface and direct file access, and provided further support of a FORTRAN interface. The software also provides "self-describing" capabilities by allowing the inclusion of auxiliary text information to describe the data and, most importantly, the software provides "network-transparent" data by providing an interface for many common computing platforms. This results in the physical data being represented independent of the computer system on which the data were written. In simpler terms, this means that a data file which would contain floating point data is readable via the netCDF software interface on both a DEC computer system and a Sun computer system without further data manipulation to transcribe the data values to the individual systems internal data representation format. Using the netCDF software as the data access interface not only allows improved accessibility of the data, but has allowed the developed software to be compiled and run on a variety of computer systems which have included a SUN Sparc-II, 386-AT compatible PC, DECstation 3100 and 5000 with ULTRIX operating system and a Data General AViiON 410 workstation with DG/UX 4.32 operating system without modification to the data files.

One question in utilizing the netCDF software was how much of a processing overhead would be added by the netCDF software systems independence. While some testing has been done utilizing the computer systems available at the Branch, the WHIPS software has not been extensively tested. It is believed, however, that for most of the image processing applications of byte or short integer data, the netCDf software does not appear to add considerable overhead to the programs processing time, and the UNIX WHIPS software implementation provides increased processing capabilities over that of the MIPS software currently utilized by the Branch. Processing of floating-point data on machines that do not use the IEEE floating-point data representation may increase processing times, but overall system performance may still show considerable improvement as hardware performance improves.

For comparison, a byte image approximately 16M-bytes in size was processed utilizing the WHIPS software on three different computer systems as well as the Branch's MicroVAX-II computer system with the MIPS software. The WHIPS implementation of the filter program is significantly different than that of the VAX/VMS MIPS version but provides the same functional options and results. Below is a comparison of executing the WHIPS **filter** program and the MIPS **filter8** program. Both programs were run with a 301 x 301 box-car kernel and the high-pass filter option selected. Time values listed below are elapsed, wall-clock time in minutes and seconds.

WHIPS: DECstation-3100 8:55
DECstation-5000 5:46
Data General AViiON-41011:09

MIPS: MicroVAX-II 31:18

The **filter** program is the most compute intensive of the currently developed WHIPS programs, and it clearly demonstrates the significant processing speed gained by the various UNIX workstations over the MIPS implementation. When appropriate, processing times of various MIPS and newly developed WHIPS programs have been compared. In general, the WHIPS software in conjunction with the netCDF data interface shows an average of fifty to seventy percent improvement in processing times.

The WHIPS image file format

A digital image may be defined as a “two-dimensional array of digital values that correspond to discrete spectral reflectivity values arranged in a checkerboard pattern over a target area” (Condit and Chavez, 1979). Each value represents the reflective brightness in the image at one discrete location. These locations are commonly referred to as pixels, which is a contraction of picture element, and the value is commonly referred to as the pixel DN (digital number). The horizontal rows of pixels will be referred to as lines and the vertical columns as samples. The image origin is located in the upper left corner of the array.

The basic *WHIPS image* file is defined utilizing the netCDF *dimension, variable,* and *attribute* components. The netCDF dimensions defines the number of lines (nl) and the number of samples (ns) contained in the two dimensional array which comprises the *image* data. These dimension variables are specified at the beginning of the data file. The image data is further defined by the image bittype (*bittyp*). The *bittyp* is used as a flag by the processing programs to indicate the datatype used to store the image data. The *bittyp* is declared a short integer variable and must contain a value of 8, 16 or 32. A value of 8 indicates byte or character image data. A value of 16 indicates image data of a 16-bit or short integer datatype. A value of 32 indicates 32-bit floating point image data. As previously mentioned, the image data is then defined utilizing the nl and ns dimensions and the *bittyp* variable. These are the only required variables needed to define *WHIPS image* data.

In addition to the image definition, one global attribute is currently in use. This netCDF attribute contains the name of the program which created the *WHIPS image* file. This global text information may be expanded in the future to contain additional information such as processing history.

A condensed example of a basic *WHIPS netCDF image* file that was printed with the netCDF utility program **ncdump** is listed below. In this example, the image data are stored as byte data, and the image DN values are printed in their octal representation.

```
netcdf mickey {
dimensions:
nl = 480 ;
ns = 472 ;

variables:
short bittyp ;
byte image(nl, ns) ;

// global attributes:
:creation_program = "raw2whips" ;

data:

bittyp = 8 ;

image =
0230, 0250, 0250, 0256, 0255, 0261, 0264, 0271, 0274, 0273, 0271, 0266,
0267, 0272, 0270, 0272, 0271, 0272, 0275, 0271, 0271, 0274, 0276, 0300,
0272, 0275, 0275, 0276, 0277, 0275, 0275, 0276, 0277, 0300, 0277, 0275,
...., ....,
```

The WHIPS side-scan sonar image format

As was stated earlier, the developed software was required to process a variety of types of side-scan sonar data collected by the Branch. The *WHIPS image* format defined using the netCDF software provided a simple format for describing the image portion but did not provide enough information for processing side-scan sonar data. To properly process side-scan sonar data additional information to detail the individual scans is required. To accommodate the side-scan sonar data, the *WHIPS image* file was expanded to include the additional information. This information, referred to as the side-scan sonar header information, is specific to the side-scan sonar processing programs and includes information such as the date, time, side-scan sonar vehicle attributes and vehicle position. Header information, which is not relevant to the specific sonar system or which may be unavailable, is flagged by inserting the netCDF infinity value for the particular data type.

The side-scan sonar header information is read and written only by the side-scan sonar specific processing programs. General image processing programs may be applied to the *WHIPS netCDF side-scan sonar image*, but only the image portion of the file will be processed and output. The user should be aware that some steps required by the sonar processing utilize the general processing programs and, for further sonar processing to take place, the header needs to be restored to the WHIPS image file.

As mentioned above, the basic WHIPS image file was expanded to handle side-scan sonar data by adding several variables. Most side-scan header variables were grouped together as one data variable to reduce the read/write access for obtaining the information. For example, it appeared easier to store all the integer date variables together which could be read with one read rather than 4 reads which would have been scattered throughout the data file. Though it would have been preferable to store all the header variables for a given swath together and reduce the file access, it was not possible to mix data variable types.

The side-scan sonar header variables are grouped as listed below:

date - date variables (year, month, day, day_of_year)
time - time variables: hour, minute, total_mininutes
seconds - time variable: seconds portion of time
sonar_attr - sonar attributes: fish_altitude, heading, pitch, roll and yaw
position - sonar position: latitude and longitude coordinates recorded as signed decimal degrees
depth - depth values: corrected and uncorrected depth in meters. Used mostly by GLORIA side-scan sonar data.

The **seconds** variable is the only single dimension variable contained within the WHIPS side-scan sonar image file. The seconds variable is dimensioned equal to the number of lines (*nl*) contained within the image file. The **date**, **time**, **sonar_attr**, **position** and **depth** are multi-dimensioned array variables. These data variables are dimensioned by the number of lines contained in the image and the number of variables for that data variable. The number of variables for the individual data groups are currently defined as:

date - four data variables
time - three data variables
sonar_attr - five data variables
position - two data variables
depth - two data variables

The number of variables for each of these data groups is stored in the WHIPS netCDF side-scan sonar image for self documentation purposes. Additionally, global attributes are defined which contain text information describing the information contained within the specific header variables. Future development may require that these data variables be increased. Due to the self-defining nature of the netCDF software, the current dimension sizes are accessible through the netCDF file itself and would not result in major revision of the WHIPS software to accommodate these changes.

An example of a small *WHIPS side-scan sonar image* was printed using the netCDF utility program ncdump and is shown below. The example shows the definition of a small sonar image file which is defined as being 5 lines by 3 samples. Note the use of the FloatInf values contained in the pitch, roll, and yaw variables of the ss_attributes data. This demonstrates the use of the netCDF fill values to indicate that valid data is not available for a specific data variable.

```
netcdf small_glo {
dimensions:
nl = 5 ;
ns = 3 ;
#date_variables = 4 ;
#time_variables = 3 ;
#ss_attributes = 5 ;
latlon = 2 ;
depth = 2 ;

variables:
short bittyp ;
byte image(nl, ns) ;
short date(nl, #date_variables) ;
short time(nl, #time_variables) ;
float seconds(nl) ;
float ss_attributes(nl, #ss_attributes) ;
double latlon(nl, latlon) ;
float depths(nl, depth) ;

// global attributes:
:creation_program = "glo2whips" ;
:date_variables = "year, month, day, day_of_year" ;
:time_variables = "hour, minute, total_minutes" ;
:seconds = "seconds portion of time variable" ;
:ss_attributes = "fish_altitude, heading, pitch, roll, yaw" ;
:latlon = "sonar position: lat/lon signed decimal degrees" ;
:depth_variables = "corrected_depth, uncorrected_depth" ;

data:

bittyp = 8 ;

image =
030, 031, 030,
033, 033, 036,
034, 033, 034,
036, 033, 036,
035, 030, 030 ;

date =
1987, 2, 25, 56,
1987, 2, 25, 56,
1987, 2, 25, 56,
1987, 2, 25, 56,
```

```
1987, 2, 25, 56 ;

time =
23, 0, 1380,
23, 0, 1380,
23, 1, 1381,
23, 1, 1381,
23, 2, 1382 ;

seconds = 0, 30, 0, 30, 0 ;

ss_attributes =
1450, 202, FloatInf, FloatInf, FloatInf,
1461, 203, FloatInf, FloatInf, FloatInf,
1472, 203, FloatInf, FloatInf, FloatInf,
1483, 205, FloatInf, FloatInf, FloatInf,
1495, 203, FloatInf, FloatInf, FloatInf ;

latlon =
35.34930038452148, -74.80359649658203,
35.34838104248047, -74.80377197265625,
35.34745025634766, -74.80394744873047,
35.34653091430664, -74.80412292480469,
35.34560012817383, -74.80429840087891 ;

depths =
1438, 1450,
1449, 1461,
1460, 1472,
1471, 1483,
1482, 1495 ;
}
```

Program Implementation

The initial program development focused on the development of a series of programs to process high and low-resolution side-scan sonar. To accomplish this, a series of task specific programs were developed along with several additional programs which could be classified as general image processing support. Whenever possible, the program was written to handle any of the three bittypes. However, most sonar specific programs were written to handle 8-bit data only because the side-scan sonar data values range between 0-255. The current programs are listed below and are grouped according to one of the three defined categories: geometric, radiometric and utility. It should be noted that several of the utility programs are side-scan sonar specific. A detailed description of each of the programs may be found in Appendix A.

<u>geometric:</u>

slant - correct a side-scan sonar image for water depth and slant-range geometry distortions
velocity - correct a side-scan sonar image for change in ship's velocity and any aspect ratio distortions that exist

<u>radiometric:</u>

derivative - compute first order difference of an image
filter - apply a low-pass, high-pass, zero replacement or divide filter to an image
lowpass2b2 - applies a 2 by 2 low-pass filter to an image
shade - applies a simple shading correction to an 8-bit side-scan sonar image
wtcombo - weighted combination of 2-4 input images

utility:

dk2dk - create a new image file by extracting a sub-area from an existing image
flip - flip the samples in an image line
glo2whips -converts a raw side-scan sonar data image and header file to a netCDF image file for use with WHIPS
listdn - print the digital values contained in an image file
listhdr - list the contents of a side-scan sonar header in a WHIPS image
mipssonar -converts a raw side-scan sonar data image and header file to a netCDF image file for use with WHIPS
mrgnav - merges navigation and altitude information into the header of a side-scan sonar WHIPS netCDF image file
raw2whips - convert a raw image file to a WHIPS netCDF image file
restorehdr - modify an WHIPS netCDF image by adding the header of a side-scan sonar image
reverse - reverse the lines in an image
whips2raw - convert a WHIPS netCDF image file to a raw image file
xhistgrm - display the histogram of an 8-bit image in an X-11 window

With the exception of programs **restorehdr** and **xhistgrm**, the processing programs open at least one input file, process the data, and outputs a new file. The output files are created utilizing the NO_CLOBBER option of the netCDF ncopen routine. Implementing the ncopen function with the NO_CLOBBER option means that the output file specified by the user must not currently exist. It was decided to open, or rather create, the output files in this manner to safeguard the user from overwritting the wrong files.

All programs have been written in the C programming language. Unlike the MIPS software, no imposed standardization has been applied to the developed programs. A library of C utility routines to support the WHIPS software development has been created, and continues to grow. These routines provide support such as opening and creating the WHIPS images, reading and writing the WHIPS side-scan sonar header information as well as additional utility functions. The lack of an enforced standardization means the programmer is free to develop programs in his or her own style. It is also the programmer's responsibility to define the developed program's parameters, level of user information, input checking and implementation of the required netCDF routines.

The WHIPS programs often read the input image data as a single line of data, process the image buffer, and output the new image line. The image buffer is dynamically allocated to accommodate the image record length based upon the *number of samples* (*ns*) and the *image bittype* (*bittyp*). Though line processing can usually be handled within one major loop with "if" statements to handle the read and writes for the file *bittyp*, it may be more efficient to create separate functions to process the individual bittypes when complex processing is required. This would elimi-nate the redundant testing and jumping within the loop to process the appropriate image *bittyp*. A short, abbreviated example which implements the netCDF software to read, process and write a line of image data follows. This is not a definitive implementation but may be considered a typical implementation. The example below assumes the input and output image sizes are identical. Also, the example does not include detailed information showing the implementation of the netCDF routines to open the input file, create the output file, and obtain the necessary information needed from the file to accomplish the data access. Since the netCDF files are considered "self-defining", the implementation of the netCDF routines to open, create, and access the files may be implemented in any manner the user chooses. The usage and inter-relationship of the netCDF variables, dimensions and data have been full documented by Unidata.

```
main(argc,argv)

char **argv;

int argc;

{

     unsigned char *cbuff;

     int i, ix;
     int cdfidin, cdfidout;
     int img_id_in, img_id_out;
     int nl, ns;
     int pos[2], len[2];

     short int *ibuff;

     float *rbuff;

/* ------------------------------------------------------------------ */
/* END OF DECLARATIONS */
/* ------------------------------------------------------------------ */

     OPEN INPUT AND OUTPUT FILES

     . . . . .
     . . . . .
     . . . . .

     GET REQUIRED NetCDF FILE INFO

     . . . . .
     . . . . .
     . . . . .

/*
** Allocate the space for the input buffer in memory.
*/

     if (bittyp == 32)
        rbuff = (float *) calloc(ns,sizeof(float));
     else if (bittyp == 16)
        ibuff = (short int *) calloc(ns,sizeof(short int));
     else cbuff = (unsigned char *) calloc(ns,sizeof(unsigned char));

     pos[0] = 0;
     pos[1] = 0;
     len[0] = 1;
     len[1] = ns;

/*
```

```
**
** MAIN PROCESSING LOOP
**
** Loop here to process all the image data (nl).
**
*/

     for (i=1; i<=nl; i++) {
         if (bittyp == 32) { /* process 32-bit data */
             ncvarget(cdfid,img_id_in,pos,len,(void *) cbuff);
             for (ix=1; ix<=ns; ix++) {

                PROCESS THE IMAGE BUFFER
                . . .
                . . .
            }
            ncvarput(cdfid,img_id_out,pos,len,(void *) rbuff);
    }
    else if (bittyp == 16) { /* process 16-bit data */
        ncvarget(cdfid,img_id_in,pos,len,(void *) ibuff);
            for (ix=1; ix<=ns; ix++) {

                PROCESS THE IMAGE BUFFER
                . . .
                . . .
            }
            ncvarput(cdfid,img_id_out,pos,len,(void *) ibuff);
    }
    else { /* process 8-bit character data */
        ncvarget(cdfid,img_id_in,pos,len,(void *) cbuff);
        for (ix=1; ix<=ns; ix++) {

                PROCESS THE IMAGE BUFFER
                . . .
                . . .
        }
            ncvarput(cdfid,img_id_out,pos,len,(void *) cbuff);
    }
    ++pos[0];
    }

    ncclose (cdfidin); /* close input file */
    ncclose (cdfidout); /* close output file */
    exit(0);
}
```

As stated earlier, it may be more efficient at times to branch to individual functions that would process a specific data type. In those cases, the processing loop could be replaced by a simple "if" statement which would test the image bittype being processed and then pass control to the appropriate function to process the image bittype. The code contained in the processing loop might be replaced by code similar to that listed below:

```
    if (bittyp == 32) do_32bit();
    else if (bittyp == 16) do_16bit();
```

```
    else do_8bit();
```

The developed WHIPS programs execute with commands entered from the run-line. It was difficult to keep the program's run-line options unique and yet meaningful from program to program. However, three key flags that have remained constant from program to program are: *-i*, *-o* and *-H*. These flags specify the input (*-i*) and output (*-o*) files to be processed. The *-H* option indicates that the user has requested the program help information be displayed. This help information is a summary of required and optional program flags along with a brief description and example of the proper usage of the flag. Unfortunately, the same key flag may be utilized by different programs with different meanings. For example, in one program the key flag *-l* may mean the number of lines while in another program the same key flag may indicate a line increment. The *-H* option was implemented to allow the user a quick way to review the required and optional input for a specific program. An example of the help output for program filter follows. The help for this specific program is somewhat long due to the complex nature of the program.

```
                     *****    FILTER    *****
         Version: 1.2                     16-October-1991

Applies a low-pass, zero replacement, high-pass or divide filter to an image.

Number of Input Files: 1               Number of Output Files: 1
Bit types: 8, 16, 32                   Bit types: 8, 16, 32


REQUIRED INPUT:
          -i   input image file to be processed
          -o   output file to be created
          -b   defines filter boxcar size
                  boxcar values must be odd integer values
                  specify as: -b nl,ns
          -l   low-pass filter changing all input values
          -L   low-pass filter changing only valid data
          -z   low-pass zero replacement filter
          -h   high-pass filter
          -d   divide filter
               NOTE: Only one of -l, -L, -z, -h or -d may be selected.

OPTIONAL INPUT:
          -v   selects range of valid data for filters
          -a   specifies fraction of original value to be added back to HPF
                  Default addback value is 0.
          -c   specifies coefficient used to expand the output results
          -f   specifies, as a real value, the fraction of the filter window
                  that must have valid data for filtering to occur
          -m   specifies minimum number of pixels of valid data needed in
                  filter window for filtering to occur
                  Default minimum value is 1.
          -n   specifies normalization value used to center HPF and Div
                  operations
                  For HPF and 8-bit input image, default normalization
                  value is 127 and is added.
                  For DIV and 8-bit input image, default normalization
                  value is 0 and is subtracted.
                  Default normalization value for other options is 0.
```

```
-H    prints this help information
```

If the user does not specify a required option or the program determines a serious error in the user's input, an error message will be displayed by the program. If an invalid option is entered, the program will display a simple usage line. An example of the usage line from program filter is as follows:

```
filter: illegal option -- x
usage: filter -i input_file -o output_file -b nl,ns [-l -L -z -h -d] options [-H]
```

The program documentation found in Appendix A describes the program options and their syntax in detail. However, program run-line options may be generally categorized as one of two types and should be quite familiar to UNIX users. Under this run-line implementation, the user may specify the program key flags in any order as long as they adhere to the individual flags's syntax rules. The first type is the simple specification of a program key flag. Some program options may be selected by entering a key flag such as *-z* in the **filter** example above. Other program options require the user to specify the key flag and additional information. One example of this is the type of pro-gram option is *-i input*. This key flag is required by all WHIPS programs, and indicates to the processing program that the following information contains the name of the input file selected for processing. Another example is the *-b nl,ns* option. This type of program flag requires the user to enter the key flag and multiple values. These values must be entered as shown in the example: separated by a comma. Though some programs will compute default values if no values or improper values are entered, the user should get in the practice of specifying all values when specifying a key flag with multiple input requirements.

Finally, each program creates a print file which contains a summary of the program commands. The print file name is created by appending a *.prt* suffix to the program name. In other words, the program **dk2dk** would create a print file named *dk2dk.prt*. The print file, if opened by the main program, contains information describing the files being processed and program execution parameters. A new print file is created if the named file cannot be opened in the user's working directory. When opening an existing print file, the file is opened in the append mode and the information is added at the end-of-file. Certain circumstances can produce a rather long print file. The print file can be quickly accessed and scrolled using available UNIX tools such as **more** or **tail.**

Future Developments

It appears that the netCDF access software provides the necessary support for future WHIPS software development. It has allowed the initial development of a general image processing software package and, more specifically, necessary side-scan sonar programs to be developed in a hardware independent environment and provides improved processing times. Future plans fall into four major areas:

1) to develop more general image processing support programs

2) to develop programs with an X-windows interface to display and interactively manipulate the images

3) to develop cartographic and registration techniques for building digital mosaics by utilizing the proj program (Evenden, 1990) that is part of the UNIX based software package MAPGEN (Evenden and Botbol, 1985) developed by the U. S. Geological Survey

4) to develop a GUI menu interface to the WHIPS programs. In addition to these plans to expand the WHIPS software, plans to install the netCDF software on the Branch's MicroVAX-II computer system to allow easier conversion of image files from the MIPS to the WHIPS format are considered.

APPENDIX A

WHIPS Program Documentation

UNIX man Pages

WHIPS Programs

dk2dk	create a new image file by extracting a sub-area from an existing image
derivative	compute first order difference of an image
flip	flip the samples in an image line
filter	apply a low-pass, high-pass, zero replacement or divide filter to an image
glo2whips	converts a raw side-scan sonar data image and header file to a netCDF image file for use with WHIPS
listdn	print the digital values contained in an image file
listhdr	list the contents of a side-scan sonar header in a WHIPS image
lowpass2b2	applies a 2 by 2 low-pass filter to an image
mrgnav	merges navigation and bathymetry information into the header of a side-scan sonar WHIPS netCDF image file
mipssonar	converts a raw side-scan sonar data image and header file to a netCDF image file for use with WHIPS
raw2whips	convert a raw image file to a WHIPS netCDF image file
restorehdr	modify an WHIPS netCDF image by adding the header of a side-scan sonar image reverse reverse the lines in an image shade applies a simple shading correction to an 8-bit side-scan sonar image
slant	correct a side-scan sonar image for water depth and slant-range geometry distortions
velocity	correct a side-scan sonar image for change in ship's velocity and any aspect ratio distortions that exist
whips2raw	convert a WHIPS netCDF image file to a raw image file
wtcombo	weighted combination of 2-4 input images
xhistgrm	display the histogram of an 8-bit image in an X-11 window

APPENDIX B

The following is an example “print” file created by program **filter**.

```
*************** filter ***************

Tue Nov 19 09:28:28 1991


INPUT FILE: triplej_16bit.cdf

      Image bittyp: 16

      Image Size:
            number of lines: 1086
         number of samples: 567


OUTPUT FILE: triplej.lpfz

      Image bittyp: 16

      Image Size:
            number of lines: 1086
         number of samples: 567

Boxcar size:
      number of lines = 5 number of samples = 5

A zero replacement filter was performed.

Valid Data Range: 1 - 32767

Minimum number of points needed for filtering was: 1

Addback = 0
Normalization Value = 0
Multiplication Value = 1
```

APPENDIX C

The following is summary list of the program key flags currently in use by the WHIPS programs and their possible meanings. The user should check the individual program documentation for a listing and definition of the available run-line options.

-i	input file (-i filename)
	(-i file1,file2,file3,file4)
-o	output file (-o filename)
-a	image area (-a sl,ss,nl,ns)
	addback values (-a 1,2,.5,-1)
-b	bittype specification: 8, 16 or 32 (-b n)
	boxcar (-b nl, ns)
-c	coefficients values (-c 1,2,.5,-1)
	course information
-f	fish_altitude
-h	heading
	header file (-h filename)
-j	julian day requested
-l	number of lines or linc (-l n or -l n.nn)
	delnl for velocity (-l delnl)
-m	minimum good
-n	normalization value (-n normval)
-p	percentage (-p minper,maxper)
-r	pixel resolution (-r resin,resout)
-s	number of samples or sinc (-s n or -s n.nn)
-t	tolerance ranges (-t val1,val2)
-v	valid data range (-v minval, maxval)
-y	year
-C	control file containing program information
-X	swab buffer before processing
-P	print file
-H	prints usage/help info
-S	sonar information requested

REFERENCES

Bothner, M.H., Parmenter, C.M., Twichell, D.C., and Polloni, C.F., 1991, A geologic map of the seafloor in western Massachusetts Bay, constructed from digital sidescan sonar images, photography, and sediment samples - assembled for CD-ROM production (abs): Gulf of Maine Scientific Workshop Jan 8-10, 1991, Woods Hole, MA.

Chavez, Pat S., 1984, U. S. Geological Survey Mini Image Processing System (MIPS): U.S. Geological Survey Open-File Report 84-880, 12p.

Condit, Christopher D., Chavez, Pat S., Jr., 1979, Basic Concepts of Computerized Digital Image Processing for Geologists: U.S. Geological Survey Bulletin 1462, 16 p.

EEZ Scan 85 Scientific Staff, 1987, Atlas of the U.S. Exclusive Economic Zone, Gulf of Mexico: U.S. Geological Survey Miscellaneous Investigation Series I-1864, p. A1-A104.

EEZ Scan 87 Scientific Staff, 1991, Atlas of the U.S. Exclusive Economic Zone Atlantic Continental Margin: U.S. Geological Survey Miscellaneous Investigations Series I-2054, 174 p.

Evenden, Gerald I. and Botbol, Joseph Moses, 1985, User's Manual for MAPGEN (UNIX version): a method of transforming digital cartographic data to a map, Open-File Report 85-706, 58 pages plus appendixes on font codes and map projections.

Evenden, Gerald I., 1990, Cartographic Projection Procedures for the UNIX Environment - A User's Manual, Open-File Report 90-284, 62 p.

Folger, D.W., Schlee, J.S., Foster, D.S., Polloni, C.F., Seekins, B.A., Brown, C.L. and Olson, A.C., 1991, Indiana Shoals: A dynamic sedimentary environment: U.S. Geological Survey Open-File Report 91-284, p. 24.

Karl, H.A., Schwab, W.C., Drake, D.E., Chin, J.L., Rubin, D.M., Twichell, D.C., and Edwards, B.D., 1990, Continental margin of the Gulf of the Farallones: geologic setting and depositional processes (abs.): AGU.

Treinish, Lloyd A., SIGGRAPH '90 Workshop Report: Data Structures and Access Software for Scientific Visualization, Volume 25, number 2, April 1991 p. 104 - 118.

Twichell, D.C., Schwab, W.C., Nelson, C.H., Lee, H.J., and Kenyon, N.H., 19XX, Characteristics of a sandy depositional lobe on the outer Mississippi Fan from SeaMARC 1A sidescan sonar images: Geology (Branch Chief approval, Sept., 1991, 13 ms pages).

Unidata Program Center, NetCDF User's Guide: An Interface for Data Access, v1.11, March 1991, 150p.