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

TABLE OF CONTENTS

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.

geometric:

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

radiometric:

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

derivative	compute first order difference of an image
dk2dk	create a new image file by extracting a sub-area from an existing image
filter	apply a low-pass, high-pass, zero replacement or divide filter to an 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
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

NAME

derivative - compute first order difference of an image

SYNOPSIS

derivative -i *input* **-o** *output* [**-h** | **-v** | **-d**] [**-a** *addback*] [**-H**]

DESCRIPTION

The derivative program computes a simple difference between adjacent pixels in an image. The derivative may be computed in one of three directions: horizontal, vertical or diagonal. The user must specify the direction of the derivative to be performed by selecting **-h**, **-v** or **-d** as part of the run-line.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8 or 16-bit image.

-o *output_file*

specifies the output file to be created.

-h | **-v** | **-d**

specifies which type of derivative to perform. Valid selections are horizontal (**-h**), vertical (**-v**) or diagonal (**-d**).

The *horizontal derivative* is processed on a line by line basis where the adjacent pixels of one single line are subtracted from each other across the image line. The image lines are processed one at a time starting with the first pixel of the image line to the last pixel of the image line with the following equation where ix is the sample of the image being processed.

line1[ix] = (line1[ix] - line1[ix+1]) + iaddback

A special case exists for the last sample of the image line because it has no adjacent value to be used in the computation. The last sample is computed as:

hder[ix] = hder[ix-1]

The *vertical derivative* is computed using two consecutive image lines. The sample locations of the image are subtracted from line to line. The first input image line is a special case scenario. This line is computed by subtracting the individual sample from itself and is computed as:

line1[ix] = (line1[ix] - line1[ix]) + iaddback

or more simply as:

line1[ix] = iaddback

The remaining image lines of the input file are processed as:

line2[ix] = (line1[ix] - line2[ix]) + iaddback

The diagonal derivative is computed by subtracting the offset sample of one line from the sample of another line. The first line of the image file is a special circumstance and is computed as a horizontal derivative. The remaining image lines are computed as:

line2[ix] = (line1[ix] - line2[ix+1]) + iaddback

A special circumstance exists for the last sample of each line. Those samples are computed as:

line2[ix] = line2[ix-1]

Options: The following run-line commands are optional to the execution of the program.

-a *addback*

specifies the addback value to be used by the program when computing the derivative. The default value is 127 for an 8-bit image and 4000 for a 16-bit image.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The program accepts only 8 or 16-bit image files.

The output file to be created must not currently exist.

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16-bit option of the program has not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

dk2dk - create a new WHIPS netCDF image file by extracting a sub-area from an existing image

SYNOPSIS

dk2dk -i *input* **-o** *output* **-a** *sl,ss,nl,ns* [**-l** *linc*] [**-s** *sinc*] [**-H**]

DESCRIPTION

The **dk2dk** (disk-to-disk) program will create a new image file by extracting a user specified sub-area from an existing WHIPS netCDF image file. The sub-area is selected by specifying the **-a** option on the program run-line. The image sub-area is specified by the starting line (*sl*), starting sample (*ss*), number of lines (*nl*) and number of samples (*ns*) to be extracted. Image area variables are specified relative to the image origin (sl=1 and ss=1) and is the upper left corner of the matrix.

The program can also be used to reduce or enlarge an input file by specifying the **-l** and/or **-s** run-line options.

The following run-line options must be specified and can occur in any order.

-i i*nput_file*

specifies the input file to be processed. The input file may be 8, 16 or 32-bit.

-o *output_file*

specifies the output file to be created.

-a *sl,ss,nl,ns*

specifies the sub-area of the input file to be extracted. The sub-area is specified by entering the origin as the starting line (sl) and starting sample (ss) of the area to be extracted and the number of lines (nl) and the number of samples (ns) to be extracted.

When specifying the sub-area, the user must specify the sl and ss values. The program will compute default values for the nl and ns parameters as the remaining lines and samples in the input image from the user specified starting position.

Option: The following run-line commands are optional to the execution of the program.

-l *linc*

specifies the line increment (*linc*) at which to output the lines from the input image sub-area. A value greater than one will reduce the input image sub-area. A value less than one will duplicate the input image lines and expand the image area selected. For example, a *line_increment* of 2 would result in every other line from the input sub-area being output. A *line_increment* of .5 would result in every line from the input image sub-area being duplicated for output. The default line increment value is 1.

-s *sinc*

specifies the sample increment (*sinc)* at which to output the samples from the input image sub-area. This option is similar to the *line increment* (**-l**) option. The default sample increment value is 1.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The output file to be created must not currently exist.

NOTES

This program should be used if the user wishes to extract and create a sub-area of the image portion of a WHIPS netCDF image file or a WHIPS netCDF side-scan sonar image. If the user wishes to extract a sub-area of the image portion of a WHIPS netCDF side-scan sonar image and retain the accompanying sonar header information for the image lines, the user should use program **ssdk2dk**.

If the user desires to extract only the image data from a WHIPS netCDF side-scan sonar image file and eliminate the sonar header, they may use program **dk2dk**.

EXAMPLE

The first example would extract the GLORIA side-scan sonar imagery from a file removing the header information.

```
% dk2dk -i gloria.slr -o gloria.sub -a 1,129
```

The second example would reduce the input file by transferring every other line and sample to the output file.

```
% dk2dk -i mickey.pic -o mickey.sub -a 1,1 -l 2 -s 2
```

SEE ALSO

ssdk2dk(1)

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16 and 32-bit options of the program have not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

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

SYNOPSIS

filter -i *input* **-o** *output* **-b** *nl,ns* [**-l** | **-z** | **-L** | **-h** | **-d**] **options** [**-H**]

DESCRIPTION

The filter program allows the user to select one of five filter operations and apply it to a WHIPS image. The filters are applied to the image by a moving boxcar. The boxcar (**-b**) is a user specified sampling size which is two odd integer values that are not necessarily equal. The values represent the number of lines and samples (*nl,ns*) to be considered when accumulating the sums. Pixel values at the center of the boxcar are modified and are affected by the surrounding valid values. The boxcar totals are applied over the image starting at line_1/sample_1 to line_n/column_n. The boxcar is shifted left to right over the image line.

Valid data are specified by the user selecting the run-line option **-v.** The data values entered by the user will define the valid data range for processing. Values less than the minimum value or greater than the maximum value are considered non-valid and are not included in the operation of computing the boxcar totals. Specifying a valid data range may have other impacts on the selected filter. See the specific filter operation described on the following pages.

Each pixel surrounding the center of the boxcar is compared to the low and high range before the filtering operation is done. If the value of the original pixel falls outside of the valid range, it is not included in the boxcar sum and count. Boxcar totals represent the total of the valid pixel values and the number of valid points surrounding the center of the boxcar. The original unchanged pixel values are used to calculate the boxcar totals.

A minimum number of valid data points (**-m**) are required to be contained within the boxcar totals before the filtering process takes place. If there are less points in the boxcar than the user specified minimum, the resulting dn value will be set to zero on output for all filters. The default minimum value for this option is 1. The user may specify a value greater than or equal to 1 and less than or equal to the total boxcar size (*nl * ns*).

The minimum valid data points which must be contained in the filter may also be specified by selecting a fraction (**-f**) of the boxcar which must contain valid data points. If this option is selected, the minimum valid points is computed as:

$$((nl * ns) * fraction) + .5$$

For example, a 5-by-5 filter with a .5 fraction specified would then have to contain a minimum of 13 valid data points in the boxcar totals for the filter to be applied. This would have the same result as specifying *-m 13,* but is easier to specify for large filters. If a fraction greater than one is specified, it is reset to one. If the computed value is less than one, it will be set to one as a default.

A coefficient value (**-c**) to expand the results of the data may also be specified. This is a real number which is used by all filters to expand the range of the results of the filter operations. The default value is 1.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the input file to be processed. The input file may be an 8, 16 or 32-bit image.

-o *output_file*

specifies the output file to be created.

-b *nl,ns*

specifies the size of the boxcar by the number of lines (*nl*) and the number of samples (*ns*).

-l | **-z** | **-L** | **-h** | **-d**

specifies the type of filter to perform. Valid filter selections are low-pass (**-l**), low-pass filter with zero replacement (**-z**), low-pass filter changing only valid data (**-L**), high-pass (**-h**) filter or divide (**-d**) filter.

The core to all the filtering operations is the computation of the *low-pass filter* (LPF). The LPF is a smoothing spatial filter which is good at reducing noise and removing the high-frequency content of an image. The LPF is computed by averaging the total valid pixels values in the pixel *neighborhood*. The *neighborhood* is the boxcar size specified by the user.

One consideration when developing a filtering program is how to deal with the edges of the image. As the boxcar begins and moves across the image, it is not properly centered and would not contain the proper sums. One possibility is to ignore the edges, thereby reducing the image content. The approach taken in this program is to compute the boxcar totals at the image edges by a folding/unfolding method. In essence, the program duplicates the neighboring pixels inside the edges, centered on the boxcar. As the boxcar moves away from the edge and across to the center of the image, these duplicated values are removed and replaced by pixel values from the center of the image. As the boxcar meets the right and bottom edge of the image, the pixel values inside the edge are slowly duplicated back as if to fold the edge of the image back over itself. The variables for the individual filter are defined as:

(i,j) - The image coordinate of the pixel being computed. For line 29 and sample 12, the image coordinate would be (29,12).

P(i,j) - The original pixel value of the input image at coordinate i,j.

S(i,j) - The sum of the points over the boxcar centered at i,j.

N(i,j) - The number of valid points within the boxcar surrounding the pixel being processed.

The LPF computation is defined below and is followed by a description of the individual filters.

$$LPF(i,j) = S(i,j)/N(i,j)$$

The ***low-pass filter*** (LPF) is a smoothing spatial filter. Only input image pixels values that fall within the user specified valid data range and processing boxcar size are totaled and averaged to produce the LPF component. If the minimum valid points (**-m** or **-f**) for the boxcar are not satisfied, the output pixel is set equal to zero. If the minimum valid points have been satisfied, the LPF is applied regardless of the original pixel value. In other words, this option will modify all input pixel values. The *low-pass filter* is computed using the following equation:

$$LPF(i,j) = S(i,j)/N(i,j)$$

$$X(i,j) = COEF*LPF(i,j)$$

If the sum of the boxcar or the number of valid points contained in the boxcar is zero, the value returned by the LPF computation will be zero.

The ***zero replacement filter*** (LPFZ) is a *low-pass filter* with one minor difference. That difference is that during the filtering process, only input pixel values equal to zero are modified. This allows the user an option to fill in “holes” based on the value of surrounding pixels. If the user does not specify a valid range for computing the boxcar totals, the minimum valid value is automatically set to 1 to eliminate zeros from the boxcar totals. The minimum valid value **MUST** be greater than zero.

The ***low-pass filter*** changing valid data only (LPFV) is also similar to the *low-pass filter* described above in the initial boxcar computations. The major difference is this option will only modify input pixel values that fall between the valid minimum and maximum values specified by the user. If an input pixel value is less than the specified minimum value, the output pixel is set to 0 for all filters. If the input pixel value is greater than the maximum value, the output pixel value is set to the maximum allowed value for the bit type. For an 8-bit image this value is 255, 16-bit is 32767 and 32-bit is set equal to FLT_MAX as defined in the file /usr/include/limits.h.

The ***high-pass filter*** (HPF) enhances the high-frequency details of an image. Edge enhancement of an image is also possible with the application of a *high-pass filter.* The HPF is computed using the low-pass filter described above. The boxcar values are computed by totaling the input image pixel values that fall within the valid data range. The *high-pass filter* (HPF) is computed as follows:

HPF(i,j) = NORM*(1-ADDBACK) + P(i,j)*COEF*(1+ADDBACK) - LPF(i,j)*COEF

Before computing the *high-pass filter,* the original pixel value is compared against the valid data range. The *high-pass filter* is applied to the image coordinate only if the original pixel value falls within the valid data range. If the original value is less than the specified minimum valid value, the output pixel is set equal to zero. If the input pixel value is greater than the maximum value, the output pixel value is set to the maximum allowed value for the bit type. For an 8-bit image this value is 255, 16-bit is 32767 and 32-bit is set equal to FLT_MAX as defined in the file /usr/include/limits.h.

The ***divide filter*** (DIV), when utilized by specifying a valid data range, will produce a binary image (0 or 255 values) similar to a mask image. When the LPF component of the filter is greater than the maximum valid value specified by the user, the input pixel will be output as 255. If the LPF component is computed as less than the minimum valid value specified by the user, the output pixel is zero. The resulting image would then be a “mask” of the valid values. The *divide filter* is computed as follows:

DIV(i,j) = COEF*(P(i,j)/LPF(i,j)) - NORM

The *divide filter* is applied to the image coordinate only if the original input pixel value falls within the valid data range. If the original value is less than the minimum value specified by the user, the output pixel value is set equal to zero. If the input pixel value is greater than the maximum value, the output pixel value is set to the maximum allowed value for the bit type. For an 8- bit image this value is 255, 16-bit is 32767 and 32-bit is set equal to FLT_MAX as defined in the file /usr/include/limits.h.

It is recommended that the user apply the resulting DIV “mask” with caution. In some cases, the “mask” outlines are not continuous. When the “mask” is applied to the image, the discontinuous lines can result in portions of the image, which are to be preserved, being dropped during the masking operation.

Options: The following run-line commands are optional to the execution of the program.

-a *addback*

specifies the addback value which is used by the *high-pass* and *divide filters* only.

-v *minval,maxval*

specifies the *minimum* and *maximum* values (minval,maxval) to be used to define the valid data range.

-c *coef*

specifies the *coefficient* (coef) to be used during the filtering process to expand the results of the data during filtering. The value specified may be a floating point value and may be any value the user desires. The default coefficient value is 1.

-n *norm*

specifies the *normalization value* (norm) used in the *high-pass* and *divide filter* computations. The default *normalization* value for 8-bit image data is 127. For 16 or 32-bit data, the default value is 0.

-m *mingood*

specifies the *minimum number of good points* (mingood) that must be contained within the boxcar before the filter is applied. The default value is 1. This option may be superseded by specifying **-f**.

-f *fraction_good*

specifies the *fraction of the boxcar* (fraction_good) that must contain valid data points before the filter is applied to the image coordinate. Specifying this option would override the **-m** option. The *fraction_good* is specified as the percentage of the boxcar that must contain valid data points before a filter can be applied.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The output file to be created must not currently exist.

SEE ALSO

/usr/include/limits.h

lowpass2b2(1), median3(1), mode3(1), mode5(1), ssfilter(1)

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 8 and 16-bit options have been extensively tested. The 32-bit option has not been fully tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

flip - flip the samples in an image line

SYNOPSIS

flip -i *input* **-o** *output* [**-H**]

DESCRIPTION

Flip will create a new output file with the order of the samples of the output image line flipped (reversed) from the order of the samples in the input image line. In other words, **flip** will output the last sample of the input line as the first sample in the output image and the first sample in the input image as the last sample of the output image. The entire line of the input image is processed by default.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file may be 8, 16 or 32-bit.

-o o*utput_file*

specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The output file to be created must not currently exist.

EXAMPLE

```
% flip -i mickey.pic -o mickey.flip
```

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16 and 32-bit options of the program have not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

glo2whips - converts a raw GLORIA sidescan sonar data image and ASCII header file to a netCDF WHIPS sidescan sonar image file

SYNOPSIS

glo2whips -i *input* **-h** *header* **-o** *output* **-l** *nl* **-s** *ns* [**-y** *base_year*] [**-H**]

DESCRIPTION

Program **glo2whips** converts a raw GLORIA sidescan sonar image and combines with it an ASCII file which contains the sidescan sonar header information to create a netCDF file for use with WHIPS. The sole purpose of the program is to assist in the transfer of GLORIA (Geologic Long-Range Inclined Asdic) sidescan sonar data processed on MIPS to a format which can be utilized by WHIPS.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

-h *header_file*

specifies the file that contains the ASCII header information.

-o *output_file*

specifies the output file to be created.

-l *nl*

specifies the number of lines (nl) contained in the image data.

-s *ns*

specifies the number of samples (ns) contained in the image data.

The program requires two input files from MIPS. The first file is the sidescan sonar header in an ASCII format file. This header information must have been printed from the MIPS image file using program **STRPHDR** or **LSTHDR** with the *secs* params option specified. The second file must contain only the sidescan sonar image data. In other words, the header information must have been removed from the MIPS file using the MIPS program **DK2DK**. The sonar image file, now minus the header information, should then be output of the MIPS program **EXPORT** using the *binnorec* params option. Program **EXPORT** will create a raw stream binary file for processing. By preparing the sidescan sonar image on MIPS in this manner, will allow the user to create the two necessary files which can then be transferred over the network to a UNIX workstation for further processing.

Actually, the data files may be created in any manner and on any system so long as they adhere to a few restrictions. The input image data (**-i**) must be contained in a raw stream file of packed byte information. The header file (**-h**) must contain one record for each line of image data. Each record must contain the following fields.

```
recno year month day hour min sec lat lon heading cor_depth uncor_depth
```

All fields must be present in each of the header records and they must be separated by one or more spaces. If there is no valid value for a field, the field must contain a zero value. All values are integer except the latitude (*lat*) and longitude (*lon*) fields which are float values. Latitude and longitude must be recorded as signed decimal degrees with west longitude and south latitude as negative. The seconds (*sec*) value of the time field may be an integer or float value.

The year value recorded in the ASCII header file is assumed to be two digits. The year value contained in the sidescan sonar header of the output image is a four digit value. This *baseyear* (**-b**) value is added to the two digit year read from the header file as it is transferred to the output header. The user may specify any value for the baseyear. If the user has 4 digit year values in the ASCII input header and does not want the baseyear value added to the value as it is transferred to the output header, they can override the value by specifying **-b** 0.

The uncorrected depth field from the ASCII input file is used to output the fish altitude field of the new sonar image. If the ASCII header file is created by some other means other than the MIPS programs **STRPHDR** or **LSTHDR**, make sure the appropriate value is recorded in the uncorrected depth field of the header records.

When processing the header file two fields, day of year (doyear) and total minutes (totmin) are computed and stored in the header for future use. These variables are needed by later processing programs.

Options: The following run-line commands are optional to the execution of the program.

-y *baseyear*

specifies the base year value to be added to the year value contained in the ASCII header file before the value is stored in the header of the output image. The default baseyear value is 1900. The main purpose of the baseyear is to convert the two digit year from the input header file to a four digit year. If the year field recorded in the ASCII header file is four digits, the user may override the addition of 1900 by specifying a baseyear value by entering -b 0 on the run-line.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

This program would be used mainly for transferring GLORIA sidescan sonar data from MIPS to WHIPS. The program is not appropriate for converting high-resolution sidescan sonar data because the time portion of the header information is recorded as minutes and seconds and generally does not provide enough accuracy for the high-resolution data unless the seconds values are recorded as float values.

No field checking is done. The program assumes all field values are correct and proper.

The program accepts only 8-bit image files.

The output file to be created must not currently exist.

EXAMPLE

```
% glo2whips -i gloria.raw -h gloria.hdr -o gloria.cdf -l 720 -s 992
```

The following is a listing of the first ten records of the g*loria.hdr* file.

```
1 85 10 9 5 0  0 25.39260 -84.82830 0 3363 3356
2 85 10 9 5 0 30 25.39135 -84.82775 0 3363 3356
3 85 10 9 5 1  0 25.39010 -84.82720 0 3363 3356
4 85 10 9 5 1 30 25.38885 -84.82665 0 3363 3356
5 85 10 9 5 2  0 25.38760 -84.82610 0 3363 3356
6 85 10 9 5 2 30 25.38635 -84.82555 0 3363 3356
7 85 10 9 5 3  0 25.38510 -84.82500 0 3363 3356
8 85 10 9 5 3 30 25.38385 -84.82446 0 3363 3356
```

```
  9 85 10 9 5 4  0 25.38260 -84.82390 0 3363 3356
 10 85 10 9 5 4 30 25.38135 -84.82337 0 3363 3356
```

SEE ALSO

mipssonar(1), raw2whips(1), WHIPS(5), whips_sonar(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

listdn - print the digital values contained in an image file

SYNOPSIS

listdn -i *input* [**-P** *print file*] [**-a** *sl,ss,nl,ns*] [**-l** *linc*] [**-s** *sinc*] [**-H**]

DESCRIPTION

The **listdn** program will display, in a tabulated form, the digital values contained in an image. By default, the information is displayed to the user's terminal (i.e. std_out), but may be output to a file by either re-directing the std_out information or by specifying the **-P** option on the runline. The program will allow the user to select a specific area of the image to be printed by utilizing the **-a** run-line option.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file may be 8, 16 or 32-bit.

Options: The following run-line commands are optional to the execution of the program.

-a *sl,ss,nl,ns*

specifies the sub-area of the input file to be printed. The sub-area is specified as the starting line (*sl*), starting sample (*ss*) of the area and the number of lines (*nl*) and the number of samples (*ns*) to be printed. If the sub-area is not specified, the entire image is printed. Depending on the size of the image, the lines printed could be very long.

If a sub-area is specified, the user must specify a minimum the *sl* and *ss* values. The program will compute default values for the *nl* and *ns* parameters as the remaining lines and samples in the input image from the user specified starting position.

-l *llinc*

specifies the line increment (linc) at which to output the lines from the input image sub-area. A value greater than one will result in the lines within the sub-area being skipped on output. A value less than one would result in the input lines being duplicated. For example, a *line_increment* of 2 would result in every other line from the input sub-area being output. A *line_increment* of .5 would result in every line from the input image sub-area being duplicated for output. The default value is 1.

-s *sinc*

specifies the sample increment (sinc) at which to output the samples from the input image sub-area. This option is similar to the line increment (*-l*) option. The default value is 1.

-P *print_file*

specifies the print file to re-direct the output information to.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

none known

EXAMPLE

The first example will display the first ten lines and ten samples from the input image.

```
% lisdn -i mickey.pic -a 1,1,10,10
```

The second example will display five hundred lines of one sample of data, sample location 100, and store the information in file core.dat.

```
% listdn -i core.pic -P core.dat -a 1,100,500,1
```

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16 and 32-bit options of the program have not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

listhdr - list the contents of a side-scan sonar header in a WHIPS netCDF side-scan sonar image

SYNOPSIS

listhdr -i *input* [**-l** *linc*] [**-j**] [**-S**] [**-P** *print file*] [**-H**]

DESCRIPTION

The **listhdr** program will list the header information from a side-scan sonar image. By default, the line number, year, month, day, hour, minute, seconds, latitude, longitude, heading and fish altitude information is displayed. If a field contained in the side-scan sonar header is printed that does not contain valid information, *inf* will be output in the appropriate field. The *inf* reflects that the field has been filled with system dependent infinity value for that data type.

The information is displayed on the user's terminal but may be output to a file by re-direction of std_out or selecting the **-P** option on the runline.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

The input file must contain the necessary side-scan sonar header information. If the user selects a file which does not contain the proper side-scan sonar header information, the program will display the message **ncvarid: variable "date" not found** and the processing will stop.

<u>Options</u>: The following run-line commands are optional to the execution of the program.

-l *linc*

specifies the line increment (*linc*) at which to output the lines from the input file. The default *linc* value is 1.

-j

flags the program to replace the day and month information with the day of year value in the output.

-S

flags the program that, in addition to the default information, output the sonar pitch roll and yaw information.

-P *print_file*

specifies the print file to re-direct the output information to.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The input file selected must contain the required side-scan sonar header information.

EXAMPLE

The example will output the side-scan sonar header from file bos13.slr and output the information to bos13.hdr. The first ten records of the header file are the listed below.

```
% listhdr -i bos13.slr -P bos13.hdr

% head bos13.hdr

 1: 1985 10  9  5  0  0.000  25.392599  -84.828300  0.0  3356.00
 2: 1985 10  9  5  0 30.000  25.391350  -84.827751  0.0  3356.00
 3: 1985 10  9  5  1  0.000  25.390100  -84.827202  0.0  3356.00
 4: 1985 10  9  5  1 30.000  25.388849  -84.826653  0.0  3356.00
 5: 1985 10  9  5  2  0.000  25.387600  -84.826103  0.0  3356.00
 6: 1985 10  9  5  2 30.000  25.386351  -84.825546  0.0  3356.00
 7: 1985 10  9  5  3  0.000  25.385099  -84.824997  0.0  3356.00
 8: 1985 10  9  5  3 30.000  25.383850  -84.824463  0.0  3356.00
 9: 1985 10  9  5  4  0.000  25.382601  -84.823898  0.0  3356.00
10: 1985 10  9  5  4 30.000  25.381350  -84.823372  0.0  3356.00
```

SEE ALSO

replacehdr(1), strphdr(1)

WHIPS(5), whips_sonar(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

ncvarid: variable “date” not found - the selected input file does not contain the proper side-scan sonar header information

BUGS

none known

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

lowpass2b2 - applies a 2 by 2 low-pass filter to an image

SYNOPSIS

lowpass2b2 -i *input* **-o** *output* [**-H**]

DESCRIPTION

Program **lowpass2b2** applies a small low-pass smoothing filter to an image. The low-pass filter consists of a 2-by-2 moving boxcar. The image is smoothed by the filter with the boxcar applied starting at the upper left origin of the image and moving across each line of input data and down through the input image. With the exception of the first line and ending sample of each line, the filter is applied by computing the average of 4 neighboring pixels with the result being stored in the lower left pixel. The filter is applied to the entire image.

An example of how the program computes the pixel averages is as follows:

```
                input          output
            -------------   ------------
    line 1:|  11  |  22  |  |  11 |  22  |
            -------------   ------------
    line 2:|  33  |  44  |  |  28 |  44  |
            -------------   ------------
```

The first input image line is a special case and is smoothed by applying a 1-by-2 low-pass filter.

```
            input                      output
        -------------------     --------------------
line 1: | 11 | 22 | 33 | 44 |    | 17 | 28 | 39 | .. |
        -------------------     --------------------
line 2: | 33 | 44 | 55 | 66 |    | 28 | 39 | 50 | .. |
        -------------------     --------------------
```

The last sample of each line is also a special case and is processed as a 2-by-1 low-pass filter.

```
            input                  output
        --------------         ---------------
line 1: | 11 | 22 | 44 |        | 17 | 33 | 44 |
        --------------         ---------------
line 2: | 33 | 44 | 66 |        | 28 | 88 | 55 |
        --------------         ---------------
line 3: | 55 | 66 | 88 |        | 50 | 66 | 77 |
        --------------         ---------------
line 4: | 77 | 88 | 99 |        | 72 | 85 | 94 |
        --------------         ---------------
```

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

-o *output_file*

specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The program accepts only 8-bit image files.

The output file to be created must not currently exist.

EXAMPLE

```
% lowpass2b2 -i gloria.vel -o gloria.2b2
```

SEE ALSO

filter(1), median3(1), mode3(1), mode5(1)

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

mrgnav - to merge navigation and bathymetry information into the header of a sidescan sonar WHIPS netCDF image file

SYNOPSIS

mrgnav -i *input* **-n** *navigation* **-o** *output* [**-f**] [**-H**]

DESCRIPTION

Program **mrgnav** will update the header information of a sidescan sonar image by merging the latitude and longitude coordinates from a user specified navigation file. The new sidescan sonar positions will be computed by a linear interpolation based upon the date and time contained in the sidescan sonar header.

The program requires two input files. The first input file (**-i**) must contain the WHIPS netCDF sidescan sonar image to be updated. The second input file (**-n**) must contain the navigation file to be used for the update. The navigation file is an ASCII formatted file with the necessary fields separated by one or more spaces. A more detailed description of the navigation file follows.

Optionally, the user may select to have the fish_altitude information updated. This is done by specifying the run-line command option **-f**.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

-n *navigation_file*

specifies the file that contains the ASCII navigation information. The navigation data must contain the following fields: latitude, longitude, year, month, day, hour, minute, seconds and, optionally, fish_altitude. The fields must be in the order listed above and separated by one or more spaces. The fields year, month, day, hour and minute must be recorded as integer values. The latitude, longitude, seconds, and fish_altitude may be recorded as integer or float values. The latitude and longitude must be specified as signed decimal degrees, and may contain any number of decimal places the user desires.

-o *output_file*

specifies the output file to be created. The output file is a modified version of the input image file with the updated header information. The actual header information modified is dependent upon the run-line option selected by the user. By default, the latitude and longitude information from the navigation file will always be merged and placed in the sidescan sonar header by the program. The user has the option to specify whether to add the fish_altitude information from the navigation file to the header area. The default is to update only the latitude and longitude coordinates of the sidescan sonar header. A sample listing of a navigation file can be found under EXAMPLE.

Options: The following run-line commands are optional to the execution of the program.

-f

flags the program to interpolate and merge the fish_altitude values from the navigation file into the sidescan sonar header. The default is **NOT** to interpolate a new fish_altitude value for the output file. This option allows flexibility in processing sidescan sonar data that may already have been slant-range corrected or for times when depth information is not required or available. This option may also be useful if the sidescan sonar header already contains fish_altitude information that the user does not wish to have overwritten. *If the* **-f** *option is specified, the fish_altitude information contained in the header information, good or bad, will be updated with newly interpolated values.*

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

Program **MRGNAV** will begin by reading the first two navigation records and the date and time of the first sidescan sonar record. The program will compare the date and time information of the image record with those of the navigation records. If the date and time match, the positions and depth values are output to the image file. If the program determines it cannot interpolate the values based upon the records it currently has in memory, it will read the next record from the appropriate file. If the appropriate file is the sidescan sonar image file, the current image record is output UNCHANGED before the next image record is read. If the image record falls within the navigation records the latitude/longitude coordinates and fish_altitude values are interpolated. The new positions and fish_altitude values are computed and added to the sidescan sonar header. These values are computed by a linear interpolation based on the date and time of the sonar record as it falls between the appropriate navigation data.

After the program execution, the user should check the print file for error messages. In the rare chance that the user has specified an incorrect navigation file it is possible to process the entire image file and not modify any records. This could yield some interesting results if the user further attempts slant-range and delta velocity corrections. MRGNAV will output to the print file the number of image records modified during the program execution. If the number of records modified does not equal the number of image records input, an error message is also output.

RESTRICTIONS

The output file to be created must not currently exist.

EXAMPLE

```
% mrgnav -i bos13.cdf -n boshrbr.nav -o bos13.mrg
```

The following is a listing of the first ten records of the boshrbr.nav file. This example does not contain the optional fish_altitude information. The seconds field may be specified as either an integer or floating point value.

```
42.34570  -70.79530  1989  4  3  5 20  0
42.34550  -70.79530  1989  4  3  5 20  6
42.34540  -70.79530  1989  4  3  5 20 18
42.34520  -70.79510  1989  4  3  5 20 30
42.34500  -70.79500  1989  4  3  5 20 36
42.34480  -70.79500  1989  4  3  5 20 48
42.34460  -70.79490  1989  4  3  5 21  0
42.34450  -70.79490  1989  4  3  5 21  6
42.34430  -70.79490  1989  4  3  5 21 18
42.34400  -70.79490  1989  4  3  5 21 30
```

SEE ALSO

WHIPS(5), whips_sonar(5)

Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System: U.S. Geological Survey Open-File Report 92-204, 11p.

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

mipssonar - converts a raw sidescan sonar data image and header file to a WHIPS netCDF sidescaon sonar image file for use with WHIPS.

SYNOPSIS

mipssonar -i *input* **-h** *header* **-o** *output* **-l** *nl* **-s** *ns* [**-c**] [**-y** *base_year*] [**-H**]

DESCRIPTION

Program **mipssonar** converts a raw sidescan sonar image and combines with it an ASCII file which contains the sidescan sonar header information to create a netCDF file for use with WHIPS. The sole purpose of the program is to assist in the transfer of sidescan sonar data processed on MIPS to a format which can be utilized by WHIPS.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

-h *header_file*

specifies the file that contains the ASCII header information.

-o *output_file*

specifies the output file to be created.

-l *nl*

specifies the number of lines (nl) contained in the image data.

-s *ns*

specifies the number of samples (ns) contained in the image data.

The program requires two input files from MIPS. The first file is the sidescan sonar header in an ASCII format. This header information must have been printed from the MIPS image file using program LSTHDR with the no params option specified. By not specifying the params option, the minutes portion of the time field is output as a floating point value. The second file must contain only the sidescan sonar image data. In other words, the header information must have been removed from the MIPS file using the MIPS program DK2DK and should be output of the MIPS program EXPORT using the binnorec params option. Program EXPORT will create a raw stream binary file for processing. By preparing the sidescan sonar image on MIPS in this manner, will allow the user to create the two necessary files which can then be transferred over the network to a UNIX workstation for further processing.

Actually, the data files may be created in any manner and on any system so long as they adhere to a few restrictions. The input image data (**-i**) must be contained in a raw stream file of packed byte information. The header file (**-h**) must contain one record for each line of image data and must contain the following fields:

```
recno year month day hour minute lat lon heading cor_depth uncor_depth
```

All fields must be present in the header records and they must be separated by one or more spaces. If there is no valid value for a field, the field must contain a zero value. All values are integer except the minutes (minute), latitude (lat) and longitude (lon) fields which are float values. Latitude and longitude must be recorded as signed decimal degrees with west longitude and south latitude as negative.

The year value recorded in the ASCII header file is assumed to be two digits. The year value contained in the sidescan sonar header of the output image is a four digit value. This baseyear (**-b**) value is added to the two digit year read from the header file as it is transferred to the output header. The user may specify any value for the baseyear. If the user has 4 digit year values in the ASCII input header and does not want the baseyear value added to the value as it is transferred to the output header, they can override the value by specifying **-b** 0.

Generally, the high-resolution sidescan sonar data processed on MIPS does not contain true depth values and the depth fields of the sidescan sonar header are filled with the netCDF float fill values. The uncorrected depth field contained in the ASCII input file is more accurately referred to as the fish altitude and is output as the fish altitude field of the new sonar image. If the ASCII header file is created by some other means other than the MIPS program LSTHDR, make sure the appropriate value is recorded in the uncorrected depth field of the header records.

When processing the header file two fields, day of year (doyear) and total minutes (totmin) are computed and stored in the header for future use. These variables are needed by later processing programs.

Options: The following run-line commands are optional to the execution of the program.

-c

flags the program to output the heading field contained in the header file (-h). In most cases the heading value contained in the header of the MIPS images for the high-resolution sidescan sonar data has been incorrect. By default this program will not read the heading information, but rather will output the netCDF fill value in the field. Outputting the field with the fill value will flag other programs that the data contained in the heading field is not available. If the heading field contained in the header file is valid and the user wishes the information to be output as part of the sidescan sonar header, the user must specify this option as part of the program run-line.

-y *baseyear*

specifies the base year value to be added to the year value contained in the ASCII header file before the value is stored in the header of the output image. The default baseyear value is 1900. The main purpose of the baseyear is to convert the two digit year from the input header file to a four digit year. If the year field recorded in the ASCII header file is four digits, the user may override the addition of 1900 by specifying a baseyear value by entering -b 0 on the run-line.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

This program can be used for transferring high-resolution sidescan sonar data because the minutes portion of the time field provides much more accuracy. In comparison, program **glo2whips** should be used for transferring GLORIA sidescan sonar data from MIPS to WHIPS. Program **glo2whips** is not appropriate for converting high-resolution sidescan sonar data because the time portion of the header information is recorded as minutes and seconds and does not provide enough accuracy for the high-resolution data.

No field checking is done. It is assumed all field values are correct and proper.

The program outputs only 8-bit image files.

The output file to be created must not currently exist.

EXAMPLE

```
% mipssonar -i bos13.raw -h bos13.hdr -o bos13.cdf -l 4853 -s 1024
```

The following is a listing of the first ten records of the file bos13.hdr file.

```
 1 89 4 6 3 40.60822 42.40889 -70.78445 79 0 12
 2 89 4 6 3 40.61508 42.40889 -70.78445 79 0 12
 3 89 4 6 3 40.62195 42.40888 -70.78446 79 0 12
 4 89 4 6 3 40.62881 42.40887 -70.78446 79 0 12
 5 89 4 6 3 40.63431 42.40886 -70.78447 79 0 12
 6 89 4 6 3 40.64117 42.40886 -70.78447 79 0 12
 7 89 4 6 3 40.64804 42.40885 -70.78448 79 0 12
 8 89 4 6 3 40.65491 42.40884 -70.78448 85 0 12
 9 89 4 6 3 40.66177 42.40884 -70.78448 85 0 12
10 89 4 6 3 40.66864 42.40883 -70.78448 85 0 12
```

SEE ALSO

mipssonar(1), raw2whips(1), WHIPS(5), whips_sonar(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

raw2whips - convert a raw image data to a netCDF file for use with WHIPS

SYNOPSIS

raw2whips -i *input* **-o** *output* **-l** *nl* **-s** *ns* [**-b** *bittyp*] [**-H**]

DESCRIPTION

Program **raw2whips** converts a raw binary stream image file to a netCDF file for use with WHIPS.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the binary stream input file to be converted. The input file may be 8, 16 or 32-bit.

-o *output_file*

specifies the netCDF output file to be created.

-l *nl*

specifies the number of lines (nl) or rows contained in the image data.

-s *ns*

specifies the number of samples (ns) or columns contained in the image data.

Options: The following run-line commands are optional to the execution of the program.

-b *bittyp*

specifies the bit type (bittyp) of the data being processed. Valid selections are 8, 16 or 32. The default bittyp is 8.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

Currently, the program assumes the image data is in the proper bit and byte order for the selected bit type. No swabbing takes place.

The output file to be created must not currently exist.

EXAMPLE

```
% raw2whips -i mickey.raw -o mickey.cdf -l 480 -s 472
```

SEE ALSO

whips2raw (1), WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

There appears to be a small problem when transferring files with less than 512 samples from MIPS unless the number of samples are evenly divisible by four. If this is a bug or merely a quirk, is not entirely known. One way around the problem is to either enlarge or truncate the file on MIPS so the number of samples is evenly divisible by four. Then create the raw image with the MIPS program **EXPORT**, transfer the file and convert it with **raw2whips**. Files with image lines greater than 512 don't seem to be a problem.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

restorehdr - modify a WHIPS image file by adding the header information from a selected WHIPS sidescan sonar image

SYNOPSIS

restorehdr -i *input* **-h** *header* [**-H**]

DESCRIPTION

Restorehdr will modify the selected input image (**-i**) by adding the sidescan sonar header from a second file (**-h**). This is necessary because some programs utilized in the processing of sidescan sonar data are general image processing programs and do not handle the processing of the header information which is stored separately from the image data. The header must be restored to the sidescan image to complete the sonar processing. The program will not create a new output file. The selected input file (**-i**) will be **MODIFIED**.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

-h *header_file*

specifies the file which is to contain the sidescan sonar header to be transferred.

The selected file must contain the necessary sidescan sonar header information. If the user selects a file which does not contain the proper sidescan sonar header information, the program will display the message **ncvarid: variable “date” not found** and the processing will stop.

Options: The following run-line commands are optional to the execution of the program.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The input file and header file must contain the same number of lines. No other comparitive checks are done to the files.

EXAMPLE

```
% restorehdr -i bos13.wco -h bos13.shd
```

SEE ALSO

WHIPS(5), whips_sonar(5)

Digital Processing of Side-Scan Sonar data with the Woods Hole Image Processing System: U.S. Geological Survey Open-File Report 92-204, 11p.

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

ncvarid: variable “date” not found - the selected input file does not contain the proper sidescan sonar header information

BUGS

None known.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

reverse - reverse the lines in an image

SYNOPSIS

reverse -i *input* **-o** *output* [**-H**]

DESCRIPTION

Reverse will create a new output file with the order of the lines reversed from the input file. In other words, reverse will make the first line of the input image the last line of the output image and the last line of the input image the first line of the output image. The entire input file is processed by default.

The following run-line options must be specified and can occur in any order.

-i *input_file*
: specifies the input file to be processed. The input file may be 8, 16 or 32-bit.

-o *output_file*
: specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-H
: displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The output file to be created must not currently exist.

EXAMPLE

```
% reverse -i mickey.pic -o mickey.rev
```

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16 and 32-bit options of the program have not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA

NAME

shade - applies a simple shading correction to an 8-bit sidescan sonar image

SYNOPSIS

shade -i *input* **-o** *output* [**-t** *minval,maxval*] [**-n** *normval*] [**-H**]

DESCRIPTION

Program **shade** applies a simple shading correction to an 8-bit sidescan sonar image and differs from program shadex in the way the coefficients are computed. The program operates in a two-pass processing phase. The first pass computes the totals for each sample position of the image and then computes the average value for each sample. Those averages are then used to compute coefficients for the sample locations and are used for image shading corrections. The second pass applies the computed coefficient to the input DN values.

The program computes the sum of each sample position by totaling the samples of every other line from the input image. Values which fall between the user tolerance limits (**-t)** are totaled into the values for the sample locations. Once the values are totalled, the average value for the sample locations are computed by dividing the sum by the number of points which fell within the specified tolerance limits (e.g. the number of valid pixels). The normalization value (**-n**) or, if not specified by the user, the average of the averages is divided by the sample average to compute the sample coefficients. A 1x11 recursive low-pass filter is then applied to the coefficients to smooth the data before the coefficients are applied to the image data.

As each input record is read during the second-phase of the processing, the input value is multiplied by the computed coefficient for that sample location.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

The input file must contain the necessary sidescan sonar header information. If the user selects a file which does not contain the proper sidescan sonar header information, the program will display the message **ncvarid: variable “date” not found** and the processing will stop.

-o *output_file*

specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-t *minval,maxval*

specifies the tolerance range for valid data which the program will use to compute the sample totals. Data values which fall outside the specified range are not totaled into the sample sums. Default values are 0,255.

-n *norm_val*

specifies the value to be used to normalize the coefficients. If a value is not specified, the average of the sample averages is used.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

Program currently accepts only 8-bit image files.

The output file to be created must not currently exist.

The program was written specifically to process sidescan sonar data, and as such, the input file must contain the necessary header information. Though the program does not require the header information for the actual processing, it is read from the input file and transferred to the output file unaltered. If it were necessary to apply the shading algorithim to the image data only, the program could be easily modified to apply the shading to the image data and eliminate the transfer of the header information.

The sample totals are computed using every other line from the input file for speed. Totalling the sample from every input line appears to have little change on the computed coefficients.

EXAMPLE

```
% shade -i gloria.slr -o gloria.shd -t 1,254
```

SEE ALSO

shadex, WHIPS(5), whips_sonar(5)

Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System: U.S. Geological Survey Open-File Report 92-204, 11p.

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

ncvarid: variable “date” not found - the selected input file does not contain the proper sidescan sonar header information

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

slant - correct side-scan sonar image for slant-range geometry distortions

SYNOPSIS

slant -i *input* **-o** *output* [**-r** *resin,resout*] [**-H**] [**-S** *scale_factor*]

DESCRIPTION

The **slant** program will correct a side-scan sonar image by removing the water column and applying a slant-to-ground range correction to the image to produce an new image with the proper across-track geometry. Using the side-scan sonar fish altitude information contained in the header, along with the image resolution values (**-r**), the program will calculate the true ground range for each pixel in a scan. These calculations are computed assuming a flat seafloor.

The user may select a *scale_factor* (**-S**) to be applied to the fish_altitude values and pixel resolutions during processing. For most cases, the *scale_factor* would simply be specified as 10. Specifying a *scale_factor* rounds-up calculations made during the water column removal and assists in the eliminating some jumps (that football affect in the water column). This option can be helpful in processing high-resolution sidescan data that is towed close to the seafloor. With fish_altitude values typicall ranging from 6.0 to 7.0 meters, a fish_altitude value of 6.2 provides considerable accuracy. However, when computing the number of pixels to be removed from the water column, only whole pixels can be removed. Scaling the fish_altitude and pixel resolution values can help in rounding-up some of the calculations thereby reducing some of the affects that might be visible from dropping a pixel from the water-colum correction and producing a better looking image. The user may wish to test this option on a sample data set before applying it to their entire data set.

The following run-line options must be specified and can occur in any order.

-i *input*

specifies the input file to be processed. The input file must be 8-bit.

The input file must contain the necessary side-scan sonar header information. If the user selects a file which does not contain the proper side-scan sonar header information, the program will display the message **ncvarid: variable “date” not found** and the processing will stop.

-o *output*

specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-r *resin,resout*

specifies the input and output pixel resolution, in meters, to be used in computing the slant-to-ground range correction. The default values for the program reflect the values required for processing 30-second GLORIA side-scan sonar data. These default values are 45 meters for the input resolution and 50 meters for the output resolution.

-S *scale_factor*

Specifies a scale factor applied to the fish_altitude and pixel resolution values. The *scale_factor* must be specified as an integer value.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

The program only accepts 8 or 16-bit image files.

The output file to be created must not currently exist.

EXAMPLE

```
% slant -i bos13.mrg -o bos13.slr -r .2,.5
```

SEE ALSO

WHIPS(5), whips_sonar(5)

Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System: U.S. Geological Survey Open-File Report 92-204, 11p.

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

ncvarid: variable "date" not found - the selected input file does not contain the proper side-scan sonar header information

BUGS

The 16-bit option has not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

velocity - correct a sidescan sonar image for changes in ship's velocity and any aspect ratio distortions that exist

SYNOPSIS

velocity -i *input* **-o** *output* [**-r** *pixres*] [**-l** *delnl*] [**-h** *delhdg*] [**-H**]

DESCRIPTION

Velocity will correct a sidescan sonar image for velocity changes and aspect ratio distortions that exist. Program variables were defined with the default values to be equivalent to those required to process 30-second GLORIA sidescan sonar data.

The velocity and aspect ratio corrections are applied, by default, for every 60 lines of data (**-l**). For a 30-second GLORIA sidescan sonar image this computes to a 30 minute data segment. As each segment is processed, the heading for the line segment is computed and compared to the heading for the previously processed segment. When the heading between the two segments differ by more than the user specified delta-heading (**-h**), the line segment to be processed is reduced to one-third the delta lines. For a 30-second GLORIA image, this value is 20 lines or 10 minutes of GLORIA data. This heading comparison is done to reduce the effects of course changes in the data. The next segment processed returns to the delta line specification unless the delta heading value is again exceeded. Processing continues in this manner till the end-of-file is encountered.

The default values are not at all appropriate for high-resolution sidescan data. In many cases, high-resolution data is recorded at a rate of several scans per second and a delta line specification of 60 is equivalent to a mere several seconds. The user should take care when specifying the program parameters for high-resolution sidescan.

The Clarke 1866 spheroid values are used to compute the distance travelled for the line segments. How much an individual file is stretched can differ slightly depending on the delta line and delta heading values specified by the user.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file must be 8-bit.

The input file must contain the necessary sidescan sonar header information. If the user selects a file which does not contain the proper sidescan sonar header information, the program will display the message **ncvarid: variable "date" not found** and the processing will stop.

-o *output_file*

specifies the output file to be created.

Options: The following run-line commands are optional to the execution of the program.

-r *pixel_resolution*

specifies the pixel resolution, in meters, in the along-track direction of the output image to be created. The default value is 50 meters.

-l *delta_line*

specifies the number of lines to be used as a processing segment. The default value is 60.

-h *delta_heading*

specifies the heading value that if exceeded from processing segment to processing segment, the processing segment will be one third the specified delta_lines.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

If the program aborts during processing, the temporary files created in /tmp will not be removed. If the program aborts, the user should check the directory and remove the unecessary files. If the files are not removed, they could impact available disk space and further processing.

RESTRICTIONS

The program currently accepts only 8-bit image files.

The output file to be created must not currently exist.

When processing begins, the program does not know how much an individual input file will be stretched during the processing and how many lines will be needed to create the output file. To accomodate this, the program creates a temporary file while applying the velocity correction. As the input file is processed, the program duplicates the necessary image and header lines and outputs the stretched image to a temporary file while it totals the number of lines which will be contained in the new output file. When the velocity correction is completed, the output file is created with the proper number of lines and the image and header lines are transferred from the temporary files to the specified output file. This approach results in the processing time being longer than it would if the data could be written directly to the WHIPS netCDF file.

EXAMPLE

The first example shows a typical run-line to process a 30-second GLORIA sidescan sonar image.

```
% velocity -i gloria.res1 -o gloria.vel
```

The second example shows the possible run-line options to process a strip of high-resolution sidescan sonar data.

```
% velocity -i bos13.res1 -o bos13.vel -r .5 -l 150
```

FILES

`/tmp/VELOCxxx` - Temporary file created during processing to hold image data.

`/tmp/VELHDxxx` - Temporary file created during processing to hold header data.

SEE ALSO

WHIPS(5), whips_sonar(5)

Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System: U.S. Geological Survey Open-File Report 92-204, 11p.

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

ncvarid: variable "date" not found - the selected input file does not contain the proper sidescan sonar header information

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

whips2raw - convert a WHIPS netCDF image file to a raw image file

SYNOPSIS

whips2raw -i *input* **-o** *output* **-l** [**-H**]

DESCRIPTION

Program **whips2raw** converts a WHIPS netCDF image file to a raw binary stream image file.

The following run-line options must be specified and can appear in any order.

-i *input_file*

specifies the netCDF file to be processed.

-o *output_file*

specifies the binary stream input file to be converted. The input file may be 8, 16 or 32-bit.

Options: The following run-line commands are optional to the execution of the program.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

none known

EXAMPLE

```
% whips2raw -i mickey.cdf -o mickey.raw
```

SEE ALSO

raw2whips(1), WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

none known

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

wtcombo - weighted combination of 2-4 input images

SYNOPSIS

wtcombo -i *input1,input2,..* **-o** *output* [**-c** *coef1,coef2,..*] [**-a** *add1,add2,..*] [**-n** *normval*] [**-H**]

DESCRIPTION

Program **wtcombo** performs a weighted combination of two or more input images. This program can be utilized to perform various operations such as ratioing input images and combining them to create one output file or subtracting one image from another. The program flexibility is in the user's selection of input file coefficients (**-c**), file add back values (**-a**) and an overall normalization value (**-n**).

In general, a new output buffer value is computed from four input files as:

```
dn_out  =  ((dn_in1 + add1) * coef1) +
           ((dn_in2 + add2) * coef2) +
           ((dn_in3 + add3) * coef3) +
           ((dn_in4 + add4) * coef4) +
           normval
```

Each input file is assigned an individual coefficient (i.e. *coef1, coef2*) and an individual addback value (i.e. *add1,add2*). The normalization value (*normva*l) is applied to the sample sum of the files. The program does not check the user entered coefficient, addback or normalization values entered by the user and may be any value the user wishes to specify.

The following run-line options must be specified and can occur in any order.

-i *input_file1,input_file2,input_file3,input_file4*

specifies the input files to be processed. A minimum of two input files must be specified and a maximum of four input files may be specified. The input files must all be the same size and bit type.

-o *output_file*

specifies the output file to be created.

Options; The following run-line commands are optional to the execution of the program.

-c *coef1,coef2,coef3,coef4*

specifies the coefficient values for the individual files. The pixel values from each input file will be multiplied by the specified coefficient. The default coefficient values are 1.0 and may be any value specified by the user.

-a *add1,add2,add3,add4*

specifies the addback values for the individual files. The pixel values from each input file will have the value added to it after it has been scaled by the appropriate coefficient. The default addback values are 0 and may be any value specified by the user.

-n *normval*

specifies the normvalization value to be added to the sums of the newly computed dn values. The default value is 0 and may be any value specified by the user.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

A minimum of two input files must be specified and a maximum of four input files may be specified.

The input images must be the same size and bit type.

The output file to be created must not currently exist.

The program processes only the image portion of the WHIPS netCDF file. If the files being processed contain additional information such as a side-scan sonar header, the output file created will have to be updated with the header information by restorehdr.

EXAMPLE

The first example shows one way to combine the 8-bit data values from two images.

```
% wtcombo -i gloria.lpf,gloria.hpf -o gloria.wco -c 1,1 -n -127
```

The second example show another possible way to combine the 8-bit data values from two images.

```
% wtcombo -i file1.pic,file2.pic -o new.pic -c .5,.5
```

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

The 16 and 32-bit options have not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

xhistgrm - display histogram of 8-bit image in an X-11 window

SYNOPSIS

xhistgrm -i *input* [**-v** *minval,maxval*] [**-H**]

DESCRIPTION

Program **xhistgrm** will compute the histogram and various statistics from the image portion of the specified 8-bit WHIPS image and display the histogram and the image statistics in an X-11 window. In addition to displaying the image statistics in the X-11 window, the statistics are output to the program print file (xhistgrm.-prt). The statistics include mean, mode, standard deviation, skew and one, fifty and ninety-nine percent values. The valid data range used to compute the statistics and the number of zeroes and 255 values contained in the image are also included for the user's information. The image statistics are computed by scanning the image and totalling only the data that falls within the user specified valid data range (**-v**). By default, this data range is 1 - 255.

The following run-line options must be specified and can occur in any order.

-i *input_file*

specifies the input file to be processed. The input file may only be 8-bit.

Options: The following run-line commands are optional to the execution of the program.

-v *minval,maxval*

specifies the minimum and maximum values to be used to define the valid data range. The default valid data range is 1-255 and eliminates any zeroes that may be contained in the image.

-H

displays the usage help information for the program. If this option is selected, the program ignores any other run-line options specified.

RESTRICTIONS

none known

EXAMPLE

```
% xhistgrm -i gloria.cdf
```

SEE ALSO

WHIPS(5)

DIAGNOSTICS

The program exit status is 0 if no errors are encountered during processing and the program completes processing.

BUGS

none known

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

WHIPS - Woods Hole Image Processing System

DESCRIPTION

The Woods Hole Image Processing System is a raster based image processing software packaged developed utilizing the netCDF software from UNIDATA. The netCDF software provides the interface for file definition and data acess for which the WHIPS software was based. The general WHIPS software which has been developed processes the image portion of the data file only.

The basic WHIPS netCDF image file is defined using two netCDF dimensions and two netCDF variables. The dimensions **nl** (*number of lines or rows*) and **ns** (*number of samples or columns*) define the two dimensional matrix which contains the image data. The image data is further defined by the **bittyp** (*bittype*) variable which may be 8, 16 or 32. The **bittyp** variable indicates the data type for the image data. A bittyp of 8 would indicate 8-bit data with data values ranging from 0-255. The 8-bit data may also be considered byte or character data. A bittyp of 16 would indicate image data stored as short integer data values ranging from 32767 thru -32768. The final bittype, 32, would indicate image data stored as 32-bit floating point values. The netCDF defined global attribute which contains the creation program name is optional. An example of the basic **image** file is listed below. In this short example, the image data are stored as byte data, and the image DN values are printed in their octal representation.

```
netcdf image_file {
dimensions:
         nl = 10 ;
         ns = 5 ;

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

// global attributes:
                  :creation_program = “program_name_goes_here” ;

data:

  bittyp = 8 ;

 image =
  0230, 0250, 0250, 0256, 0255,
  0230, 0244, 0250, 0255, 0254,
  0235, 0246, 0252, 0257, 0260,
  0235, 0253, 0253, 0256, 0256,
  0236, 0257, 0256, 0256, 0255,
  0237, 0253, 0255, 0260, 0262,
  0232, 0250, 0251, 0260, 0267,
  0235, 0252, 0252, 0265, 0267,
  0240, 0255, 0261, 0264, 0271,
  0240, 0261, 0263, 0272, 0271 ;
}
```

Though the WHIPS software was intended to provide general image processing capabilities, the need to process side-scan sonar data was also an issue. The basic WHIPS image did not provide the necessary information required for various sonar processing steps and was expanded to include additional information. This information, which is referred to as the side-scan sonar header, includes date, time and sonar fish position was added. The header information is read and written by the side-scan sonar specific processing programs only. General image processing programs may be applied to the WHIPS netCDF 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. When data for a specific header variable is not available, the field should be filled with the netCDF fill value for that data type. This fill value may then be used as a flag to indicate invalid or missing data rather as required by various processing programs.

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

date - date variables (year, month, day, day_of_year)

time - time variables: hour, minute, totmin

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 data variables 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. Future development may require that these variables be increased and current dimension sizes would be accessible through the netCDF file.

An example of a small (5 lines by 3 samples) WHIPS netCDF side-scan sonar image printed using the netCDF utility program **ncdump** is listed below. Note the FloatInf values contained in the pitch, roll and yaw variables of the ss_atributes data. This is an example of utilizing the netCDF fill values to indicate that valid data is not present for a specific data variable.

```
netcdf side-scan_image {
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 = “program_name_goes_here” ;
                  :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 ;
}
```

SEE ALSO

whips_sonar(5)

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

Woods Hole Image Processing System Software Implementation: Using NetCDF as a Software Interface for Image Processing: U. S. Geological Survey Open-File Report 92-25, 74p.

NAME

whips_sonar.h - sidescan sonar header structure and constants

SYNTAX

#include "whips_sonar.h"

struct whips_sonar_info *sonar = &whips_sonar_info_struct;

struct sonar_cdf_info *sscdf = &sonar_cdf_info_struct;

DESCRIPTION

This file contains constant values and two structures utilized by the WHIPS software when processing the sidescan sonar header. The first structure is used to store the sidescan sonar header information for the specific image record being processed. The second structure is used to contain the netCDF variable id's for reading or writing the header information to and from the WHIPS netcdf image files. The file contains the following information.

```
/*
** whips_sonar.h:
**
** Declaration of sonar header info for WHIPS
**
** ***************************************************************
** *               WHIPS_SONAR                                   *
** ***************************************************************
*/

/*
** This section defines the sonar header information for
** any given scan. The variables for the specific
** record should be stuffed into this structure for processing.
** The structure may be filled by read_sshdr or by the user
** during processing. The information contained in the structure
** is output to the WHIPS netCDF side-scan sonar image file by
** sshdr_out.
*/

struct whips_sonar_info
{
/*
**         date information **
*/

  short int year;                 /* should be 4 digit year */
  short int month;
  short int day;
  short int doyear;               /* day of year */

/*
**         time information **
```

```
*/

  short int hour;
  short int minute;
  short int totmin;               /* total minutes of day */

  float seconds;

/*
**        side-scan sonar attributes **
*/

  float fish_alt;                 /* fish altitude */
  float heading;
  float pitch;
  float roll;
  float yaw;

/*
**        depth information for GLORIA side-scan sonar **
*/

  float depth_corrected; /* meters */
  float depth_uncorrected; /* meters */

  double lat;                     /* lat & lon in decimal degrees */
  double lon;

} whips_sonar_info_struct;

/*
** ---------------------------------------------------------------
*/

struct sonar_cdf_info
{

/*
** This section declares the ‘id’ variables used for the
** various header information contained in the netCDF
** files and required for the side-scan sonar header information.
*/

  int date_idin;
  int time_idin;
  int secs_idin;
  int ss_attr_idin;
  int latlon_idin;
  int depth_idin;
```

```
    int date_idout;
    int time_idout;
    int secs_idout;
    int ss_attr_idout;
    int latlon_idout;
    int depth_idout;

} sonar_cdf_info_struct;
```

The *year* is stored as a 4 digit value due to the coming new century and to allow proper recording of the year values.

The *month* and *day* are re-computed and stored as the *day_of_year* for convenience to avoid conversion which might take place during sonar processing.

The *totmin* is computed from the variables *hour* and *minute* and stored for convenience.

SEE ALSO

glo2whips(1), listhdr(1), mipssonar(1), restorehdr(1), sshead(1), sumss(1)

WHIPS(5)

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

Woods Hole Image Processing System Software Implementation: Using NetCDF as a Software Interface for Image Processing: U. S. Geological Survey Open-File Report 92-25, 74p.

Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System Software: U. S. Geological Survey Open-File Report 92-204, 11p.

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.