UNITED STATES DEPARTMENT OF THE INTERIOR
U. S. GEOLOGICAL SURVEY

MAPIT: An improved method for mapping digital sidescan sonar data using the Woods Hole Image Processing System (WHIPS) Software

by

Valerie Paskevich[1]

Open-File Report 96-281

July 1996

[1] Marine and Coastal Geology Program
Woods Hole, MA 02543

CONTENTS

Abstract

The Marine and Coastal Geology Program, Northeast Section located in Woods Hole, MA has been involved in collecting, processing and digitally mosaicking sidescan sonar data utilizing the Woods Hole Image Processing System (WHIPS) software since 1992. Program MAPIT, which is described here, replaces the previously used WHIPS programs for digital mapping of sidescan sonar data while reducing the required processing time to 20-25% over the previously used method. A detailed description of the programs utilization along with a mapping example is included.

Introduction

Since 1992 the Marine and Coastal Geology Programs, Northeast Section has utilized the Woods Hole Image Processing System (WHIPS) software (Paskevich,1992a) to process and digitally mosaic sidescan sonar data. This locally developed software package was written specifically for processing high and low-resolution sidescan data collected by the field office, and has been installed on a variety of UNIX workstations. The focus of the WHIPS software is to provide a suite of programs that may be initiated from a run-line command to accomplish pre-processing requirements and digital mapping of the unique data set. When a production run of collected and processed field data is required, a UNIX script file containing the processing requirements for a given data set is created and executed by the user.

This new mapping program, **mapit**, conceptually is similar to the procedure described in U.S. Geological Survey Open-File Report 92-536 (Paskevich,1992c) but is more efficient computationally. As in the previous mapping method, the sonar pixels are processed on a one-by-one basis. In other words, the geographic coordinate for each sonar pixel from the input file must be calculated. The computed geographic coordinate must then be projected to it's Cartesian coordinate based on the map projection and earth model specified by the user, and once the Cartesian coordinate is know, it's position in the output file (i.e. it's image coordinate) must then be computed. This can easily result in the need to compute the geographic to Cartesian to image position for millions of points in a single input file. The new program and old mapping method have been compared, and the new program has reduced the processing time to 20-25% over the previous method.

There are two major changes and improvements over the previously described method. The first major change is that program **mapit** combines the features of programs **gss_vel** and **projss** and incorporates cartographic projection routines from **proj** (Evenden, 1990). This change alone is responsible for the significant increase in speed and time required to map a sidescan sonar swath. The second change is the definition of the mapping control file. Previously the user needed to create a file that contained the four corner geographic bounds of the image to be created. Now the user need only define the upper left and lower right geographic coordinates of the map area, but they must now also include the map projection information (e.g. projection name, ellipsoid) in **proj** format.

To begin the mapping procedure, the sidescan sonar data must be in the WHIPS format (Paskevich,1992) which utilizes the Unidata NetCDF data access software (Unidata, 1991). The WHIPS software was developed utilizing the NetCDF data access software to provide platform independent data files due to the wide variety of UNIX workstations being utilized at the Northeast field center. It is assumed that prior to starting this procedure, the user has properly formatted and processed the sidescan data to be mapped. The initial processing should include all radiometric and geometric corrections required by the data set such as slant-to-ground range correction, striped noise removal, shading, and beam angle correction. This document does not discuss the pre-processing requirements of sidescan data as that can be found elsewhere, for example, Chavez (1986), Miller et al (1991), Paskevich (1992b). Paskevich (1992b) provides an in-depth discussion of pre-processing sidescan data utilizing the UNIX based WHIPS software.

To digitally map the sidescan data swath, the sidescan file must include the sonar data header with the necessary sonar fish information. More specifically, the sidescan header must have the fish position recorded in the proper coordinates and a heading for each scan. Since a portion of program **mapit** is based on program **gss_vel** (see documentation in Appendix A), the consecutive positions stored in the sonar header must be unique. Program **mapit** will compute how many times to duplicate a given sonar swath to properly fill gaps between consecutive swaths.

In addition to the fish position, a heading value stored in the sidescan header is also required to properly compute the line orientation from the fish position. A heading logged from the sidescan fish would be desirable. However, if no heading is logged by the sidescan system, ship heading or heading computed from the fish position stored in the header may also be used. This can be computed using **sshead** (Appendix A).

As with the previous mapping method, one drawback to the digital mapping still remains. This is the holes or gaps that may occur from the placement of consecutive lines of sonar data. Since the heading values are critical in placing the sonar pixels, fluctuations in headings which can occur between the adjacent lines can cause the lines to overwrite one another rather than being placed neatly alongside each other. The overwriting of the adjacent lines and the resulting gaps can be minimized by smoothing the fish navigation using **avg_position** (Appendix A) and heading using **avg_heading** (Appendix A) contained in the sonar header before beginning the digital mapping procedure. However, this will not provide a perfect solution and filtering of some sort to fill in the holes in the final image is still required.

As stated earlier, one major change in the new mapping procedure involves program **proj** (Evenden, 1990). The previous mapping method utilized **proj** as a UNIX filter. By utilizing **proj** in that manner all projections available in **proj** were available to the user wanting to map their sidescan data. However, not all the projections were appropriate to our mapping needs. Therefore, program **mapit** has selected specific projections from the **proj** cartographic library and made only those available to the user. The sub-set of the available projections was implemented to reduce the size of the compiled program and therefore the memory required to run the program. A list of the selected projections available in **mapit** are listed in Table I with the required **proj** parameters if any. The list is not meant to be a definitive description of the various projections. It is included as a quick reference. For more detailed information on the projection requirements, the user is directed to Evenden, 1990 and 1994. As usual, the user should fully understand the requirements of the selected projection.

The user should also be familiar with the operation of the UNIX program **proj** as they will be required to specify the projection parameters utilizing **proj** parameters. When specifying the projection parameters for program **mapit**, the user may reference the **proj** documentation. The basic parameter specifications for **mapit** are similar to the run-line options for **proj**. The major change is that the '+' prefix for the **proj** commands are eliminated. For example, when running the UNIX program **proj** and the user wanted to specify the mercator projection, the projection was specified as ***+proj=merc*** on the **proj** run-line. This **proj** parameter would now be specified simply as ***proj=merc*** in the **mapit** control file (see example in Appendix C).

It is also the user's responsibility to specify the ellipsoid to be used in the mapping procedure. A list of available ellipsoids is shown in Table II. If no ellipsoid is specified as part of the **proj** initialization, **proj** will default to the Clarke 1866 spheroid.

The mapping example provided here is the same example (mapping GLORIA sidescan sonar data) as described in the previous digital mapping documentation, U.S.G.S. OpenFile Report 92-536. The GLORIA data processed has an input pixel resolution of 45 meters. The final map/image created is a 100 meter resolution for a 2° by 2° area bounded by -114° to -112° longitude and -34° to -36° latitude. Utilizing the same example will allow for easier comparison and clarification of changes in the mapping procedure and programs.

Program Summary

The following is the list of WHIPS programs that the user may need to complete the sidescan sonar digital mapping. Some of the listed programs are not used in the mapping procedure but have been included for reference.

avg_heading	apply a running average to the heading values contained in a WHIPS netCDF sidescan sonar image header
avg_position	apply a running average to the fish positions contained in a WHIPS netCDF sidescan sonar image header
filter	applies a low-pass, high-pass, zero replacement or divide filter to an image
listhdr	list the contents of a sidescan sonar header in a WHIPS image
mapit	computes the geographic coordinates for each pixel contained in a WHIPS sidescan sonar image and projects the geographic coordinates to it's Cartesian equivalent and places the pixel in a WHIPS image file
median3	applies a 3-by-3 median filter to a WHIPS image
mode3	applies a 3-by-3 mode filter to a WHPS image.
mode5	applies a 5-by-5 mode filter to a WHIPS image
qmos	mosaics (overlays) the specified input file over the specified output file. The program will either overlay where the input file has priority over the existing output file, where the output file has priority over the input file, or average non-zero pixel values together from the input and existing output file.
sshead	computes a simple heading for a WHIPS netCDF sidescan sonar image
whips2raw	converts a WHIPS NetCDF image file to a raw, binary image file

Digital Mapping Procedure

To begin the mapping procedure, the user must make some preliminary decision that will define the map area to be created. These decisions include selecting the map projection, spheroid and map bounds. As previously mentioned, program **mapit** utilizes a sub-set of the **proj** cartographic library. The user should reference Table I to select from the list of available map projections.

The **proj** specifications and geographic bounds of the map area must be entered into a text file by the user. Program **mapit** will access this file as it's control file (***-c***) and utilize the information in the **proj** initialization and image creation. The control file must contain three lines of information that are summarized as follows:

1) the **proj** (projection) parameters
2) the upper left geographic coordinates of the map
3) the lower right geographic coordinates of the map

The geographic coordinates specified on lines two and three are required to define the map area and may be defined in the ***DMS*** (Degrees, Minutes and Seconds) system acceptable to the **proj** cartographic library. The ***DMS*** system has been previously defined (Evenden, 1990) but will be summarized here for convenience.

The ***DMS*** system is defined as two numeric values separated by "white space" (either blanks or tabs). In it's simpler format signed degrees (e.g. -72, 72W or 72w) is acceptable. Signed decimal degrees (e.g. 42.5, 42.5N or 42.5n) is also acceptable. However, for more accuracy a typical geographic coordinate may be defined as 42°25'15.22"N. Since there is no degree symbol available in the ASCII character set and imbedded blanks are not allowed, the equivalent DMS value for the coordinate may be expressed as 42d25'15.22"N or 42d25'15.22. The second example illustrates that the suffix letter and trailing seconds (") sign may be eliminated because the coordinate is located in the northern hemisphere and therefore positive. If the same latitude coordinate were in the southern hemisphere, it may have been specified as 42d25'15.22"S, -42d25'15.22, or 42d25'15.22s. Please note that the d and ' are only required when respective minute and second subfields are employed.

For the mapping example, the **mapit** control file referred to as ***mapit.dat*** is shown below. A comparison of the mapping control files and mapping procedure from the previous mapping procedure and the control file required by mapit is shown in Appendix B.

```
proj=utm ellps=clrk66 south lat_0=-110
-34 -114
-36 -112
```

A second example of defining a simple **mapit** control file follows. This example illustrates a map defined as a mercator projection utilizing the WGS84 ellipsoid, and geographic coordinates defined in the ***DMS*** format acceptable to **proj**.

```
proj=merc ellps=WGS84 lat_ts=41
41d13'n 71d54'w
41d11'n 71d50'w
```

***NOTE: WGS84** is the ellipsoid used by the AMG navigation system.

Once the user has the **mapit** control data file created, they may execute the mapping program. The program has 4 required keywords that are listed below. There are additional, optional keywords the user may specify on the program run-line. The user may reference the **mapit** program documentation found in Appendix A for a full description of the program's keywords and their usage.

-i ***filein***	specifies the WHIPS sidescan sonar input file
-o ***fileout***	specifies the WHIPS output file
-c ***control_file***	specifies the mapit control file
-r ***resin,resout***	specifies the pixel resolution of the sidescan input data and the output file to be created

The run-line specification to process one line of GLORIA data (pass12.avg_head) from the mapping example may now be specified as:

```
mapit -i pass12.avg_head -o east.map -c mapit.dat -r 45,100
```

The GLORIA swath mapped from this run-line is shown in Figure 1. The mapped swath illustrates small gaps that can be left between adjacent lines. These holes are due to the heading fluctuation and may have been minimized by further smoothing the heading values prior to mapping. But additional smoothing of the heading values will never entirely eliminate the digital mapping gaps because the lines are not perfectly straight. These gaps are best filled by filters designed to properly fill the gaps, and can be applied as a final processing step once all the sonar swaths are mapped. The entire modified processing script utilized to produce the three (east.map, west.map and south.map) final 2° by 2° GLORIA mosaic is shown in Appendix C.

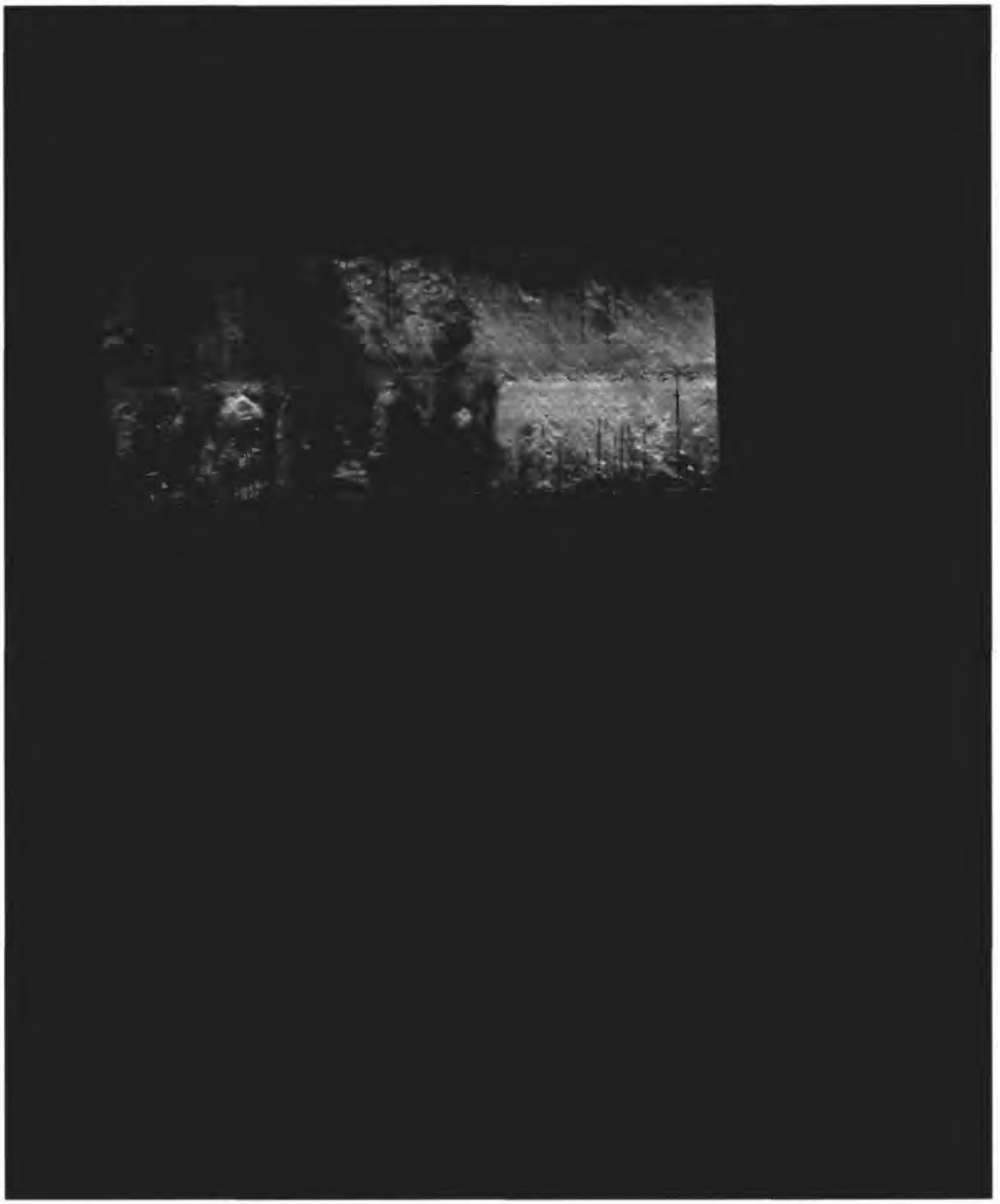


Figure 1 - GLORIA pass12 placed in image/map space

Another change from the previous mapping procedure involves the output file specification (**-o**) in program **mapit**. The new program will allow the user to write to a file that may already exist. Previously, the user needed to create a new output file each time the mapping procedure ran. This resulted in numerous files, all of the same size, being created. Though there were trade-offs in that approach, program **mapit** will now allow the user to open an existing output file and, if it passes two initial tests, allow the file to be updated. An existing output file will be opened and updated if:

- the selected output file contains the same number of lines and pixels as computed by program **mapit** from the user specified **proj** parameters
- the bittype of the output file matches the bittype of the input file

If the existing output file fails either of these two tests, the program will display an error message and stop. Allowing the user to update an existing output file will reduce the number of files that are needed during the mapping phase. Updating an existing file also eliminates the need to execute program **qmos** to combine map files and additionally reduces the amount of disk space needed and the time to complete the digital mapping process. The user will now be able to create one data file for a sonar line that may consist of two or more input files, or create one file for all lines oriented in the same direction. As before, it is the user's responsibility to maintain the bookkeeping to ensure data are not incorrectly overwritten in the output file.

The user should also take note of the program's print file, ***mapit.prt***, created during the program execution. The program print file, as with all the WHIPS program print files, provides an overview of the program's execution parameters and specific program information. The print file should not be deleted until the digital mosaicking procedure is completed. An example of the **mapit** print file is shown on the following page.

```
***************   mapit   ***************

 Wed Jun 12 12:00:23 1996


INPUT FILE: pass12.avg_head

      Image bittyp: 8

      Image Size:
            number of lines:   720
          number of samples:  1024


OUTPUT FILE: east.map

      Image bittyp: 8

      Image Size:
            number of lines:  2256
          number of samples:  1870


Map corner locations:

     37,    1 --------------------     1, 1848
              |                    |
              |                    |
              |                    |
              |                    |
              |                    |
              |                    |
              |                    |
              |                    |
   2256,   67 --------------------  2219, 1870

Pixel Resolution: 45.00(input)     100.00(output)

MAPIT CONTROL FILE: mapit.dat

   Proj parameters: proj=utm ellps=clrk66 south lon_0=-111
   Upper left lat/lon: -34 -114
         (y/x meters): 6233979 222902
   Lower right lat/lon: -36 -112
          (y/x meters): 6015789 409868

The -R option was selected.
```

The ***mapit.prt*** file is of particular importance because it provides two important sets of information the user will need in the digital mosaicking phase. The first is the size of the created map file, in lines and pixels. The user will need this information to create the PCI database file when beginning the digital mosaicking procedure. Secondly, the print file also reports the location of the map bounds within the image file. Depending on the map projection specified, the map area may not be rectangular as would be the case when specifying a mercator projection. The defined map area may be cylindrical as with a UTM projection. If this is the case, the geographic bounds will not occur exactly at the image corners. Some map bounds may

map bounds may fall within the image area while others may fall exactly on the image corners. Because the defined image must be rectangular, the map image file is created with the largest size possible to allow placement of the map within the image file. The ***mapit.prt*** file will tell the user where the geographic coordinates are located in the image. The transformed x and y meter coordinates of the map area's latitude and longitude coordinates are also included. Knowing where the specified geographic coordinates are in the image file will allow the user to properly specify the georeferencing parameters of the map in the PCI database when the digital mosaicking procedure is completed.

The recommended approach to accomplishing the digital mapping phase is to focus on creating separate mosaics with alternating lines. This means initially creating two, or more if necessary, image map files. This approach is taken to keep adjacent lines from overwriting each other, and keeping data which might be good from being overwritten by bad data or noise.

When all the sonar swaths for a given map area have been digitally mapped, any gaps or holes that may exist between the lines of the individual swaths must be filled. The user has the option of running a variety of filters (e.g. mode, median, zero replacement) to fill the gaps. The gaps are considered zero values as is the image area where no data is mapped. By filtering the image to fill these 'holes', the image will be slightly enlarged along the far range of the swaths. This is an area of noise that will be removed from the image during the digital mosaicking phase and is not of concern. Figure 2 is the east component (shown reduced) of the GLORIA mosaic example after all appropriate east lines have been combined and a series of filters have been applied to fill the gaps. Note the swaths now show a more distinct edge at the far range. This is where the noise has been enhanced as a result of the filtering.

Figure2 - GLORIA east passes mapped & filtered

When all the sonar swaths for a given map have been digitally mapped and the gaps filled, the image maps can be combined by stenciling unwanted data from the swaths and mosaicked to build the final digital mosaic. In the case of the GLORIA data used in the example, three mosaics were created. One for each of the lines (ship's) major heading direction: east, west and south.

The final mapped images for the remaining two areas, west and south, are shown, reduced, below. These images have also been filtered to fill any 'holes' left from the mapping procedure. The three files, east, west and south, will be combined to create the final digital mosaic. An overview of the digital mosaicking procedure follows in the next section.

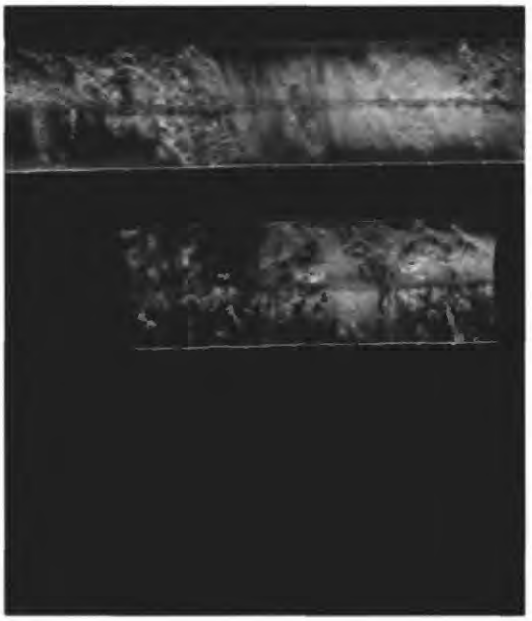


Figure 3 - GLORIA west passes mapped & filtered

Figure 4 - GLORIA south pass mapped & filtered

Digital Mosaicking Procedure

Mosaicking is the process of combining several arbitrarily shaped images to form one large, radiometrically balanced image. In the case of sidescan sonar, the images are the adjacent swaths collected during the survey. The mapped swaths are combined together along user-specified cut lines defining the polygon and image precedence (what part of the image to keep; what part of the image to remove). Ideally, the cut-line boundaries can be blended together to reduce the boundary between images so they are not easily seen.

Examining the east, west and south images the user can see where data contained in adjacent swaths would overwrite each other if they had been simply composited together as part of the mapping procedure. Additionally, the noise at the far range of the images makes a simple composite method undesirable. Typically each mapped swath may have some undesirable data that could be replaced by overlapping data from an adjacent swath. The user may wish to remove the unwanted portion of the imagery and select data which he or she feels is best. To support digital mosaicking, a cut and paste method to composite the images is preferred. A variety of methods and computer programs are available to accomplish a *simple* cut and paste compositing and, if needed, tone matching of the files. Two such programs are Adobe Photoshop (Adobe Systems Incorporated, 1994) and Corel PhotoPaint (Corel Corporation, 1994). However, these programs do not support the more sophisticated geometric corrections required to register images, and are simply mentioned as alternative methods of a elementary cut and paste method. Detailed description of these programs is outside of the scope of this report and will not be included here.

Geometric Correction where the registration of one image to another takes place. Ground Control Points (GCPs) are collected, and the uncorrected (slave) image is transformed to the georeferenced (master) image. To accomplish this, more sophisticated computer programs that allow for rubber sheeting and transformation of the slave image according to the derived polynomial are needed. To complete geometric correction and mosaicking, the Northeast field center utilizes the **PCI Remote Sensing** software which includes program **GCPworks**. This program is one of several components of the PCI Remote Sensing software package, and is specifically written to support georeferencing, rectification and mosaicking of remote sensing imagery.

In addition to **GCPworks**, the Northeast field center utilizes a number of additional PCI software packages to support general remote sensing enhancement and analysis. These programs are available to the user via either an X-11 interface, X-Pace, or a simple ASCII interface named EASI. Many of the PCI programs are used as analytical tools after completing the digital mosaic. However, a few programs are needed to set up the PCIdatabase and import the WHIPS created mosaic prior to beginning the digital mosaicking process with **GCPworks**. These programs, **CIM** and **WHIPSRD**, are described below as part of the PCIdsk creation prior to beginning **GCPworks**.

PLEASE NOTE: The following notes relate specifically to completing the digital mosaicking using the PCI Remote Sensing software package. For sites that may have other remote sensing software packages, such as ERDAS or ERMAPPER, changes to the process will be required. However, some general information and suggestions are included that may be useful if utilizing other software packages.

To begin the digital mosaicking phase, the user must import the individual images into a PCI database file for processing. This is done after all the sonar swaths have been mapped. To begin, the user must define a simple PCI database file of the same size (number of lines and pixels) as was created in the mapit mapping procedure. This is done using the PCI X-Pace program **CIM** (Create **Im**age database). When defining the PCI database, the user must also specify the number of image 'channels' to be contained in the database. It is suggested, that the user create the database with enough channels to hold the mapped images, **plus one additional working channel** (i.e. in the example shown in Figs 2, 3 & 4, 4 channels would be needed). The working channel will be used to produce the final digital mosaic. When the mosaic is completed, the channels holding the mapped data may be deleted or overwritten. An annotated example of defining the PCIdsk data file for the GLORIA mosaic is shown below utilizing the EASI parameters. The filename suffix, '*pix*', is the default identifier for a PCI database file and must be specified as part of the filename. For more information regarding the EASI keywords, the user may reference the PCI program documentation or utilize the on-line HELP feature of the PCI X-Pace programs.

```
EASI> FILE="gloria.pix"      <-- name of file to be created
EASI> DBSZ=1870,2256         <-- number of pixels & lines in image from mapit.prt file
EASI> PXSZ=100,100           <-- output pixel resolution in meters also from mapit.prt file
EASI> TEX1 = "GLORIA sidescan sonar mosaic"
EASI> TEX2= "Juan Fernandez microplate"
EASI> DBNC = 4               <-- 3 maps + 1 working channel
EASI> DBLAYOUT = "Band"      <-- band or pixel; band recommended
```

After the PCI database file has been created, each of the maps created by **mapit** must be imported into the PCI database. A program has been added to the **PCI** Image Interchange category to support the exchange of the data from WHIPS to **PCI**. This program, **WHIPSRD**, has been added to the **PCI** program library and may be found in the Image Interchange category of the X-Pace programs.

If the **WHIPSRD** program is not available, a second, two-step procedure is available to import the file to the PCI database. First, the user must convert the WHIPS program to a raw binary stream image file. This is done by utilizing the program **whips2raw**. The output file created from this program may then be directly imported to the PCI database utilizing their **IMAGERD** program. For sites that do not have the **WHIPSRD** program or the PCI software, most other software packages provide some similar method of importing foreign data files.

When all the maps have been imported to the PCI database file, the user must make a decision as to which of the mapped images will be considered the 'registered' or master image. This means that when doing the geo-referencing and stenciling procedure features from the other images and swaths will be tied and matched to the features in master image. This may result in warping and rubber sheeting of some of the mapped imagery in an attempt to align features but minimizes the amount of warping by keeping one of the images constant.

Once the user has selected which of the images will be the master, the image should be copied to the remaining empty channel in the PCIdsk file. This is the channel that will be used to build the final composite mosaic. One of the existing image channels may be selected as the master and used without copying. However, copying the image to a second work channel provides a backup in case an error occurs in the processing and can eliminate the need to reload the image into the database.

To copy the image to the working channel, the user has easily three methods. They may select either of the two methods used above and re-read the selected WHIPS file directly into the working channel. The third and more direct option is to utilize the X-Pace program **III** to accomplish a simple image transfer. Program **III** will allow the user to specify the input channel to be read and the output channel to store the image copy thereby producing a quick copy of the imagery.

Once the PCIdsk file has been created and the WHIPS images have been imported to the database file, the user may begin the process of digital mosaicking and registering utilizing program **GCPworks**. Though sidescan sonar mosaicking was not a typical application of the **GCPworks** software developers, the program has proven highly effective in fulfilling the needs of the Marine and Coastal Surveys Program to produce final digital mosaics.

Program **GCPworks** provides a number of functions critical to completing the final mosaic. In addition to the basic function of collecting ground control points to align features on adjacent swaths, the program provides support for mosaic stenciling, image feathering along the stencil cut-line to make seams disappear, and interactive registration preview. Real-time Root Mean Square error scatter plot reports along with model fit reports also provide useful information to determine best model and order fit before applying the registration and mosaicking to the disk files. For a more detailed description of program GCPworks, the user is referred to the available GCPWorks Reference Manual (November 1995). This PCI manual provides detailed description of the program's theory, usage and application.

The last step of building the completed mosaic is highly interactive and requires more of the user's time and attention than was needed in the digital mapping phase. The final digital mosaic is built in a repetitive process of registration and stenciling by selecting the necessary pieces from the slave image(s) (one at a time) to incorporate into the master image. When selecting the mosaic cut-line, the user may select what portion of the image to show or replace with overlapping data. Additional options to reduce the effect of the mosaic cut-line on the image and tone matching to produce a more uniform image are also available.

An example of the windows and usage of **GCPworks** is shown below. The screen 'snapshot' was taken during a mosaicking phase of the program's application to the GLORIA data. Windows in the upper right of the 'image screen' show GCPworks main menu and the 'Mosaic Area Collection' pop-up window. To the left of this area, the two top windows show an overview of the slave (unreferenced) image and the master image (georeferenced image). Two windows below these show the imagery at full resolution around the area of the cursor from each of the overview windows.

This view of GCPworks was taken after some of the digital mosaicking steps were completed. First, the slave file has been geo-referenced to the master. Note the +nn marks denoting the location of the selected ground control points. Secondly, a stencil line (yellow line) can be seen in the overview and full resolution windows. The image data contained within the stencil area from the 'Uncorrected Image Overview' (slave) will be placed within the stencil area of the 'Georeferenced Image Overview' (master) file. Only one stencil line can be drawn at a time, so only one track from the slave database can be registered and mosaicked at a time. It also has been found to be difficult to pick GCP's for more than one track at a time.

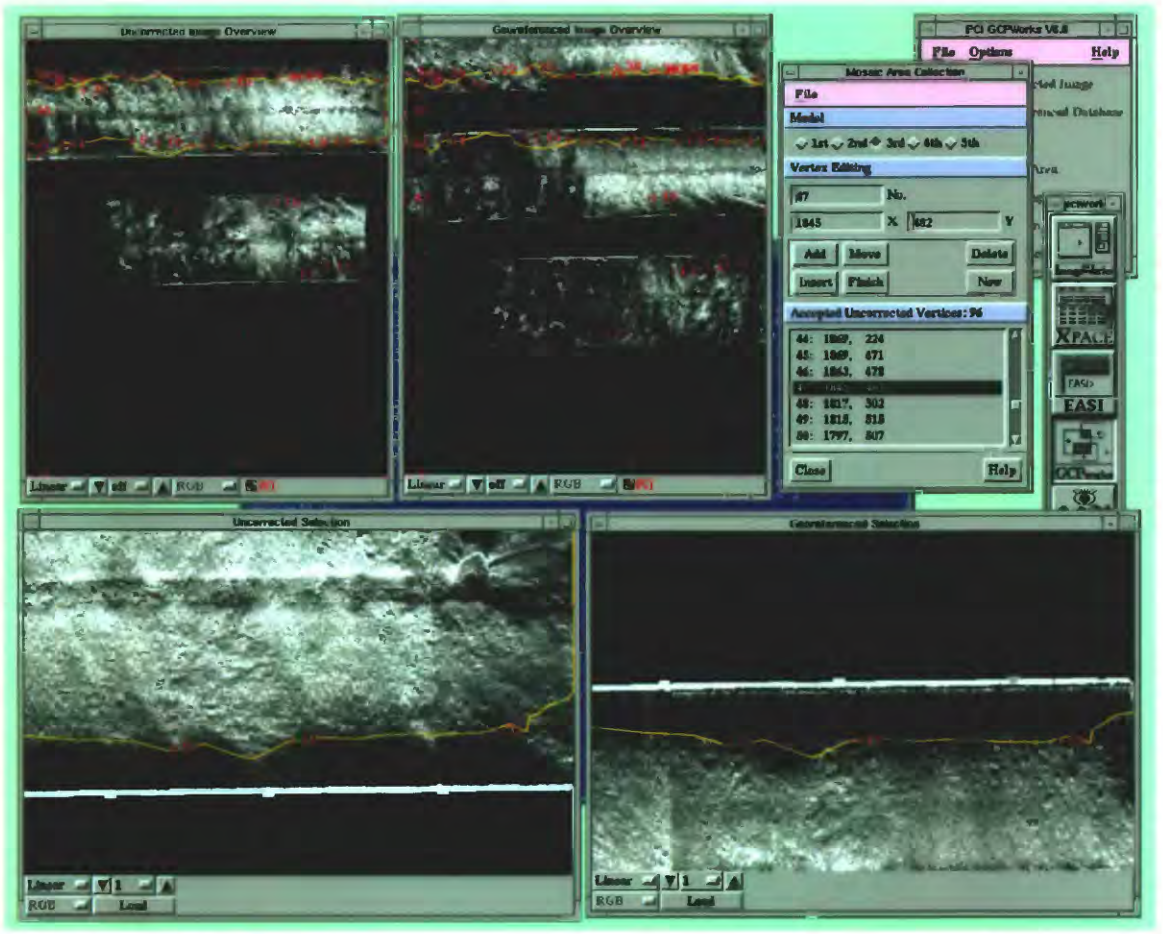


Figure 5 - GCPworks windows example

Selecting Ground Control Points

It is best to select GCPs at well-defined and easy to recognize points on both the georeferenced and uncorrected data sets. Unlike georeferencing requirements of traditional remote sensing data, such as aerial photographs, sidescan sonar data presents a specific set of unique problems. At times locating common points on adjacent sidescan sonar images can be difficult due to the different look angles from the adjacent swaths and subsequent shadows generated and the small amount of overlap in the swaths. Patience and perseverance are important traits when learning the 'art' of GCP selection and registration and digital mosaicking of sidescan sonar imagery.

When collecting GCPs, it is important to select points that are distributed over the entire image. Most points will be from overlaping areas with adjacent swaths, but some points away from the swaths are need as well. The selected GCPs will be utilized to compute a 1st to 5th order polynomial describing how the uncorrected image has to be warped to make it register (match) the georeferenced image. Collecting GCPs simply along the swaths would result in the image being distorted at it's bottom, sides and corners. Collecting well distributed GCPs will help reduce unwanted image distortion.

In the GCPworks example on the previous page, the sonar image being georeferenced was in the upper quarter of the image. GCPs were first collected in the four corners of the image to tie the image to each other and provide the initial first order transformation. Next, GCPs were collected along the image swath followed by identifying features from other portions of the image to provide more evenly distributed GCPs. Finally, additional points were selected as tie points in the area where no imagery was available. This final set of points are added to distribute points over the image and reduce the clustering of points in the upper half of the image. The placement of these points are based on the polynomial transformation computed from the previously selected points and as such errors in those points will affect the placement of these final points. Since there are no features associated with these points, they are accepted as they are placed by GCPworks.

One very useful option of GCPworks is the GCP Scatter Plot Report that can be obtained while in the GCP Selection and Editing Panel. This report can assist the user in identifying the GCPs which contain the greatest error. The reports can be viewed by selecting "GCP Scatter Plot" from the Reports menu found on the top menu line of the panel. The report is updated each time the GCP list is changed by adding or removing control points and shows the X and Y residual errors for each GCP point. The report can be used to determine the accuracy of the derived coefficients of the GCPs and easily identify points which contain the greatest errors.

The four panels comprising Figure 6 a through d shows the GCP Scatter Plot Reports for the 1st to the 4th polynomial based on the selected GCPs in the example. It is important to note that while selecting a higher order polynomial will result in a more accurate fit in the immediate area of the GCPs, it may introduce new and larger errors in those parts of the image away from the GCPs. In this specific application, the first and second order polynomial show the greatest errors. The 3rd order polynomial was selected and applied to the image. The completed digital mosaic is shown at the end of this section.

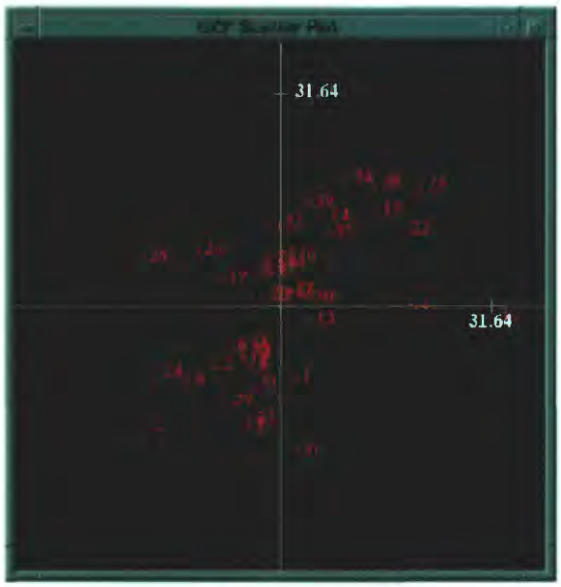


Figure 6a - GCPs with 1st order polynomial

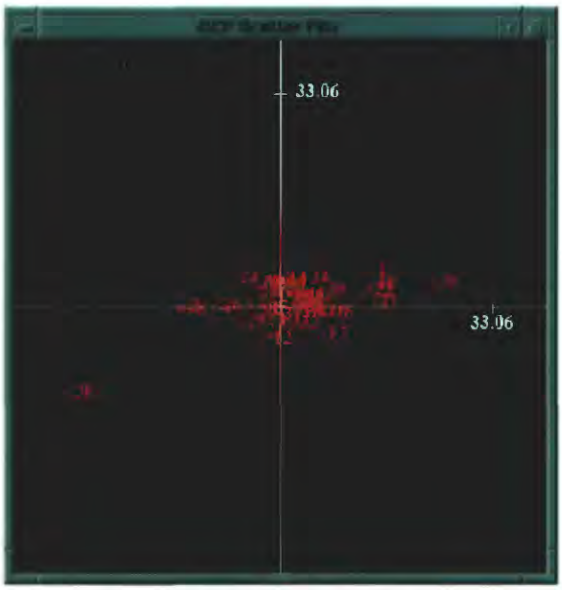


Figure 6b - GCPs with 2nd order polynomial

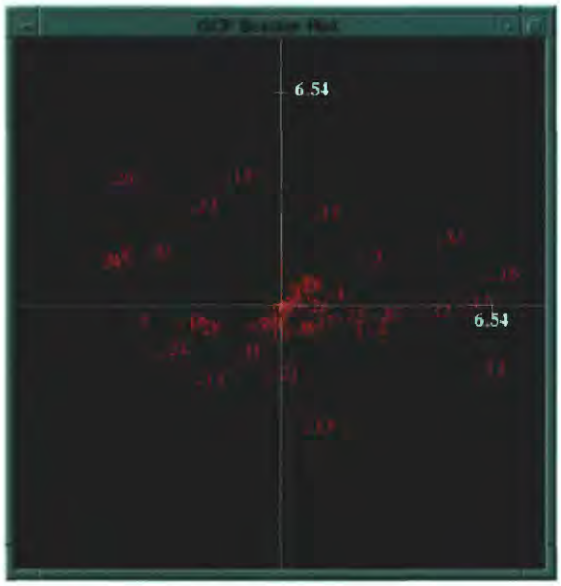


Figure 6c - GCPs with 3rd order polynomial

Figure 6d - GCPs with 4th order polynomial

Software Requirements

Program **mapit** is one of the dozens of programs comprising the Woods Hole Image Processing System (WHIPS) software developed at the U.S. Geological Survey, Woods Hole, MA. The focus of the WHIPS software is for initial processing (radiometric and geometric corrections) and digital mapping of sidescan sonar data. This program, along with the entire WHIPS software package, has been installed on a variety of UNIX platforms. However, before attempting to install the software package, there are a few characteristics of the WHIPS software that must be addressed.

First, the WHIPS software utilizes the NetCDF Data Access Software as described in the 1991 Unidata Report (Unidata Program Center, 1991) and USGS Open-File Report 92-25 (Paskevich, 1992a). This software provides the underlying data file layout and access. The NetCDF data access software was chosen because of the variety of UNIX workstations in use at the Woods Hole field center. Utilizing the NetCDF software provided the ability to create platform independent data files. In other words, the NetCDF software provided the ability to create files on one computer system that may be read on a system with a different byte order.

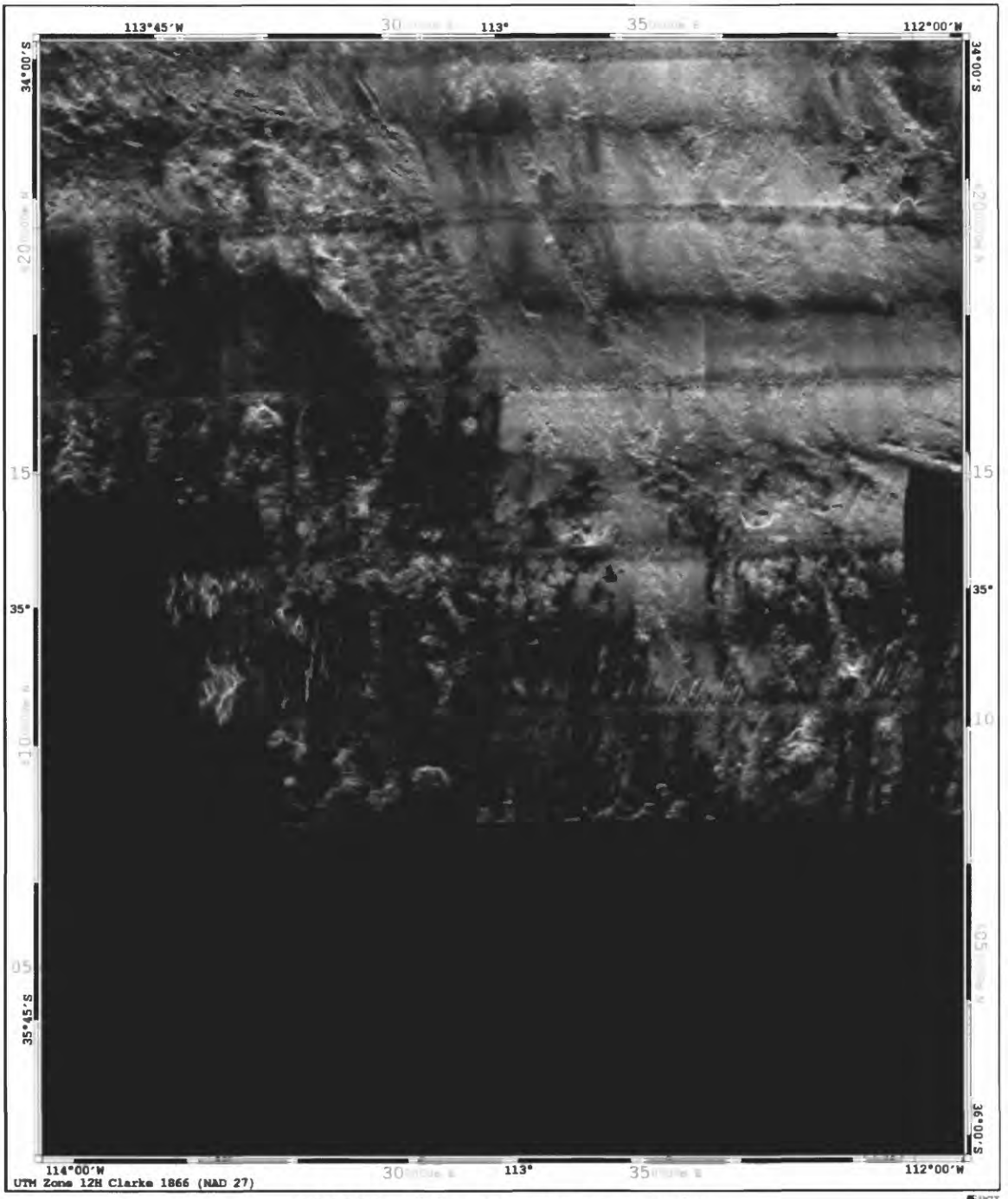
A second requirement of program **mapit** is the ***proj cartographic library*** (Evenden, 1990). To properly compile the program, the proj v4.0 or greater cartographic library must be available. The proj cartographic library currently contains 100+ cartographic projections. Only a sub-set of the cartographic library was included in program **mapit** and is defined in the header file *mapit-pjlist.h*. The included projections are also summarized for the user in Appendix A. If a projection must be added to the program, the user should add the information to the *mapit-pjlist.h* file and recompile the program.

Program **mapit** also has two minor restrictions regarding the projection initialization information. First, the program has been limited to reading a character string of 100 maximum characters. This means that the specified projection line cannot be greater than 100 characters. This should not be a problem under normal circumstances. If the projection specification will be longer than 100 characters, the program should be modified. To allow projection lines longer than 100 characters, the variable *pjline* should be increased and the program recompiled.

Secondly, **mapit** has assumed that there will not be more than 20 separate projection parameters specified. This should not be a problem since most of the currently included projections require at most three or four projection specific variables. If more than two separate projection parameters are needed, the variable *params* should be increased and the program recompiled.

The WHIPS software, proj cartographic library, and the NetCDF data access software has been compiled and tested on a variety of computer systems. All software is in the public domain and may be obtained via anonymous ftp. The WHIPS software and PROJ 4.0 is available from the U.S. Geological Survey Seafloor Mapping server (kai.er.usgs.gov). Because of security features installed on the Seafloor Mapping server, some users may experience difficulty in connecting via anonymous ftp. If problems occur, they may wish to try accessing the system via the World Wide Web at http://kai.er.usgs.gov and going to the Seafloor Mapping server's homepage. Access to the ftp directories can be made through here. The NetCDF data access software is available from Unidata (ftp.unidata.edu).

In the past, the compiled WHIPS software has been made available for specific systems. However, it has become impractical to provide executables for specific systems. Only the WHIPS source code is now made available. User's wishing to use the software must now compile the software for their specific computer systems.



GLORIA sidescan sonar mosaic

Juan Fernandez microplate

1:1 250 000 Scale

Kilometres 50 0 50

****** BLANK *****

TABLE I

The table below summarizes the projections which have been include in program **mapit** from ***the proj 4.0 cartographic library***. The first column contains the common name/identifier for the projection followed by the **proj** keyword for the projection. The third and final column summarizes the required **proj** parameter(s) for the selected projection.

The information contained in this table were summarized from various **proj** documentation. This table is not meant to be a definitive description of the projection characteristics. It is simply meant to be a summary of the **required proj** parameters so the user may specify the selected map properly. For a detailed description of any map projection or the optional parameters, the user is advised to reference Snyder, 1987 and 1989 or Evenden, 1990.

Projection Name	**proj name**	**proj parameters***	**Notes**
Albers Equal Area	aea	lat_1, lat_2	Normal default values when creating maps of the conterminous United States are 33°N and 45°N for **lat_1** and **lat_2**.
Azimuthal Equidistant	aeqd	lat_0	
Modified Stereographics of Alaska	alsk		
Central Cylindrical	cc		
Equal Area Cylindrical	cea	lat_ts	
Equidistant Conic	eqdc	lat_1, lat_2	
Modified Sterographics of 48 U.S.	gs48		
Modified Sterographics of 50 U.S.	gs50		
Lambert Azimuthal equal Area	laea		
Lambert Conformal Conic	lcc	lat_1, lat_2 or lat_0	Normal default values when creating maps of the conterminous United States are 33°N and 45°N for **lat_1** and **lat_2**.
Lambert Equal Area Conic	leac	lat_1, south	Select parameter **south** for south polar aspect.
LANDSAT - Space Oblique	lsat	lsat, path	
Mercator	merc	lat_ts	
Miller Oblated Sterographic	mil_os		The central meridian (**lon_0**) and parallel (**lat_0**) are fixed at 20°E and 18°N respectively.

Projection Name	proj name	proj parameters*	Notes
Miller Cylindrical	mill		
Mollweide	moll		
Near-sided perspective	nsper		
Orthographic	ortho		
Polyconic (American)	poly	lat_ts	
Robinson	robin		
Rectangular Polyconic	rpoly		
Sinusoidal (Sanson-Flamsteed)	sinu	lat_ts	
Sterographic	stere		
Transverse Central Cylindrical	tcc		
Transverse Cylindrical Equal Area	tcea		
Transverse Mercator	tmerc		
Two Point Equidistant	tpeqd	lon_1, lat_1, lon_2, lat_2	
Universal Transverse Mercator	utm	zone or lon_0, south	For maps in the southern hemisphere, the keyword **south** should be specified. If both **zone** and **lon_0** are specified, **zone** will take precedence.
van der Grinten (I) van der Grinten (II) van der Grinten (III) van der Grinten (IV)	vandg vandg2 vandg3 vandg4		

* The following list describes the **proj** cartographic parameters.

lon_0 - central meridian. When specified with lat_0, determines the geographic origin of the projection.
lat_0 - central parallel
lat_ts - latitude of true scale
lat_1 - first standard parallel
lat_2 - second standard parallel
lsat - LANDSAT satellite number, ***n***, must be in the range 1-5
path - LANDSAT satellite path number, ***p***, must be in the ranges 1-251 for ***n***=1,2,3 or 1-233 for ***n***=3,4
south - to be specified for southern hemisphere mapping applications
zone - One of the 60 zone numbers required by the Universal Transverse Mercator projection.

TABLE II

The table below lists the various ellipsoid names and their respective values contained within the **proj** cartographic library. The first column contains the ellipsoid descriptor. The second column contains the **proj** keyword to identify the ellipsoid. The last column represent the semi-major axis in meters. The list was compiled from program proj v4.4 and may not reflect additions to later versions of the software.

When running **mapit**, the user must specify the ellipsoid and may chose one from the table below. To specify a spheroid, the user must enter the command **ellps=*proj_name*** as part of the **proj** command line in the **mapit** control file.

Descriptor	proj *ellps* keyword	Semi-Major Axis (A)
MERIT 1983	MERIT	a=6378137.0
SGS 85	SGS85	a=6378136.0
GRS 1980(IUGG, 1980)	GRS80	a=6378137.0
IAU 1976	IAU76	a=6378140.0
Airy 1830	airy	a=6377563.396
Appl. Physics. 1965	APL4.9	a=6378137.0
Naval Weapons Lab., 1965	NWL9D	a=6378145.0
Modified Airy	mod_airy	a=6377340.189
Andrae 1876 (Den., Iclnd.)	andrae	a=6377104.4
Australian Natl & S. Amer. 1969	aust_SA	a=6378160.0
GRS 67(IUGG 1967)	GRS67	a=6378160.0
Bessel 1841	bessel	a=6377397.155
Bessel 1841 (Namibia)	bess_nam	a=6377483.865
Clarke 1866 [1]	clrk66	a=6378206.4
Clarke 1880 mod.	clrk80	a=6378249.145
Comm. des Poids et Mesures 1799	CPM	a=6375738.7
Delambre 1810 (Belgium)	delmbr	a=6376428.
Engelis 1985	engelis	a=6378136.05
Everest 1830	evrst30	a=6377276.345
Everest 1948	evrst48	a=6377304.063
Everest 1956	evrst56	a=6377301.243
Everest 1969	evrst69	a=6377295.664
Everest (Sabah & Sarawak)	evrstSS	a=6377298.556
Fischer (Mercury Datum) 1960	fschr60	a=6378166.
Modified Fischer 1960	fschr60m	a=6378155.
Fischer 1968	fschr68	a=6378150.
Helmert 1906	helmert	a=6378200.
Hough	hough	a=6378270.0
International 1909 (Hayford)	intl	a=6378388.0

Descriptor	proj *ellps* keyword	Semi-Major Axis (A)
Krassovsky, 1942	krass	a=6378245.0
Kaula 1961	kaula	a=6378163.
Lerch 1979	lerch	a=6378139.
Maupertius 1738	mprts	a=6397300.
New International 1967	new_intl	a=6378157.5
Plessis 1817 (France)	plessis	a=6376523.
Southeast Asia	Seasia	a=6378155.0
Walbeck	walbeck	a=6376896.0
WGS 60	WGS60	a=6378165.0
WGS 66	WGS66	a=6378145.0
WGS 72	WGS72	a=6378135.0
WGS 84 [2]	WGS84	a=6378137.0

[1] standard for many USGS maps

[2] ellipsoid used by AMG navigation acquisition system

TABLE III

The table below lists the 60 UTM zones which may be specified when defining a map in the Universal Transverse Mercator (UTM) projection. One of the 60 zones may be specified in lieu of specifying the central meridian (lon_0). To specify the zone, the user must include the **proj** initialization command of **zone=*n*** where ***n*** is in the range of 1 to 60. For zones in the southern hemisphere the **proj** keyword ***south*** should also be specified as part of the **proj** initialization..

UTM Zone	Zone Range	Central Meridian	UTM Zone	Zone Range	Central Meridian
1	180W - 174W	177W	31	0E - 6E	3E
2	174W - 168W	171W	32	6E - 12E	9E
3	168W - 162W	165W	33	12E - 18E	1SE
4	162W - 156W	159W	34	18E - 24E	21E
5	156W - 150W	153W	36	24E - 30E	27E
6	150W - 144W	147W	36	30E - 36E	33E
7	144W - 138W	141W	37	36E - 42E	39E
8	138W - 132W	135W	38	42E - 48E	45E
9	132W - 126W	129W	39	48E - 64E	51E
10	126W - 120W	123W	40	64E - 60E	57E
11	120W - 114W	117W	41	60E - 66E	63E
12	114W - 108W	111W	42	66E - 72E	69E
13	108W - 102W	105W	43	72E - 78E	75E
14	102W - 96W	99W	44	7SE - 84E	81E
15	96W - 90W	93W	46	84E - 90E	87E
16	90W - 84W	87W	46	90E - 96E	93E
17	84W - 78W	81W	47	96E - 102E	99E
18	78W - 72W	76W	48	102E - 108E	10SE
19	72W - 66W	69W	49	108E - 114E	111E
20	66W - 60W	63W	50	114E - 120E	117E
21	60W - 54W	57W	51	120E - 126E	123E
22	54W - 48W	51W	52	126E - 132E	129E
23	48W - 42W	45W	53	132E - 138E	13SE
24	42W - 36W	39W	54	138E - 144E	141E
25	36W - 30W	33W	55	144E - 150E	147E
26	30W - 24W	27W	56	150E - 156E	153E
27	24W - 18W	21W	57	156E - 162E	159E
28	18W - 12W	15W	58	162E - 168E	16SE
29	12W - 6W	9W	59	168E - 174E	171E
30	6W - 0W	3W	60	174E - 180W	177E

— PAGE 25 FOLLOWS —

APPENDIX A

The following is a list of the WHIPS program that may be utilized in the sidescan sonar digital mapping procedure. A detailed description of the listed programs follows the listing.

avg_heading	apply a running average to the heading values contained in a WHIPS NetCDF sidescan sonar image header
avg_position	apply a running average to the fish positions contained in a WHIPS NetCDF sidescan sonar image header
filter	applies a low-pass, high-pass, zero replacement or divide filter to an image
gss_vel	computes the geographic coordinates of the sidescan sonar pixels and applies the necessary velocity correction to the sonar swath
listhdr	list the contents of a sidescan sonar header in a WHIPS image
lowpass2b2	applies a 2-by-2 low-pass filter to an image
median3	applies a 3-by-3 median filter to an image
mode3	applies a 3-by-3 mode filter to an image
mode5	applies a 5-by-5 mode filter to an image
projss	place projected sidescan sonar data in a map space
qmos	Mosaics (overlays) the specified input file over the specified output file. The program will either overlay where the input file has priority over the existing output file, where the output file has priority over the input file, or average non-zero pixel values together from the input and existing output file.
sshead	computes a simple heading for a WHIPS NetCDF sidescan sonar image
whips2raw	converts a WHIPS NetCDF image file to a raw (binary) image file

NAME

avg_heading - apply a simple running average to the heading values contained in a WHIPS netCDF sidescan sonar image header

SYNOPSIS

avg_heading -i *input* **-o** *output* [**-l** *nl*] [**-H**]

DESCRIPTION

The **avg_heading** program allows the user to apply a simple running average over the heading values contained in a WHIPS netCDF sidescan sonar image header and to effectively smooth the values. The number of values averaged may be specified by the user by selecting the **-l** option along with the number of lines (*nl*) to average on the run-line. The specified number of lines must be 3 or greater and must be an odd integer value. If this option is not specified, the program defaults to averaging the heading values from 3 lines. Special processing takes place to handle the beginning and end of the sidescan sonar files. However, general processing is done by accumulating the heading values for an equal number of lines before and after the actual line being processed, averaging the value, and outputting the new heading value in the output file.

Special processing takes place to handle the beginning and ending lines contained in the files. It is not possible to have an equal number of lines before and after the line being processed unless processing is taking place in the center of the image, away from the beginning and ending lines. To compute the total to be averaged at the beginning of the file, the first line is weighted by an additional 1/2 the number of lines to be totaled. This means when nl = 3 and the first line is to be processed, the first heading value is computed as:

```
new_heading1 = (head1 + head1 + head2) / 3
```

In this simple case, processing the second line will produce an equally spaced number of heading values before and after the record being processed. Even spacing of the lines will continue until the last line is to be processed. To process the last line, as with the first line, the heading from the last line is double weighted.

As the number of lines to be processed is increased by the user, the weighting of the first line is increased. For example, when the user specifies nl=5 the new heading for the first record is computed as:

```
new_heading1 = (head1 + head1 + head1 + head2 + head3) / 5
```

In the example above using 5 lines and processing moves from the first to the second record, the first line is not weighted as heavily.

```
new_heading2 = (head1 + head1 + head2 + head3 + head4) / 5
```

As described earlier, moving away from the beginning of the file will eventually produce line processing where the heading values are weighted evenly before and after the specific line to be processed. The even processing will continue until processing nears the end-of-file. At that time,

any necessary weighting of the last line in the image is similar to that done for the first line in the file.

Special processing was added to correct a problem that occurred when averaging north heading values. During times when the heading value would alter between high values (approximately 325° - 360°) to low values (approximately 0° - 35°) averaging the low and high values produced accurate results though wild heading values. These shifts in the heading value typically occur in one of two scenarios: 1- a ship is attempting to follow a northerly course. and it's heading waivers; or 2- the ship is executing a turn. Regardless of the cause, averaging the high and low values together produces undesirable results. In simplest terms, if the program is attempting to compute an average heading from two values of 359° and 1°, the mathematically correct result would be 180° though not the desired heading value of 0°. To correct for this problem and to compute an accurate heading value, additional checks were added to the program to test when the heading values were approaching 0° from either direction. When the program has detected heading values both in the high and low range, it focuses on those headings that will be used to compute the average heading for a given line. As the total is computed, 360° is added to the low values to bring the headings into a similar data range. The average heading is then computed from these values. If the average heading is greater than 360°, 360° is subtracted from the average and becomes the new heading value for the line being processed. As the processing continues to the next line, the program checks to see if both high and low heading values continue to exist. If so, the special computation applies once again. When the heading values being totaled fall entirely within a high or low range, the default method of computing the new heading value by a running-average is resumed.

The special handling required to properly compute the heading values when mixing the high and low heading values will slow file processing in those files were the heading values shift frequently in the north direction. The heading computation and program execution is more efficient when processing can be accomplished without having to address the special handling for the north heading values or for a few cases such as a valid course turn in a file.

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 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.

-l *nl*

specifies the *number of lines* (nl) for which the heading value is to be totaled and averaged.

-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 sidescan sonar header information.

The output file to be created must not currently exist.

EXAMPLE

The example below shows a simple execution of the program.

```
% avg_heading -i pass29.hdr -o pass29.avg_hd
```

The example below shows a possible execution of the program to compute the running average using seven heading values.

```
% avg_heading -i pass29.hdr -o pass29.avg_hd -l 7
```

SEE ALSO

avg_position(1)

WHIPS(5), whips_sonar(5)

"Digital Mapping of Side-scan Sonar Data with the Woods Hole Image Processing system Software": U.S. Geological Survey Open-File Report 92-536, 90p.

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

avg_position - apply a simple running average to the fish positions contained in a WHIPS netCDF sidescan sonar image header

SYNOPSIS

avg_position -i *input* **-o** *output* [**-l** *nl*] [**-H**]

DESCRIPTION

The **avg_position** program allows the user to apply a simple running average over the fish positions contained in a WHIPS netCDF sidescan sonar image header and to effectively smooth the latitude and longitude positions. The number of positions to be totaled and averaged may be specified by the user by selecting the **-l** option along with the number of lines (*nl*) to average on the run-line. The specified number of lines must be 3 or greater and must be an odd integer value. If this option is not specified, the program defaults to averaging the fish position for 3 lines. Special processing takes place to handle the first and last records of the sidescan sonar image file as well as the beginning and end of the sidescan sonar files. However, general processing is done by accumulating the latitude and longitude coordinates for an equal number of lines before and after the actual line being processed, averaging the position, and writing the new coordinates to the output file.

The first and last record are handled uniquely. The fish positions for those records are output unchanged. This was implemented to ensure that consecutive, separate file/line segments would match. In addition to the special processing for the first and last records of the sonar file, special processing takes place to handle the beginning and ending lines contained in the files. When *nl* (**-l**) is specified as 5 or greater, it is not possible to have an equal number of lines before and after the line being processed unless processing is taking place in the center of the image, away from the beginning and ending lines. To compute the totals to be averaged at the beginning of the file, the first line is weighted by an additional 1/2 the number of lines to be totaled. This means when nl = 5 and the first line is to be processed, the second positions are computed as:

```
new_lat = (lat1 + lat1 + lat2 + lat3 + lat4) / 5
new_lon = (lon1 + lon1 + lon2 + lon3 + lon4) / 5
```

In this simple case, processing the third line will produce an equally spaced number of heading values before and after the record being processed. Even spacing of the lines will continue until the end-of-file is approached. As processing nears the end-of-file, the fish positions from the last line are weighted accordingly.

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 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.

-l *nl*

specifies the number of lines (nl) for which the sidescan sonar fish positions are to be totaled and averaged.

-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 sidescan sonar header information.

The output file to be created must not currently exist.

EXAMPLE

The example below shows a simple execution of the program.

```
% avg_position -i pass29.mg2 -o pass29.avg_pos
```

The example below shows a possible execution of the program to compute the running average using seven values.

```
% avg_position -i pass29.mg2 -o pass29.avg_pos -l 7
```

SEE ALSO

avg_heading(1)

WHIPS(5), whips_sonar(5)

"Digital Mapping of Sidescan Sonar Data with the Woods Hole Image Processing system Software": U.S. Geological Survey Open-File Report 92-536, 90p.

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

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. Optional program keywords are listed and described following the detailed filter descriptions.

-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

gss_vel - compute geographic coordinates of sidescan sonar pixels and corrects the sonar imagery for changes in ship's velocity

SYNOPSIS

gss_vel -i *input* **-r** *resolution* [[*options* ...] **[-E** *ellips*] ... **[-H]**] >std_out

DESCRIPTION

The **gss_vel** program will compute the geographic coordinates of each pixel in the scan line of a WHIPS netCDF sidescan sonar image. The input file MUST be a sidescan sonar image that has NOT been velocity stretched. **Gss_vel** combines the functions of programs **gss** and **velocity** because it will apply the velocity corrections as each swath is processed. The velocity correction is applied by the program by first computing the distance travelled between consecutive swaths and then the number of times an individual line should be duplicated to fill the image to the position of the next swath. Velocity stretching of a sidescan sonar image is accomplished by duplicating individual swath lines in an attempt to produce an image with the same across and along-track resolution. The velocity correction may not be necessary for mid or low-range sidescan sonar which have `*square*' pixels.

The program begins by calculating the number of times the individual line to be processed should be duplicated so it will meet the following line and reduce the gap in the map space between the consecutive lines. The duplication factor is calculated first by determining the distance travelled between a pair of consecutive lines contained in the sidescan sonar image. The distance travelled between the consecutive swaths is determined by comparing the position coordinates contained in the sidescan sonar header. The user may select the spheroid (**-E**) to be used in the distance computation from one of five available ellipsoids. With the distance travelled between the consecutive lines known, the program then computes the number of times a line should be duplicated to fill the image space up to the placing of the next swath based on the sidescan sonar image resolution (**-r**) specified by the user.

As each individual swath is processed, the geographic coordinate of each pixel is computed using the heading value contained in the sidescan sonar attribute field, the sonar position associated with the scan line, and the user supplied resolution value. The resolution value (**-r**) is the pixel resolution, in meters, of the input image file. As each individual line is duplicated, a new nadir position is calculated based on the sonar resolution (**-r**) and the heading contained in the sidescan sonar header. Geographic coordinates are then computed for the pixels contained in the duplicate lines to allow the proper placement of the swaths and to reduce the gaps which can occur between swaths.

The program computes the geographic coordinates of the pixels and, if necessary, the new positions of the duplicate lines by a simple distance computation utilizing one of five earth ellipsoids. By default, **gss_vel** uses the Clarke 1866 spheroid in these computations. The user may select an alternate earth ellipsoids by specifying the **-E** option on the program run-line. The available ellipsoids are: 1) Clarke 1866 (default); 2) International 1909 (Hayford); 3) Geodetic Reference System 1980; 4) WGS 1984; and 5) WGS 1972.

By default, the program will process every image line and sample contained in the input file. The user may select which lines of the image to process by specifying the **-l** option. When this option is specified, the user must specify the starting line and ending line to process within the image. The user may also optionally specify the number of samples (nsamps) or the distance (distance) from nadir to be processed within each scan by selecting either the **-s** or **-d** program option. Each sonar image line is processed from nadir to the specified far range with the port side being processed first, followed by the starboard side. As the geographic coordinates of the pixels are

computed, the position, along with the associated pixel value, are output to the std_out device. The coordinates are recorded as decimal degree values and are output in latitude/longitude order.

The fish_heading value contained in the sidescan sonar header fish attributes field may be adjusted by the user by selecting the **-h** option. When this option is selected, the user must also specify a heading adjustment value (heading_adjustment) to be added to the fish heading value before the pixel coordinates are computed. This option, along with the ability to select a portion of the input image to be processed, provides a simple way to "tweak" a segment of a sonar line when the navigation data may not accurately reflect the track of the sonar fish. Hopefully this option will assist in adjusting the sonar swaths and provide a simple way to lineup adjacent swaths.

As stated above, the program output is to std_out. This option was implemented to allow for piping of the data through a series of steps to facilitate a stream processing to place the pixels directly into a defined map space. If the input pixel value is equal to zero, the program will NOT output the coordinate and pixel information. This was implemented to help reduce the amount of pixels output and later processed.

The maximum and minimum geographic coordinates computed by the program are output to the program print file.

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 or 16-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 processing will stop.

-r *resolution*

specifies the *pixel resolution*, in meters, of the input image. This value is used to compute the position of the pixel values from the known position at nadir.

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

-l *sl,el*

specifies a *sub-area* of the input file to be processed. The sub-area is specified by entering the starting line (sl) and the ending line (el) of the sonar image file to be processed. This option, along with **-s** or **-d**, may be selected to specify a sub-area of the sonar image to process.

-h *heading_adjustment*

specifies a user supplied value to be added to the fish heading value before computing the pixel coordinates. This option may be used to adjust the fish heading and swath orientation if the user suspects the fish coordinates to not accurately reflect the direction of the fish.

-s *nsamps*

specifies the number of samples (nsamps), port and starboard, to be processed from nadir. For example specifying samp as 225 would flag the program to process 225 samples from either side of nadir resulting in a total of 450 samples being output per scan. This option,

along with **-l**, may be selected to specify a sub-area of the sonar image to process. If the user specifies samp as greater than the number of samples actually contained in each side of the image scan, the value will be set to half the actual number of samples contained in a line of the image file.

The user may optionally select the **-d** option to specify the distance from nadir to be processed.

-d *distance*

specifies the *distance* (distance), in meters, from nadir for which to process the image samples. The user may specify this option, along with **-l**, to select a sub-area of the sonar image to process.

The user may optionally select the **-s** option to specify the number of samples from nadir to be processed.

-E *ellips*

specifies the *earth ellipsoid* to be used in the computation of the pixel and line coordinates. The ellips value may be specified as one of the following values:

1 = Clarke 1866
2 = International 1909 (Hayford)
3 = Geodetic Reference System 1980
4 = WGS 1984
5 = WGS 1972

-R

flags the program to round-up the number of times a sidescan sonar swath is to be duplicated to accommodate the velocity stretching. For example, when the program computes a line duplication factor of 3.5, the program, by default, will output the line only three times. If the **-R** option has been selected on the program run-line, line duplication factor would be rounded-up and the specific line being processed would be duplicated four times.

This option may help reduce the gaps which can exist between adjacent lines as they are placed in the map space. It will also increase the processing time and will affect the resultant image by increasing the blockiness of the sonar map 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 only accepts 8 or 16-bit image files.

The output file to be created must not currently exist.

The **-s** or **-d** options are exclusive. Only one may be selected.

NOTES

This program combines the functions of programs gss and velocity. In addition to the pixel coordinate computation which program gss provides, this program will correct the image for changes in ship's velocity by duplicating the line being processed up to the position of the following line.

This program also differs slightly from **gssv**. Program **gssv** requires the sidescan sonar input file to have been previously processed through program velocity.

EXAMPLE

The example below shows a simple application of the program to compute the coordinate values for a sidescan sonar image processed at .5 meters using the default Clarke 1866 spheroid.

```
% gss_vel -i bos13.wco -r .5 >bos13_cord.dat
```

The example below shows the program usage to select a specified number of lines (275 lines) and width (375 samples per side). The -R option has been selected to help reduce the gaps between the placement of the consecutive sidescan sonar swaths. The **-E** option has also be specified to utilized the International 1909 spheroid.

```
% gss_vel -i gloria.head -r 50 -l 201,475 -s 375 -E 2 -R >pass74b.dat
```

SEE ALSO

avg_heading(1), avg_position(1), gss(1), gssv(1), projss(1), sshead(1), sumss(1)

WHIPS(5), whips_sonar(5)

"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.

"Digital Mapping of Side-Scan Sonar Data with the Woods Hole Image Processing system Software": U.S. Geological Survey Open-File Report 92-536, 90p.

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

The 16-bit option has not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

listhdr - list the contents of a sidescan sonar header in a WHIPS netCDF sidescan 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 sidescan 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 sidescan 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 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.

-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 sidescan sonar header information.

EXAMPLE

The example will output the sidescan 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 sidescan 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

mapit - computes the geographic coordinates of sidescan sonar pixels and maps the pixels in a user specified cartesian space. Mapping of the pixels includes additional correction to the sonar imagery for changes in ship's velocity.

SYNOPSIS

mapit -i *input* **-o** *output* **-c** *control_file* **-r** *resin,resout* [[*options* ...] [**-H**]]

DESCRIPTION

The **mapit** program is a composite of WHIPS programs **gss_vel** and **projss** and provides the functionality of both with the integrated projection features of program **proj**. As with **gss_vel**, this program will compute the geographic coordinates of each pixel in the scan line of a WHIPS netCDF sidescan sonar image from the known fish position and heading value contained in the sonar header. Rather than passing the geographic coordinates to **proj** and **projss**, this program will project the geographic coordinate internally and place the pixel value in it's proper cartesian coordinate to build a digital sidescan sonar mosaic within one program.

The output file (**-o**) is defined by a user specified cartographic projection by specifying typical **proj** parameters and the geographic coordinates of the map area. A sub-set of the **proj** cartographic library has been made available and the user should reference Table 1 for a complete list of the available projections. Table 1 also includes an overview of the required **proj** parameters for the specific projection. The list is not intended to be a definitive description of the projections. Rather it is a brief summary of the required **proj** parameters. User's should be completely familar with the cartographic requirements of their selected projection.

The input file (**-i**) MUST be a sidescan sonar image that has NOT been velocity stretched. **Mapit** will apply the velocity corrections as each swath is processed. The velocity correction is applied by the program by first computing the distance travelled between consecutive swaths and then the number of times an individual line should be duplicated to fill the image to the position of the next swath. Velocity stretching of a sidescan sonar image is accomplished by duplicating individual swath lines in an attempt to produce an image with the same across and along-track resolution. The velocity correction may not be necessary for mid or low-range sidescan sonar which have `*square*' pixels.

Because the output file (**-o**) to be created is an image map, the user must supply the cartographic information to the program and the geographic bounds of the map area. The information is supplied to the program in the **mapit** *control file* (**-c**). The file must contain three lines of information. The first line is the projection parameters required by **proj**. The second and third line must contain the geographic coordinates of the upper left and lower right bounds respectively. The map bounds may be specified in the ***DMS*** format acceptable to **proj**. The final piece of information needed to define the map image is the output image resolution (**-r**) specified by the user on the program run-line.

The program begins by calculating the number of times the individual line to be processed should be duplicated so it will meet the following line and reduce the gap in the map space between the consecutive lines. The duplication factor is calculated first by determining the distance travelled between a pair of consecutive lines contained in the sidescan sonar image. The distance travelled between the consecutive swaths is determined by comparing the position coordinates contained in the sidescan sonar header. With the distance travelled between the consecutive lines known, the program then computes the number of times a line should be duplicated to fill the image space up to the placing of the next swath based on the sidescan sonar image input resolution (**-r**) specified by the user.

As each individual swath is processed, the geographic coordinate of each pixel is computed using the heading value contained in the sidescan sonar attribute field, the sonar position associated with the scan line, and the user supplied resolution value. The resolution value (**-r**) is the pixel resolution, in meters, of the input image file. As each individual line is duplicated, a new nadir position is calculated based on the sonar resolution (**-r**) and the heading contained in the sidescan sonar header. Geographic coordinates are then computed for the pixels contained in the duplicate lines to allow the proper placement of the swaths and to reduce the gaps which can occur between swaths.

The program computes the geographic coordinates of the pixels and, if necessary, the new positions of the duplicate lines by a simple distance computation the ellipsoid specified by the user on the proj parameter line of the control file. This is a significant change from program gss_vel because the user know may select from any of the ellipsoids available to proj. Additionally, the user MUST specify an ellipsoid as part of the proj initialization. A default ellipsoid is no longer used. Not specifying an ellipsoid may cause unknown results.

As each of the geographic coordinates for the sonar pixels are computed, they are projected to their cartesian values based on the user map specifications. The projected coordinates are then compared to see if they fall within the image space. If the coordinate is outside of the map area it is ignored and processing continues with the next sonar pixel. Pixels outside of the map area can be common due to heading fluctuations or where track lines are oriented along a map edge resulting in a portion of the swath being outside the map area. By checking each pixel, the entire image map area can be filled and easily produce a clipped sidescan image without the user being concerned with what portion of a file may fall within the image area.

If the sonar pixel's cartesian coordinates falls within the map, it's value is placed directly in the output file. It is important to note that the pixel placement is done on a pixel-by-pixel basis. When multiple dn values are computed for the same location, **mapit** will always place the last pixel input in the output location regardless of the coordinates previous content.

By default, the program will process every image line and sample contained in the input file. The user may select which lines of the image to process by specifying the **-l** option. When this option is specified, the user must specify the starting line and ending line to process within the image. The user may also optionally specify the number of samples (nsamps) to be processed within each scan by selecting either the **-s** program option. Each sonar image line is processed from nadir to the specified far range with the port side being processed first, followed by the starboard side.

The fish_heading value contained in the sidescan sonar header fish attributes field may be adjusted by the user by selecting the **-h** option. When this option is selected, the user must also specify a heading adjustment value (heading_adjustment) to be added to the fish heading value before the pixel coordinates are computed. This option, along with the ability to select a portion of the input image to be processed, provides a simple way to "tweak" a segment of a sonar line when the navigation data may not accurately reflect the track of the sonar fish. Hopefully this option will assist in adjusting the sonar swaths and provide a simple way to lineup adjacent swaths.

The maximum and minimum geographic coordinates computed by the program are output to the program print file along with the exact location of the map bounds and their location in the image. The contents of the **mapit** control file is also logged to the program print file for the user's reference.

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 or 16-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 processing will stop.

-o *output_file*

specifies the output file to be created. If the file exists, it will be openned for updating if it passes two simple checks. The first check is to verify that the number of lines and pixels in the existing file are equal to the number of lines and pixels computed from projection parameters. The second check is to verify that the input image and output image are the same bittype. When creating the output file, the file bittype is determined from the input file.

-c *control_file*

specifies the **mapit** control file which contains the projection information and geographic coordinates of the map bounds. The file must contain three lines of information and are summarized as follows:

1) the **proj** (projection) parameters
2) the upper left geographic coordinates of the map in ***DMS***
3) the lower right geographic coordinates of the map in ***DMS***

-r *resin,resout*

specifies the *resin* and *resout*, in meters, of the sidescan sonar input data and the desired pixel resolution of the output image. The *resin* value is used to compute the position of the pixel values from the known position at nadir.

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

-l *sl,el*

specifies a *sub-area* of the input file to be processed. The sub-area is specified by entering the starting line (sl) and the ending line (el) of the sonar image file to be processed. This option, along with **-s**, may be selected to specify a sub-area of the sonar image to process.

-h *heading_adjustment*

specifies a user supplied value to be added to the fish heading value before computing the pixel coordinates. This option may be used to adjust the fish heading and swath orientation if the user suspects the fish coordinates do not accurately reflect the direction of the fish.

-s *nsamps*

specifies the number of samples (nsamps), port and starboard, to be processed from nadir. For example specifying *nsamps* as 225 would flag the program to process 225 samples from either side of nadir resulting in a total of 450 samples being output per scan. This option, along with **-l**, may be selected to specify a sub-area of the sonar image to process. If the user specifies *nsamps* as greater than the number of samples actually contained in each side of the image scan, the value will be set to half the actual number of samples contained in a line of the image file resulting in the entire swath being processed

-R

flags the program to round-up the number of times a sidescan sonar swath is to be duplicated to accommodate the velocity stretching. For example, when the program computes a line duplication factor of 3.5, the program, by default, will output the line only three times. If the **-R** option has been selected on the program run-line, line duplication factor would be rounded-up and the specific line being processed would be duplicated four times.

This option may help reduce the gaps which can exist between adjacent lines as they are placed in the map space. It will also increase the processing time and will affect the resultant image by increasing the blockiness of the sonar map 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 sidescan data.

The created output file will have the same bittype as the input file.

NOTES

This program combines the functions of programs **gss_vel**, **proj** and **projss**. By combining the individual programs and eliminating the need for piping the information from program to program, program **mapit** has been timed and reduces the overall processing time by 20 - 25% from previous methods.

EXAMPLE

The example below shows a simple application of the program to one swath of GLORIA sidescan sonar data and create an output image named *east.map*. The data resolution is specified as 50meters for the GLORIA sonar data and the desired output image will be created with a 100meter pixel resolution. The contents of the mapit control file (*mapit.dat*) is shown first followed by the program run-line.

```
% more mapit.dat
proj=merc lat_ts=27.25n ellps=clrk66
29d30' -94
25.0 -88

% mapit -I T2F10.hdg -o odd.map -c mapit.dat -r 50,100
```

The example below shows the program usage to select a specified number of lines (275 lines) and width (375 samples per side) from a second GLORIA sidescan sonar file. The **-R** option has been selected to help reduce the gaps between the placement of the consecutive sidescan sonar swaths.

```
% more mapit2.dat
proj=utm ellps=clrk66 south lon_0=-111
-34 -114
-36 -112

% mapit -i pass12.avg_head -o east.map -c mapit2.dat \
   -r 45,100 -l 201,475 -s 375 -R
```

SEE ALSO

avg_heading(1), avg_position(1), gss(1), gssv(1), gss_vel(1), projss(1), sshead(1), sumss(1)

WHIPS(5), whips_sonar(5)

"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.

"Digital Mapping of Side-Scan Sonar Data with the Woods Hole Image Processing system Software": U.S. Geological Survey Open-File Report 92-536, 90p.

"MAPIT: An improved method for mapping digital sidescan sonar data using the Woods Hole Image Processing System (WHIPS) Software": U.S. Geological Survey Open-File Report 96-xxx, nnp.

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

The 16-bit option has not been completely tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

median3 - apply a 3-by-3 median filter to an image

SYNOPSIS

median3 -i *input* **-o** *output* [**-z**] [**-H**]

DESCRIPTION

The **median3** program allows the user to apply a 3-by-3 median filter to a WHIPS image. Median3 will modify the value centered on a 3-by-3 boxcar with the median value computed from the neighborhood distribution. The neighborhood, n, consists of 9 values (3x3), and the median value is computed as i,j = 5 = (n+1)/2 after the data has been arranged in increasing order.

The user may apply this program to fill zero values contained within an image by selecting the **-z** option on the run-line. When this option is selected, only those pixel values from the input file equal to zero are modified. However, all input pixel values, including those equal to zero, are used to compute the median value.

Median3 is most suitable for data that has a skewed distribution. However, the value obtained for the median may not be representative if the individual items do not tend to cluster at the center of the distribution.

Special processing takes place to handle the first and last lines of the image file. Adjacent lines are weighted to allow for unfolding to take place during the processing. When computing the 3-by-3 median of the first image line from the input file, the second line is read twice and used in the computation. To process the last line contained in the image line, the next to last line is read twice and used for the computation.

In addition to the special line processing, the program applies a similar overlapping procedure to the samples at the beginning and ending of each line. For the first and last samples contained in the image lines, the neighboring pixels are doubly weighted to allow for the foldover computations.

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 8, 16 or 32-bit image.

-o *output_file*

specifies the output file to be created.

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

-z

flags the program to apply the filter only when the center value of the neighborhood is zero. This will allow the user to apply the 3-by-3 median filter as a zero only replacement filter.

-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

```
% median3 -i map.cdf -o map.med3
```

NOTES

Though a histogram method could be employed to calculate the median value for 8-bit data, the histogram method would be more difficult to implement for 16 and 32-bit data. Therefore median3 is set-up to sort (via the UNIX library function qsort) the data values and can be quickly applied to 16 and 32-bit data as well as 8-bit.

SEE ALSO

lowpass2b2(1), filter(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.

BUGS

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

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

mode3 - apply a 3-by-3 mode filter to an image

SYNOPSIS

mode3 -i *input* **-o** *output* **[-z] [-Z] [-H]**

DESCRIPTION

The **mode3** program allows the user to apply a 3-by-3 mode filter to a WHIPS image. **Mode3** will modify the value centered on a 3-by-3 boxcar with the mode value computed from the neighborhood distribution. The neighborhood, *n*, consists of 9 values (3x3), and the mode value is the value which occurs most often within the neighborhood. In a neighborhood of 9 values it is possible that no mode value can be determined. For example, 9 different values may occur within the neighborhood or 2 different values may occur 4 times. When no mode value can be computed for the neighborhood, the input pixel value for that location is output unchanged.

The user may apply this program to replace zero values contained within an image by selecting the **-z** option on the run-line. When this option is selected, only those pixel values from the input file equal to zero are modified. However, all input pixel values, including those equal to zero, are used to compute the mode value.

In addition to the zero replacement option (**-z**), the user may select not to include the zero values from the image when computing the mode value by selecting the **-Z** program option. When this option is selected, the zero values for the neighborhood are not included when totalling the occurrences of the unique values for the neighborhood. The **-z** and **-Z** options are not mutually exclusive and may be selected individually or together during a single execution of the program. Selection of these program options is at the user's discretion depending on the results he or she wishes to achieve.

Special processing takes place to handle the first and last lines of the image file. Adjacent lines are weighted to allow for unfolding to take place during the processing. When computing the 3-by-3 mode of the first image line from the input file, the second line is read twice and used in the computation. To process the last line contained in the image line, the next to last line is read twice and used for the computation.

In addition to the special line processing, the program applies a similar overlapping procedure to the samples at the beginning and ending of each line. For the first and last samples contained in the image lines, the neighboring pixels are doubly weighted to allow for the foldover computations.

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 8, 16 or 32-bit image.

-o *output_file*

specifies the output file to be created.

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

-z

flags the program to apply the filter only when the center value of the neighborhood is zero. This will allow the user to apply the 3-by-3 mode filter as a zero only replacement filter.

-Z

flags program not to include zero values when computing the mode value. This option can be helpful when trying to apply the mode as a zero replacement filter and the mode in some neigborhoods are zero.

-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

The example below shows a simple execution of the program.

% mode3 -i map.cdf -o map.mode3

The example below shows a possible execution of the program to replace zero values within the image while excluding any zero values from the mode computation.

```
% mode3 -i map.cdf -o map.mode3zr -z -Z
```

NOTES

Though a histogram method could be employed to calculate the mode value for 8-bit data, that method would be more difficult to implement for 16 and 32-bit data. Therefore mode3 is set-up to sort (via the UNIX library function qsort) the data values and then count the number of times a unique value occurs within the neighborhood. The program will then compute the mode value and this technique can be applied to 16 and 32-bit data as well as 8-bit.

SEE ALSO

lowpass2b2(1), filter(1), median3(1), mode5(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 have not been thoroughly tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

mode5 - apply a 5-by-5 mode filter to an image

SYNOPSIS

mode5 -i *input* **-o** *output* [**-z**] [**-Z**] [**-H**]

DESCRIPTION

The **mode5** program allows the user to apply a 5-by-5 mode filter to a WHIPS netCDF image. Mode5 will modify the value centered on a 5-by-5 boxcar with the mode value computed from the neighborhood distribution. The neighborhood, n, consists of 25 values (5x5), and the mode value is the value which occurs most often within the neighborhood. In a neighborhood of 25 values it is possible that no mode value can be determined. For example, 25 different values may occur within the neighborhood or 5 different values may occur 5 times. When no mode value can be computed for the neighborhood, the input pixel value for that location is output unchanged.

The user may apply this program to fill zero values contained within an image by selecting the **-z** option on the run-line. When this option is selected, only those pixel values from the input file equal to zero are modified. However, all input pixel values, including those equal to zero, are used to compute the mode value.

In addition to the zero replacement option (**-z**), the user may select not to include the zero values from the image when computing the mode value by selecting the **-Z** program option. When this option is selected, the zero values for the neighborhood are not included when totalling the occurrences of the unique values for the neighborhood. The **-z** and **-Z** options are not mutually exclusive and may be selected individually or together during a single execution of the program. Selection of these program options is at the user's discretion depending on the results he or she wishes to achieve.

Special processing takes place to handle the first two and last two lines of the image file. Adjacent lines are weighted to allow for unfolding to take place during the processing. When computing the 5-by-5 mode of the first image line from the input file, the second and third lines are read twice and used in the computation. To process the second line in the input file, the third and fourth lines are read once and the first line is read twice to allow for the foldover processing. Similarly, the last two lines contained in the image are handled with unique weighting done to the adjacent lines.

In addition to the special line processing, the program applies a similar overlapping procedure to the samples at the beginning and ending of each line. For the first and last two samples contained in the image lines, the neighboring pixels are doubly weighted to allow for the foldover computations.

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 8, 16 or 32-bit image.

-o *output_file*

specifies the output file to be created.

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

-z

flags the program to apply the filter only when the center value of the neighborhood is zero. This will allow the user to apply the 5-by-5 mode filter as a zero only replacement filter.

-Z

flags program not to include zero values when computing the mode value. This option can be helpful when trying to apply the mode as a zero replacement filter and the mode in some neigborhoods are zero.

-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

The example below shows a simple execution of the program.

```
% mode5 -i map.cdf -o map.mode5
```

The example below shows a possible execution of the program to replace zero values within the image while excluding any zero values from the mode computation.

```
% mode5 -i map.cdf -o map.mode5zr -z -Z
```

NOTES

Though a histogram method could be employed to calculate the mode value for 8-bit data, that method would be more difficult to implement for 16 and 32-bit data. Therefore mode5 is set-up to sort (via the UNIX library function qsort) the data values and then count the number of times a unique value occurs within the neighborhood. The program will then compute the mode value and this technique can be applied to 16 and 32-bit data as well as 8-bit.

SEE ALSO

lowpass2b2(1), filter(1), median3(1), mode3(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 have not been thoroughly tested.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

projss - place projected side-scan sonar data in a map/image space

SYNOPSIS

projss -o *output* < std_in [**-H**]

DESCRIPTION

Projss will take the sidescan sonar data output from one of the sidescan sonar geographic coordinate computation programs (**gss, gss_vel** or **gssv**) and **proj** and create a WHIPS netCDF image which represents a map of the sonar data for a specific area, projection and scale. The first four input records must contain the map area bounds, in meters, for the user selected desired map area. Subsequent data records must contain the sidescan sonar pixel coordinate and their associated pixel value. Each pixel coordinate contained in the sidescan sonar data must have been previously converted to geographic coordinates by either program **gss, gss_vel** or **gssv**, and, further, converted to meter coordinates for the selected map area by program **proj**. The map coordinates must be in y x (latitude longitude) order and the coordinate pairs must be integer values.

Program input is through std_in. As stated above, the actual input to program projss is the sidescan sonar pixel coordinates which have been converted to meter values based on a specific map projection and scale. The first four records of the input file **MUST BE** the map corner coordinates, in meters, for the map projection and scale. The map corner coordinates must be specified in the following order:

upper left
upper right
lower right
lower left

For example, if the user desires to create a 2° map at a scale of 1:100 for an area bounded by 35° to 37° latitude and -75° to -77° longitude for a simple mercator map, the first four records of the input file would contain the following information:

```
44132 -85717
44132 -83491
41391 -83491
41391 -85717
```

Program **projss** will begin by reading the first four pairs of coordinates from the input file. Once the program has obtained this information it will calculate the size of the image (number of lines and number of samples). When the size of the WHIPS netCDF image has been computed, **projss** will then create the image file on disk and fills the image with blank lines before beginning to place the pixel values in the image.

The remainder of data in the input file must be the pixel coordinates and pixel values which projss will place in the appropriate map space. The information which follows must be the pixel coordinates, in meters, along with the 8-bit pixel value. Program projss accepts and calculates the location (the actual image line and sample coordinate) within the map space for the input pixel dn values. The program then places, on a pixel-by-pixel basis, the sidescan sonar dn values. It is important to note that the pixel placement is done on a pixel-by-pixel basis. When multiple dn

values are computed for the same location, **projss** will always place the last pixel input in the output location regardless of the coordinates previous content.

Program **proj** is essential in creating the final map product. The user should be familiar with its usage and various options. Some **proj** options must be specified to create the proper data output for program **projss**.

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

-o *output_file*

specifies the output file to be created. The output file will be an 8-bit WHIPS image.

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.

The input meter coordinates must be recorded as integer values.

EXAMPLE

The example below shows a simple execution of the program. The first four pairs of coordinates contained in the file, project.dat, must be the projected map area bounds.

```
% projss -o map2.cdf <projec.dat
```

The following is a typical example using program **gss** and **proj** as filters to complete the processing and mapping of a sidescan sonar image. The input image to program **gss**, gloria.wco, contains a pixel resolution of 50 meters. The data is to be mapped at 100 meter resolution to an Albers Equal Area projection using standard parallels and a central longitude of 85°. The file, bounds.dat, must contain the map corner coordinates and is show below.

```
% gss -i gloria.wco -r 50 | \
  proj +proj=aea +lon_0=-85 +lat_1=29.5 +lat_2=45.5 -r -s \
  -m 1:100 -f `%.0f' bounds.dat - | projss -o map2.cdf

% more bounds.dat
25.5 -85      # upper left corner
25.5 -84      # upper right corner
24.5 -84      # lower right corner
24.5 -85      # lower left corner
```

SEE ALSO

gss(1), gss_vel(1), gssv(1), proj(1)

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

User's Manual for MAPGEN (UNIX version): a method of transforming digital cartographic data to a map: U. S. Geological Survey Open-File Report 85-706, 134 p.

Cartographic Projection Procedures for the UNIX Environment - A User's Manual: U. S. Geological Survey Open-File Report 90-284, 62p.

"Digital Mapping of Side-Scan Sonar Data with the Woods Hole Image Processing System Software": U.S. Geological Survey Open-File Report 92-536, 90p.

NOTE

There are three major drawbacks to the pixel-by-pixel method employed in this program/mapping procedure and they are discussed below. Note of the drawbacks are made in an attempt to warn the user of possible problems which could affect the quality of their final mosaic when using the **gss | proj | projss** processing scenario, and to discuss possible future processing alternatives.

The first and second drawbacks are due to the random order of the input pixels being placed. This random input results in a last-in placement for coordinates which may have two or more valid pixels. Since the program can not place in order the multiple values for a given coordinate, the program must deal with each pixel as a unique value and is placed in its coordinate location regardless of any pixel values which may have been previously placed. Even a simple comparison for placement of pixels in non-zero locations and averaging of the pixels would be restrictive due to the overhead placed in computing. Secondly, if the input pixels could be placed in sorted order from the image origin down to the last line and last sample to be placed, processing could be accomplished by "building" a complete image line. This would result in fewer writes to the output file since a complete line of image data could be written rather than the many single writes which must be done for each pixel input Fewer writes would hopefully speed up processing. However, when processing over a half million pixels at a time (a modest file at best), speed may be best accomplished by increased workstation performance. The sorted order of the data would also be beneficial by allowing the program to compare multiple pixel values for a specific coordinate and select the final output pixel value by averaging or selecting either the minimum or maximum value.

The third drawback is the "holes" which may develop as the pixels are placed in the map space. These "holes" may be best eliminated, or reduced in number, by the user carefully selecting the resolution of the sonar data being processed with the goal in mind of the final resolution of the map to be created. If the user selects to process their sonar strips at .5 meters and their final map resolution is 1 meter, fewer "holes" will develop from the pixel placement. An obvious drawback is the volume of pixels that will have to be processed resulting in longer execution time. The user must consider the trade-off of processing such enormous volumes of data or what "holes" may be created in the mapping process. These "holes" must be filled in some manner. The preferred method would be some form of interpolation based on the orientation of the scan line and the appropriate neighboring pixels. To accomplish such a task, the sonar data must be processed on a swath-by-swath basis with two consecutive swath's being processed at a time. Unfortunately, the ability to handle the data on a pixel-by-pixel basis is the only option currently available. This results in the "holes" or gaps that may by created to be filled in some manner after the sonar map has been created. Possible options are either a low-pass filter replacing zero values, a 3-by-3 mode or median filter, a 5-by-5 median filter or some combination of the former. The "filtering" options are less than perfect since the zero pixels are filled relative to the orientation of the boxcar passing over the image, not the orientation of the scan lines. It may be with careful selection and application of the filters, the image will not be degraded significantly.

DIAGNOSTICS

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

BUGS

The program was written to assist in testing the necessary programs and steps to utilize program **proj** while prototyping the sidescan sonar line-by-line mapping. Program **projss** was not intended to be a final product program and does not contain sufficient user input checks. Therefore, the program may appear to run successfully and yet produce wild results based on the user's incorrect input.

AUTHOR/MAINTENANCE

Valerie Paskevich, USGS, Woods Hole, MA.

NAME

qmos - quick mosaic of two WHIPS netCDF images

SYNOPSIS

qmos -i *input* **-o** *output* **[-I | -O | -A] [-H]**

DESCRIPTION

Program **qmos** will mosaic (overlay) the specified input file (**-i**) over the specified output (**-o**) file. The program will either overlay where the input file has priority over the existing output file (**-I**, program default), where the output file has priority (**-O**) over the input file, or average (**-A**) non-zero pixel values together from the input and existing output file.

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 modified. The output file must currently exist.

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

-I

flags the program that non-zero pixel values in the *input file* are to take precedence. This is the program default. When the input file takes precedence, the non-zero value of a specific pixel coordinate will be replaced by the non-zero input pixel value for that location.

-O

flags the program that non-zero pixel values in the *output file* are to take precedence. When the output file takes precedence, the non-zero value of a specific pixel coordinate will not be replaced by the non-zero input pixel value for that location.

-A

flags the program to *average* non-zero pixel values from the input and output files. When averaging of the files is selected, the non-zero pixel values for a specific image coordinate are averaged together and the output pixel value is replaced with the new 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 selected input and output file must be the same size.

Unlike the majority of WHIPS programs where the output file must not exist and is created by the application program, the output file to be modified must currently exist for this application to execute successfully.

EXAMPLE

```
% qmos -o map.comp -i 126.map
```

SEE ALSO

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

sshead - computes simple heading for a WHIPS netCDF sidescan sonar image

SYNOPSIS

sshead -i *input* **-o** *output* [**-h** *heading_adjustment*] [**-H**]

DESCRIPTION

Program **sshead** reads a WHIPS netCDF sidescan sonar file and computes the heading value from swath to swath. The new heading value is then recorded in the sonar-attribute field in the output file. The heading is computed from the sonar position values contained in the header information of a WHIPS sidescan sonar image.

The new heading value is computed from consecutive coordinate pairs. Since the first valid computed heading value is for the second swath of the sonar image, a valid heading cannot be computed for the first swath. Therefore, the new heading of the first record is assumed to be the same as the heading for the second swath and is set accordingly.

The user may specify a value to be added to the computed heading value by selecting the program option -h. This allows the user a simple way to adjust the heading value computed from the navigation if they feel the navigation does not accurately reflect the track of the sonar fish.

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. The output file contains the sidescan sonar header information from the input file with the new heading values.

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

-h *heading_adjustment*

specifies a heading adjustment value to be added to the computed heading 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

Program currently accepts only 8-bit image files. The input file must contain the necessary sidescan sonar header information.

The output file to be created must not currently exist.

EXAMPLE

```
% sshead -i gloria.wcox -o gloria.head
```

SEE ALSO

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 sidescan sonar header information

atan2: DOMAIN error - may be displayed during programming. Processing continues until the end-of-file is encountered and the file is useable. The source of this error is assumed to be generated when trying to compute a heading between consecutive records with the same position.

Version 1.3 of the program was modified to compare consecutive positions, and when duplicates are found, output the last good heading value computed. This additional check will add some time to the processing but will hopefully eliminate the error problem. This change, however, does not eliminate the need to clean-up the navigation data set as much as possible to eliminate the large number of consecutive duplicate positions that can be found in something like the EG&G data sets. It is also highly recommended that the dataset have good unique positions at the beginning of the file so the heading can be computed accurately to start.

AUTHOR/MAINTENANCE

Valerie Paskevich
U. S. Geological Survey
Marine and Coastal Geology Program
384 Woods Hole Rd.
Quissett Campus
Woods Hole, MA 02543

NAME

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

SYNOPSIS

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

DESCRIPTION

Program **whips2raw** converts a WHIPS netCDF image file to a raw binary stream image file. The output of the program may then be converted to other image formats or imported to other software packages.

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.

APPENDIX B

The control file for program **mapit** is simpler to define than the previously used mapping procedure. Program **mapit** requires only one control file which contains the projection information and map area bounds. The previous mapping procedure required two control files: 1- contained the projection information; 2- contained the map area bounds. Below is an example of defining the **mapit** control file followed by the same mapping program specifications from USGS OPF-92-536. The contents of the ***mapit.dat*** file are shown and it's placement in the **mapit** run-line is highlighted for easier identification.

```
% more mapit.dat
proj=utm ellps=clrk66 south lon_0=-111
-34 -114
-36 -112

% mapit -i pass12.avg_head -o east.map -c mapit.dat -r 45,100
```

```
% more bounds.dat
-34 -114
-34 -112
-36 -112
-36 -114

% more utm-proj
proj +proj=utm +lon_0=-111 -r -s -m 1:100 -f '%.0f' bounds.dat -


% gss_vel -i pass12.avg_head -r 45 -s 490 | utm-proj | projss -o pass12.map
```


- PAGE 71 FOLLOWS -

APPENDIX C

Below is a summary of the steps used to complete the mapping and processing of the three components (east, west and south) of the two degree GLORIA sidescan sonar map. Mapping was accomplished in a sub-directory from where the individual passes were stored. The required programs and their parameters, along with any UNIX required commands, were entered into a file and executed as a single script. The required map control file, ***mapit.dat***, is shown first.

The script files closely matches the script file utilized in USGS Open-File Report 92-536 (Paskevich,1992c). However, it would be more practical to apply all pre-processing to the sonar swaths before beginning the mapping procedure. In other words, if the sonar heading values are to be smoothed by program **avg_heading**, this program should be applied to the swaths as part of the pre-processing and should be completed before starting the digital mapping. Unecessary processing will only increase the time required to complete the digital mapping process.

```
% more mapit.dat
proj=utm ellps=clrk66 south lat_0=-110
-34 -114
-36 -112

% more do-map
# ************************************************************************
#   do_east component
# ------------------------------------------------------------------------
#
avg_heading -i ../pass41.head -o pass41.avg_head -l 9
mapit -i pass41.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass42.head -o pass42.avg_head -l 9
mapit -i pass42.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass43.head -o pass43.avg_head -l 9
mapit -i pass43.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass11.head -o pass11.avg_head -l 9
mapit -i pass11.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass12.head -o pass12.avg_head -l 9
mapit -i pass12.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass13.head -o pass13.avg_head -l 9
mapit -i pass13.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass26.head -o pass26.avg_head -l 9
mapit -i pass26.avg_head -o east.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass27.head -o pass27.avg_head -l 9
mapit -i pass27.avg_head -o east.map -c mapit.dat -r 45,100
#
# ------------------------------------------------------------------------
#   fill gaps in east component map
# ------------------------------------------------------------------------
#
mode3 -i east.map -o east.mode3 -z -Z
mode5 -i east.mode3 -o east.mode5 -z -Z
```

```
#
rm east.mode3
#
mode3 -i east.mode5 -o east.mode3
#
filter -i east.mode3 -o east.lpfz -z -b 3,3
rm east.mode3 east.mode5
#
# ************************************************************************
#   do_west component
# ------------------------------------------------------------------------
#
avg_heading -i ../pass37.head -o pass37.avg_head -l 9
mapit -i pass37.avg_head -o west.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass38.head -o pass38.avg_head -l 9
mapit -i pass38.avg_head -o west.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass39.head -o pass39.avg_head -l 9
mapit -i pass39.avg_head -o west.map -c mapit.dat -r 45,100
#
avg_heading -i ../pass25.head -o pass25.avg_head -l 9
mapit -i pass25.avg_head -o west.map -c mapit.dat -r 45,100
#
mapit -i pass26.avg_head -o west.map -c mapit.dat -r 45,100 -l 1,302
#
# ------------------------------------------------------------------------
#   fill gaps in west component map
# ------------------------------------------------------------------------
#
mode3 -i west.map -o west.mode3 -z -Z
mode5 -i west.mode3 -o west.mode5 -z -Z
#
rm west.mode3
#
mode3 -i west.mode5 -o west.mode3 -z -Z
#
filter -i west.mode3 -o west.lpfz -z -b 3,3
rm west.mode3 west.mode5
#
# ************************************************************************
#   do_south component
# ------------------------------------------------------------------------
#
mapit - pass26.avg_head -o south.map -c mapit.dat -r 45,100 -l 324,588
#
# ------------------------------------------------------------------------
#   fill gaps in south component map
# ------------------------------------------------------------------------
#
mode3 -i south.map -o south.mode3 -z -Z
mode5 -I south.mode3 -o south.mode5 -z -Z
#
rm south.mode3
#
mode3 -i south.mode5 -o south.mode3 -z -Z
#
filter -i south.mode3 -o south.lpfz -z -b 3,3
rm south.mode3 south.mode5
```

REFERNCES

Adobe Systems Incorporated[2], 1994, Adobe Photoshop 3.0 User Guide.

Chavez, Pat S., 1986, Processing Techniques for Digital Sonar Images from GLORIA, Photogrammetric Engineering and Remote Sensing, vol. 52, No. 8, pp. 1133-1145.

Corel Corporation[3], 1994, CorelDraw User's Manual, Vol. 1, version 5.0, pp. 423-478.

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

Evenden, Gerald I., 1994, Cartographic Projection Procedures Release 4 Interim Report, 42 p.

Evenden, Gerald I., 1994, Cartographic Projection Procedures Release 4 Second Interim Report, 21 p.

Mazel, C., 1985, Side scan sonar training manual, Klein Associates, Inc., Salem, N.H.

Miller, Richard L., Dawn, Fa S. and Cheng, Chiu-Fu, 1991, Digital Preprocessing Techniques for GLORIA II Sonar Images, Geo-Marine Letters, 11:32-31.

Paskevich, Valerie, 1992a, Woods Hole Image Processing System Software Implementation: Using NetCDF as a Software Interface for Image Processing, Open-File Report 92-25, 72 p.

Paskevich, Valerie, 1992b, Digital Processing of Side-scan Sonar data with the Woods Hole Image Processing System Software, Open-File Report 92-204, 9p.

Paskevich, Valerie, 1992c, Digital Mapping of Side-Scan Sonar Data with the Woods Hole Image Processing System Software, Open-File Report 92-536, 89 p.

PCI[1], GCPWorks Reference Manual: Version 6.0, November 1995, 145 p.

Snyder, J.P., 1987, Map projections - A working manual: U.S. Geological Survey Professional Paper 1453, 249 p.

Snyder, J. P. and Voxland, R.M., 1990, An album of map projections: U.S. Geological Survey Professional Paper 1453, 249 p.

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

---

[1] PCI Remote Sensing Corporation http://www.pci.on.ca

[2] Adobe Systems Incorporated http://www.adobe.com

[3] Corel Corporation http://www.corel.com