SEMI–AUTOMATED EXTRACTION OF STRUCTURAL ORIENTATION DATA FROM AEROSPACE IMAGERY COMBINED WITH DIGITAL ELEVATION MODELS

FRANS BRESLER SLABBER

Thesis submitted in fulfilment of the requirements for the Degree of Master of Science in the Department of Geology, Rhodes University, Grahamstown

January 1996

ABSTRACT

A computer–based method for determining the orientation of planar geological structures from remotely sensed images, utilizing digital geological images and digital elevation models (DEMs), is developed and assessed.

The method relies on operator skill and experience to recognize geological structure traces on images, and then employs software routines (GEOSTRUC©) to calculate the orientation of selected structures. The operator selects three points on the trace of a planar geological feature as seen on a digital geological image that is coregistered with a DEM of the same area. The orientation of the plane that contains the three points is determined using vector algebra equations. The program generates an ASCII data file which contains the orientation data as well as the geographical location of the measurements. This ASCII file can then be utilized in further analysis of the orientation data.

The software development kit (SDK) for TNTmips v5.00, from MicroImages Inc. and operating in the X Windows environment, was employed to construct the software. The Watcom C\C++ Development Environment was used to generate the executable program, GEOSTRUC© .

GEOSTRUC© was tested in two case studies. The case studies utilized digital data derived from the use of different techniques and from different sources which varied in scale and resolution. This was done to illustrate the versatility of the program and its application to a wide range of data types. On the whole, the results obtained using the GEOSTRUC© analyses compare favourably to field data from each test area. Use of the method to determine the orientation of axial planes in the case study revealed the usefulness of the method as a powerful analytic tool for use on a macroscopic scale.

The method should not be applied in areas with low variation in relief as the method proved to be less accurate in these areas. Advancements in imaging technology will serve to create images with better resolution, which will, in turn, improve the overall accuracy of the method.

TABLE OF CONTENTS

LIST OF FIGURES

FONT CONVENTIONS

C program instructions are typeset in a typewrite font, for example,

`printf("Hello");`.

X Window function calls are typeset in a slanted, bold font, for example,

XtAddCallback().

TNTmips menu options are typeset in a bold font with hyphen connecting options descending the menu hierarchy, for example,

prepare-raster.

ACKNOWLEDGEMENTS

I would like to express my gratitude to the Anglo–American Corporation of South Africa for funding this research, Dr Robin Harris for supervising the research and suggesting the fundamental concept, and the Rhodes University Geology Department for providing the necessary computing facilities as well as the opportunity to complete a Master of Science degree.

Finally, I would like to express my sincere gratitude to Jane Don–Wauchope without whose support and encouragement this project would still be unfinished.

CHAPTER ONE

INTRODUCTION

Traditionally, structural information of a particular area is obtained by a combination of photo–interpretative analysis and extensive fieldwork. The fieldwork consists of the measurement of structure orientation by hand–held devices, such as Brunton compasses. Sometimes, the fieldworker has to access rugged terrain, which makes this method difficult, time consuming and costly.

The photo–interpretation phase, for obtaining structural information, was previously limited to an analysis of the available imagery, usually in hardcopy format, by a skilled photogeologist . The photogeologist would identify surface expressions of structures on imagery. If stereo–coverage of the particular area was available, the photointerpreter would painstakingly produce structural orientation data by utilizing photogrammetric methods, such as elevation–by–parallax–differences (EPD), as described by Wolf (1974). The EPD method involves the calculation of the elevation of an arbitrary point by comparison with the known elevation at a particular point, using the parallax difference between the two points. The parallax for a particular point may be measured using a parallax bar.

Three dimensional data obtained in this manner provided insight into the orientation of structures. In general, the photo–interpretation phase proved time consuming, especially when the information volume increased due to better imaging technology.

Both the fieldwork and the photo–interpretation phases are labour intensive and require a great deal of time.

A process that would minimize the amount of fieldwork, thus reducing the cost and time factor, and speed up the accurate assessment of the structural orientation data for a specific area, was required.

This thesis descibes the development and subsequent evaluation of a semi–automated

process for analyzing the orientation of subsurface geological structures, using a combination of aerospace imagery and digital elevation models. Usage of the term semi–automated indicates the use of interactive photointerpretative skills, augmented with computer routines, in the process. Aerospace imagery is a collective term for digital aerial photography and satellite imagery, both stored in a digital format accessible by computer.

1.1. Preamble to Process Development

The progress in computer technology, that is, in low–cost hardware with sufficient speed and processing power, together with the occurrence of aerospace imagery in digital format, makes the development of a computer–based technique a viable option. The progress in hardware goes hand–in–hand with advancements in computer languages, providing system software engineers with the necessary high–level interaction with operating systems. Together, these two factors supply the medium to develop complex image processing routines such as the one proposed in this thesis.

For the past few years the use of high resolution aerospace imagery has become common place amongst photogeologists, mainly due to easier access to such imagery by the public. The quality of such imagery, in terms of resolution and noise levels, has improved greatly and the promise of products covering a greater spectral range on future platforms justifies the incorporation of aerospace imagery into structural analysis methods. An example of high resolution imagery suitable for structural analysis are SPOT (Systeme Pour l'Observation de la Terre) panchromatic images, which have a 10m ground resolution (Lillesand & Kiefer, 1987).

The two dimensional nature of imagery, however, is restrictive for the analysis of three dimensional structures. Techniques, such as shape–from–shading described by Wadge *et al.* (1990), do, to some extent, provide orientation data from imagery, but have several disadvantages as outlined below. Wadge *et al.* (1990) proposes the extraction of structural information via the mapping of topographic "edges" that correspond to lithological boundaries and quantifying the plane orientation by photoclinometric methods. Photoclinometric methods are ways of quantifying the relationship between variations in the reflected light and local surface orientation.

The drawbacks of the method proposed by Wadge *et al.* (1990) are, the restrictive nature of the specific geological enviroment in which it can be used (the process assumes that the topography is mainly structurally controlled as opposed to stratigraphically controlled), the homogeneous reflectance of the surface cover, and the assumption that the sun is the only source of illumination.

These disadvantages could be overcome by the addition of information to the geological image. A simple way to do this is to try and produce a three dimensional image, either by direct superimposition or by creating a link between the geological image and a Digital Elevation Model.

Definition 1. *A Digital Elevation Model (DEM) is a digital image with the cell values corresponding to absolute or true elevation of the geographical area covered by the image.*

A source of relatively cheap DEMs, at 200m ground resolution, covering most of South Africa, is available from the South African Government Survey Office. This source is constantly being expanded and the existence of such a central DEM facility ensures that the data format conforms to an international standard[1], making it portable. Furthermore, most modern geographic information system computer packages provide routines for the production of DEMs from topographical maps, stereopairs of aerial photographs and satellite images.

The existence of the above-mentioned technology motivated the development of an image processing routine to measure subsurface orientation of geological structures. Such a tool needs to be fast, cost-effective and consistent in its results. It could minimize time spent on field work as the geologist need only validate the results of the process with minimal fieldwork. Large areas could be structurally assessed faster than conventional methods. Areas with large-scale complex geological structures may be easier to decipher as the geologist would have a regional overview of the geology at all times, compared to a fieldworker who is restricted to viewing local structures only.

[1] DEMs in the NDEM format are available from the Government Survey Office.

1.2. The Role of GIS and Image Processing

Geographical Information Systems (GIS) provide an existing environment for the creation of an analysis process. These systems usually contain most of the necessary image manipulation tools and mapping routines which provide a foundation for the construction of a structural analysis routine.

This section introduces GIS in a brief overview, as well as existing image processing routines, placing emphasis on those tools and routines that are of relevance to the development of a computer–based structural remote sensing routine.

1.2.1. An Introduction to GIS.

A Geographic Information System is an information system applied specifically to geographical data. The system incorporates combinations of geographically referenced data and non-spatial data, and includes operations for the analysis of the data, to reveal proximity relationships that may be used in a decision making process.

Defining the scope of a GIS is a study on its own and a lot of research has been done. Cowen (1988) categorizes four general approaches to defining GIS;

(1) Process–Oriented – A GIS is a collection of routines that convert geographic data into useful information.

(2) Application – A GIS is a collection of subsystems, each one designed to manipulate specific types of information.

(3) Toolbox – A GIS is a set of sophisticated computer algorithms for the handling of spatial data.

(4) Database – A GIS is a set of computer routines for the management of a spatial database.

More information on the scope and definition of GIS is available in Burrough (1986), Maguire (1991), and Marble (1990).

1.2.2. Image Object Types.

The progress in image and GIS processing has benefitted greatly from the advancement in computer technology. Many commercial packages, such as ArcInfo and Erdas, are now available to the user, providing a multitude of tools, sufficient for most GIS

related operations. Although these packages differ greatly in certain areas, such as storing data in a particular file format, they share universal principles in terms of image types. Therefore raster and vector–based images are described, and their advantages and drawbacks outlined.

Definition 2. *A Raster is defined as a digital computer image with cell values stored in consecutive addresses in computer memory, such that there exists a one–to–one correspondence between the pixel position on screen and the memory address. The size of the storage area in memory, assigned to one pixel, may vary according to the data type used.* (**Fig. 1a.**)

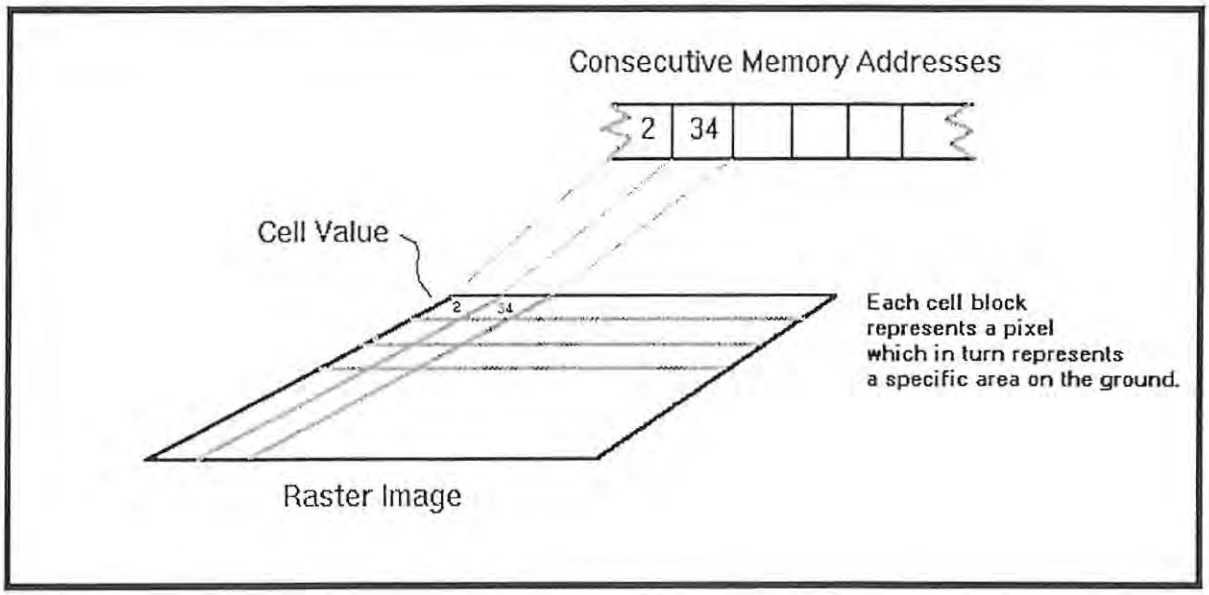


Fig. 1a. The raster image and its relation to computer memory.

The amount of colour variation available depends on the data type used. For a 8–bit raster, there are 256 values available, and this kind of raster is typically used for monochrome images. The 16–bit raster has 2 bytes per cell available, and therefore can assume 65 536 values. This type of raster is commonly used for the display of elevation data. A colour composite raster, with 16–bits per cell, will have 5 bits for each of the colour components red, green and blue (the 16th bit is unused). Each cell in a 16–bit colour composite raster can therefore take on one of 32 768 possible colours. Lastly, the 24–bit raster, has 8 bits available for each of the RGB colour components (Red, Green,

Blue), that is, each component can take on one of 256 levels, giving the cell a choice of 16 777 216 (256x256x256) colours.

Rasters enable easy manipulation of imagery data and lend themselves to matrix operations such as filtering, as well as lending themselves to georeferencing, a process that geographically locates any given digital image on the surface of the earth in a predetermined projection system. This operation is termed georeferencing or registering an image. It involves assigning geographic location, in a projection system, to specific pixels in the image, thus enabling the GIS system to "place" the image on the surface of the earth.

The disadvantages of the raster structure include the large amounts of storage memory required compared to other image types, the high computing time necessary to perform operations on it, and the large amount of time required to display images with the raster structure.

Definition 3. *A vector is a digital computer image, with each geometric surface classified into one of the following classes; point, line, or polygon. This means that a vector image consists of a collection of point coordinates, lines, and polygons, together with their attributes.*

A line is a set of ordered points, and a polygon is a set of ordered lines that form a closed curve. Element attributes are characteristics that determine the display style of the element, for example outline style and colour. As opposed to rasters, vectors require less memory space and can contain much more information.

The aerospace imagery used in this study are all in the raster image format. This enables the easy and fast access of elevation values in the DEM. To retain the structural information a vector image object will be used by the structural analysis routine.

1.2.3. Digital Elevation Models.

As DEMs form an integral part of the development of a structural analysis routine, more time is spent in this section on the origins of DEMs. DEMs may be produced in a number of ways. The quality of the resultant DEM, however, is a function of the data capture method and the source material. Commonly, there are three ways of preparing a DEM, namely ground surveys, photogrammetric methods, and cartographic techniques

(Weibel & Heller, 1991).

Ground surveys usually produce DEMs of the highest quality. A modern operator would use a GPS[2] and measure sample points in the field. Depending on the GPS used, this kind of DEM generation produces DEMs with a high accuracy. As the operator directly controls the data gathering, more sample points could be added in areas of rapid variation in the relief. The technique is labour intensive, and usually restricted to covering small study areas.

For the generation of DEMs covering large areas, the utilization of photogrammetric methods are more effective than ground surveys. Photogrammetric methods require stereopairs of aerospace images and some kind of device for measuring parallax differences, usually a parallax bar. Mechanical stereoscopic plotters, used in the production of topographic maps, could also be used. Using this setup, the area is sampled at various locations for elevation. In such a way, a 3D image of the area is constructed. This manual process is time consuming, especially as satellite stereo–imagery, that is, large volumes of data, become more readily available.

Most GIS–related computer packages include photogrammetric software that automatically generate DEMs from image stereopairs. For example, SPOT satellite scenes of the same location, viewed from different orbital positions, provide image pairs. The high resolution of SPOT imagery and near global coverage make these stereopairs ideal for the generation of DEMs (Sasowsky *et al.*, 1992; Simard *et al.*, 1988). Generally, automated stereomatching techniques employ an image matching process as a first step. This matching process identifies corresponding pixels on both pairs that represent the same geographical location. Once this correlation process is completed, the elevation of the area is calculated, pixel–by–pixel, by converting parallax values to elevations through spatial intersection. The latter step depends on *a priori* knowledge of the orbital parameters of the SPOT satellite, logged in the Leader File for each scene.

Computer automated stereomatching techniques, however, do still require further development and have some problems. Time–elapse between left and right image acquisition creates problems in the image correlation step of the process. That is, seasonal variation

[2] A Global Positioning System determines geographical location from signals received from a series of satellites.

in vegetation, for example, changes the reflectance values for a particular point and its neighbourhood. This causes pattern mismatching between two geographically similar points. Finally, noisy digital data could also introduce errors (Day & Muller, 1988).

The final method of DEM generation is the cartographic method. Topographical maps of an area provide contour data which is a reflection of the area's relief. There are two approaches to generating DEMs from topographical maps. In the first approach, a DEM raster is compiled by digitizing points on contour lines on the topographical map.

The second approach makes use of a trace of the contour lines, compiled manually, and an automatic line–following routine. The trace of the contours is converted to digital format through scanning. A vector object is generated by means of a line–following routine and the individual contour heights are attached to the respective lines. A vector to raster conversion is the final step to produce a DEM.

Both the approaches outlined above have the disadvantage of oversampling elevation data along the contour line, and undersampling elevation data in the direction perpendicular to the contour line.

The three methods outlined above produce an image with irregularly spaced sample points. This is rectified by resampling the image to a rectangular grid. A multitude of interpolation techniques used in the resampling process exist (Watson, 1992a), and computer packages usually provide a selection to choose from.

Interpolation (in a GIS context) is defined as follows:

Definition 4. *Suppose there exists a given set of observation points in a plane, each with its own xy–coordinates, then interpolation is a rule by which we calculate the value of an observation at arbitrary coordinates.*

Interpolation techniques assume that the surface being approximated is continuous (mathematically), which is the case with relief. They also work with a finite set of data points in a fixed spatial position, so that the technique cannot be expected to resolve relief features if the dimensions of such features is less than the minimum distance between two observations. On the other hand, interpolation techniques can introduce features that do not actually exist on the original surface.

Examples of common interpolation techniques that could be applied in the resampling

process are; inverse distance methods, Delauney triangulation and kriging.

Inverse distance methods use the assumption that the further a known observation is from an interpolation point the less influence it has on the calculation of the interpolation point. Horton (1923) was the first to suggest making the weight inversely proportional to the distance. Weight is a factor multiplied with an observed value and is a measure of the influence of that observation at an arbitrary coordinate. For example, given 3 points x_i, let $f(x_i)$ denote the observed value at x_i, then the interpolated value at an arbitrary point x_a is given by,

$$f(x_a) = \sum_{i=1}^{3} w_i f(x_i) \quad where \; w_i = \frac{\frac{1}{d_i}}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3}}$$

and d_i denotes the distance from x_a to x_i. Note that the weights w_i are normalized,

$$\sum_i w_i = 1$$

The disadvantages of this method are that the surface is discontinuous at datums and that the method cannot infer surface extensions beyond the maximum and minimum of the observed values.

The Delauney triangulation involves two stages. The first stage is the determination of the natural neighbourhood of a known elevation point by means of Delauney triangulation. For each triplet of known elevation points, a circumcircle is constructed, that is, a circle that passes through each of the three points and does not include any other elevation point. The centres of the circles contructed over the whole data set are connected by lines in such a manner that the resultant lines outline polygons about each elevation point. These proximal polygons define the so-called natural neighbourhood of an elevation point.

The second stage in the Delauney triangulation method fits a plane function to each triplet respectively and in this manner supplies a way of determining the elevation of an arbitrary point located in the data set. Conversely, the elevation of an unknown point could be determined by assigning a planar surface to each proximal polygon. The height of these planar surfaces being equal to the elevation of the observation point within the polygon.

Once the Delauney triangulation for a data set has been constructed, any function (not necessarily linear) could be introduced for the interpolation.

Kriging involves the fitting of a surface to the whole data set or, more commonly, to a subset at a time. The fitted surface is constructed by building a surface from basis functions, user or pre–defined, that describe minimum variance of the surface. The surface function is a linear combination of these basis functions. The accuracy of the kriging method relies on the choice of basis functions.

Current research in using laser beams to determine elevation from orbiting or airborne platforms might in the future lead to the production of precise DEMs. Unfortunately the current public access to DEMs produced in this manner is restricted, due to control by the U.S. military[3] (Drury, 1987a).

1.2.4. Databases.

A collection of data filed in an ordered manner, together with a set of access and manipulation functions, is called a database. Most GIS packages have custom databases as part of the system, or have the ability to interact with existing ones, such as DBaseIV and Paradox. In GIS, databases can be linked directly to images providing user–friendly access to image information. In other words, in GIS, databases provide a way of relating non–spatial data to geographical location.

The development of a process to determine the orientation of geological structures relies on the capability to assign resultant measurements to geographical location. This assignment involves the construction of a database with the relevant structural information linked to a vector image object that displays the location of these measurements.

1.3. Previous Research

The groundwork for processes that measure dip and dip–direction of geological structures from aerospace imagery was established in papers by Chorowicz *et al.* (1991) and Morris (1991). In these papers the authors demonstrate different approaches to developing techniques for the systematic extraction of the required information.

[3]The DEMs produced in this manner are used in the programming of Cruise missile guidance systems.

1.3.1. Chorowicz' Method.

As source material, Chorowicz (1991) uses a 1:50 000 geological map that was digitized using a scanner. Possibly due to poor reproduction, geological contacts were redrawn so that they were more visible. The scanning resolution was set such that an equivalent ground resolution of 5x5m per pixel was achieved for the maps.

A DEM was derived by digitizing 20m contour lines obtained via photogrammetric stereoscopic modelling on 1:60 000 aerial photographs. The elevation values were then interpolated using the elastic grid method to generate a 50m square grid. A resampling of this DEM, so that the ground resolution would be consistent with the geological map, was performed. The next step in the process involved the superimposition of the digital geological map on top of the DEM providing a 3D geological map.

Since any three points in R^3 (Three Dimensional Eulcidean Real Space) uniquely define a plane, one needs only to identify three points on a geological surface in order to obtain an unique solution of the dip/dip–direction of the geological surface.

Chorowicz' (1991) method relies on the ability to automatically extract accurate traces of thalwegs from the DEMs.

Definition 5. *The line connecting the lowest points along a stream bed or valley is called a thalweg (Bates & Jackson, 1984).*

This is done by scanning the DEM raster along each row, filtering the resultant profile for thalwegs. The filtering process involves analysing the DEM row, which is a row of numbers, by means of an algorithm that detects local minima utilizing the gradient of the profile. The filtering process excludes flat–bottomed valleys so that the precise location of thalweg/contact intersection may be determined. The thalweg/contact intersection forms the location of one of the points in a 3–point cluster. (**Fig. 1b.**)

The reason for this choice of location, according to Chorowicz (1991), is the relatively high difference in relief that thalwegs offer, ensuring the non–colinearity of points selected for one plane. A plane is fitted to the three points, and the orientation of the plane calculated. For each thalweg/contact intersection seven different 3–point clusters are found. The average of these seven measurements is taken as the resultant dip/dip–direction of the geological surface.

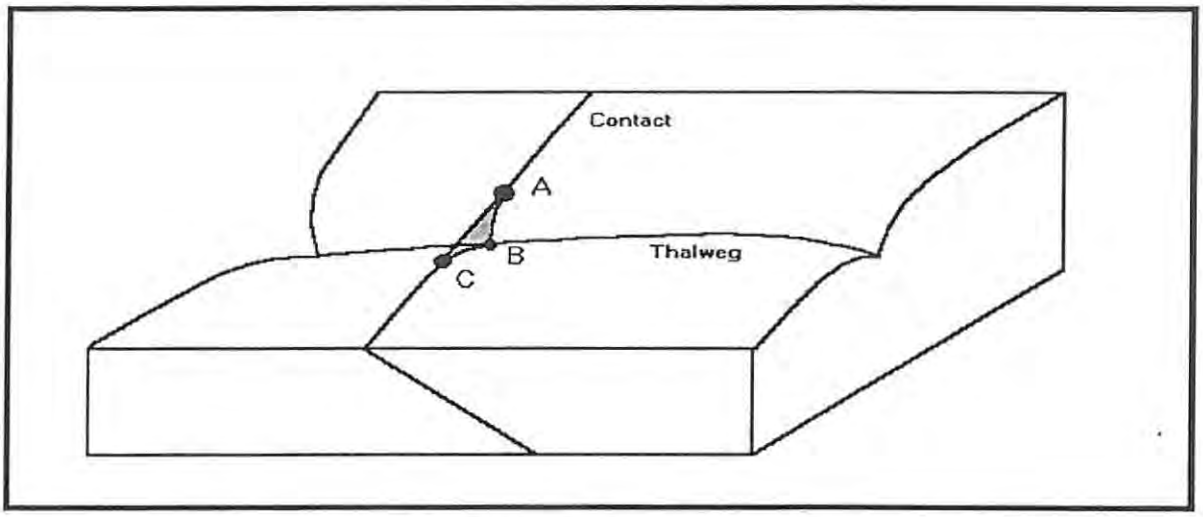


Fig. 1b. Location of three points on the geological contact. The middle point (B) is located on the thalweg/contact intersection

Chorowicz (1991) concludes his paper by illustrating the use of such a technique in a case study, which brings to light several shortcomings of the technique. When compared to field measurements, his method indicates accurate results in areas of high relief and steeply dipping bedding planes, however, the results in low relief areas are less reliable, and dips calculated tend to be steeper than field measurements. Reasons cited for these errors are the poor registration of the image and the DTM, and the occurrence of small scale folding that will induce large variations of local dip/dip–direction in small areas. Chorowicz (1991) also suggests the manual redrawing of contacts across thalwegs to ensure that the apex of the V is positioned within the thalweg.

Chorowicz' (1991) method, as with any fully automated process, is prone to problems in the initial pattern–recognition phase. That is, thalweg detection by row filtering depends on the local minima detection algorithm being able to differentiate between thalwegs and noisy data. More research needs to be done in this area.

Another drawback is the usage of a scanned geological map which introduces location errors since the reliability of contact position is dependent on the original mapper's positioning and the scale of the map.

As an automated process it fails as a lot of operator control is required to ensure the production of accurate data.

1.3.2. Morris' Method.

Morris (1991), describing a technique developed from aerospace imagery, suggests intensive image processing on the DTM before analysis. His process relies on a knowledge–based system to assess the accuracy of orientation data produced.

Digital geological imagery was obtained from a Daedadus Airborne Thematic Mapper sampling the scene in eleven bands, from visible to infra–red. Geometric correction of the data proved difficult, and this Morris (1991) attributed to the wide scan angle of 86 degrees. Ground resolution obtained for the raster image was 5x5m per pixel.

A DEM was generated by digitizing a 1:10 000 topographical sheet with 10m contour intervals. The DEM was resampled in order to obtain the same spatial resolution as the geological image.

The first image processing operation Morris (1991) performed was Lambertian shading (MicroImages, 1993) of the DEM raster image from two perpendicular directions to exclude directional preference in the enhancement process. The motivation for Lambertian shading was the highlighting of geologically complex areas it produced, so that the image could be segmented into areas of varying geological complexity.

The next step involved filtering both the DEM and the geological image with a Sobel filter (Drury, 1987b). A Sobel gradient filter is a matrix operator that enhances oriented linear features (edges). The preferred orientation is dependent on the matrix entries.

The end result of this preprocessing was a 3D geological image with amplified linear geological features, from which the orientation could then be determined. Several points from this 3D geological image were selected along a lithological boundary or a fault. A plane was fitted to these selected points, using a least–squares approximation procedure. **(Fig. 1c.)**

A major difficulty with this method, as pointed out by Morris (1991), is the colinearity of data along a linear feature. Mathematically, points that lie on a straight line in R^3 define an infinite number of planes. Although this problem could be overcome by statistically assessing the linearity of groups of points, this would mean more calculation being performed which all adds to the complexity of the process and creates even more opportunity for error.

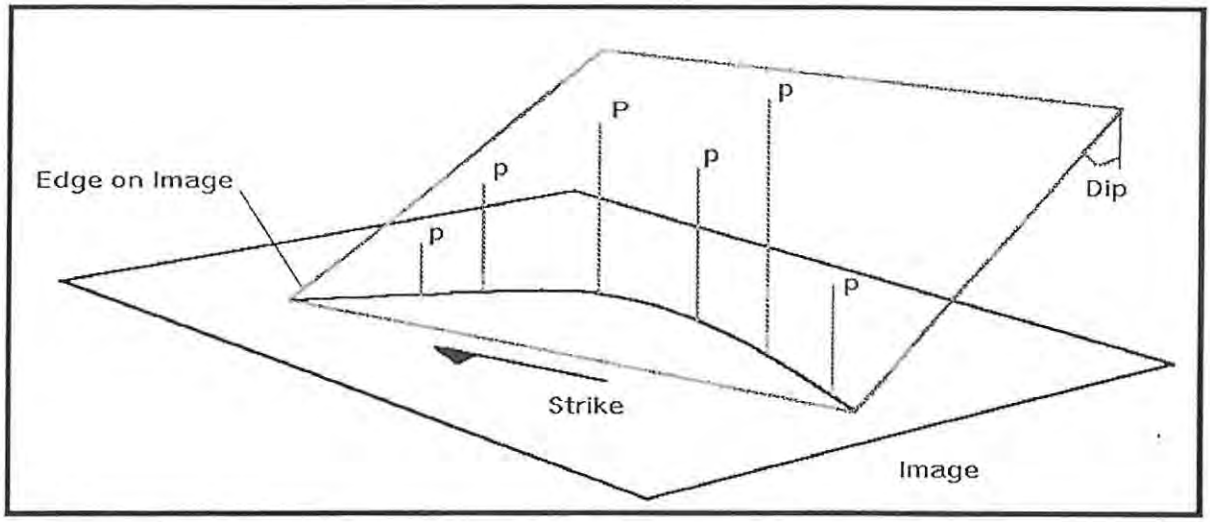


Fig. 1c. Planar surface fitted to points (P) on image.

1.3.3. General Critique.

Both these authors attempted to develop a process that is fully automated, or that at least necessitates minimal operator interference. Although Morris' (1991) method has been utilized with reasonable success (Mahon & North, 1993), the technology, in terms of the hardware, software and methods of pattern recognition, is not sufficiently developed for the creation of a fully automated system. As can be seen from the previous research in this area, the method is either highly specialized for specific geological conditions, or the operator has to spend a lot of time correcting errors or validating process results.

1.4. Existing Software Routines

Several image processing and GIS software packages exist that provide the means for extracting subsurface structural orientation data from imagery. Examples are MICRODEM and VUE3D.

MICRODEM is a PC MSDOS based menu-driven image processing system. The source code is written in Pascal and the program is readily available from Petmar Trilobite Breeding Ranch Software. Written by Peter L. Guth, it incorporates a routine that calculates the orientation of a plane which intersects three points.

Very little is known about VUE3D. The program is mentioned in papers authored by Reynes *et al.* (1993) and Grimaud *et al.* (1994). Attempts to contact these authors have been unsuccessful at the time of writing this thesis.

1.5. The Aims of this Project

The basic principles behind Chorowicz' (1991) and Morris' (1991) techniques point the way for the development of a semi–automated process that will determine the orientation of geological structures such as faults, joints, and axial planes. To be more specific, the aims of this project are;

A. To develop the necessary software for the direct measurement of orientation data of geological structures that is user–friendly, interactive and assigns complete control to the operator at all times, satisfying the following specifications;

- The process must be able to store data in a format accessible to external databases, or geological analysis routines such as StereoNet.
- The developed software must not be restricted to operating on a single platform, that is, it must be fully portable. Ideally the chosen platform must be accessible to the field geologist.
- The process itself must allow for a wide variety of source material formats. Digital imagery, usable in a geological context, are available from a wide range of sources such as aerial photographs (panchromatic or colour) and satellite images with a wide variety of wavelengths. DEMs can be generated by any of the previously discussed methods.
- The results must be accurate within geologically acceptable limits, so that they may be incorporated into structural maps, databases and structural models.

B. To test the resultant process on areas with different geological environments, that is, areas that differ in lithology, structural complexity and geomorphology.

C. To evaluate the effect of scale changes, raster cell resolution variation and DEMs of different accuracy on the validity of the results obtained from the process. This testing of the process will be done by performing two case studies in different areas with different source material, comparing the results to existing fieldwork. Ultimately, the most cost–effective way to accurately analyse a region's structure by remote sensing methods will be suggested.

CHAPTER TWO

SOFTWARE SYSTEMS UTILIZED IN PROCESS DEVELOPMENT

To satisfy desired portability and ease of operation requirements specified in chapter one, the TNTmips GIS and image processing system, operating in the X Windows environment, was chosen for the development of the structural analysis routine. In this chapter the afore mentioned systems are discussed.

The fundamentals of the X Windows system, which was selected as a developing environment for its functionality across a large choice of platforms, is briefly described.

The second section in this chapter is devoted to the TNTmips software package and the Watcom C/C++ Compiler, which are both utilized in the development process.

2.1. The X Windows System

The X Windows System or X is a Graphics User Interface (GUI) that is platform and hardware independent. The system was developed jointly by the Massachusetts Institute of Technology and Digital Equipment Corporation. The aim was to produce a windowing system that was independent of operating platform or hardware architecture and that was freely available to anyone. This would in turn encourage standardization of windowing systems.

The X Windows System is currently maintained by the X Consortium, an independent, non–profit organization that was formed in 1993 as a successor to the MIT X research group. The X Consortium acts as an administrator of all X related software. It actively promotes the evolution and development of the X windowing standard and participating members of the consortium are drawn from a broad spectrum of interested parties.

2.1.1. X Window Principles.

X directly controls each picture element (pixel) of a display device (bit–mapped graphics), enabling simultaneous display of graphics and text on the screen. It divides the

screen into multiple input and output areas called windows, providing a powerful multi-tasking environment.

X is a network-based system. Each machine using X runs an *X Server*, which is a program that acts as a translator between hardware and X instructions. The X server manages one screen, some pointing devices (mouse, trackball), and the keyboard. User programs that operate in this X server environment are known as *clients*. A client sends drawing requests and information requests to the server, and the server sends back user input to the client, replies to information requests, and reports errors.

X provides the means for clients running on one machine to have output on another, without the two machines being of the same type. This motivates the use of *server* as a name for the software that acts as a go-between between clients and the resources of the local system.

Tasks that the server performs include: allowing access to system resources by multiple clients; interpreting network messages from clients; displaying two-dimensional graphics; maintaining windows, cursors, fonts and graphics.

2.1.2. X Protocol Messages and Xlib.

Communication between the server and the client is in the form of a set of X protocol messages (Nye, 1990). These messages are designed to communicate all information necessary to operate a window system over a network system.

X protocol uses a programming library to interface with underlying network protocols, such as TCP/IP or DECnet, in cases where the client and server are not resident in the same machine, and local interprocess communication channels in cases where the client and server are resident in the same machine. The X protocol supports four kinds of messages that may be transferred over the network;

(1) A protocol *request* generated by the client asking for information on display characteristics, or inquiring about the state of the system.

(2) A *reply* sent from the server in response to a request, containing specified information.

(3) An *event* generated by the server notifying the appropriate client about a device action (mouse click) or about a side-effect of a previous request.

(4) An *error* event, which is similar to an event but handled differently by clients.

When developing X clients the programmer rarely deals with the X protocol explicitly, because the X protocol is embedded in higher level function calls found in toolkits such as Xt Intrinsics and Motif.

Xlib is a low–level C language interface with the X protocol. It translates C data structures and procedures into the special form of X protocol messages and thereby provides a C language interface between client and server. Xlib also performs several important functions with respect to the managing of the X operating enviroment as outlined below (Nye, 1992).

2.1.2.1. Event Buffering.

Client requests to the server are buffered by Xlib so that the client may continue operating and not have to wait for a server response. This is possible since not all requests made by the client require immediate action. The event queuing feature of Xlib increases the performance of the network. That is, communication between client and server can be longer since the messages are queued and released simultaneously, and are, therefore, less numerous.

2.1.2.2. The Window Manager.

This is a special client written with Xlib, which has the responsibility of managing the display area and manipulating windows on the screen. The window manager performs the following operations;

(1) Placement and movement of windows.

(2) Iconification[1] of windows.

(3) Starting and manipulation of windows.

(4) Input to windows.

These operations are implemented by the addition of features such as titlebars, resize and icon tools to every window opened by an application.

Finally, Xlib provides the means for reducing network traffic by utilizing data abstraction methods, that is, identifying window properties by numerical identification

[1] Iconification is the process whereby an open window is replaced by a smaller icon to save space on screen.

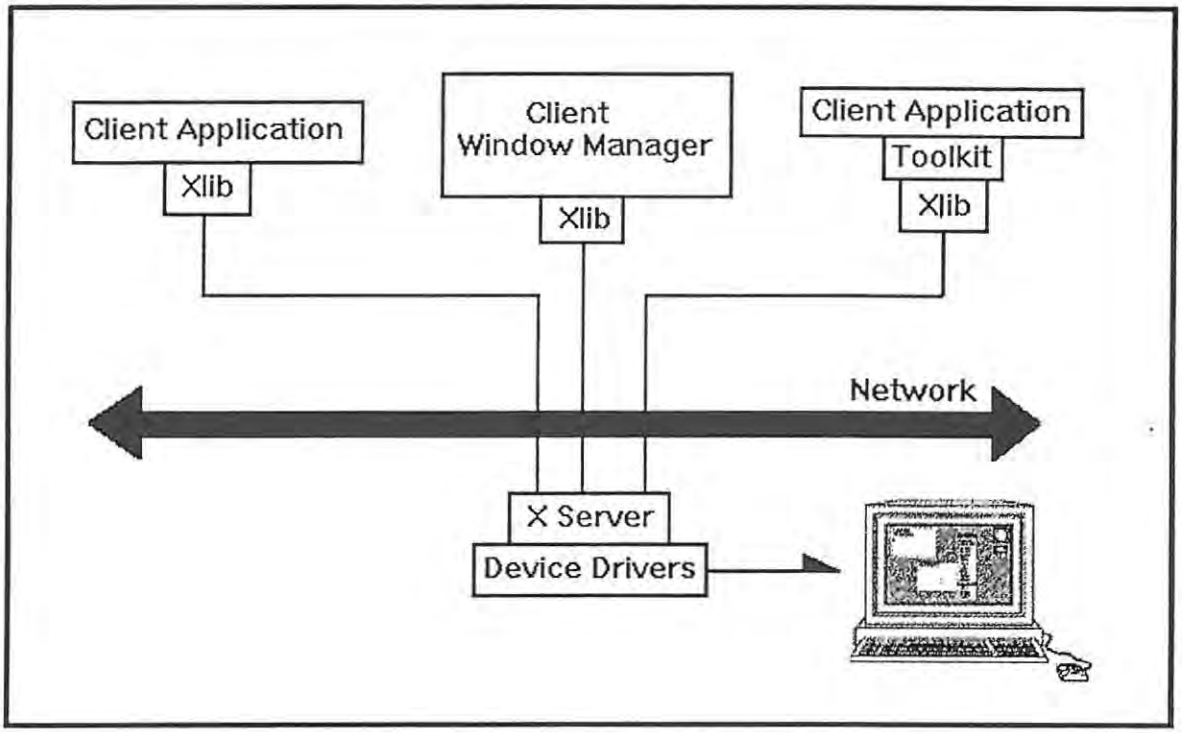


Fig. 2a. The Client–Server path via Xlib (Nye, 1992).

tags. **Fig. 2a.** illustrates client communication with the server via Xlib.

2.1.3. Xt Intrinsics and the OSF/Motif.

Xt Intrinsics or X Toolkit Intrinsics is a high level C subroutine library developed to make the writing of complex GUI programs easier (Nye & O'Reilly, 1990). Xt Intrinsics does not have a specific "look and feel", that is, the general style of the graphics interface on screen is not universal in appearance for different clients utilizing the Xt toolkit. It provides the foundation structures to commonly used user–interface elements, called *widgets*. More formally, a widget is defined as a reusable, configurable piece of code that operates independently of the application except through prearranged interactions. Examples of widgets frequently used are menus, dialog boxes, scrollbars, and command buttons.

A ready–to–use widget can be constructed by an application programmer, or developed by companies, and is compiled in libraries of ready–made widgets. These libraries are called *Third Party Toolkits* and cover most instances of user interaction with an ap-

plication program. Motif, developed by the Open Software Foundation[2] (OSF), is such a toolkit.

Motif provides the application programmer with commonly used user–interface components tied together with a consistent appearance. This means that applications utilizing the Xt Toolkit and OSF/Motif have the same look and feel, making it easier for a user to learn new applications.

2.1.3.1 Motif Widget Types. The widgets available to the programmer consist of two basic types, primitive and manager. A primitive widget is designed to work as a single entity. It provides active interface objects such as,

(1) Pushbuttons – These widgets supply the user with a square button, on screen, that is connected to a certain predefined process, executed when the user points to it with a pointing device.[3] There are different variations to the pushbutton, eg. DrawnButton, CascadeButton and ToggleButton.

(2) Scrollbar – Designed to enable the user to scroll through large amounts of data. A fraction is visible on the screen in a display window at any one time. The thumb position indicates the position of the visible data in a larger buffer. By either dragging the thumb or pointing in the scrollbar, the user may change the position of the visible data.

(3) Textfield – Allowing the user to input data to the application. Examples of such data entry widgets are, TextField and Text.

Manager widgets are designed to be containers and will have primitive widgets placed under their control. These widgets are less visible to the user. Primitive widgets placed under the control of a manager widget are referred to as child widgets. The managing widget (parent) performs important functions such as ensuring the correct placement of children when a window is resized. Important manager widgets are,

(1) Frame – Creates the frame outline for every displayed window.

(2) Mainwindow – A typical top level container for an application.

(3) DrawingArea – Provides an area for displaying graphics.

---

[2] 11 Cambridge Center, Cambridge MA 01754.

[3] This entails positioning the arrow on the button and, for example, pressing the left mouse button down.

(4) RowColumn – Used for neatly aligning identical primitive widgets.

(5) BulletinBoard – Which is further divided into three important sub–classes. The Form type (for better control over widget geometry), the MessageBox and SelectionBox dialogs for prompting user input.

There is a third type of widget, called a shell widget, which is not visible to the user. When the Motif toolkit is first initialized, a shell widget is created which contains all other widgets in the application. The main task of this top level widget is to interact with the window manager client. The top level shell is not visible to the user as it is always overlaid by a manager widget of the same size.

The other shell widget types are, the Override shell used in pop–up menus that must be at top–level, and Transient shells which are used for dialogs.

2.1.3.2. Widget Classes.

To understand the origin of widgets from a C language perspective, knowledge of the object–oriented programming (OOP) style is required. OOP is formally defined as a type of programming that provides a way of modularizing programs by establishing partitioned memory areas for both data and procedures, which can be used as templates for spawning copies of such modules on demand (Tello, 1991).

In OOP a class (object) is defined as a collection of functions (formally called methods) and data (supplied to the class), arranged in such a way that the functions act on the data. The data members can either be private, in which case only class member functions may access them, or public, where the data members can be set from outside the class.

OOP also provides a way of storing data and executable routines in specifically allocated memory areas, called code *encapsulation* (Smith, 1991), which simplifies the writing of application code for complex systems. This physical separation ensures the class independence from outside influence. In practical terms this means that as long as the programmer knows the input and output parameters of a class, the programmer is insulated from the internal operations in a class.

Objects can receive input or communicate with each other by receiving or passing messages. They can only receive messages directed at them and they have no access to anything else. This feature of objects excludes the possibility of external code interfering

with the functionality of objects.

The class structure in OOP supports the *inheritance* (Smith, 1991) property. When a class is derived from another class, the derived class inherits all the characteristics of the original. This means that a class A derived from B will have all the functions and data structures occurring in B, plus its own new data and function members. The derived class A is called a *subclass* of B, and B is called the *superclass* of A. In a large system that contains a large amount of subclasses the class hierarchy can be represented by a network tree showing the uppermost superclass to the specialized bottom class.

Finally, in connection with OOP the concept of *overloading* (Schildt, 1987) is mentioned. In most programming languages, functions that perform the same operation on different data types, must be named differently. The term *overloading* describes the use of different classes for different data types, with each class containing the appropriate member function to deal with the class's specific data type. This means that similar functions operating on different data types can carry the same name–tag. Overloading is then just a way of using the OOP environment for the simplification of writing software.

The Xt library retains widgets as C class structures. The data objects in a class are represented by specific values related to a widget's appearance or response, and these may be set at specific times. The function objects are either sets of functions that operate in a predetermined manner, usually in response to Xt function calls made by the application, or the widgets' internal functions that, for example, manage the widgets' appearance on screen. Since the C language does not include any special support for OOP, X relies on programming style to maintain a semblance of class structure.

Third party toolkits, like Motif, consist of a library of subclasses derived from the Xt widget class set. This means that widgets used by the application programmer share characteristics which have been inherited from a common basic superclass. **Fig. 2b.** illustrates the class hierarchy for the Xt Intrinsics and OSF/Motif toolkits.

When a widget is created, an *instance* of the widget class is created. This is similar to the creation of an instance of an integer, "a", in C as follows,

```
int a;
```

In other words, `a` is of the type integer. An instance of the pushbutton class is

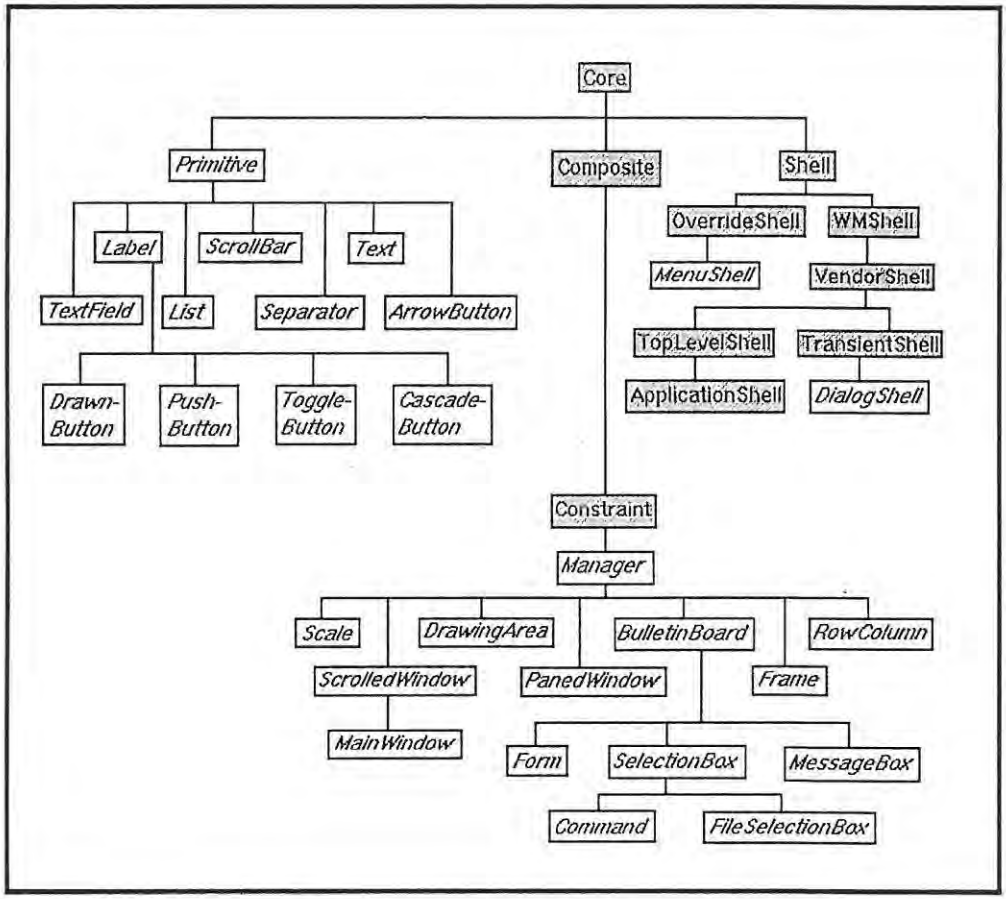


Fig. 2b. The Xt Intrinsics and OSF/Motif Class Hierarchy.
The Xt classes are represented by shaded boxes
(Nye & O'Reilly, 1990).

created, for example, when we execute the Motif command ***XmCreatePushbutton()***. Two instances of the same widget may have different names, but they share the same characteristics of operation.

2.1.3.3. Event-driven Programming.

The normal top–down approach to programming cannot be applied to the writing of windowing systems. Since different applications may be running at the same time, a program is required to respond to *events* rather than follow a set path of execution. Events may be in the form of user actions, such as mouse clicks, or keyboard input, or events may be triggered by the application itself as a side–effect of an executed routine. The event–driven programming style supplies the necessary structure. A program written

in this style initializes windows, registers responses to specific events, and then enters a loop, waiting for an event that will require action from the application. In other words, the application continually checks what the user wants and performs the appropriate tasks to fulfill those requests.

Generally, the structure of an event–driven program can be divided into three parts. Firstly, there is an initialization process which sets up the window system for user interaction. This includes procedures such as the opening of a connection to the display, font and colour setup, top–level window creation, and the display of this top–level window. Then the program structure enters an event loop implemented as a `while` loop. The program reads, interprets and processes events, using the Xlib routine ***XNextEvent*** to obtain event reports from the server. The event loop is usually terminated by the change in status of a variable in response to user action (for example, activating the "Quit" button). Finally, a clean–up occurs, freeing memory space allocated to window resources and resetting X server parameters that may have been set during the execution of the program.

2.1.3.4. Callbacks.

To enable widgets to be useful, graphical artifacts execute custom routines at certain times. The way in which a widget responds to a particular event is governed by a *translation table*. Every widget has a translation table supplying it with events and corresponding reactions. By way of example, the translation table for a pushbutton, "BSelect Press", corresponds to the left mouse–button being pressed down, and the action associated with this event is the Arm() function which causes the display of the button to appear pressed.

The Arm() function is an example of a predefined *callback* function. Custom functions may be added to widgets. For example, to register a certain function routine to a pushbutton the ***XtAddCallback()*** command is used.

2.1.3.5 Widget Resources.

Every widget has resources such as font, dimensions and colour that can be set by an application itself, or, subsequently, by a user.

A widget inherits resource values from a higher widget class on creation. These re-

sources have default values to lessen the programmer workload. Otherwise, for every widget instance, the programmer would have to set over 30 resources. To reset some of these from within the program, the Xt commands *XtSetArg()* and *XtSetValues()* could be utilized. For example, to reset the width and height of a pushbutton the Xt commands are used as follows,

```
Arg args[10];
int n = 0;

XtSetArg( args[n], XmNheight, 500); n++;
XtSetValues( args[n], XmNwidth, 750); n++;

XtSetValues( button, args, n);
```

The above code resets the resources of the PushButton class instance, button. This type of dynamic resetting may be done at any stage during application execution, provided the button instance has been created.

An alternative is the configuring of resources within the program when the pushbutton is created. Again, an argument list is set with the Xt command *XtSetArg()* and then these arguments are specified in the Motif function *XmCreatePushButton()*.

```
Arg args[10];
int n = 0;

XtSetArg( args[n], XmNheight, 500); n++;
XtSetValues( args[n], XmNwidth, 750); n++;

XmCreatePushButton( TopLevel, "Push_me", args, n);
```

Or the Xt function *XtVaCreateManagedWidget()* could simply be used to set the resources and create a managed widget in one function call,

```
button = XtVaCreateManagedWidget( "Push_me",
                                  XmPushButtonWidgetClass,
                                  TopLevel,
                                  XmNwidth, 750,
                                  XmNheight, 500,
                                  NULL);
```

Finally, there is also a way of setting resources from outside the application. When Motif initializes the application it looks for a special file in the current directory, named .Xdefaults. This file is a plain ASCII file that contains commands to reset resources in the following manner,

```
[application_name_path].resource_name:  value.
```

So, for example, the entries for the .Xdefaults file would be,

```
push.button.width:  750
push.button.height:  500
```

where "push" is the name of the application. Note that the prefix XmN– has been dropped from the resource name (Barkakati, 1991).

2.2. The TNTmips Operating System

2.2.1. TNTmips : Introduction.

TNTmips by MicroImages[4] is a platform–independent X windowing system designed for editing, displaying and filtering of digital imagery on the one hand, and as a Geographical Information System (GIS) on the other.

MicroImages' first entry into image processing software was with a product called HOTLIPS (Home and Office Techniques for Local Image Processing Systems) and was written in fortran operating on a Z–80. The next step in the development was MIPS (Map and Image Processing System) written in C for IBM compatible machines. MIPS made use of a dual monitor system, where one colour monitor handled the graphics output, and the other the user and text interface.

Finally MicroImages decided to port MIPS to the Unix environment for speed and portability, and rewrote it in the X Window environment. This move went hand–in–hand with a name change to TNTmips, TNT being an abbreviation for "the new thing".

Since TNTmips is an X Window client it can be installed on any system that can run an X server. Up to date MicroImages supplies executables to the following platforms; MicroSoft Windows, MicroSoft Windows NT, Macintosh, SunOS 4.X, HP (HP/UX), Dec Alpha, SGI (Irix), and IBM RS/6000 (AIX). For the MicroSoft Window platforms, MicroImages supplies their own proprietary X server (MI/X) that totally insulates the user from the underlying MicroSoft operating system. The X Window base for TNTmips ensures portability of the program and the identical appearance of the graphics interface across various platforms.

[4] MicroImages, Inc. 201 North 8th Street, Lincoln, Nebraska 68508-1347 USA.

The hardware setup used by the Rhodes University Geology Department is TNTmips V5.00 on a 486DX Intel machine running at 33MHz, with a 21inch NEC colour monitor driven by a Matrox Graphics Accelerator. The MicroImages X server is installed on MS Windows V3.1.

2.2.2. TNTmips : Functionality.

TNTmips provides an easy–to–use windows interface with menu–driven routines for most image processing and geographical manipulation needs. It facilitates the creation of the standard GIS/Image Processing image objects, such as rasters, vectors and computer aided design (CAD) objects from various sources, in the RVC (raster–vector–CAD) file format. The RVC file format is particular to the TNTmips operating system.

In the following section some of the main routines present in the TNTmips operating system are discussed.

2.2.2.1. Image Processing.

Imagery may be imported into the TNTmips system through the scanning of hard-copy originals or the importation of existing digital imagery from other file formats such as GIF, LandSat TM, and the SPOT CCT format. The Rhodes University Geology Department has a Howtek 3+(A3) scanner at its disposal. After initial importation the images are in a raster format. Several processes may then be applied to a raster image;

(1) Enhancement of the image involving the transformation of the greyvalue histogram to a desired distribution.

(2) Filtering of the raster by user–defined matrices or by predefined filters, such as a Sobel filter.

(3) Manual editing of the raster cell values using the TNTmips raster editing routine.

(4) Classification (manual or automatic) of rasters.

For multiple raster processing the following processses are available;

(1) Raster logic and mathematics, such as addition and subtraction.

(2) Colour composite creation from separate RGB colour components.

(3) Progressive transformation.

2.2.2.2. GIS Tools.

This selection of tools operates on all of the image object types and contains modules for transforming one into the other, that is, rasters to vectors, CAD objects to vectors and vice versa. The following GIS techniques, amongst others, are available for implementation;

(1) Georeferencing of objects in a specified projection system.
(2) Stereoscopic modelling of image stereopairs for the production of DTMs/OrthoImages.
(3) Database attachment to CAD and vector elements, as well as querying options.
(4) A multi–layer display function for viewing a series of superimposed objects or data covers.
(5) Editing facilities for all types of objects (rasters, vectors and CAD).
(6) A Software Manipulation Language (SML) toolkit that gives additional functionality to the system.
(7) Vector object merging by mosaic, overlay and intersection methods.
(8) Hardcopy output of all imagery.

2.2.3. TNTmips : Software Development Kit from MicroImages.

The SDK is purchased as an optional extra to the operating package. It provides the opportunity for an experienced programmer to develop new custom processes that may be utilized as an add–on feature to the existing system, or as a stand–alone routine (Ghormley, 1993a).

MicroImages also provides a macro utility, called the Spatial Manipulation Language (SML), that can be used to solve problems peculiar to a specific project. SML, however, is limited in usefulness and does not provide full flexibility in development, as does the SDK.

The SDK consists of platform specific header files that contain function prototypes, and platform specific run–time library files. These form the building blocks for the TNTmips operating system. This setup provides the application programmer with immediate access to the RVC file format, so that objects created and manipulated by newly developed processes are compatible with TNTmips. Indirectly, this also means that process

source code is readily portable to other platforms that can run TNTmips.

The SDK is oriented towards experienced professional programmers and assumes prior skills in the C language, OOP, and the X Window environment.

2.3. The SDK Run–Time Libraries

The SDK is supplied as a set of include files (*.h), documentation files (*.txt), and library files (*.lib) which must be installed in compiler accessible directories before the building of new applications can begin. In the next sections the library files pertaining to the development of the structural analysis routine are listed, with examples from each library given.

2.3.1. The MIPSLIB Library.

This library contains general utility tools for manipulating the TNTmips system. The collection varies widely from complicated macro routines to simple function routines. Examples;

(1) ***TransCADElemGen2D*** - This macro translates CAD elements through a generic 2D transformation.

(2) ***uctostr*** - The function for the conversion of a `UNICODE`[5] type array to a `char` type array.

2.3.2. The MIXLIB Library.

This library includes all functions with the Mx- prefix and is known as the X Function Library. It provides the programmer with all the necessary functions for user interaction, with the Motif interface, with any TNTmips oriented routine. In addition it provides special groups for displaying objects and using graphics tools. A brief description of the main groupings will be presented in the following subsections.

2.3.2.1. Prompt Field Widgets.

These widgets are used in the creation of input fields that accept a specified data type in a display window. For example, ***MxCreatePromptFloat*** creates a form[6] containing a label and a text widget prompting for a `float` value.

[5] UNICODE is a defined data type in the TNTmips source.

[6] A container widget.

2.3.2.2. Standard Pop–up Dialogs.

These consist of functions for the creation of transient pop–up windows while the application is executing. Pop–up windows can prompt the user for an input value, or display a text message. An example of such a window is ***MxPopupMessageMt***, which displays a message from the messages.txt ASCII file resident in the TNT directory. The specific message is identified according to a specific class and key.

2.3.2.3. Menubars and Option Menus.

This library contains functions that provide fast realization (display) of menus: An example is ***MxCreateMenuBar***, which displays text items on a bar in the display window. Each text item has a pull–down menu attached to it, which becomes active when pointed to.

2.3.2.4. Display Window Toolkit.

All functions in this group start with the prefix Mxdw- and support the display and manipulation of raster, vector and CAD objects. This toolkit also provides the necessary utilities for communication with the TNTmips display window tools, such as the crosshair tool. An example is the function ***MxdwCreateLayerRaster*** which creates the necessary structure for the display of a selected raster object.

2.3.2.5. Tool Protocols.

This group of functions handles the operation of the tools supplied with the TNTmips display routines. All functions in this group have the Mxt- prefix. An example is ***MxtCreateCrossHair*** which adds a crosshair option to the existing tool option in the display window menubar.

Finally, the MIXLIB library also contains miscellaneous functions, such as ***MxInit*** which initializes the toolkit.

2.3.3. The RVCLIB Library.

This library contains all the RVC–related functions used to manipulate raster, vector and CAD objects in TNTmips. The RVC file format is specific to the TNTmips operating system, and use of the functions resident in this library are critical for proper access to these objects. All functions in this library have the prefix Mf-. In the following sections the main groups of these functions are described and an example from each group is

listed. In general, the different groups consist of ways to open and close RVC objects, to read and to write information from them, to retrieve sub–objects related to them, and to access database attributes attached to RVC objects.

2.3.3.1. CAD Objects.

An example of such a function is ***MfSetCADMode***, which sets a point data conversion mode. This function is used when reading data from a CAD object in order to ensure a specific data type as output.

2.3.3.2. Raster Objects.

An example is ***MfSetDftNullVal***, which sets the default null value for a raster object to a specific value. The null value is the value that will be assigned to "empty" raster cells.

2.3.3.3. Vector Objects.

An example is ***MfGetVect***, which retrieves a vector object through a pop–up dialog box, prompting the user to a select a particular vector.

2.3.3.4. Database Objects.

These objects contain all the necessary functions for maintaining a database attached to a vector or CAD object. An example is ***MfDBListItemSize***, which returns the number of entries in the list of records attached to an element.

2.3.3.5. Georeference Subobjects.

An example is ***MfReadGeorefMatrix***, which retrieves the relevant 3x3 matrix from a georeference sub–object, describing the transformation between object coordinates and geographical coordinates.

2.3.4. Miscellaneous Libraries.

The run–time libraries also include;

(1) OFont routines that provide the TNTmips system with the capability to write text to a video output device. For instance, when a vector object contains text labels and the vector is either displayed or edited the OFont routines supply the screen font.

(2) Port routines enable the system to receive input from scanners and digitizers and to send output to printers and plotters.

2.4. The Watcom C\C++ Development System

The Watcom Integrated Development System[7] is a professional, multi-platform C and C++ compiler with a comprehensive suite of tools for developing and debugging 16/32-bit applications for DOS, Novell NLMs (Netware Loadable Modules), OS/2, MS Windows, and Windows NT (Watcom, 1994).

The development of a customized program for TNTmips using the SDK requires the use of the Watcom compiler, linker and bind utility. MicroImages advises programmers to use the Watcom system to produce TNTmips compatible software.

2.4.1. The Watcom Compiler.

The process of creating an executable can be divided into three definite steps. The first step is the compilation of the program. This involves the use of the Watcom C compiler which consists of a preprocessor, a main compiler and an assembler. Each of these perform a certain task in the compilation process.

The preprocessor receives the source code in ASCII format and performs the following tasks;

(1) Removes comments in the source code. Comments are demarcated with /* */ and // characters.

(2) Interprets special preprocessor directive commands in the source code that are preceded by ♯, such as ♯include<stdio.h> which will include the header file, stdio.h, in the compilation process.

The C compiler then translates the source code into assembly code. Finally, the assembler receives the assembly code and produces object code, in the form of an .obj (object) file when working in MSDOS. Object code is the language understood by the operating system, whereas assembly code is architecture specific to, and not dependent on, the operating system.

[7]WATCOM International Corporation, 415 Phillip Street, Waterloo, Ontario, Canada N2L 3X2.

2.4.2. The Watcom Linker.

The Watcom linker is a linkage editor that takes program object and library files as input and produces executable files as output. In this case, for the development of a 32–bit MS Windows application (remember that the MicroImages X Windows Server is resident in MS Windows), the inputs are the run–time libraries supplied by MicroImages, as described in section 2.3., and the object file produced by the compilation process, as discussed in the previous section.

The output is an executable file with a .rex extension. To produce a windows executable, the resultant .rex file is combined with the Watcom Windows–Extender to produce an MS Windows executable. For this final step the Watcom Bind Utility is used.

Various command line options are needed to make an .exe (executable) file compatible with the TNTmips system. These options are combined into a .lnk (link) file, which is accessed the moment the linker is activated by typing ***wlink***. See Appendix A for the options used to link the custom program created in this project.

CHAPTER THREE

GEOSTRUC V3.13 DESIGN AND IMPLEMENTATION

In this chapter the design concepts for the GEOSTRUC©[1] analysis routine are described. Appendix B contains a complete source code listing for the program. The implementation and usage of the developed routine is also discussed.

3.1. Mathematical Preliminaries

Determination of the orientation of a plane in R^3 is fundamental in the operation of the GEOSTRUC© analysis routine. The general equation of a plane in R^3 is given by,

$$ax + by + cz = 1 \tag{1}$$

where the coefficients (a,b,c) of the equation are the components of a vector that is normal to the plane at all points on the plane.

The cross–product between any two vectors in R^3 produces a normal vector that is perpendicular to both vectors.

Definition 6. *(Lang, 1983a) Given two arbitrary vectors* (A_x, A_y, A_z), $(B_x, B_y, B_z) \in R^3$, *the result* (C_x, C_y, C_z) *of the cross–product between the two vectors is defined by,*

$$\begin{aligned} C_x &= A_y B_z - A_z B_y \\ C_y &= A_z B_x - A_x B_z \\ C_z &= A_x B_y - A_y B_x \end{aligned} \tag{2}$$

If the two vectors are contained within the plane, the cross–product between them will produce a normal vector to the plane. The orientation of the plane may be determined

[1] The name given to the custom routine.

directly from the orientation of the normal vector. Furthermore, the plane equation (1) can now immediately be written down. What we have actually described in this paragraph is the "fitting" of a plane to three points in R^3 and the subsequent determination of the orientation of that plane.

Note that the method outlined above is not the only one for fitting a unique plane to three points in R^3. Substituting the three points in R^3 into equation (1), one at a time, results in three distinct linear equations. These equations can now be solved simultaneously by employing either Gauss Reduction (Burden & Faires, 1988) or Cramer's Rule (Lang, 1983b).

The cross–product method was chosen for the GEOSTRUC© routine as it minimizes computation time and complexity.

3.2. Design Objectives

The aim of the GEOSTRUC© routine is to convert theoretical concepts, as outlined above, into a practical, user–friendly, interactive process that gives the operator complete control over each step. An approach to system design favoured by Ward and Mellor (1985) was used. They proposed the use of prototyping in the design of systems which have a significant interaction with users. This means that the system designer constructs a basic user interface system with no working functions connected to it. The user is then given the chance to operate this prototype and make suggestions on design modifications. This "interleaved" designing and modification process, together with the addition of functions, enables the basic prototype to evolve into the final product.

An operating environment, namely X Windows, and a GIS/image processing system, namely TNTmips, have already been decided upon. The process is to be developed, using the TNTmips SDK, so that the user can select, by means of a pointing device, three points on the trace of a geological structure, on a georeferenced digital geological image. The geographical location of these points, that is, latitude and longitude (x,y), is now used to extract the true elevation (z) of the points from an underlying, co–referenced DEM of the same area. Once these xyz–coordinates of the three points are known, the plane calculation may be performed. The process must also be able to determine the thickness of a geological bed. Data gathered in this manner then needs to be saved

in formats accessible to other routines in TNTmips, external databases and external programs, such as structural mapping systems and stereonet programs.

3.3. Program Design

To translate these design objectives into program code an events list was established as follows;

01. Select Raster and DEM objects (co–georeferenced) for analysis in the same latitude/longitude (lat/long) system.
02. Select Vector and ASCII objects for data output.
03. Display Raster Object.
04. Display DEM extents on raster image.
05. Select point on raster and obtain xy–coordinates.
06. Access DEM for elevation coordinate (z) at selected xy–coordinates.
07. Repeat events 05–>06 until three points are located on the raster.
08. Calculate the orientation of a plane defined by the three points.
09. If thickness calculation is required then do events 10–>11.
10. Repeat events 05–>06 until two points are located on the raster bounding a geological bed.
11. Calculate the thickness of the feature outlined by points selected in 09.
12. Accept thickness for saving to output objects.
13. Write orientation, location, and optional thickness to output objects.
14. Repeat events 05–>12 until finished.

The next step was to compile a structure diagram (**Fig. 3a.**) which translates the events list into a structure compatible with the X Windows event–driven programming style.

Finally, the code was written, firstly as a graphics prototype, and then as a fully fledged analysis routine.

3.4. Code Description

A full code listing is available in Appendix B and the line numbers given in the following discussion refer to line numbers of the code in this Appendix. The code may be divided roughly into three parts;

(1) The first section of code defines which header files are included, defines custom structures, and specifies global variables.

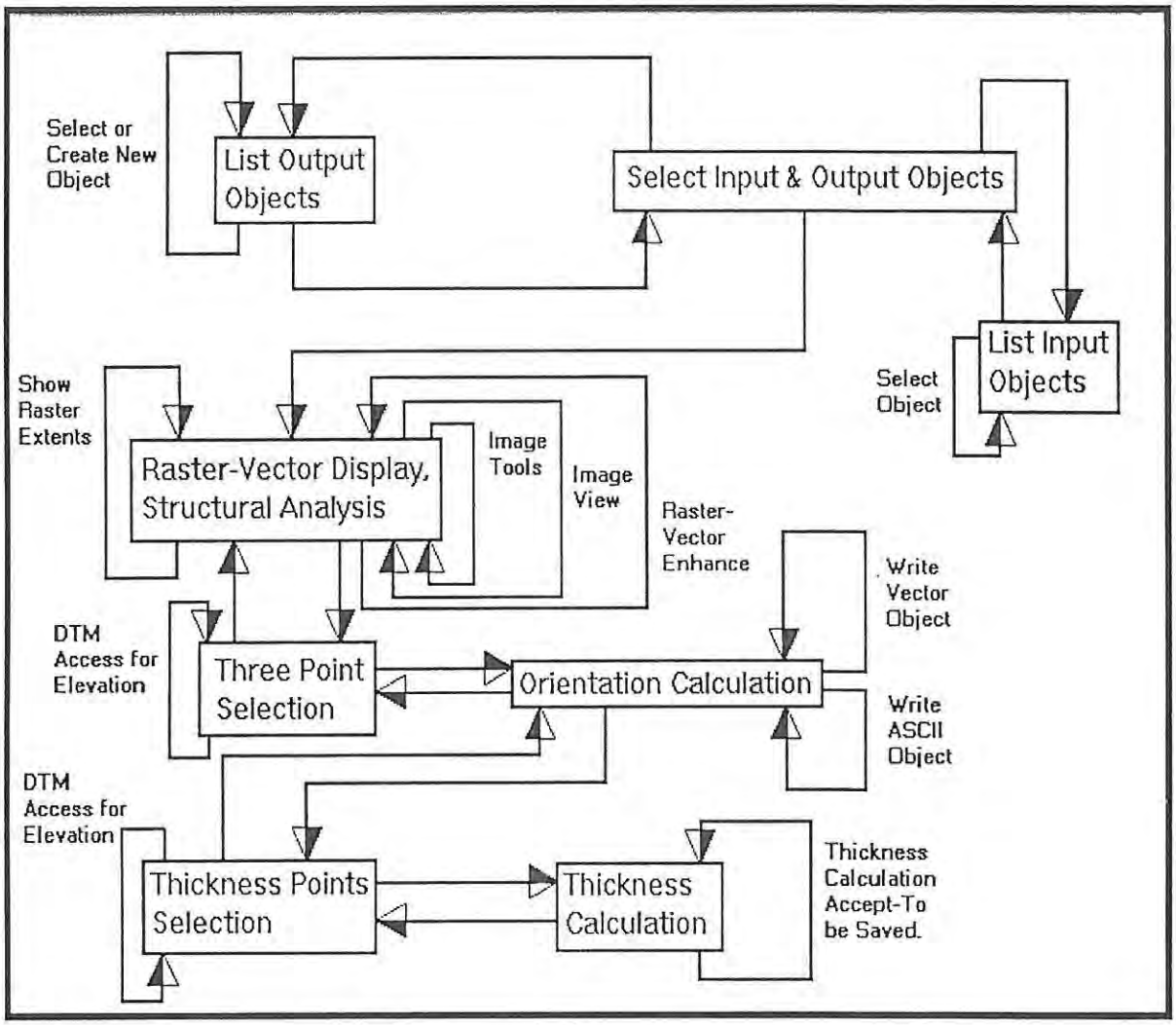


Fig. 3a. The structure diagram for the analysis routine. Boxes represent user–interfaces and closed flows represent available user actions.

(2) The larger body of code consists of callback functions that perform all the necessary operations for the analysis.

(3) The last section is the C language `main()` function, containing the program control loop, which is the initialization point for the process.

To aid a better understanding of the logical flow of the program, the code is discussed beginning at the highest line number and ending at the lowest line number. That is, the three sections, as outlined above, appear in reverse order. In Appendix B, however, the code is listed in ascending line numbers as required by the C programming convention.

3.4.1. Code Section One.

In this section the main function, which is characteristic of all C programs, is described. The function is the root of all callback functions.

The main function *(Appendix B:1636-1716)* contains routines for the graphics window that manages the input of the four objects required for the analysis routine (**Fig. 3b.**).

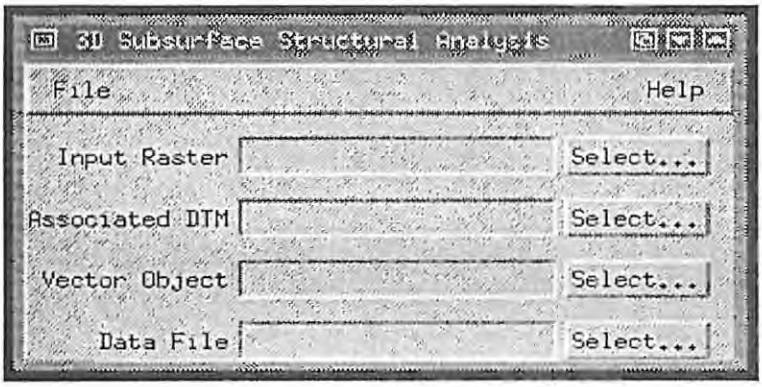


Fig. 3b. The Input window.

It provides the starting point for the interactive process, initialized via pull-down menus.

3.4.2. Code Section Two.

This section contains all the callback functions, which are described below. They are defined before the `main()` function as part of the preferred X convention. Note that each function has parameters passed to it which are listed immediately after the function name. For this specific case the parameters usually include a pointer, `DDATA *ddp`, to the custom DDATA structure, and an X callback structure, usually denoted by `XmAnyCallbackStruct`.

The body of each function begins with locally declared variables that are used within the function operational sphere. These local variables are transient and do not exist after a function has been executed.

3.4.2.1. Object Selection Functions.

The analysis routine requires the user to pre-select four objects before the analysis can begin. This section describes the four necessary functions. Note that some of the code is devoted to ensuring that the user can only continue with the process if all the necessary objects have been selected *(Appendix B:1664)*.

The first function in this section is the ASCII output file selection window which prompts the user to select an output data file that will contain the results of the analysis in comma–delimited format (Section 3.4.2.7.) *(Appendix B:1615–1634).* The ASCII data file can be a new one, or an existing one, in which case the file is appended.

The vector object is selected via the vector selection window *(Appendix B:1584–1613).* The user can either create a new vector object or re–open an existing one. If the user selects an existing vector, the user is prompted to select a georeference sub–object.

The DEM selection window prompts the user to enter the name of the overlapping DEM *(Appendix B:1556–1582).* The selection of a lat/long georeference subobject is also required.

The final input window is the raster selection window. The user is required to select a raster (the image of the area to be analyzed), after which an existing lat/long georeference subobject for that particular raster is selected *(Appendix B:1529–1554).*

Note that, as in all of the above selection functions, a new object is not created at this time, only the name is acquired.

3.4.2.2. Display Window Management Functions.

The following five functions control the display window for the raster image overlaid with a vector object overlay. The correct placement of these two objects is critical for the success of the procedure. An extensive amount of code is devoted to the precise placement of the two objects.

The raster and vector objects are displayed in a graphics window that forms the basis for the analysis routine. The window contains a menu–bar with pull–down items for tool selection (cross–hair, zoom–box), view selection (full–view, zoom–in/out, slide), object enhancement, and analysis. The other feature in this window is the display area for viewing the raster and vector objects *(Appendix B:1393–1514)* (**Fig. 3c.**).

Certain actions may be accessed from the main window menu, such as drawing the outline of the DEM overlap on the raster image. This is accomplished by writing point and line elements to the vector object. The point coordinates are obtained by transforming the DEM corner coordinates from file format (row/column) to geographical coordinates *(Appendix B:1213–1391).*

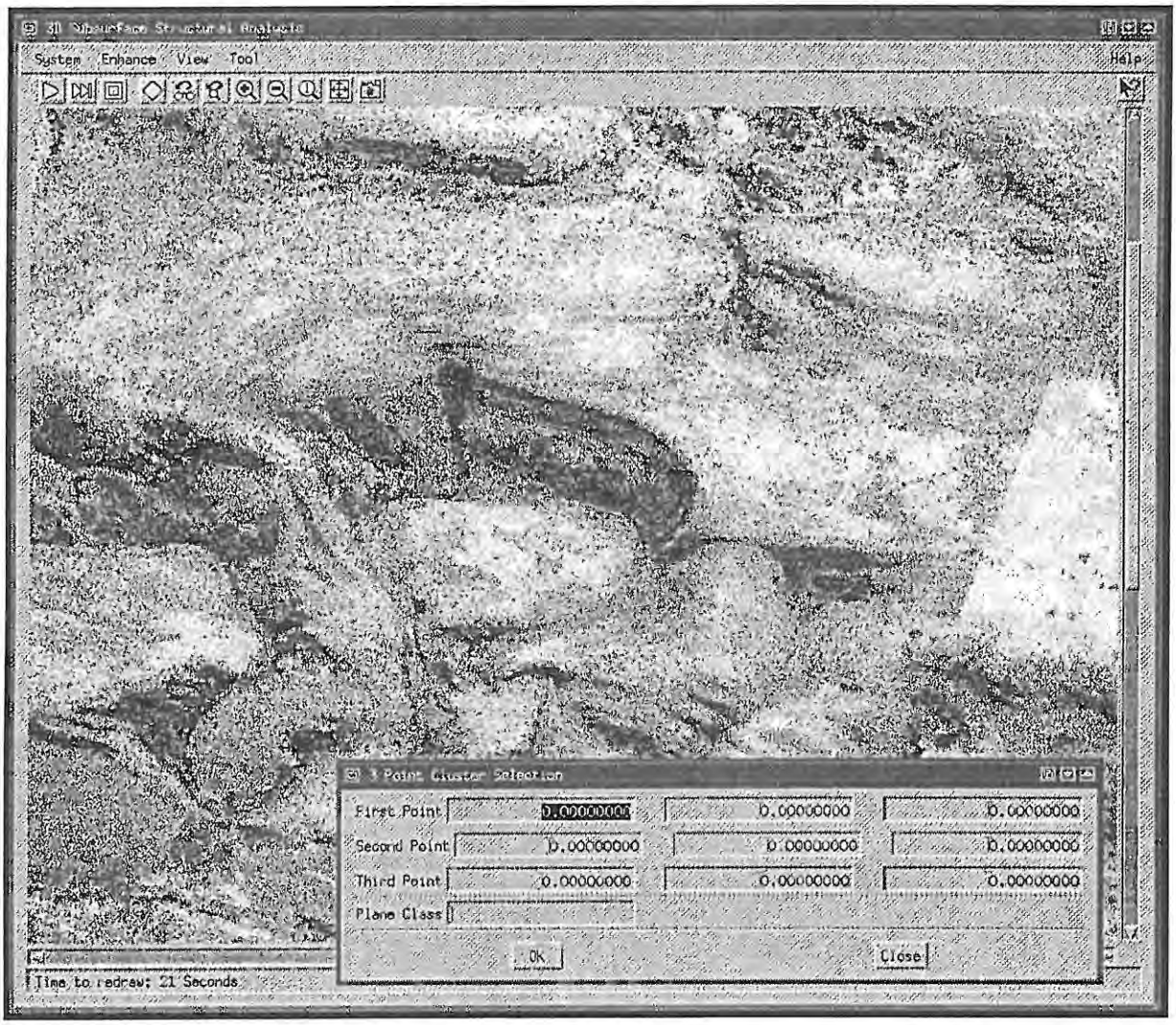


Fig. 3c. The object display window and 3–Point selection window.

Before the vector object can be displayed the program has to either create a new object, or initialize an existing one. Vector creation has to take place after the raster has been displayed so that the appropriate display characteristics can be added to the vector object header. If the vector object already exists the internal program variables for point numbers, line numbers and label numbers are updated *(Appendix B:1143–1211)*.

Functions that provide the user with menu options for enhancing the raster and vector objects in the display window exist *(Appendix B:1127–1133)*.

3.4.2.3. 3–Point Acquisition Functions.

This section describes the functions responsible for the acquisition of three point data clusters that are used in the plane–fitting procedure. The functions perform tasks such

as: pointing device data retrieval, accessing the DEM for the elevation coordinate (z), and driving the functions that fit a plane to the three points.

When coordinate data is generated by the pointing device, certain transformation processes are required before the data can be displayed in the prompt window. The data is transformed from screen coordinates to raster file coordinates (row/column) and then to geographical coordinates (lat/long). The function that executes the above transformations also distributes the information, in memory, to the three points *(Appendix B:1026-1125)*.

Once the lat/long coordinates have been acquired, a separate function is called upon to retrieve an elevation value for this specific geographical location. That is, given lat/long information, the function accesses the DEM and retrieves the cell value (elevation) at that specific lat/long. Clearly, the DEM needs to be georeferenced in the same lat/long system to enable the conversion from geographical coordinates to DEM file coordinates (row/column) *(Appendix B:0962-1024)*.

A window is opened in order to enable the user to select three points, by means of a pointing device, on the raster image of an area. The window has entry areas for the three sets of coordinates and a text entry area where the user can optionally enter the class of the geological feature being investigated *(Appendix B:0824-0960)*.

On exiting the analysis procedure, the program ensures that all open objects are closed *(Appendix B:0815-0822)*.

The plane-fitting procedure is initiated when the user activates the OK button *(Appendix B:0807-0813)*.

The three point selection window is destroyed when the user activates the close button *(Appendix B:0794-0805)*.

3.4.2.4. Plane Location Function.

This function is essential to the post-analysis process, in that it provides an external database or display routine with a geographical location at the centre of the 3-point triangle, thereby positioning the result of each plane orientation calculation in xyz space. This data could, therefore, be used, for example, to annotate a digital geological map with structure ticks.

The centroid of the 3–point data triangles is determined in the following manner: the centre of each triangle is located by determining the maximum distance between x–coordinates and the y–coordinates of the three vertices. These distances, divided in half, supply the xy–coordinates for an approximate centre of the triangle *(Appendix B:0656–0754)*.

3.4.2.5. Plane Orientation Calculation Functions.

The calculation of plane dip and dip direction is the central feature of the program. The five functions discussed in this section calculate the dip/dip direction in two stages and display the relevant information in a window.

The first step in determining the orientation of the selected plane is an affine transformation of the three selected point coordinates from geographical coordinates to local system coordinates. The first point of each 3–point cluster is mapped as the origin of a new, local system. The coordinates of the other two points are transformed to preserve distance *(Appendix B:0756–0792)* (**Fig. 3d.**). Note that the curvature of the earth is not taken into account, as the difference it introduces is negligible over the short distances used in 3–point plane solutions.

The two non–zero points in the local coordinate system represent vectors attached to the origin. By means of a cross–product calculation between two vectors in R^3, a normal vector is determined *(Appendix B:0595–0611)* (See section 3.1.). The normal vector, which is perpendicular to both original vectors and, hence, to the geological surface, is then normalized to length one. It is important to ensure that the normal vector has a positive z component, in other words, that it points "upwards" *(Appendix B:0607)*.

Next, the orientation of this normal vector to the plane is converted into trend and plunge of the dip line of the plane in standard angular parameters (bearing from North for trend and angle from horizontal for plunge) The dip is calculated by determining the angle between the plane normal and the positive z–axis (**Fig. 3e1.**). The dip direction is calculated by first establishing the quadrant in which the plane normal occurs, and then performing a trigonometric calculation *(Appendix B:0613–0654)* (**Fig. 3e2.**). Note that, depending on the quadrant, the trend is corrected in different ways.

The result of the orientation calculation is displayed in a text window *(Appendix*

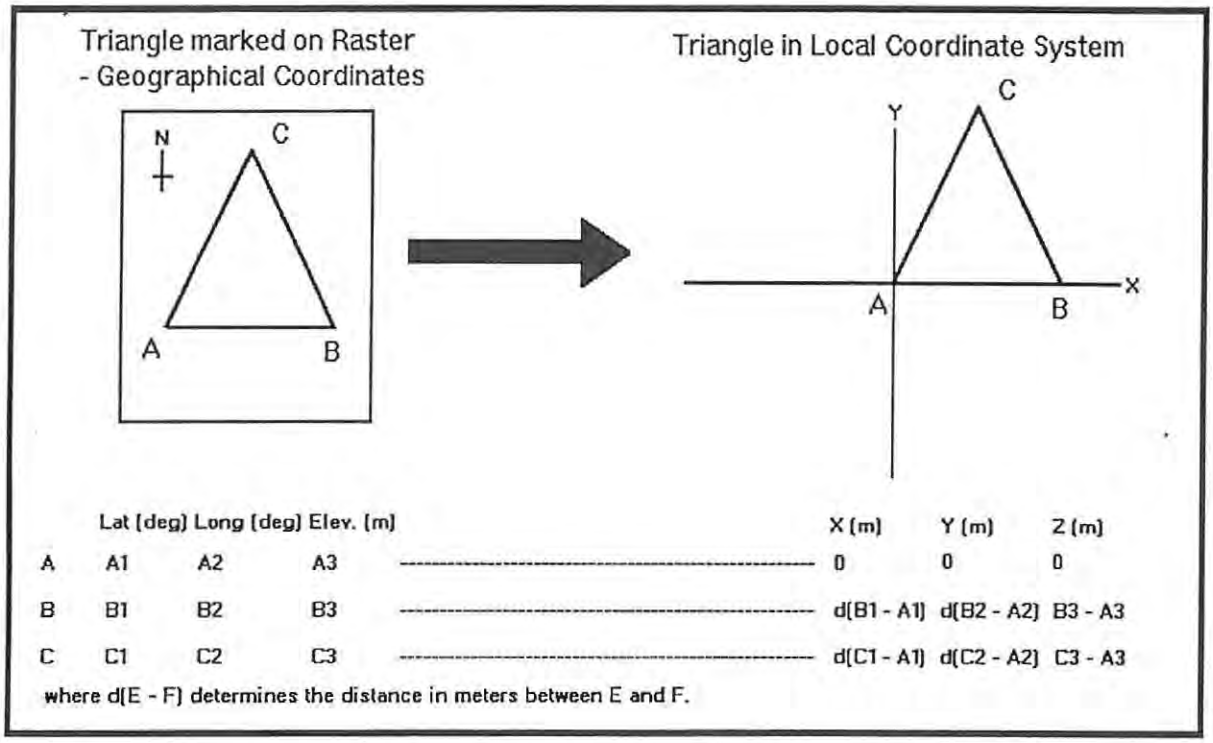


Fig. 3d. The coordinate transformation between map coordinates and a local cartesian coordinate system. The first point selected by the user is always designated A.

B:0508–0559) together with centroid coordinates of the 3–point triangle and a user selected structure class, for example bed, fault or dyke *(Appendix B:0560–0594)* (**Fig. 3f.**).

If an orientation measurement is cancelled the text window is destroyed *(Appendix B:0491–0506).*

3.4.2.6. Thickness Option Functions.

There are five functions assigned to creating an option for the measurement of bed thickness. They enable the user to select two points on either side of the surface trace of a bed and perform a thickness calculation on the given data.

A window opens when the user selects the thickness option which prompts the user to enter two points that bound the geological bed, for which the thickness calculation has to be performed *(Appendix B:0393–489).*

After the data entry, a new function transforms the point coordinates from geographical to local coordinates, as outlined in section 3.4.2.5., and calculates the true thickness

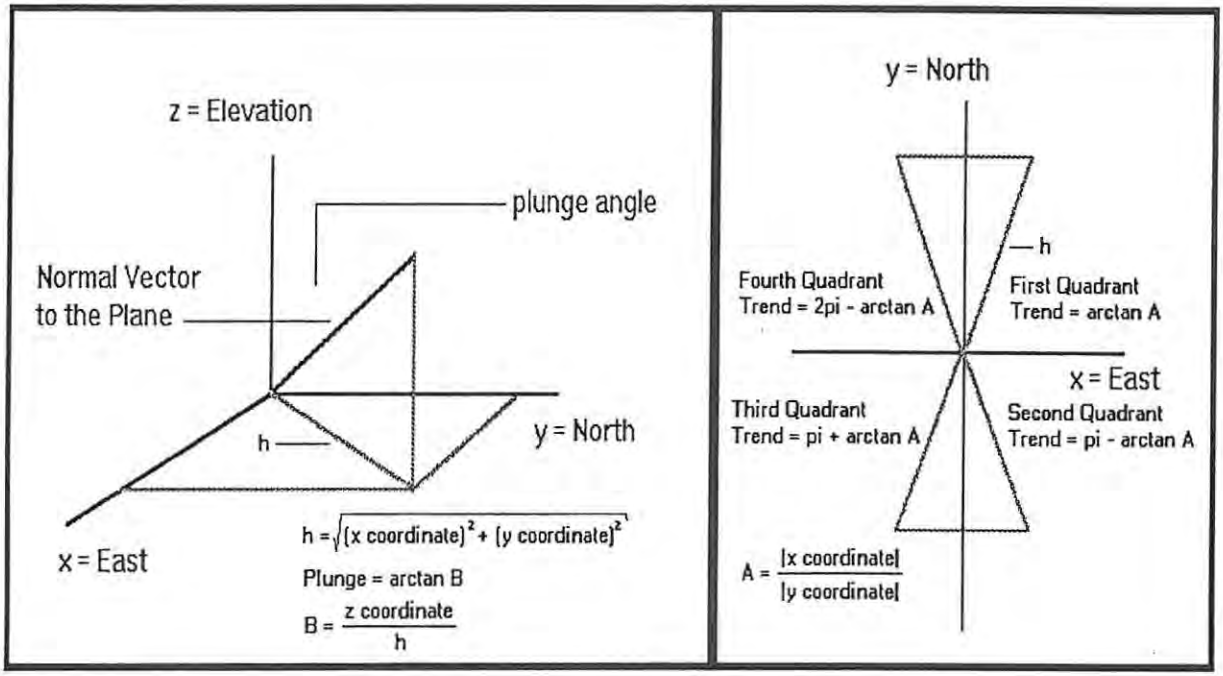


Fig. 3e1. and **Fig. 3e2.** The calculation of plane trend and plunge from the normal vector in the local coordinate system.

of the bed. The result of the calculation is displayed in a text window *(Appendix B:0283–0367)* (**Fig. 3f.**). This section of code includes Xt Intrinsics and Motif function calls in order to structure the graphics window.

The program checks if the toggle button for saving thickness value has been set *(Appendix B:0266–0273)*, in which case the thickness value will be written to the ASCII data file, and closes the thickness result display window *(Appendix B:0275–0281)*.

A function exists that closes the window opened for selecting thickness bounding points *(Appendix B:0381–0391)* upon completion of the thickness calculation.

3.4.2.7. Vector/ASCII Object Writing Function.

This function updates the ASCII file in a format outlined below, and writes three points, with an interconnecting line, to the vector object.

The function opens the ASCII data file and appends data to the file in a comma delimited format as, measurement number, x–coordinate for triangle centre, y–coordinate for triangle centre, elevation for triangle centre, dip direction, dip, geological structure classification, and thickness[2]. The comma delimited format simplifies the import of this

[2]If the thickness option is not activated, the value written will be zero.

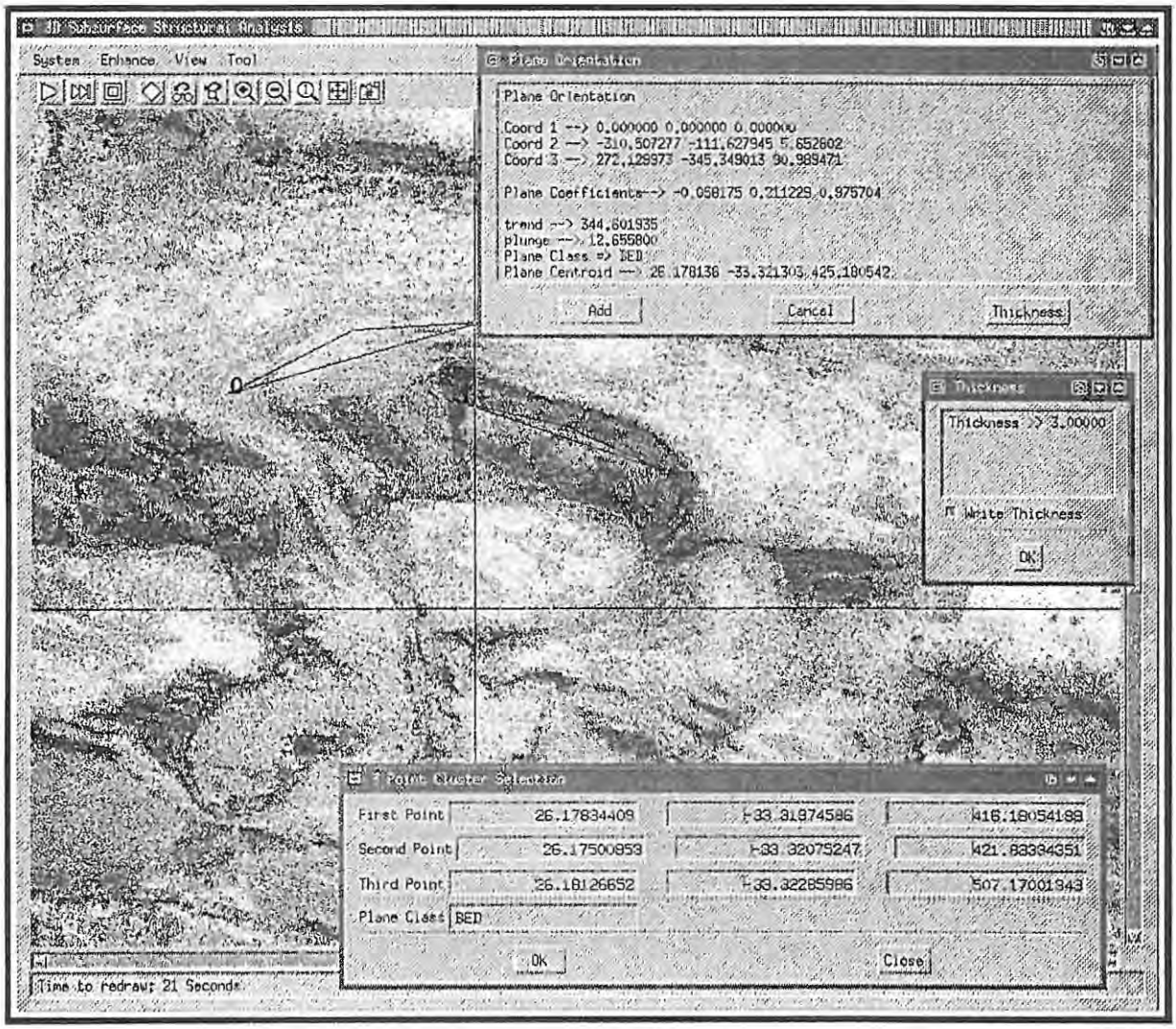


Fig. 3f. Selected geological structure with text window and thickness popup.

ASCII file into external database programs, such as dBaseIV *(Appendix B:0108-0132).*

The function then writes the three points selected by the user to the user selected vector object, using the geographical coordinates (lat/long) of the points. Further vector elements written to the vector object are;

(1) A triangle label. This label is the number of the 3–point triangle and is displayed on screen *(Appendix B:0221-0238).* Each 3–point calculation is marked on screen by a numbered triangle starting at 1 and numbered consecutively. This allows the user to check calculation results, as listed in the same order in the ASCII data file, locating potential errors and spurious results.

(2) A line to the vector object. This line interconnects the three selected points so

that a triangle appears on screen *(Appendix B:0239-0252)*.

Finally, the function updates the vector object header to accommodate the new minimum or maximum xy extents, if required, and then redisplays the vector object so that the triangle defined by the 3 points shows up on screen *(Appendix B:0253-0264)*.

3.4.3. Code Section Three.

The final listing in the source code includes the header files that contain all the TNTmips related functions for the manipulation of RVC objects *(Appendix B:0001-0006)*, the header files of the X Windows/Motif System *(Appendix B:0007-0012)*, and a standard C header file *(Appendix B:0013)*.

Also listed in this section is the declaration of the appcontext variable as a type XtAppContext variable *(Appendix B:0014)*.

The main vehicle for passing information between functions is the DDATA structure *(Appendix B:0015-0080)*.

3.5. Implementation of the GeoStruc Routine

3.5.1. Data Preparation.

The routine makes use of a remotely sensed image of the required locality, as well as a Digital Elevation Model (DEM) covering the same area. Both the image and the DEM must be in the RVC–raster format supported by TNTmips. Examples of input images are;

(1) SPOT Panchromatic images processed to level S.
(2) Digital aerial photographs (corrected for distortion).
(3) Orthophotos.
(4) Landsat TM images.

and examples of DEMs are;

(1) DEMs derived by stereo–correlation of a SPOT stereo–pair (Lemmens, 1988; Ghormley, 1993b).
(2) National Digital Elevation Model (NDEM) data purchased from Government Survey.

(3) DEMs derived from topographical sheets by digitizing contours, resampling and interpolation (see section 1.2.3.).

(4) DEMs derived from Radar Altimetry (Drury, 1987b).

(5) DEMs produced by photogrammetric methods.

TNTmips offers all the necessary routines for the construction of DEMs and the importing of images from source formats.[3] It is critically important that the image and the DEM overlap in the desired area of investigation. Furthermore, both rasters must be georeferenced in the Latitude/Longitude projection system. If the rasters are not initially in this system, then particular processes in TNTmips can be used to convert them into the required projection system.

It is recommended that the raster image and the DEM raster have the same cell dimensions, although this is not crucial. Similar cell dimensions will ensure greater accuracy in the cross–referencing of xy–coordinates between the image raster and the DEM raster.

In areas with low variation in relief the process might give inaccurate results. This is due to the fact that the three points selected will be almost in a straight line, thus effecting the plane–fitting procedure (An infinite number of planes can be fitted to three colinear points.).

3.5.2. Operational Summary[4].

The operation is initiated by selecting four objects, namely;

(1) A digital geological image (raster object, RVC format) and the attached georeference subobject (Lat/Long Projection).

(2) A digital terrain model (raster object, RVC format) and the attached georeference subobject (Lat/Long Projection).

(3) A transparent overlay (vector object, RVC format), either existing or new, and the attached georeference subobject (Lat/Long Projection) if a new object was created.

(4) A data file (text file, ASCII format), either existing or new.

[3] Not completely true as the NDEM test data used in this study had to be preprocessed before importing it into TNTmips. See Appendix C for a description of the C routine used in this preprocessing.

[4] A complete user manual is included in Appendix D

The digital geological map and the transparent overlay are displayed and the user may optionally display the extents of the DEM on the transparent overlay.

The user interactively selects three points on any geological structures of interest, using a pointing device. For each of these three points the latitude (in degrees), longitude (in degrees) and elevation (in meters) is known. Before any calculation is done the routine transforms the coordinates of the three points into a local coordinate system with an origin at one of the points, using an affine transformation. Each point is now represented in terms of its xyz–coordinates.

To determine the orientation of the plane, two vectors are first constructed with the selected points as endpoints. These two vectors will lie in the plane whose orientation is required. Taking the cross–product of these two vectors gives another vector perpendicular to both of the originals, representing the normal to the plane. The orientation of the normal vector is then used to determine the dip and dip direction of the plane. Note that error trapping has to be performed on the end result as the normal vector can have two directions, with the difference between the two being a scalar factor -1.

The user may now calculate the thickness of a geological bed, provided the orientation for that particular bed has been determined. The thickness determination process requires that the user select two points, with the pointing device, on the upper and lower contact of the bed in question. These two points, however, do not have to be aligned along any geographical direction. For better accuracy, however, they should be in the same vicinity of the region where the orientation of the bed was determined.

Finally, if the operator is satisfied with the analysis results, the orientation and thickness (if calculated) is saved to the data file and to the transparent vector overlay. The 3–point cluster is represented by a triangle on the transparent overlay and labelled with an identification number that references the result in the ASCII data file by line number.

The process is then reset and the orientation of a new structure may be determined.

CHAPTER FOUR

TWO APPLICATIONS OF THE GEOSTRUC PROGRAM

To assess the optimal combination of source material the GEOSTRUC© program was tested in two areas with the source material in the two instances derived from different origins. By design different ground resolutions for the different studies were selected to try and show the effect this factor has on the accuracy of the overall result.

4.1. Program Application One : Rectified aerial photograph on DEM derived from digitized topographic contours

4.1.1. Introduction.

This first case study involves a test area selected in the initial stages of GEOSTRUC© development. The geology of the region is relatively simple with clear bedding traces evident on aerial photographs and simple fold patterns. Fieldwork in the area was performed by Millad (1993) which meant that field measurements were available for comparison with remotely sensed data. The area selected was reasonably small, but provided a suitable test site for a specific combination of source material used in GEOSTRUC© , that is, a scanned aerial photograph for the digital geological image and a contoured orthophoto for DEM generation.

Furthermore, the area has significant topography in the form of deeply incised valleys, which allows for reasonable variation in the z-values of 3–point clusters.

4.1.2. Geological Setting.

The locality of the area selected for this study is approximately 8km east of Alicedale in the Eastern Cape Province, South Africa (**Fig. 4a.**). This area contains outcrops of the Witpoort and Kweekvlei Formations, both members of the Witteberg Group, which

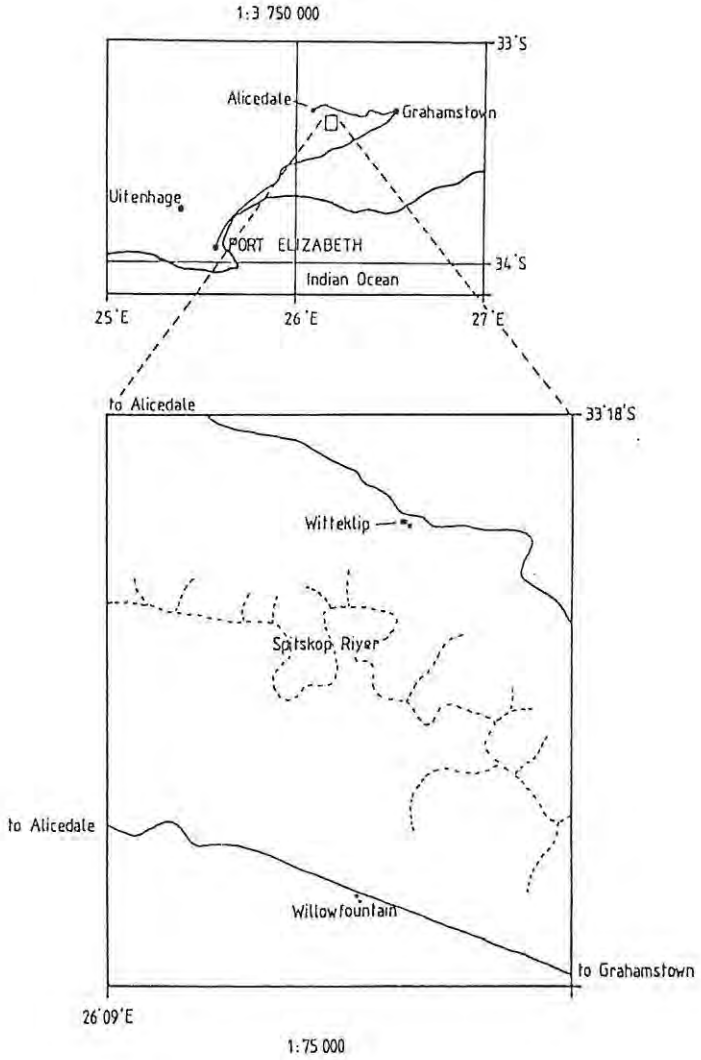


Fig. 4a. Locality of the Alicedale Study Area (Millad, 1993).

is in turn part of the Cape Supergroup (SACS, 1980). Note that the Kweekvlei Formation is also part of the Lake Mentz Subgroup.

The Witpoort Formation consists almost entirely of fine to meduim grained quartz arenite or metasandstone, sporadically interlaced with micaceous black shale bands of thicknesses less than three metres. This formation is relatively resistant to weathering processes and, therefore, forms topographic "highs". The lithology thus controls the

geomorphology of the landscape. Together with a medium level of reflectance, attributed to the sandstone component, the unit, and structures within it, are easily distinguishable on imagery.

The Kweekvlei Formation consists of a black, fissile shale, that weathers relatively easily. This, coupled with poor exposure partly due to agricultural activities, makes the recognition of internal structures difficult in this formation. The Kweekvlei Formation has a darker reflectance than the Witpoort Formation and the contact between the two formations is, thus, easily discernible where outcrops occur.

Regionally, the strike of the bedding planes is roughly oriented WNW-ESE. Gentle folding occurs in the area with the axial plane striking in a WNW–ESE direction. The main geological structure covered by the study area is an anticline in the Witpoort Formation. Several fractures occur in the region, although the surface expression of such structures is not very clear on the aerial photographs used.

4.1.3. Preparation.

4.1.3.1. DEM Generation.

A DEM for the area was generated in the following manner. Firstly, contours and lat/long grid intersections were transferred onto tracing paper by hand from a contoured orthophoto.[1] The contour interval was 10m on a scale of 1:10 000.

The resultant trace was scanned into TNTmips to produce a raster image of the trace and geographic coordinate grid intersections in the RVC file format. The raster image was enhanced by means of threshold resampling, which converts cell values representing lines and gridmarks to 0 (black) and background cell values to 255 (white). In this case, threshold resampling works as follows; a threshold cell value t, where $0 \leq t \leq 255$ for 8–bit rasters, and two outcomes, 0 and 255, are selected. The raster is now operated on cell–by–cell, and for each cell the current cell value, t_{old}, is compared to the threshold value to determine the new value, t_{new}, by the following equation;

$$t_{new} = \begin{cases} 255 \; if \; t_{old} > t \\ 0 \; if \; t_{old} < t \end{cases}$$

Further processing involved thinning the thresholded contour lines to one pixel width.

[1] 3326 AC9 WILTON from the Government Survey Office, Mowbray, Cape Town.

Georeferencing the DEM raster was simplified by the inclusion of grid marks on the original trace. Geographical coordinates obtained from the topographical sheet provided the input data for the referencing process in TNTmips.

In order to attach elevation values to the digital contour lines, the raster was converted to a vector object by means of the following procedure; a semi–automatic line following process in TNTmips was employed so that consecutive raster cells, with zero greyvalue, on a particular contour line were sampled. The coordinates of these cells were stored as point elements, in the vector object, and, collectively, they constituted a line element. The resultant vector had the same georeferenced subobject as the raster.

A DEM was then generated by sampling the vector object at intervals that define a rectangular grid. Grid nodes, for which no contour elevation could be found, were assigned an elevation value determined by a minimum curvature interpolation process (Briggs, 1974; Watson 1992b). A minimum curvature surface, fitted to known elevation points, distributes the change in slope evenly across the whole surface such that the change in slope is minimized at all points. The DEM raster obtained had a cell size of 1.47m.

4.1.3.2. Digital Geological Image Generation.

An aerial photograph of the area scanned at 300dpi[2] to a greyscale raster object, in the RVC format, provided a suitable digital map. The raster was georeferenced by comparison with the orthophoto used in the DEM generation. To correct for photographic distortion, the raster was warped to a UTM projection system, and finally, resampled to have the same cell size as the DEM.

4.1.4. Operation.

The geological image raster and DEM raster, as described in the previous section, were analyzed using the GEOSTRUC© routine. The orientation of bedding planes (S_0), contact planes (S_1), axial planes (S_2) and fracture planes (S_3) were measured (**Fig. 4b.**).

4.1.5. Results and Discussion.

The ASCII file generated by the GEOSTRUC© process (Appendix E) was imported into a TNTmips database utilizing the **Import/Export-Database-ASCII Text** facility. A

[2]dpi stands for **dots per inch** and is a measure of resolution.

user–defined format file (Appendix F) was used to instruct TNTmips on the ordering of data in the ASCII file.

The resultant TNTmips database object was then built into a vector object containing point elements, using the **Import/Export-Vector-RVC Database** routine.

The point elements in this new vector object represented the geographical coordinates of the centroids for each measurement taken. Attached to these points were the dip direction/dip values obtained and the structure classification. For display purposes, symbols were attached to the vector points in order to display the appropriate orientation "structure ticks" at the measurement location. The symbols varied according to the dip amount, and were generated by a style query object in the TNTmips **display-map & poster layout** process (Appendix F).

Measurements where divided into the following categories; bedding planes(S_0), contact planes(S_1), axial planes(S_2) (See Appendix G for instructions on the measurement of axial plane orientation.), and fracture planes(S_3). **Figures 4c–f.** depict the four categories overlaid on the digital aerial photograph of the area.

Fig. 4b. GEOSTRUC© measurements for the Alicedale Case Study. See Appendix H for a high quality reprint of the Alicedale Case Study.

Fig. 4c. S_0 plane orientations for the Alicedale Case Study.
(Data derived by GEOSTRUC©)

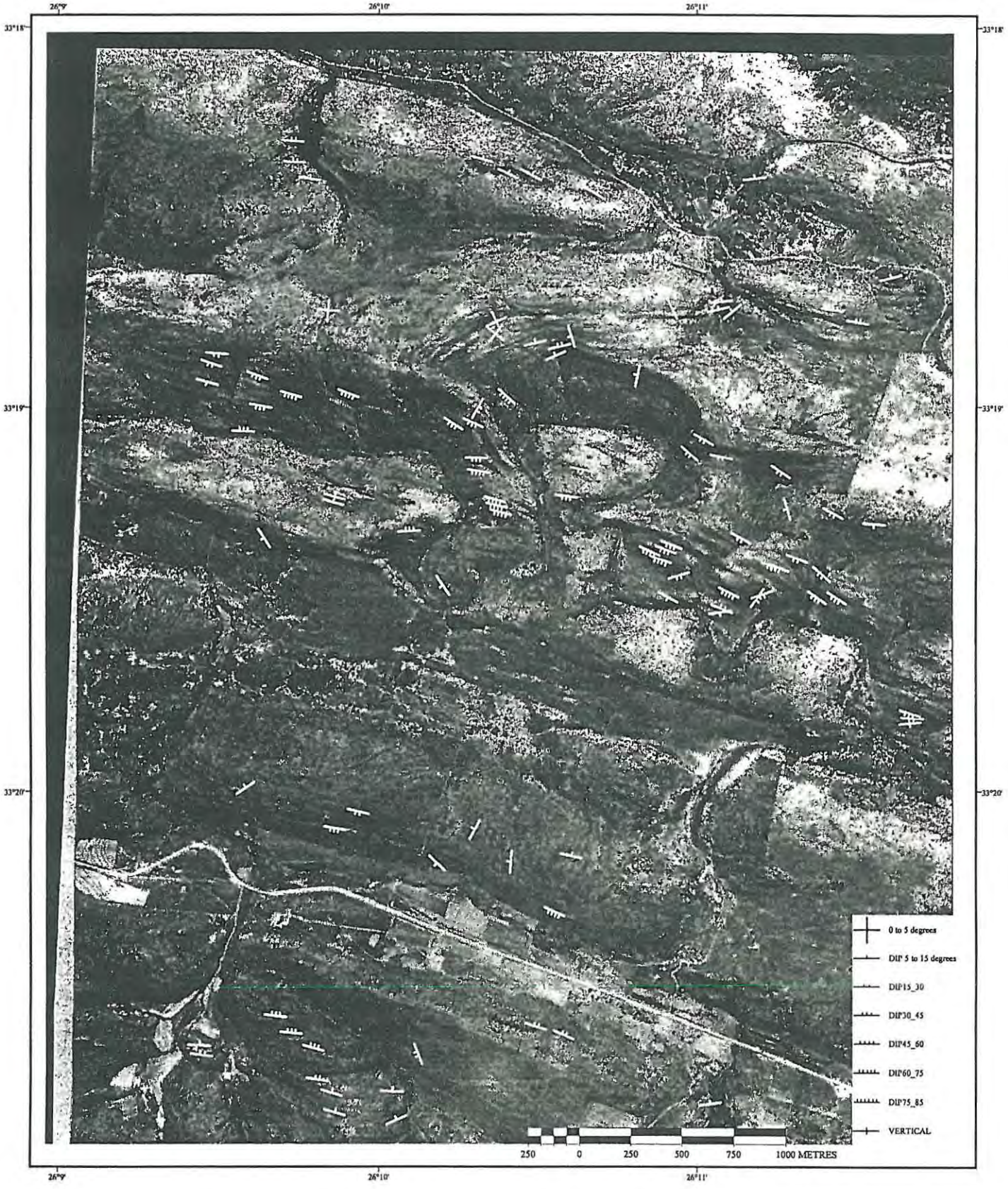


Fig. 4d. S_1 plane orientations for the Alicedale Case Study.
(Data derived by GEOSTRUC©)

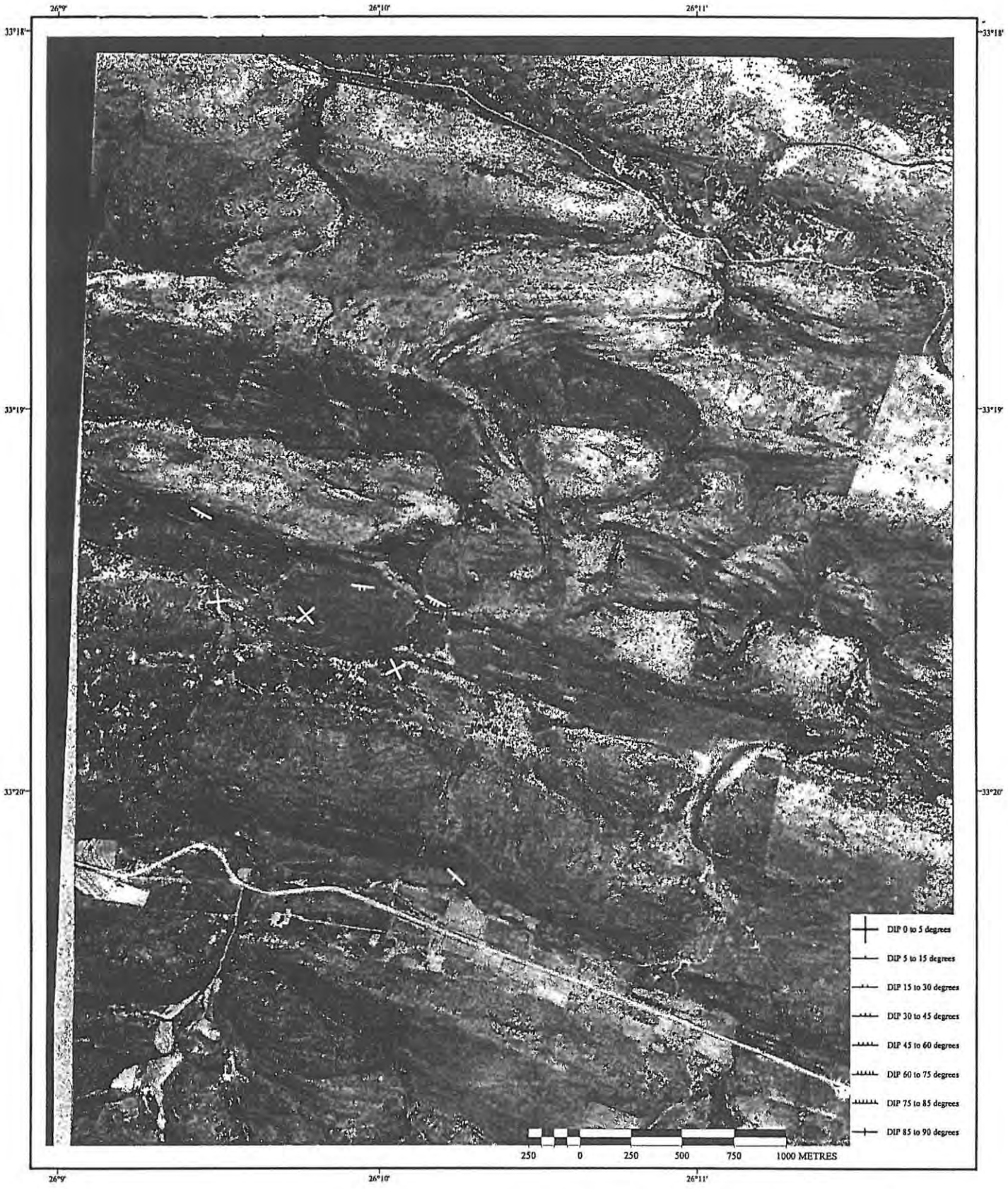


Fig. 4e. S_2 plane orientations for the Alicedale Case Study.
(Data derived by GEOSTRUC©)

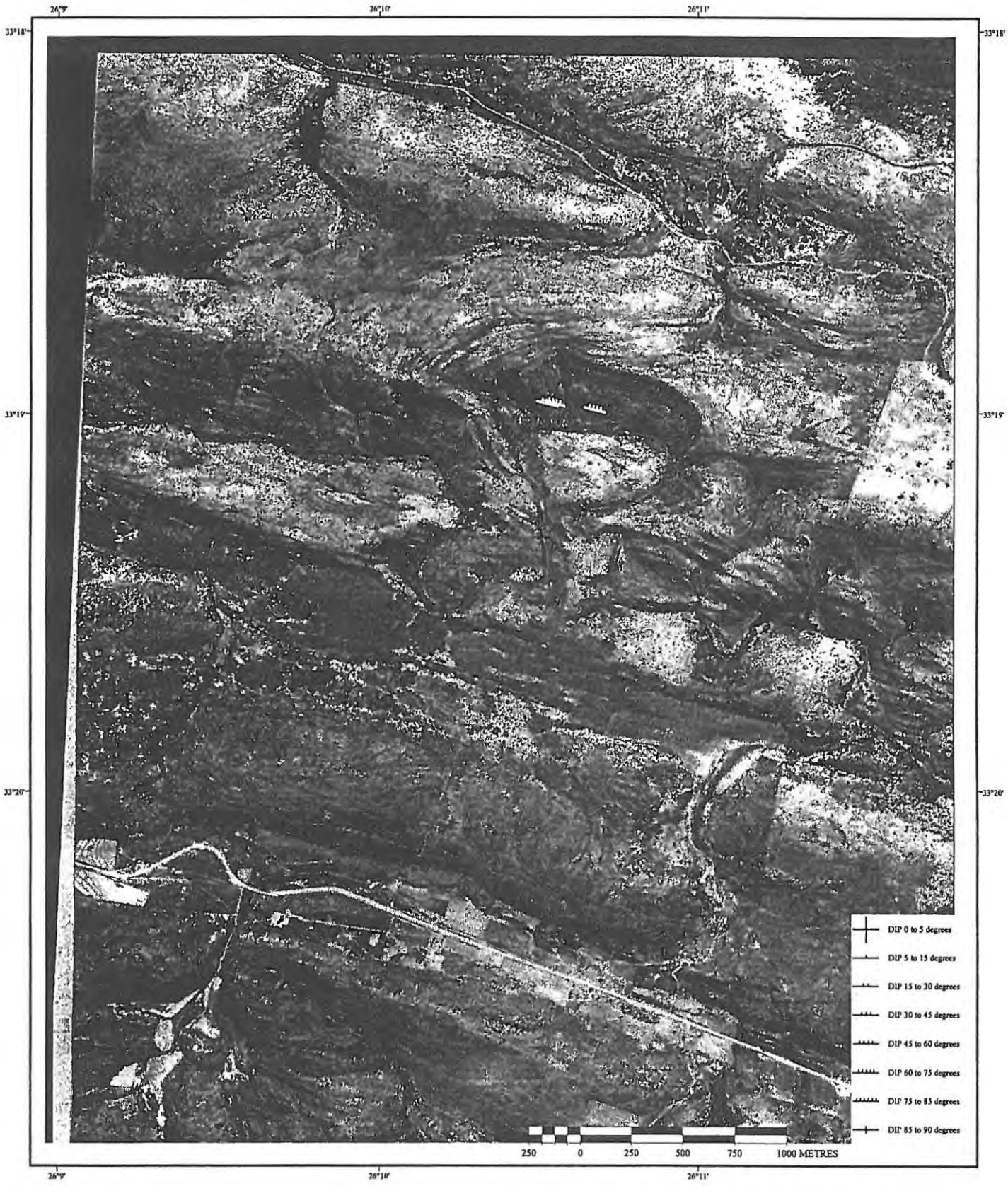
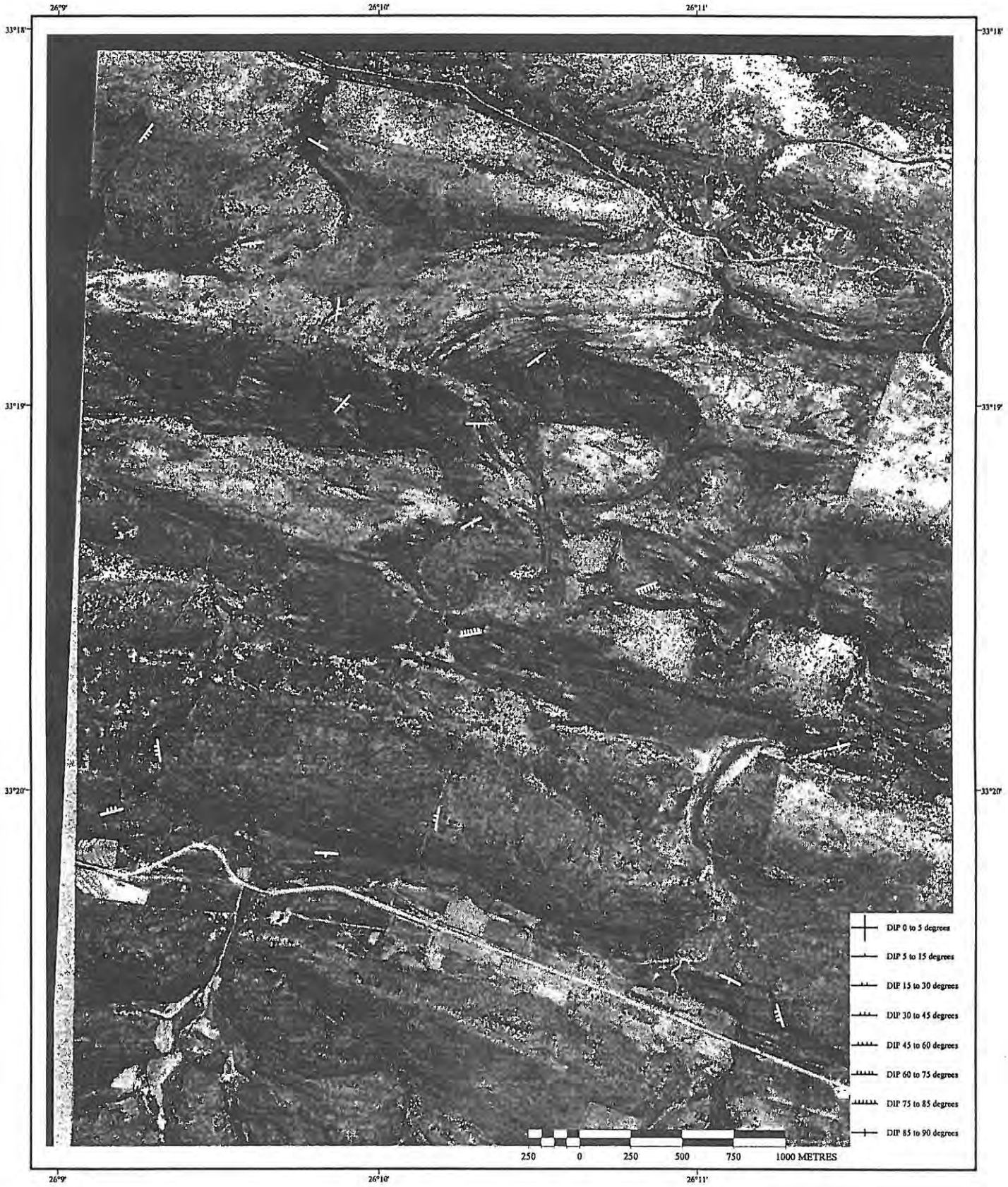


Fig. 4f. S_3 plane orientations for the Alicedale Case Study.
(Data derived by GEOSTRUC©)



Note that the dip direction/dip symbols are positioned at a triangle centroid and, hence, do not always plot at the outcrop position of the measured structure.

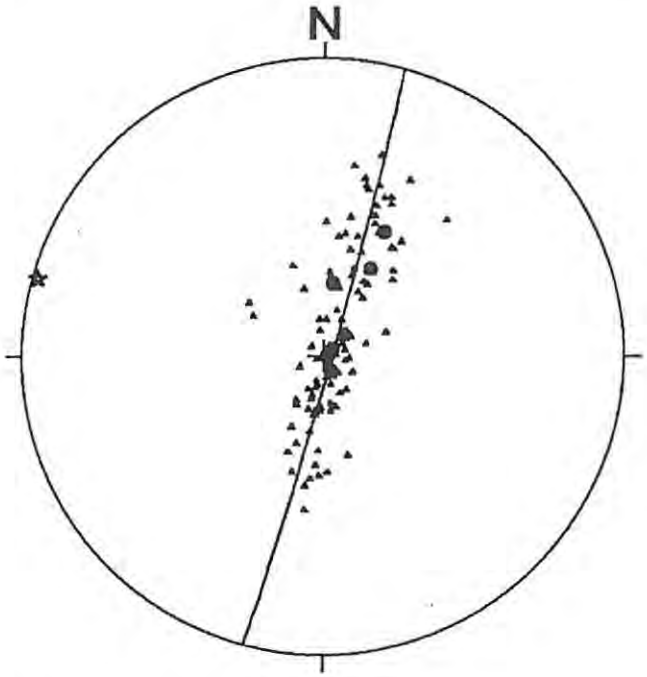


Fig. 4g. S_{0-1} planes for the study area showing the best–fit girdle and pole. Triangular icons mark poles to S_0 planes and circular icons mark poles to S_1 planes. The star icon marks the pole to the best–fit girdle representing the regional fold axis orientation. Sample Size = 104.
(Data derived by GEOSTRUC©)

The accompanying stereonet for the poles to S_{0-1} is shown in **Fig. 4g.** The stereonet shows that the dominant strike of the bedding in the area is WNW–ESE. The best–fit great circle to these poles, also depicted on the stereonet, strikes 029/81 SE with a pole located at 299/09 representing the best–fit fold axis for the region. This result compares favourably with field measurements in this area by Millad (1993) who calculates a best–fit fold axis orientation of 284/09 which is within 15 degrees of the result obtained in this study by remote sensing methods.

Eigenvector statistics performed on the S_{0-1} data, as proposed by Woodcock (1977), are tabulated in **Fig. 4h.** The K parameter (fabric shape parameter) reveals a girdle–like shape for the distribution of the poles to the S_{0-1} planes and indicates that the fold axis determined is statistically significant. This girdle–like concentration about the best–

```
                EIGENVECTORS
      l       m       n       Strike  Dip
V1=  0.136  0.076  0.988   29.306 81.019
V2= -0.954 -0.258  0.151  195.119  8.711
V3=  0.266 -0.963  0.038  285.451  2.167

                EIGENVALUES
Lambda1= 85.834  S1=  0.825
Lambda2= 16.973  S2=  0.163
Lambda3=  1.193  S3=  0.011

S1/S2=  5.057
S2/S3= 14.224
S1/S3= 71.932

Ln(S1/S2)=  1.621  Ln(S2/S3)=  2.655

C=  4.276
K=  0.610
N=104
```

Fig. 4h. Eigenvector statistics determined for the distribution of S_{0-1} poles. (Data derived by GEOSTRUC©)

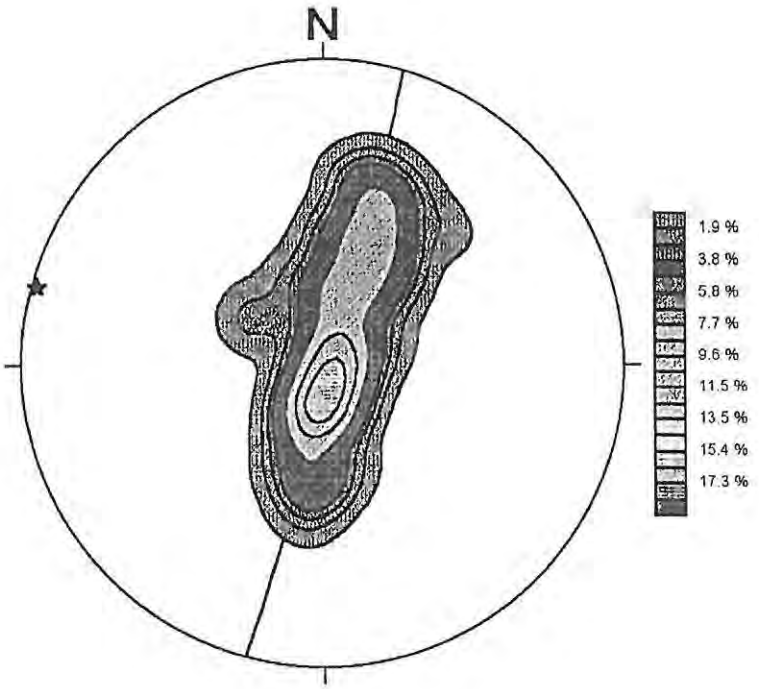


Fig. 4i. Contoured stereonet of S_{0-1} poles. (Data derived by GEOSTRUC©)

fit great circle indicates fold limbs that dip to the NNE and SSW respectively. Again this result is validated by field measurements taken by Millad (1993). An asymmetrical concentration of poles to bedding about the stereonet centre mark, as illustrated by a

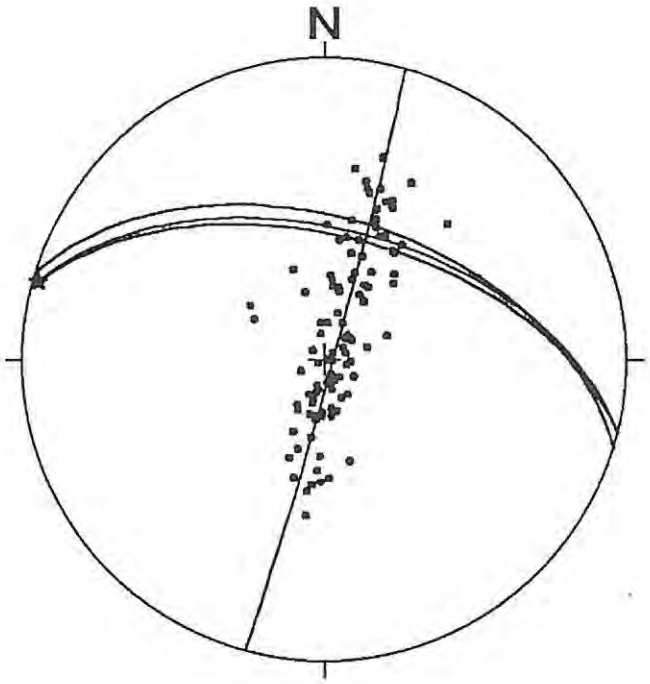


Fig. 4j. A stereonet plot of the S_{0-1} poles and the S_2 planes. The axial planes (S_{0-2}) determined using GEOSTRUC© are shown as ESE–WNW striking great circles. Poles to bedding and the best–fit data to these are also shown for comparison. (Data derived by GEOSTRUC©)

contoured stereonet (**Fig. 4i.**), may be an indication of asymmetrical folding. Millad (*opp. cit.*) does mention that folds in the area possess southern limbs dipping steeply to the south and northern limbs dipping gently to the north, that is, a SW vergence which is uncharacteristic of the CFB in general. This geometry is clearly evident on the stereonet (**Fig. 4i.**) and on the image map (**Fig. 4c.**).

Fig. 4j. shows the S_2 (axial plane) measurements plotted on a stereonet with the poles to the S_{0-1} planes also shown. The axial plane measurements derived by the GEOSTRUC© method appear to be valid since the fold axis obtained from GEOSTRUC© analysis plots on the great circles representing the axial planes, hence satisfying the criterion that the fold axis of a fold is always contained within the axial plane. Note that the axial planes dip to the north (as validated also by Millad's (1993) fieldwork) confirming the unique SW vergence of the folds in this region of the CFB.

Few fracture planes could be distinguished on the aerial imagery, thus making the sample too small (N=18) for a formal evaluation of joint sets and fault geometry. In **Fig.**

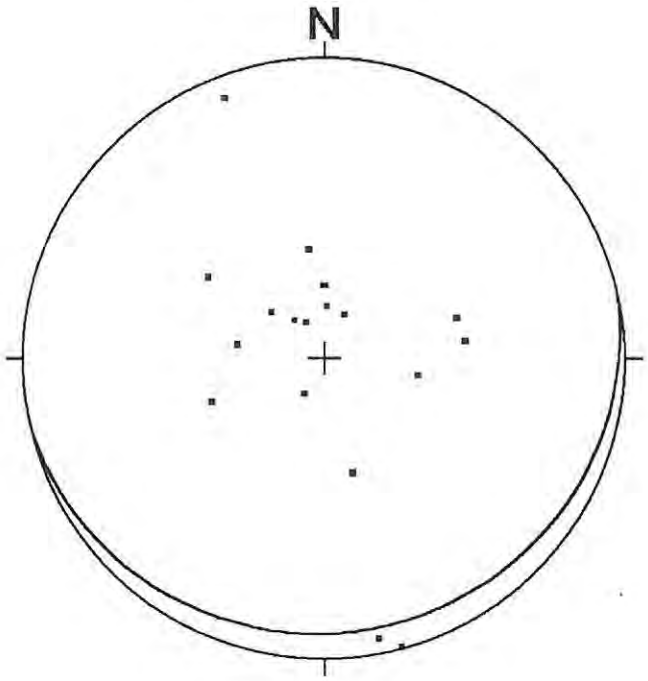


Fig. 4k. S_3 poles for the study area showing the great circle mean. Sample Size = 18. (Data derived by GEOSTRUC©)

4k. a stereonet is presented with the poles to the fracture planes plotted. Eigenvector analysis of the sample indicates a cluster–like distribution of poles with a mean of 346/80. Visual inspection of this stereonet also shows joint sets striking ENE–SSW with a dip of approximately 20 degrees, and possibly another striking NNW–SSE and dipping at 45 degrees. This configuration of joint sets conforms to commonly occurring joint sets in folded bedding planes (Hobbs *et al.*, 1976).

4.2. Program Application Two: SPOT Panchromatic image on NDEM data

4.2.1. Introduction.

A study area in the Baviaanskloof–Kareedouw district, Eastern Cape, South Africa, was selected (**Fig. 4l.**).

Situated in the Cape Fold Belt (CFB), this structurally complex mountainous region provides an excellent opportunity to test the GEOSTRUC© routine in an area characterized by high relief. Other considerations taken into account in the selection of the study area were meduim vegetation cover, the availability of a cloud–free SPOT image

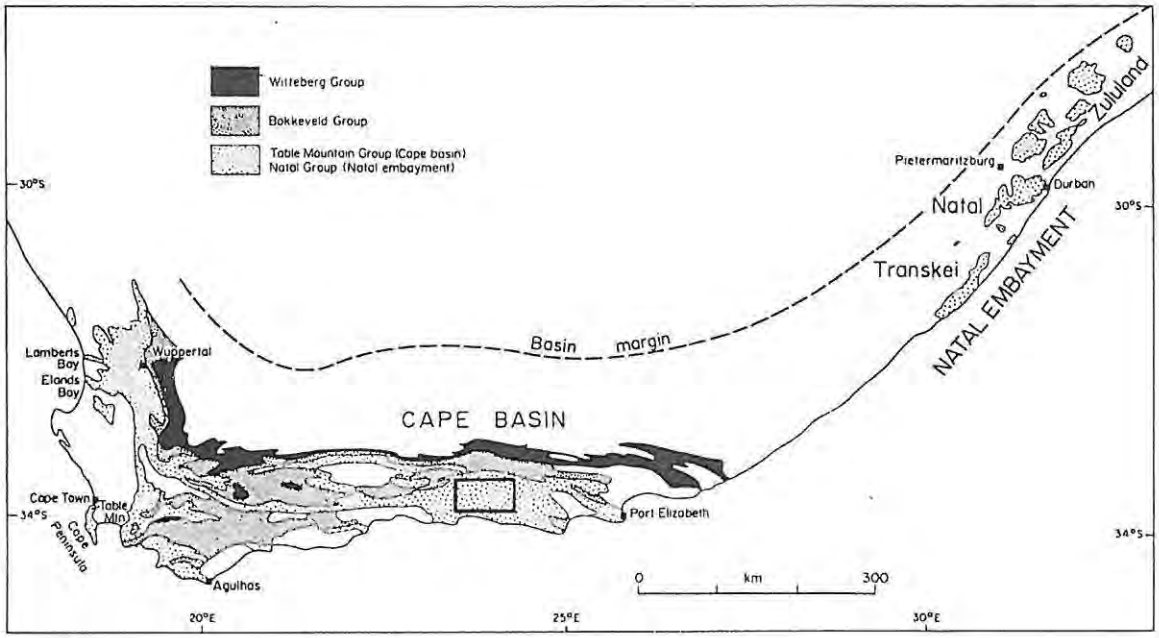


Fig. 41. Locality of the Kareedouw Study Area (Tankard *et al.*, 1982).

and access to existing field data.

4.2.2. Geological Setting.

4.2.2.1. Lithostratigraphy.

In the area of interest the CFB consists of East–West striking regional folds with the rocks belonging to the Cape and Karoo Supergroups. The main geological formations present in the study area are (Tankard, 1982);

(1) Peninsula Formation - Medium– to coarse–grained quartz arenite with quartz pebbles and trace fossils.

(2) Cedarberg Formation - Shale, mudstone and fine–grained sandstone.

(3) Tchando Formation - Quartz arenite.

(4) Baviaanskloof Formation - Shale, mudstone, quartz arenite with marine invertebrates.

(5) Nardouw Formation - Coarse–grained quartz arenite with trace fossils.

(6) Gydo Formation - Shale, siltstone and fine–grained sandstone.

The Gydo Formation is part of the Bokkeveld Group and the other formations belong to the Table Mountain (TM) Group. The formations follow a chronological depositional sequence with age increasing northwards. The material for the TMS (Table Mountain Sandstone) group was derived almost exclusively from the north and its age is bounded by marine fossil assemblages of the late Ordovician in the Cedarberg Formation and brachiopods of the early Devonian in the upper Nardouw (Truswell, 1977). The Bokkeveld group overlays the TMS group in the Eastern Cape, with its lithology represented by the Ceres and Traka subgroups. This group contains shallow water marine fauna of the late Early Devonian, and is characterized by interbedded sandstone and shale formations.

4.2.2.2. Structure.

The CFB consists of two geographically distinct structural branches. The interest of this case study lies in the south–eastern branch, characterized by tightly folded beds, sliced by thrusts and normal faults trending roughly eastward. Large variation in the dip amount of bedding planes occurs. This is attributed to original differences in bedding thickness and to fold repetition of layered sequences (Hälbich, 1992).

A strongly developed joint set is present, striking roughly N–S, and nappe structures have been identified in the Baviaanskloof Range to the north of the study area (Theron, 1969).

The exact nature of the orogenic process involved in the formation of the CFB is still a debatable issue (Hälbich, 1983). The regular, co–linear style of folding is atypical of soft–sediment folding and indicates a horizontal N–S shortening as a deformation mechanism.

4.2.3. Preparation.

4.2.3.1. Digital Geological Image Generation.

A SPOT pan–chromatic scene with 10m ground resolution, preprocessed to level S of the study area, was used as the input image raster.[3] The preprocessed SPOT image has the following corrections applied to it;

(1) the image is resampled to exclude the effects of malfunctioning detectors and to geometrically correct the image for distortion introduced by the imaging set–up,

(2) the registration of corner pixels to map coordinates,

[3]Scene ID : 129-0-418 910812.

(3) a second geometric correction to allow the image to be registered with another SPOT image, allowing for multitemporal studies.

This level of preprocessing produces a high precision image for pixel–to–pixel referencing.

Due to the lack of a CCT tape reader another method of importing SPOT imagery had to be used in this project. TNTmips v5.00 provides an import utility especially designed to translate imagery from the SPOT CCT format to raster objects in the TNTmips RVC format. To implement the **prepare-import/export** utility, the SPOT header file for this particular scene was accessed in order to obtain the number of rows and columns. A simple C program (Appendix C) was written in order to read specific lines from the SPOT Header file according to the CCT format specifications (SPOT, 1990).

The next step in data preparation was the registration of the SPOT image using the georeferencing process found under **prepare-georeference**. For the initialization of this process projection information had to be extracted from the SPOT Header file supplied with the image. This was accomplished by utilizing the same C routine discussed above. The following information was extracted;

(1) Image projection model (Universal Transverse Mercator).

(2) Projection spheroid (International Hayford 1909).

(3) Coordinates for the corner pixels of the scene.

The corner pixels of the SPOT scene, together with their supplied coordinates, were the first four points used in the georeferencing process. Further reference points were identified on 1:50 000 topographical sheets[4], mounted on a Summagraphics digitizer tablet. The selected geographical points used in registering the satellite image satisfied the following three criteria;

(1) A point must be recognizable as a distinguishable variation in the pixel pattern of the image.

(2) A point must be recognizable as a variation in the contour pattern, or other mapped feature, on the topographical map.

(3) A point must either have a spot–height or a height easily extrapolated from nearby contour lines.

[4] 3323DB, 3324CC, 3324CA, 3324CB.

Examples of typically selected points are mountain peaks, road intersections and river valley intersections. The 10–by–10m ground resolution of SPOT panchromatic images makes the selection of such points a relatively straight forward process. A total of 27 evenly distributed georeferencing points were selected for the image. The residuals for the points, that is, a measure of the accuracy of the georeferencing process, were kept below 300m using the affine surface fitting model.

4.2.3.2. DEM Generation.

Elevation data covering roughly the same area as the SPOT scene were obtained from the National Land Information Services, Mowbray, South Africa, as part of the National Digital Elevation Model database.[5] Before being imported into TNTmips, the raster image data files had to be converted from an ASCII format (CCNLIS, 1990) to binary format using a custom C routine designed for this purpose (Appendix C). The import facility in TNTmips, with **simple array** selected, was used to translate the NDEM data, in binary format, to the RVC format.

Image parameter information from the header of each image file, used for the registration of the data, was extracted using a custom C routine. The DEMs were registered in the Gauss Conformal projection system. For use with the GEOSTRUC© analysis routine, they were converted to the lat/long projection system. This was done using the **prepare-raster-resample-automatic** process in TNTmips. Finally, using the same process in TNTmips, the DEMs were resampled from their original 200m ground cell resolution to 10m cell resolution to conform to the SPOT image raster resolution.

4.2.4. Operation.

For the sake of brevity, the test area was divided into four sub–areas according to the available DEM divisions. The ASCII files generated by GEOSTRUC© for each of the four sub–areas are given in Appendix E. Note that the ASCII files for bedding orientation, fracture orientation and axial plane orientation, for each of the sub–areas, are separated for ease of operation. **Fig. 4m.** shows the measurements on the 3323DD section for the bedding plane orientations.

The measurement of orientations of structures in the Langkloof Valley, in the study

[5] 3323DB, 3323DD, 3324CA, 3324CC.

area, were hampered by agricultural activities in the valley. A strongly developed drainage pattern in the northern parts of the study area aided the measurement of joint plane orientations.

4.2.5. Results and Discussion.

The ASCII files were processed in a similar fashion as outlined in the beginning of section 4.1.5. in order to produce 12 structural maps of the study area. That is, four maps showing the bedding orientation (**Figures 4n–q.**), four maps showing axial plane orientation (See Appendix G for instructions on the measurement of axial plane orientation.) (**Figures 4r–u.**) and four maps showing fracture orientation (**Figures 4v–y.**).

Fig. 4m. GEOSTRUC© measurements for the Kareedouw Case Study, 3323DD. See Appendix H for a high quality reprint of the Kareedouw Case Study.

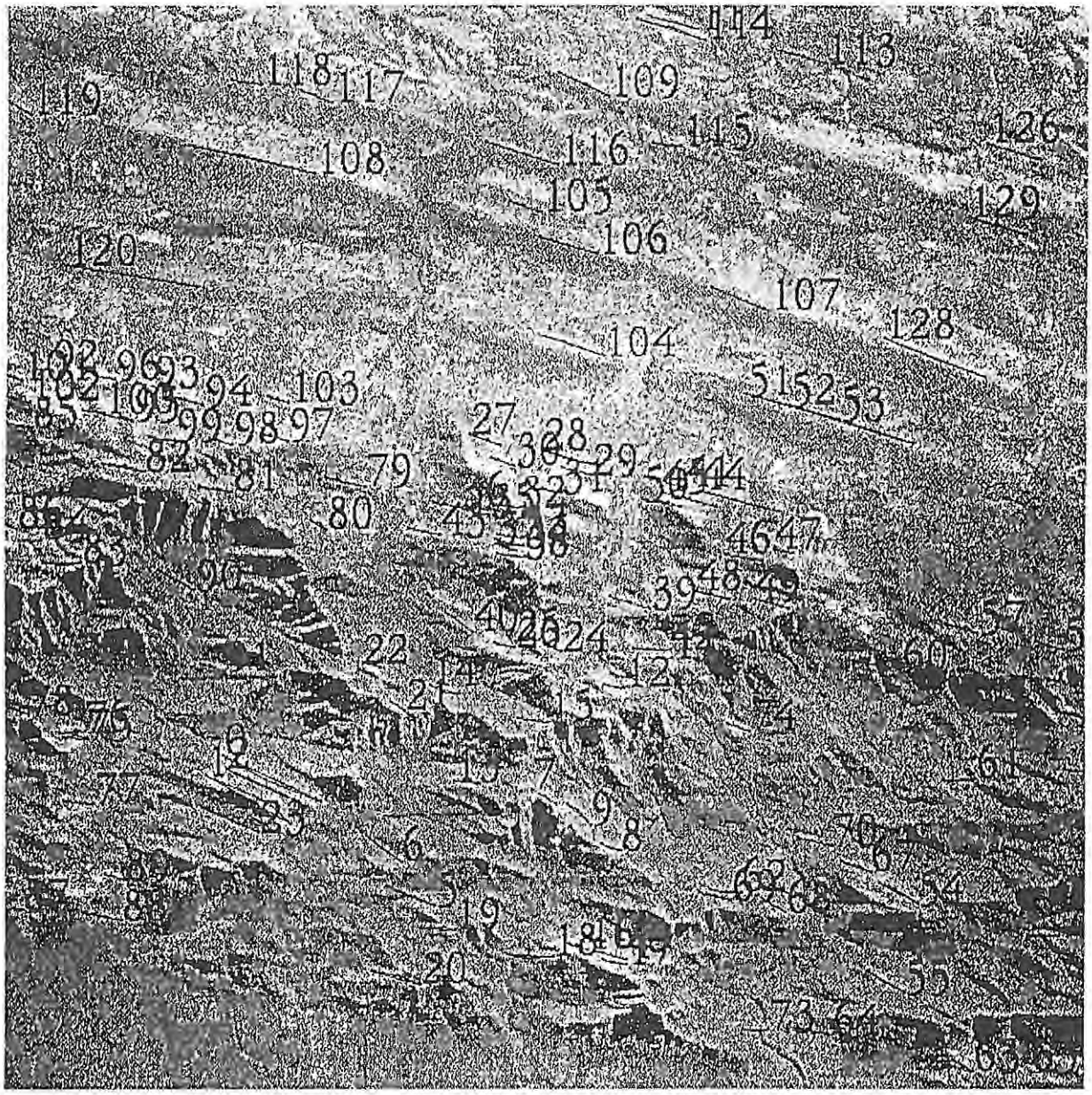


Fig. 4n. S_0 plane orientations for the Kareedouw Case Study, 3323DB. (Data derived by GEOSTRUC©)

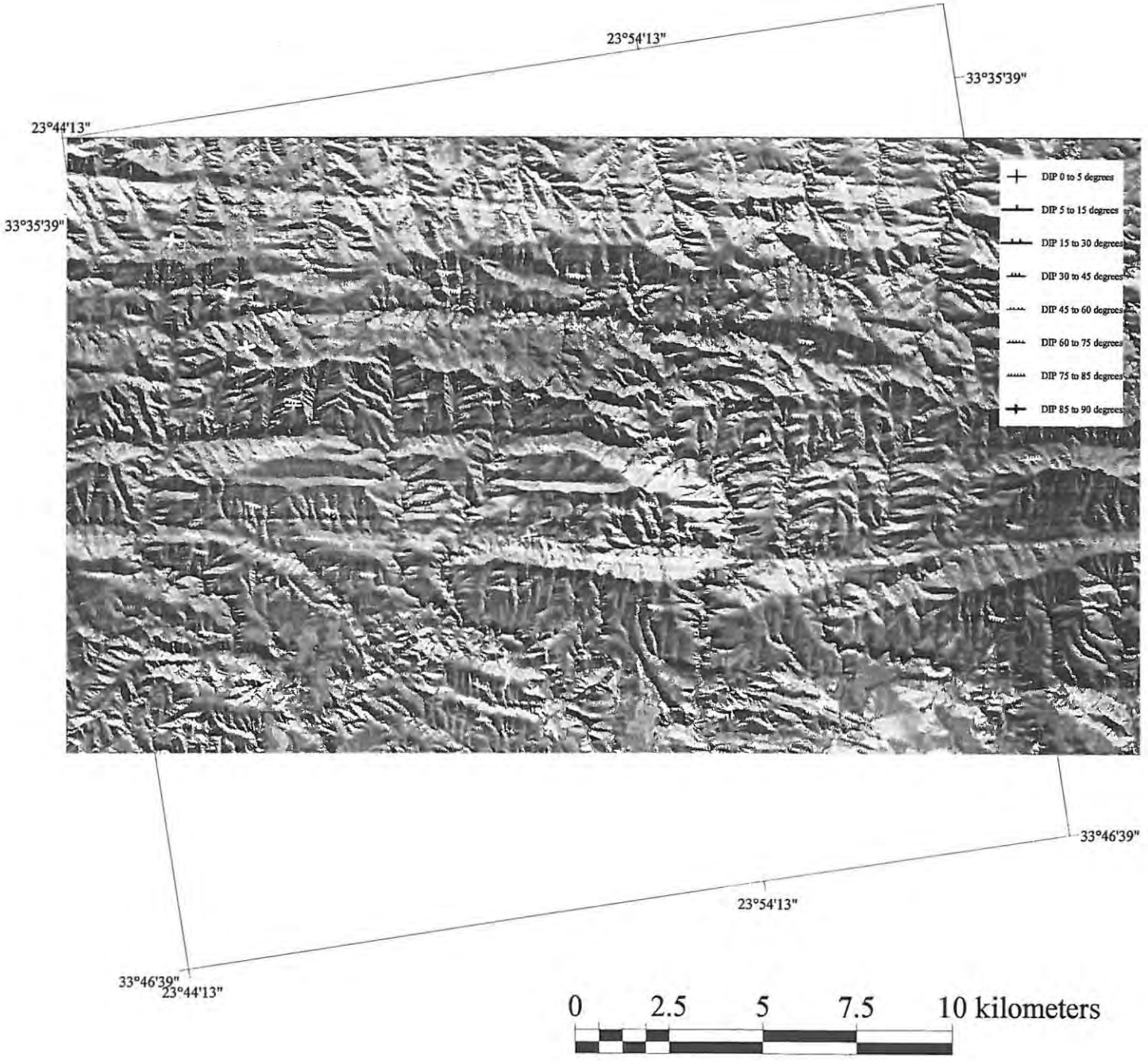


Fig. 40. S_0 plane orientations for the Kareedouw Case Study, 3323DD. (Data derived by GEOSTRUC©)

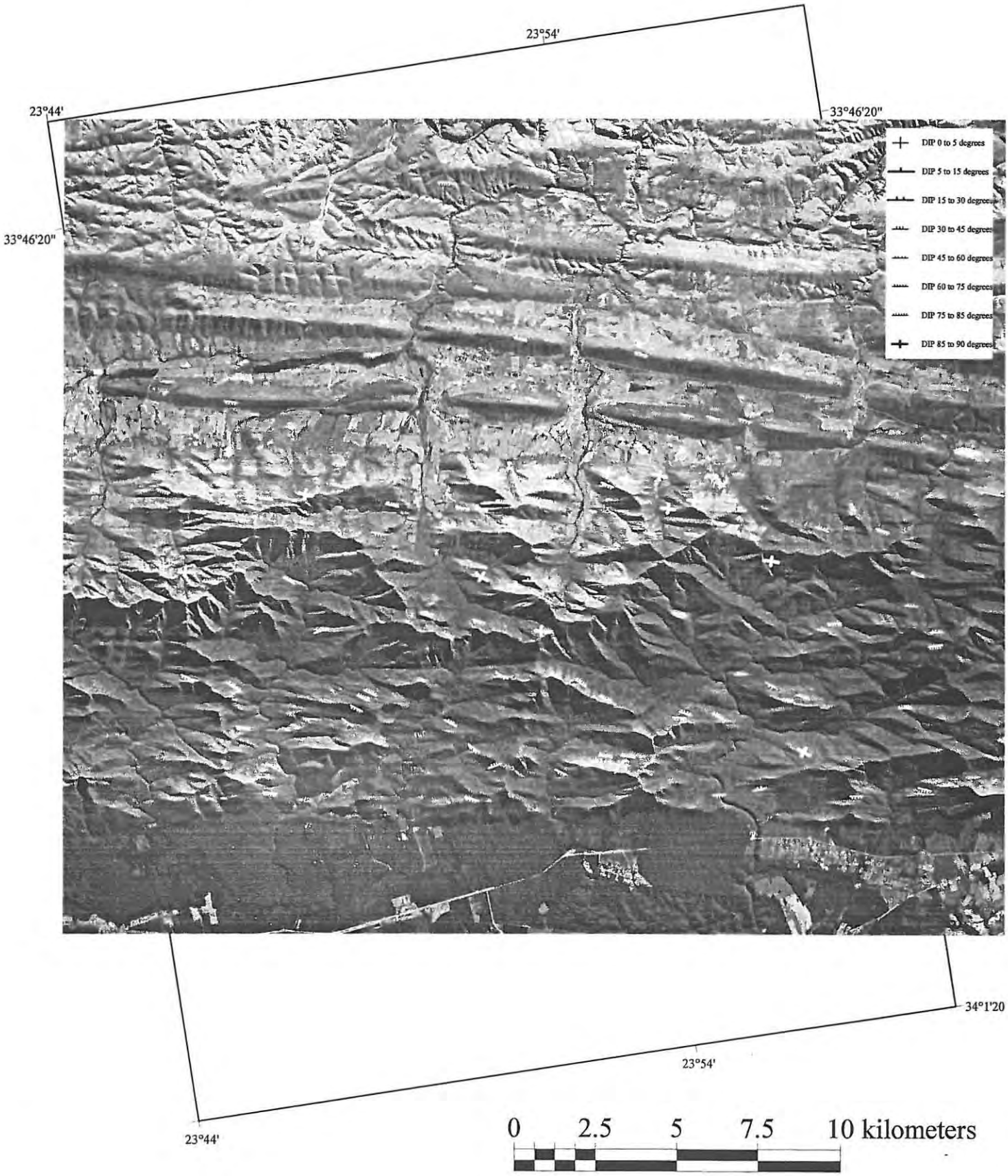


Fig. 4p. S_0 plane orientations for the Kareedouw Case Study, 3324CA. (Data derived by GEOSTRUC©)

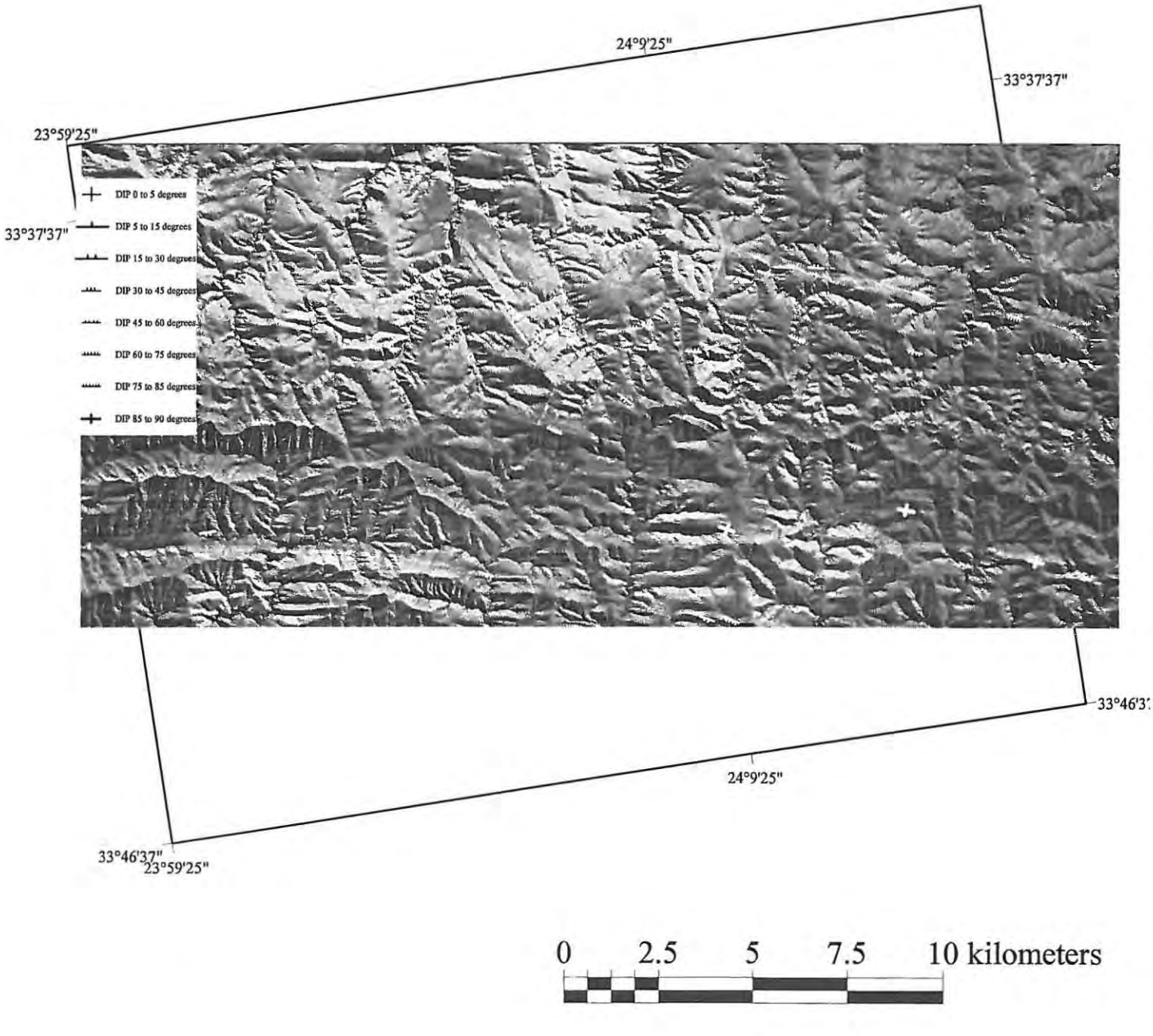


Fig. 4q. S_0 plane orientations for the Kareedouw Case Study, 3324CC. (Data derived by GEOSTRUC©)

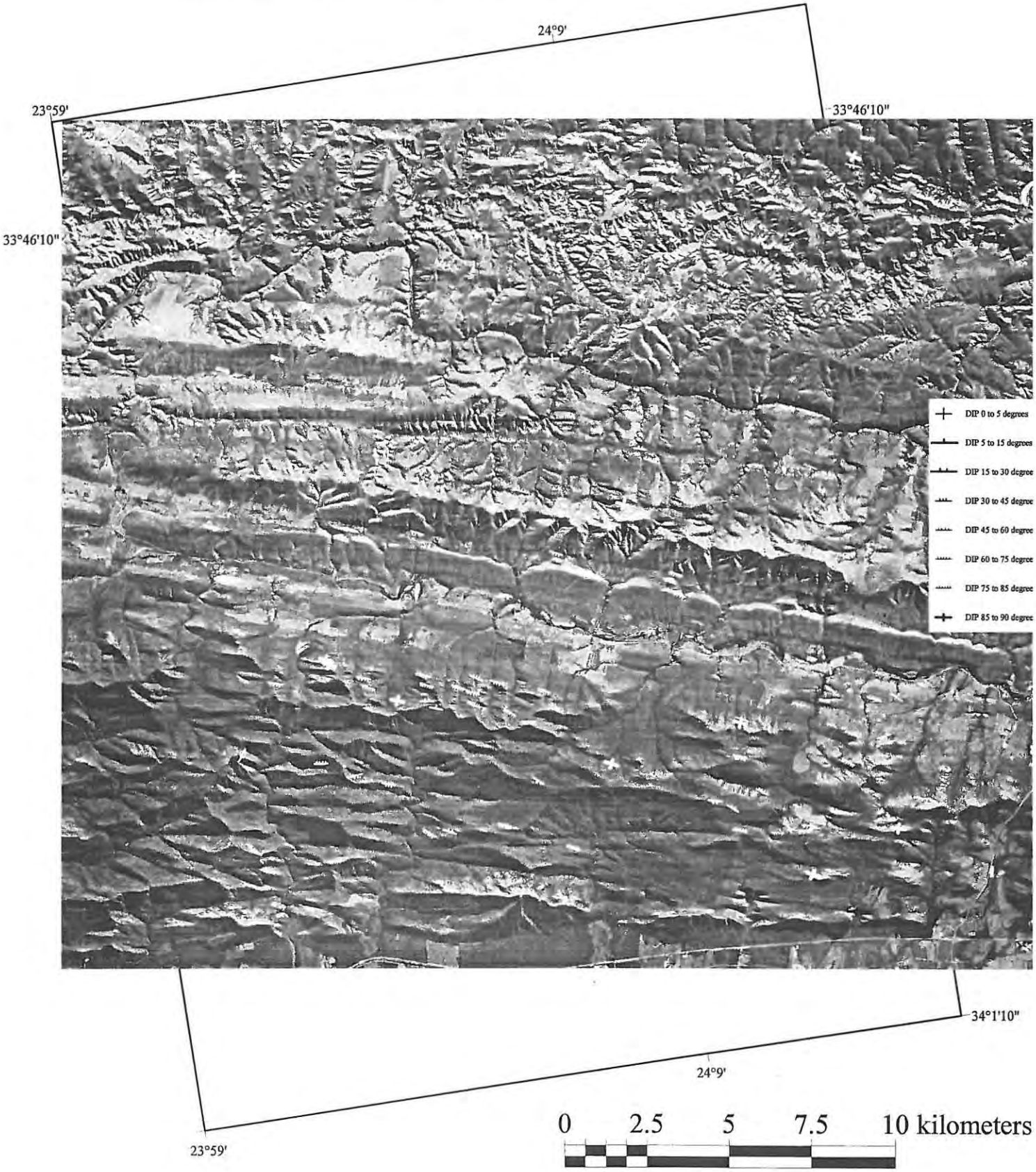


Fig. 4r. S_2 plane orientations for the Kareedouw Case Study, 3323DB. (Data derived by GEOSTRUC©)

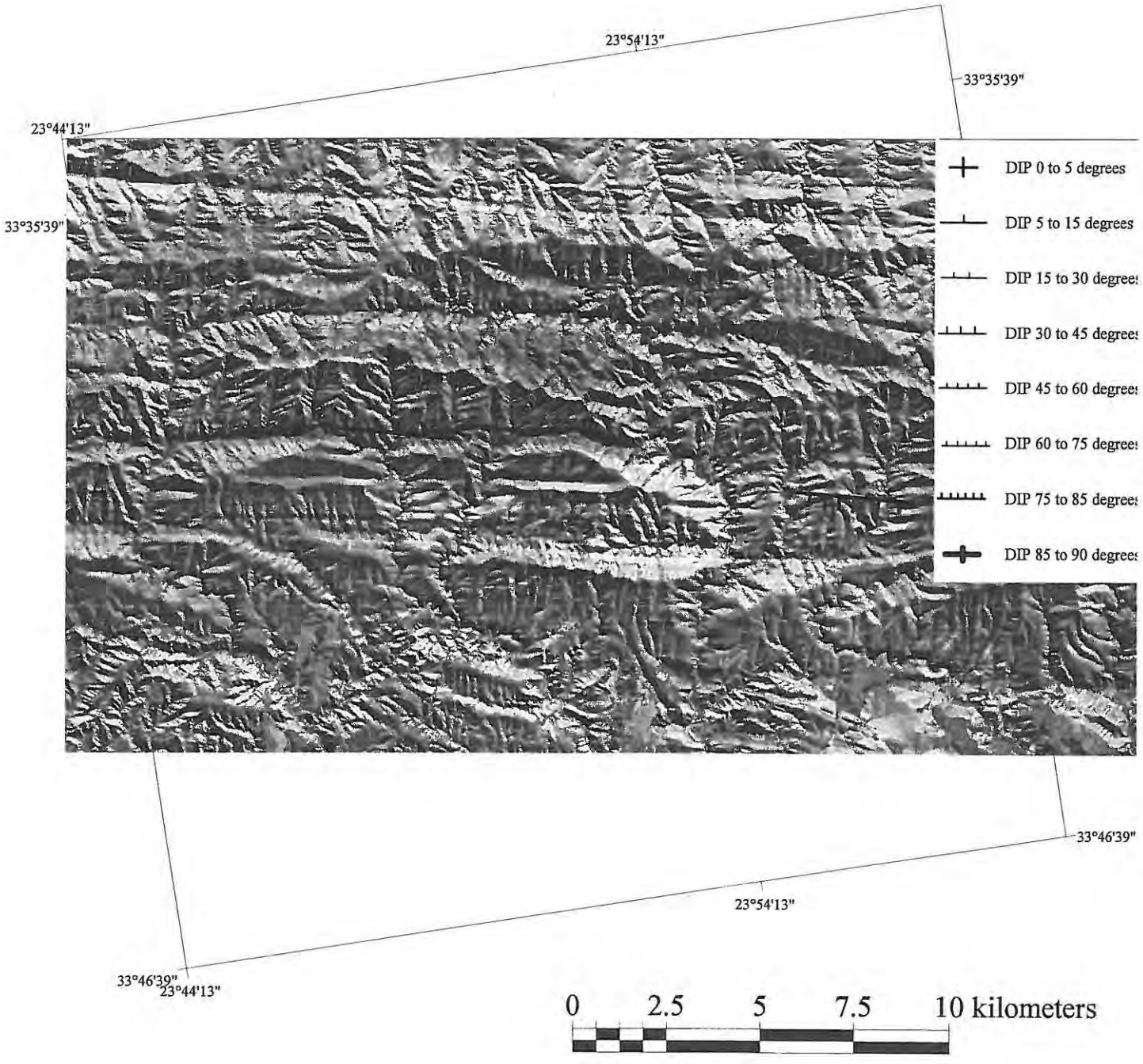


Fig. 4s. S_2 plane orientations for the Kareedouw Case Study, 3323DD. (Data derived by GEOSTRUC©)

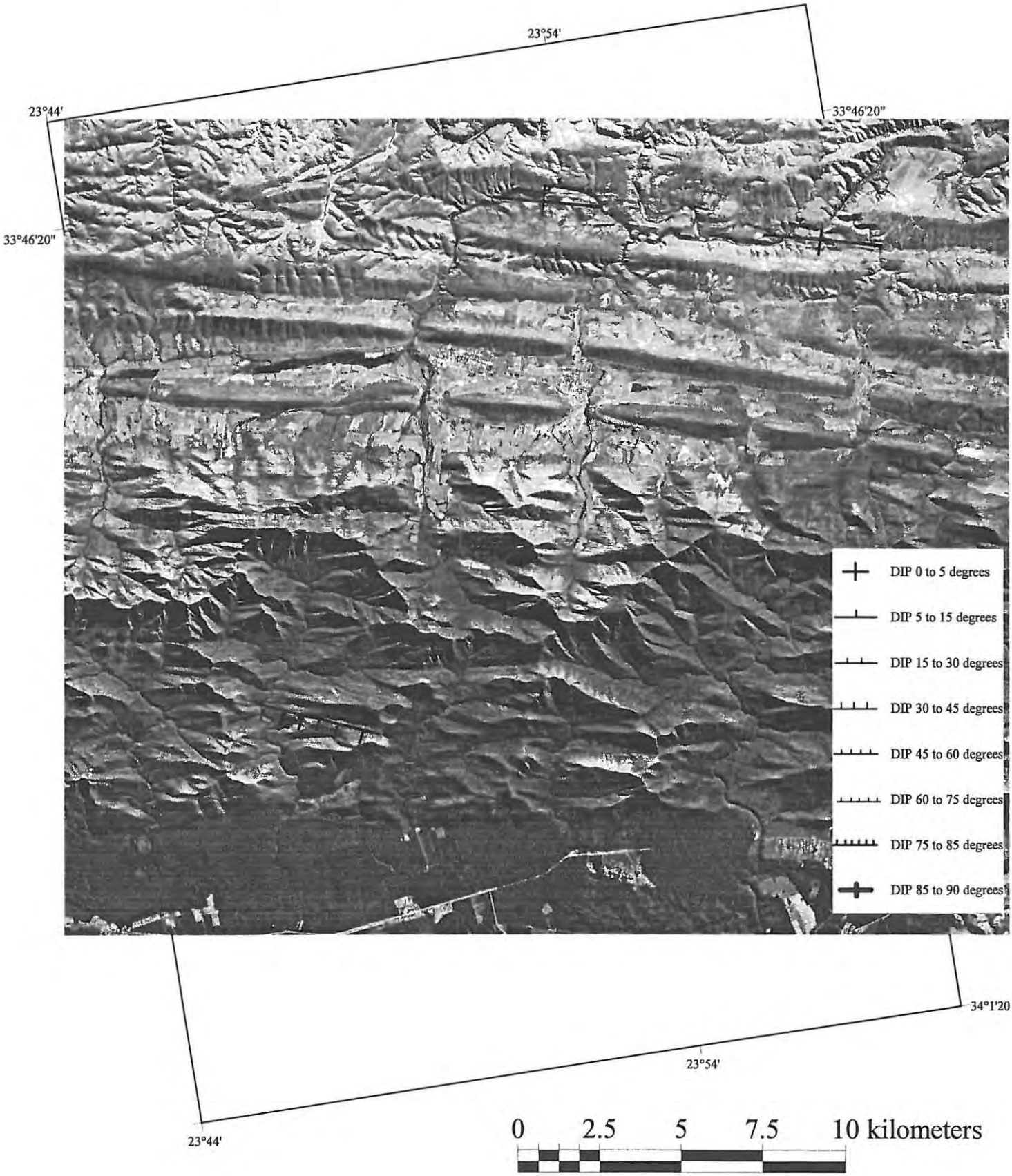


Fig. 4t. S_2 plane orientations for the Kareedouw Case Study, 3324CA. (Data derived by GEOSTRUC©)

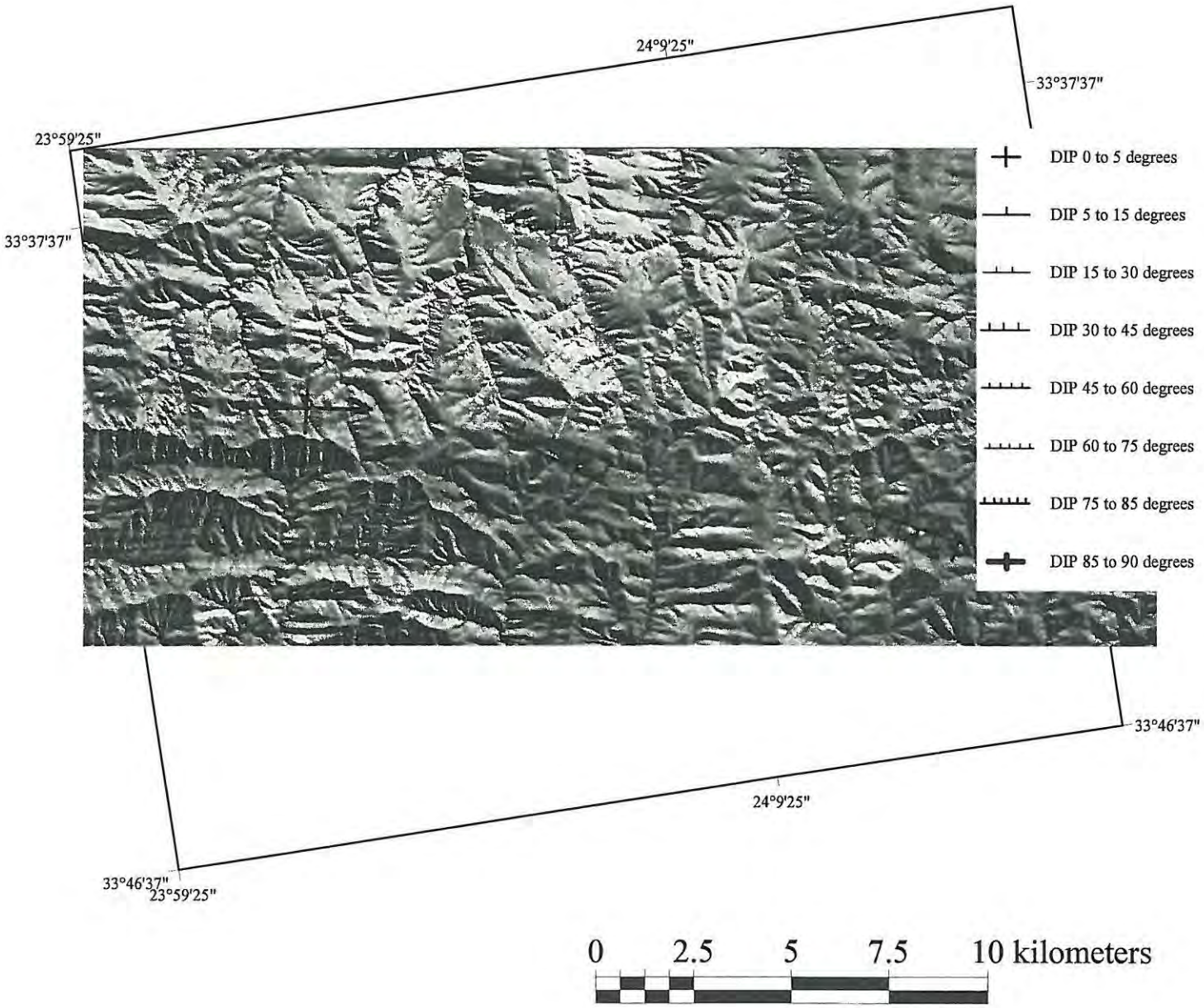


Fig. 4u. S_2 plane orientations for the Kareedouw Case Study, 3324CC. (Data derived by GEOSTRUC©)

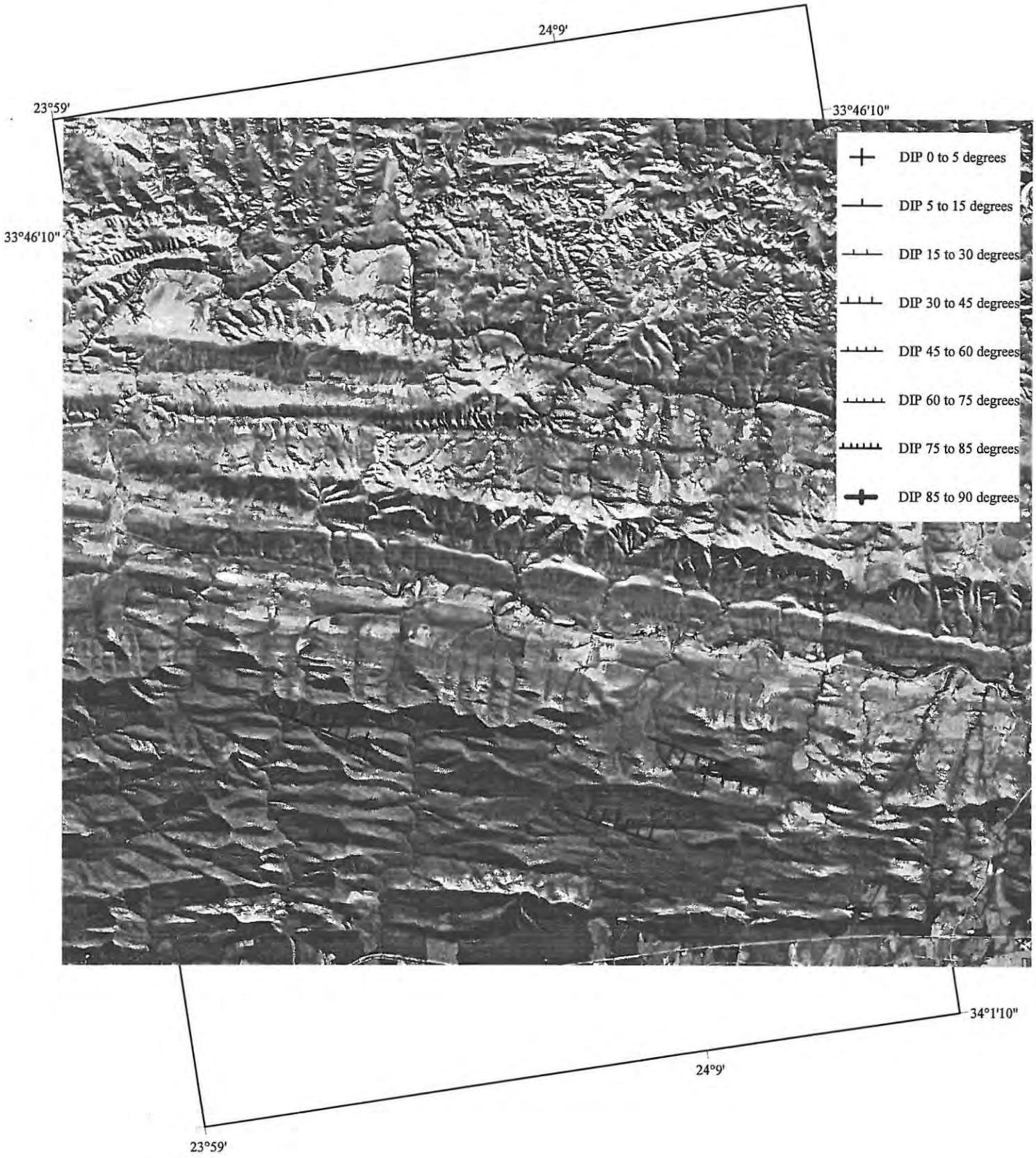


Fig. 4v. S_3 plane orientations for the Kareedouw Case Study, 3323DB. (Data derived by GEOSTRUC©)

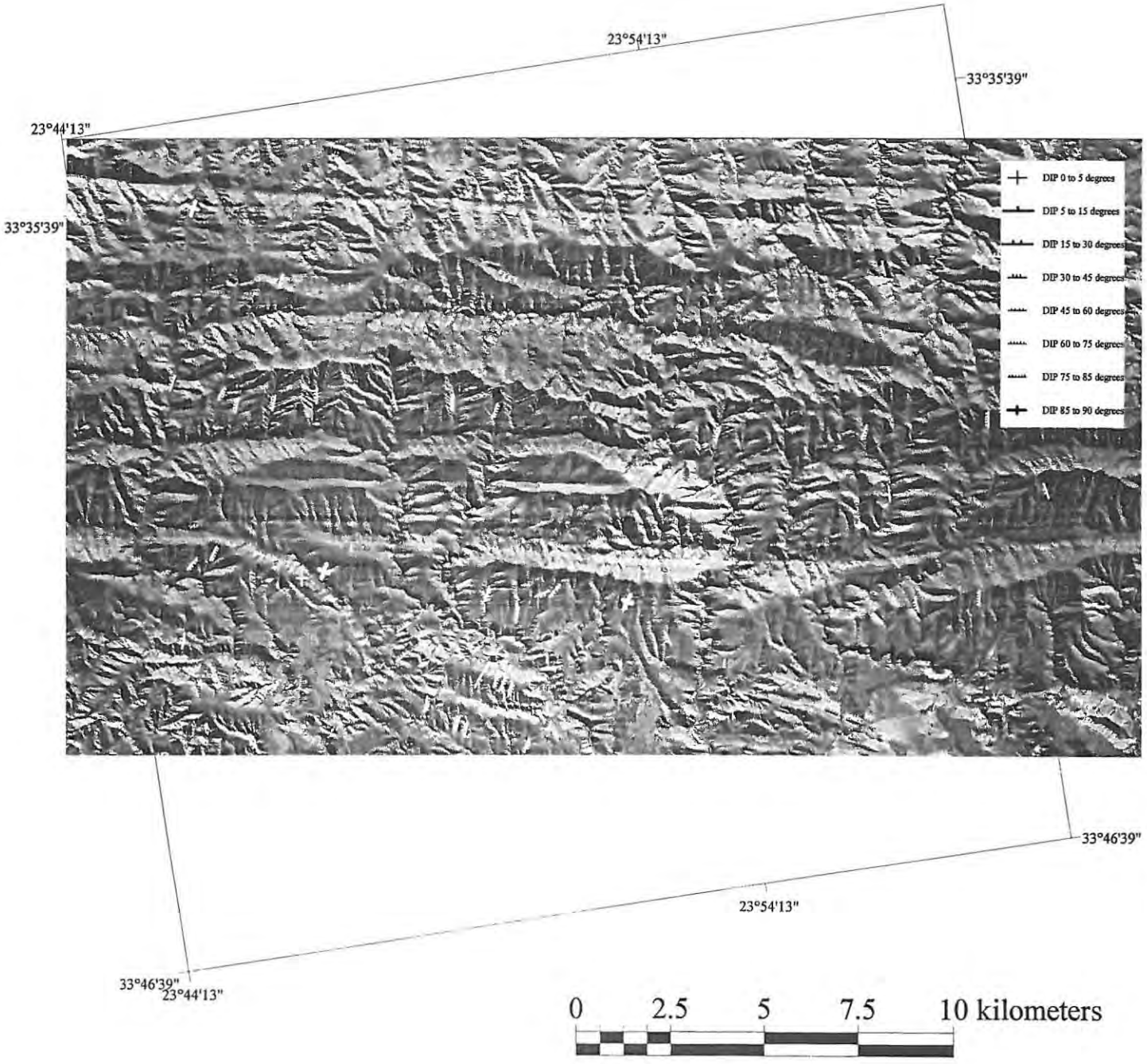


Fig. 4w. S_3 plane orientations for the Kareedouw Case Study, 3323DD. (Data derived by GEOSTRUC©)

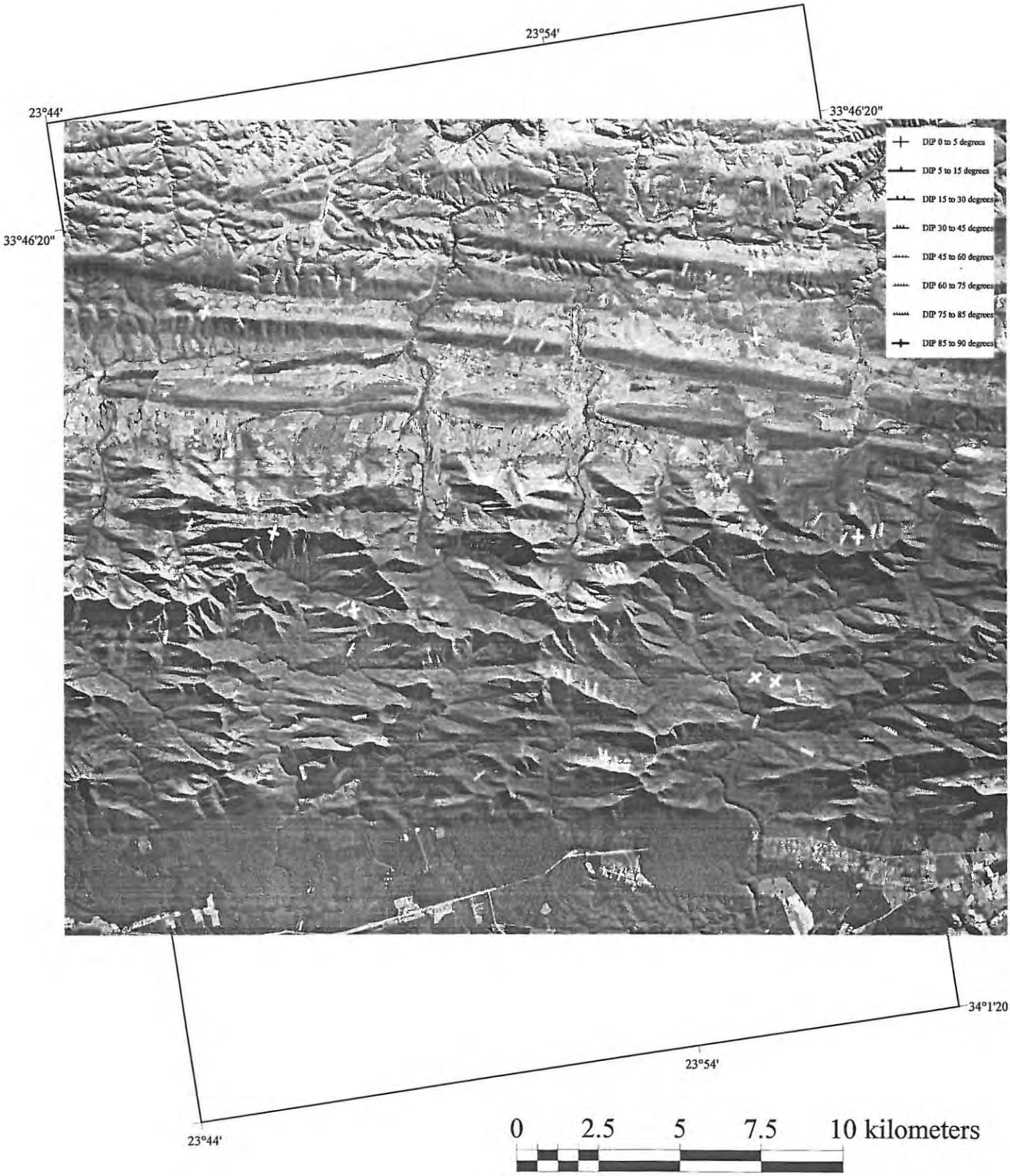


Fig. 4x. S_3 plane orientations for the Kareedouw Case Study, 3324CA. (Data derived by GEOSTRUC©)

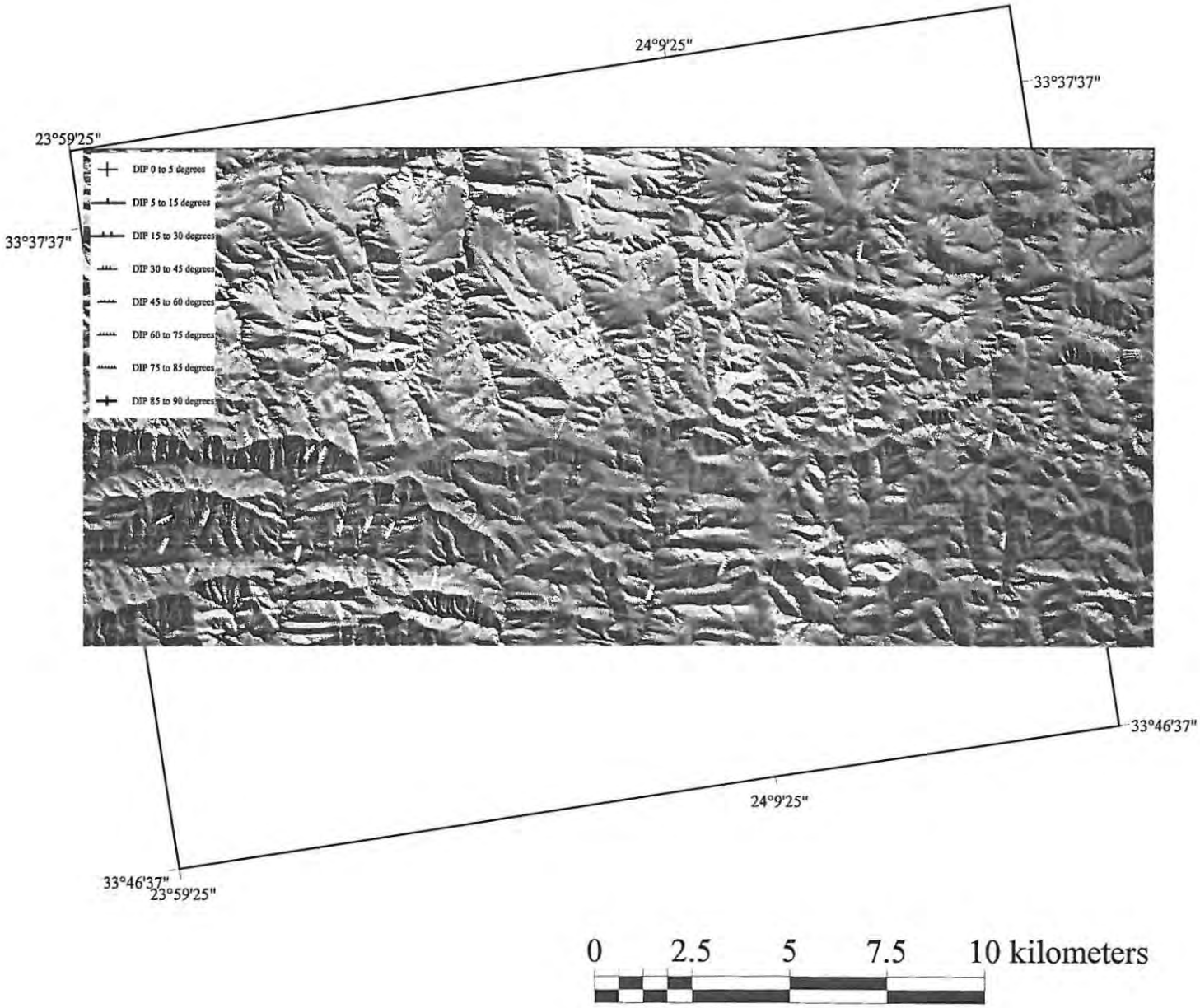
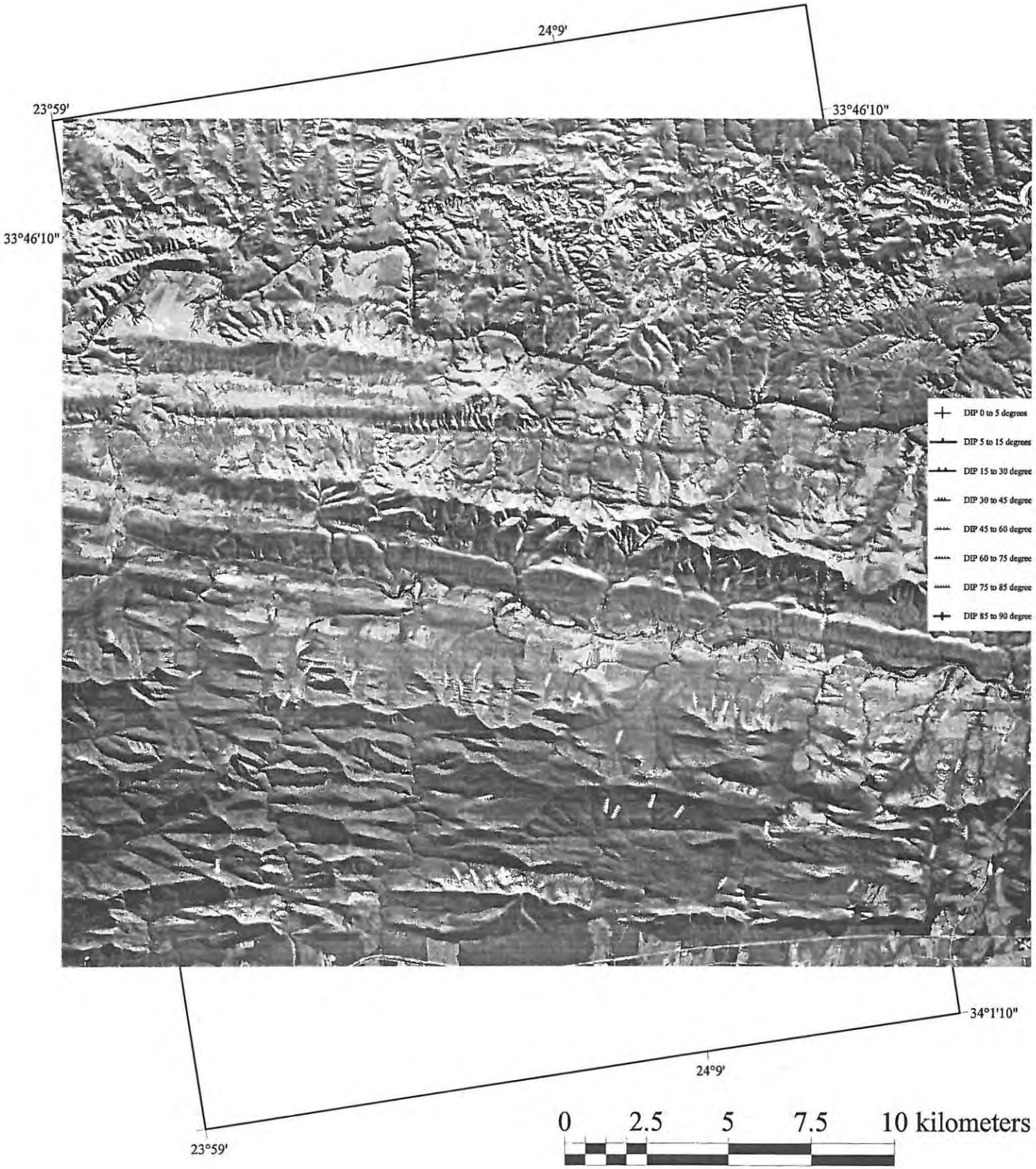


Fig. 4y. S_3 plane orientations for the Kareedouw Case Study, 3324CC. (Data derived by GEOSTRUC©)



The accompanying stereonet for the GEOSTRUC© bedding orientation measurements (**Fig. 4z.**) shows the dominant strike of the bedding in the area to be WNW–ESE. Eigenvector analysis (Woodcock, 1977) of the data (**Fig. 4aa.**) gives a value of 0.194 for the K fabric parameter. The poles to the S_0 planes are, thus, distributed in a girdle–like manner. The best–fit girdle to the poles of the S_0 planes, also depicted on the stereonet, strikes 012/90 with a pole at 102/00. Fieldwork done in the region (Sephton, 1995) is presented in **Fig. 4ab.** together with eigenvector statistics for this data (**Fig. 4ac.**)

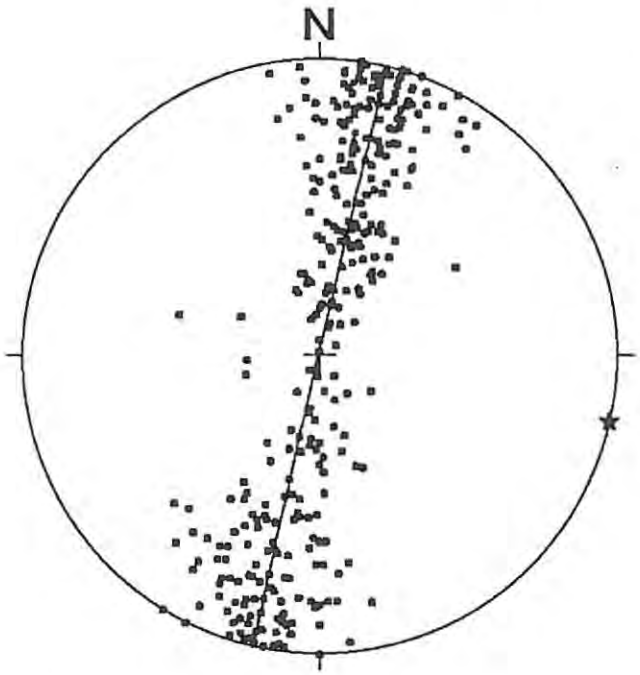


Fig. 4z. S_0 planes for the study area showing the best–fit girdle and pole. Square icons mark poles to S_0 planes. The star icon marks the pole to the best–fit girdle representing the regional fold axis orientation. Sample Size = 348. (Data derived by GEOSTRUC©)

The K fabric parameter for Sephton's (*opp. cit.*) data is 0.754, indicating a girdle–like distribution for the poles. The best–fit girdle strikes 044/76 with a pole at 134/14. The pole derived from Sephton's fieldwork differs by 32 degrees in dip direction and four degrees in dip amount. The difference may be due to the fieldwork not having been performed in exactly the same region. Contoured stereonets for the GEOSTRUC© derived data (**Fig. 4ad.**) and the fieldwork data (**Fig. 4ae.**) are also given.

The GEOSTRUC© derived contoured stereonet indicates limbs dipping steeply to the

```
                EIGENVECTORS
        l       m       n       Strike   Dip
V1=   0.957   0.220   0.188    12.927  10.848
V2=  -0.182  -0.049   0.982   195.135  79.144
V3=  -0.225   0.974   0.007   103.005   0.408

                EIGENVALUES
Lambda1=214.914   S1=   0.618
Lambda2=125.348   S2=   0.360
Lambda3=  7.738   S3=   0.022

S1/S2=   1.715
S2/S3=  16.200
S1/S3=  27.775

Ln(S1/S2)=   0.539   Ln(S2/S3)=   2.785

C=   3.324
K=   0.194
N=348
```

Fig. 4aa. Eigenvector statistics determined for the distribution of S_0 poles. (Data derived by GEOSTRUC©)

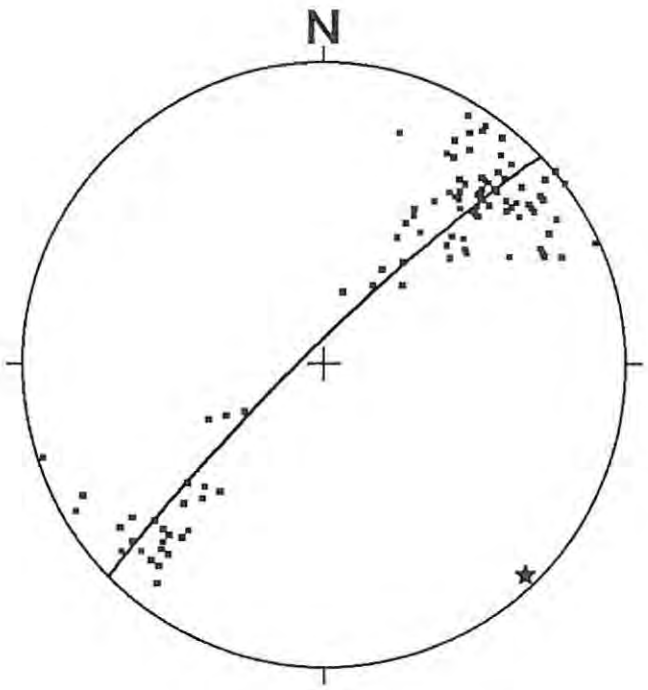


Fig. 4ab. S_0 planes for the study area showing the best–fit girdle and pole. Square icons mark poles to S_0 planes. The star icon marks the pole to the best–fit girdle representing the regional fold axis orientation. Sample Size = 117. (Sephton (1995) fieldwork data)

north and limbs dipping steeply to the south. Furthermore, a second maximum on the

EIGENVECTORS

	l	m	n	Strike	Dip
V1=	0.688	0.682	0.246	44.748	14.258
V2=	-0.112	-0.236	0.965	244.650	74.876
V3=	-0.717	0.692	0.086	136.006	4.938

EIGENVALUES

Lambda1= 95.244 S1= 0.814
Lambda2= 19.403 S2= 0.166
Lambda3= 2.352 S3= 0.020

S1/S2= 4.909
S2/S3= 8.248
S1/S3= 40.487

Ln(S1/S2)= 1.591 Ln(S2/S3)= 2.110

C= 3.701
K= 0.754
N=117

Fig. 4ac. Eigenvector statistics determined for the distribution of S_{0-1} poles. (Sephton (1995) fieldwork data)

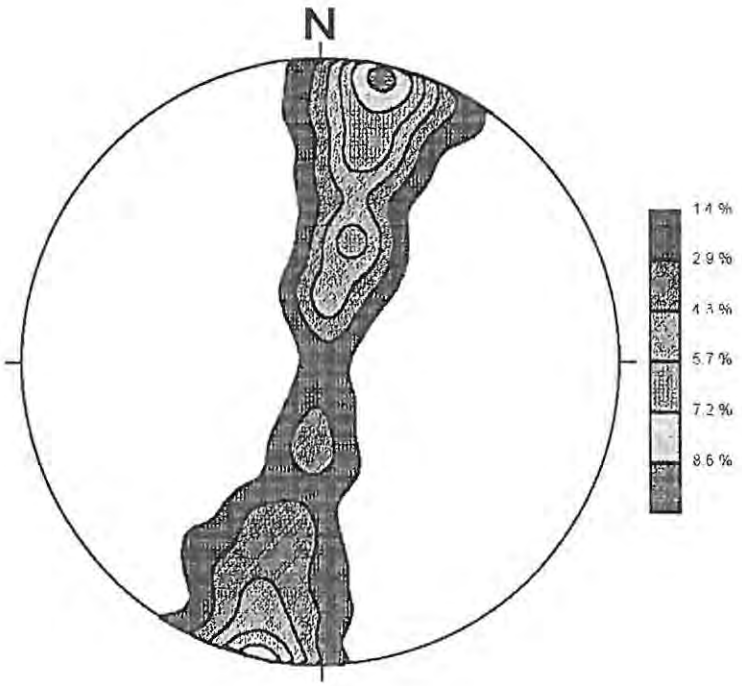


Fig. 4ad. Contoured stereonet of S_0 poles. (Data derived by GEOSTRUC©)

stereonet shows limbs dipping 40 degrees and less, to the north. The contoured stereonet for Sephton's (*opp. cit.*) data shows limbs dipping steeply to the north–east and between 80–45 degrees to the south–west.

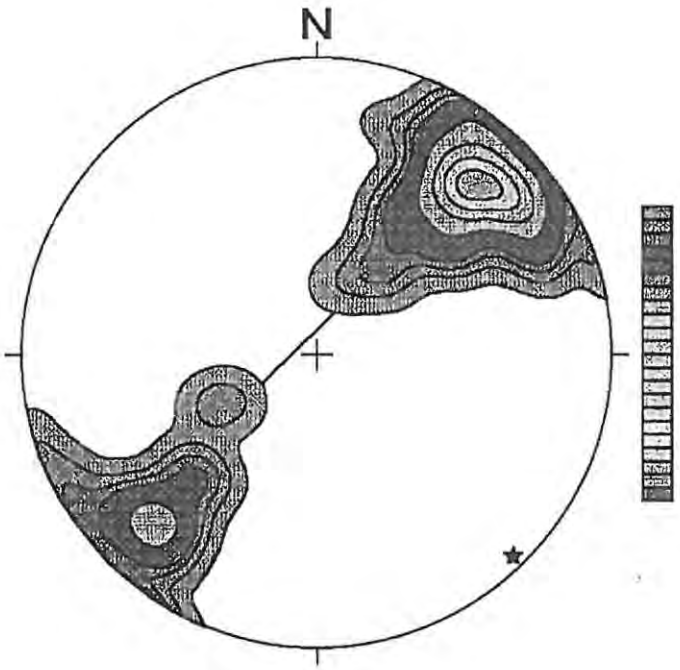


Fig. 4ae. Contoured stereonet of S_0 poles. (Sephton (1995) fieldwork data)

Axial plane (S_2) orientations for the area were also measured using the GEOSTRUC© routine. The results are shown in **Fig. 4af.** Note the close correlation between the best–fit pole to the S_0 planes (**Fig. 4z.**) and the orientation of the GEOSTRUC© derived axial plane measurements (**Fig. 4af.**). The distribution of S_2 planes in this stereonet indicates that the axial planes strike WNW–ESE. The axial planes vary in dip direction and dip from 020/45 to 200/45.

The stereonet depicting fracture orientations for the study area, as derived by GEOSTRUC© , is shown in **Fig. 4ag.** and data from fieldwork is shown in **Fig. 4ah.**

The eigenvector analysis on the GEOSTRUC© derived data is shown in **Fig. 4ai.** and the eigenvector analysis of the fieldwork data is shown in **Fig. 4aj.**

The eigenvector analysis for the GEOSTRUC© data gives a K fabric parameter of 0.275 which indicates a girdle–like data distribution. This is possibly due to the relatively large area covered by the study, thus introducing joint and fracture sets with varying dip. Eigenvector analysis of the fieldwork data indicates a cluster–like distribution of the data with mean girdle at 135/86. This result reflects a strongly developed joint set that is possibly one of a conjugate pair that is commonly associated with folded strata (Davis,

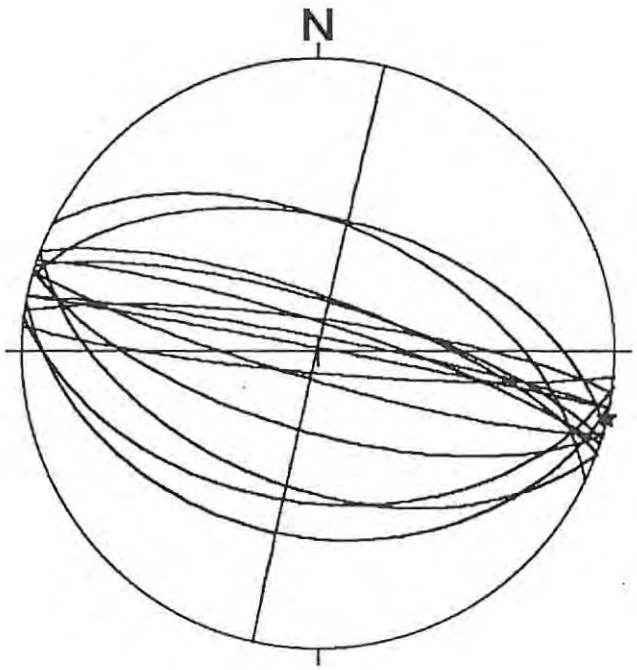


Fig. 4af. S_2 planes with best–fit girdle to the S_0 poles plotted. (Data derived by GEOSTRUC©)

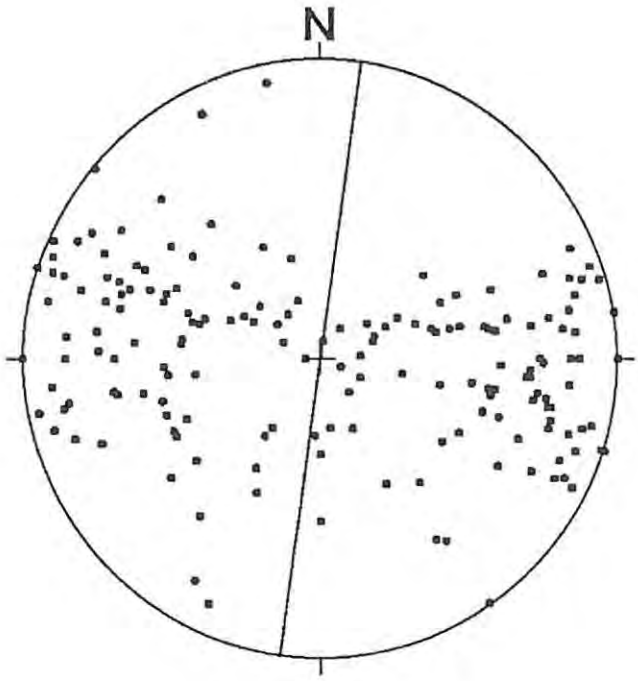


Fig. 4ag. S_3 poles for the study area showing the great circle mean. Sample Size = 175. (Data derived by GEOSTRUC©)

1984). A conjugate pair of joint sets commonly occurs with a symmetrical distribution about the axial plane of a fold. Davis (1984) states that oblique joints are classically

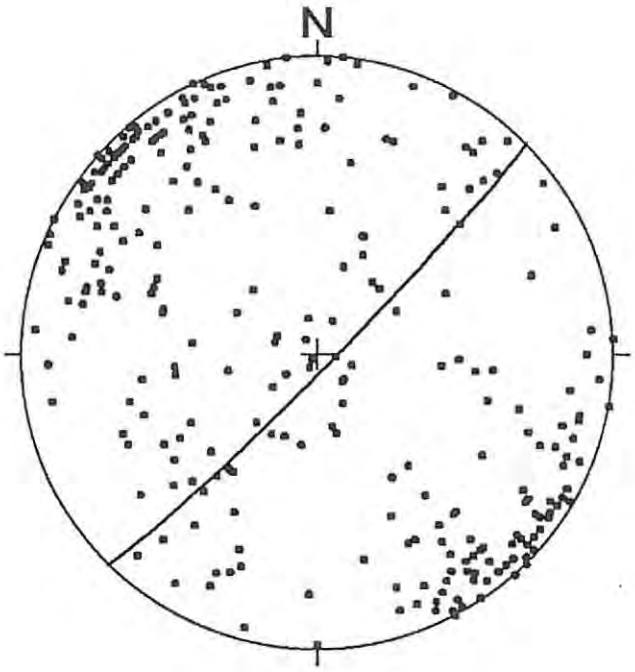


Fig. 4ah. S_3 poles for the study area showing the great circle mean. Sample Size = 305. (Sephton (1995) fieldwork data)

EIGENVECTORS

	l	m	n	Strike	Dip
V1=	-0.131	0.991	0.001	97.548	0.075
V2=	0.002	-0.001	1.000	327.205	89.885
V3=	-0.991	-0.131	0.002	187.548	0.088

EIGENVALUES

Lambda1= 92.809 S1= 0.530
Lambda2= 64.741 S2= 0.370
Lambda3= 17.450 S3= 0.100

S1/S2= 1.434
S2/S3= 3.710
S1/S3= 5.319

Ln(S1/S2)= 0.360 Ln(S2/S3)= 1.311

C= 1.671
K= 0.275
N=175

Fig. 4ai. Eigenvector statistics determined for the distribution of S_3 poles. (Data derived by GEOSTRUC©)

interpreted as conjugate shear joints that form in folded layers as a response to shortening perpendicular to the axial surface of the fold. Thus, this result supports the deformation model as suggested in section 4.2.2.2. A comparison of the contoured stereonets of the

```
                EIGENVECTORS
        l       m       n       Strike  Dip
V1=   0.708  -0.703   0.067   315.196   3.815
V2=  -0.554  -0.494   0.670   221.745  42.074
V3=   0.438   0.511   0.739    49.395  47.671

                EIGENVALUES
Lambda1=183.143   S1=   0.600
Lambda2= 65.650   S2=   0.215
Lambda3= 56.207   S3=   0.184

S1/S2=   2.790
S2/S3=   1.168
S1/S3=   3.258

Ln(S1/S2)=   1.026   Ln(S2/S3)=   0.155

C=   1.181
K=   6.606
N=305
```

Fig. 4aj. Eigenvector statistics determined for the distribution of S_3 poles. (Sephton (1995) fieldwork data)

poles to the S_3 planes for the GEOSTRUC© derived data (**Fig. 4ak.**) and the fieldwork data (**Fig. 4al.**) shows a similar distribution with a rotated difference (possibly due to different areas). The rotated difference in the strike of the joint planes is approximately 40 degrees. The coverage by the GEOSTRUC© routine of a larger area than the fieldwork performed by Sephton (1995) could be responsible for this difference.

Three thickness measurements on the Kouga Formation were performed. The GEOSTRUC© results were 450m on the northern limb of a syncline and 199m and 144m on the southern limb of the same syncline. See Appendix E, section E.2.3., for the lat/long coordinates of these measurements which can be found in **Fig. 4o.** The thickness, estimated by SACS (1980), for the Kouga Formation is 380m.

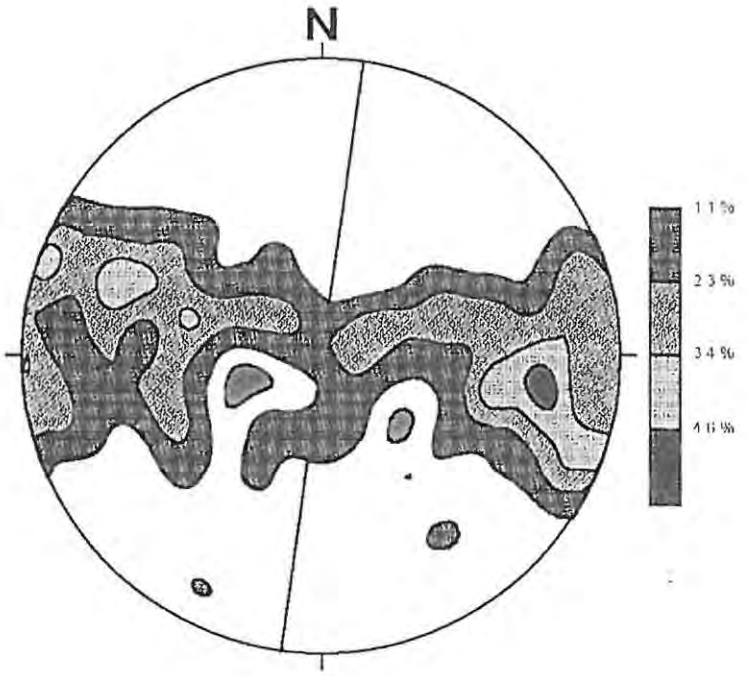


Fig. 4ak. Contoured stereonet of S_3 poles with mode at 098/90. (Data derived by GEOSTRUC©)

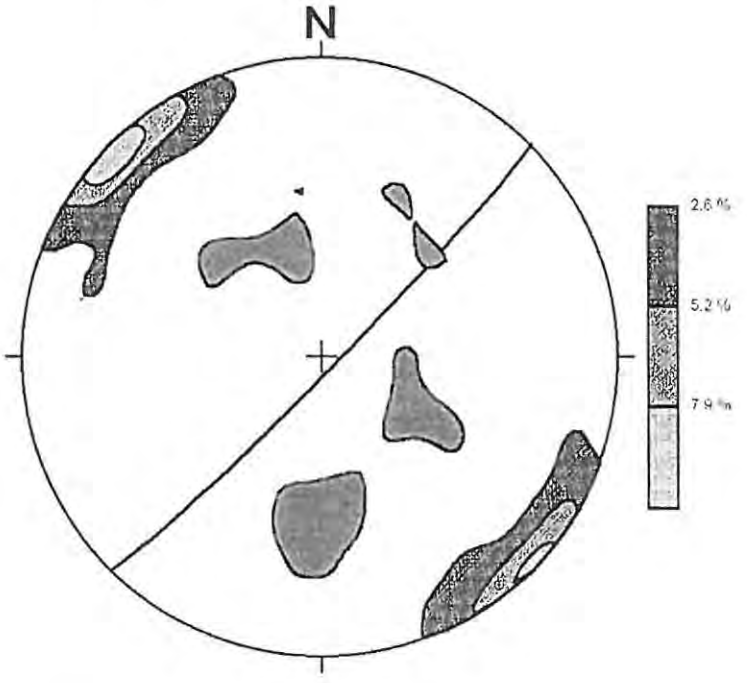


Fig. 4al. Contoured stereonet of S_3 poles with mode at 135/86. (Sephton (1995) fieldwork data)

CHAPTER FIVE

DISCUSSION AND CONCLUSION

5.1. Discussion

5.1.1. The GEOSTRUC© Program.

The GEOSTRUC© analysis routine provides the user with a fast and effective tool to determine, on a regional scale, the orientation of subsurface geological structures.

The interpretation and recognition of geological patterns on digital imagery are assigned to the operator, so that the operator has complete control over the selection of points on geological stuctures which will be used in the calculation of their orientation. An operator also has the option of accepting or rejecting an orientation result depending on his/her judgement and geological experience. An operator, therefore, needs to be skilled in geological photo-interpretation and familiar with the regional geology of an area being examined.

The GEOSTRUC© routine is user-friendly. Utilizing a windows graphics interface, together with a pointing device, the routine minimizes keyboard usage and it is easy to navigate through the system of menus, submenus and subroutines.

The use of X Windows (Section 2.1.) as an operating environment makes the GEOSTRUC© routine a fully portable analysis package. Using a PC-based X server, the package could be operated during fieldwork. Furthermore, future improvement of the X Windows System could be used for upgrading the speed and efficiency of the GEOSTRUC© routine.

Source materials of various kinds, such as digital geological images and DEMs, may be used in the routine. The construction of the GEOSTRUC© routine using the SDK for TNTmips, supplied by MicroImages, has made this possible since TNTmips contains all the necessary functionality for the manipulation and generation of digital geological images and DEMs from a broad spectrum of sources. The GEOSTRUC© routine is, thus,

an add–on feature to TNTmips.

The ASCII data file, generated during operation of the GEOSTRUC© process, contains the necessary orientation and location data which is accessible to both the TNTmips database construction facility and the MS–Windows based program StereoNet (as illustrated in sections 4.1.5. and 4.2.4.) and can be imported into any standard spreadsheet or database program.

5.1.2. GEOSTRUC© Application.

The two program applications performed (Sections 4.1.& 4.2.) illustrate the use of the GEOSTRUC© routine in two geographically distinct areas. The two areas differ significantly in structural complexity and geomorphology, thus enabling the evaluation of program performance over a range of geological settings.

5.1.2.1. Source Material.

The DEMs used in the studies were derived by different techniques from different kinds of source materials. The Alicedale case history (Section 4.1.) used a DEM at 1.47m cell resolution derived from digitized topographic contours. The Kareedouw case history (Section 4.2.) used a DEM at 10m cell resolution, resampled from the original 200m cell resolution of the National Digital Elevation Model format and purchased from the Surveyor General's Office.

The cell resolution of the SPOT image used in the Kareedouw case history was 10m. Note that resampling the accompanying DEM to a cell resolution less than 10m would have been unnecessary and would not have improved the accuracy of the results in the Kareedouw case history.

When using aerial photographs as a source for the generation of digital geological images, the resultant image resolution is dependent on the scale of the photogaphy and the resolution of the scanner used in the hardcopy–to–digital conversion process. The accompanying DEM can, thus, be of a resolution less than 10m for more accurate results.

5.1.2.2. Usefulness of the GEOSTRUC© Routine.

GEOSTRUC© has proved to be a cost–effective way to assess the regional (or large–area) geology of an area especially in cases where the area consists of rugged, inaccessible terrain. For example, the cost in terms of manpower and time involved in collecting data

for the Kareedouw case history by fieldwork is far greater than the equivalent cost of employing the GEOSTRUC© routine. The actual data collection for the case history was completed within two days.

The determination of the orientation of large–scale structures is much easier and reliable through the GEOSTRUC© routine. For example, field measurements of the orientation of large faults are often unreliable as measurements on small–scale variations in the fault plane are not a true reflection of the actual regional orientation of the fault plane. In this case GEOSTRUC© provides more accurate results because of the "bird's eye" perspective.

Areas with complex structural geology can be deciphered in much less time than field measurements using the GEOSTRUC© routine. A GEOSTRUC© operator can decide, during operation, that a certain area with complex structures needs more measurements, thus ensuring an easier analysis of that particular area. This flexibility is not a feature present during fieldwork.

The optional thickness calculation gives the GEOSTRUC© user immediate access to bedding thickness during area analysis. This is another advantage over fieldwork.

5.1.2.3. Axial Plane Orientation.

A powerful feature of this routine that came to light during the application phase was its ability to directly measure the orientation of axial planes of large–scale folds. By simply selecting three points that lie on hinge points along the axial plane trace of a fold the orientation may be determined. A comparable field technique would be to take three GPS readings at three places along the axial plane trace of a fold. This kind of fieldwork would usually be time–consuming and often impossible as certain hinge points might be totally inaccessible. The GEOSTRUC© routine thus has a powerful additional application for calculating the orientations of axial planes.

5.1.2.4. Problem Areas.

Usage of the GEOSTRUC© routine in areas with little variation in relief may lead to inaccurate results, because the three points selected for a calculation will approach colinearity which, in turn, affects the plane calculation (three points in a straight line define an infinite number of planes). Thus, in these areas, the plane fitting procedure

will not function properly.

Areas where traces of geological structures are indistinct, due to either lack of topographic expression or dense vegetation cover, will not be suitable for analysis with the GEOSTRUC© process as locating the 3–point cluster is impossible.

Finally, the principle of scale of structure with respect to scale of topography is mentioned. If the occurrence of geological structures is on a scale smaller than the rate of topographical variation the GEOSTRUC© process will not be able to resolve the trace of the structure, hence, making the measurement of orientation impossible.

5.2. Conclusion

5.2.1. The GEOSTRUC© Routine.

The GEOSTRUC© routine provides geologists with a tool to analyze the orientation of geological structures quickly and effectively. The method operates on a wide variety of source materials and can be used on a regional as well as a local scale.

5.2.2. Further Development.

Possible future modifications and enhancements to the GEOSTRUC© routine could be;

(1) Enabling the user to select several 3–point clusters on one structure trace, then calculating the average orientation of the structure, in this way improving the accuracy of results.

(2) Adding automatic stereonet construction routines to the menu options. This stereonet option could be engineered such that it is displayed in real–time within a stereonet window that is updated each time an orientation reading is accepted.

(3) Constructing semi–automated geological cross–sections.

Advancements in pattern recognition methods and software could greatly benefit the GEOSTRUC© routine in the following way; pattern recognition methods would be used to extract the trace of geological structures, such as bedding planes, regional joint sets, faults and axial planes, from digital imagery. The pattern recognition method would probably make use of existing geological information of the area under investigation (some kind of expert system) and advanced mathematical techniques in image processing (fuzzy

mathematics). The procedure would determine the location of geological structures on the image. Once determined, the GEOSTRUC© routine would provide a way to determine its orientation (currently, the operator acts as the pattern recognition tool). In this way pattern recognition methodology could turn the GEOSTRUC© routine into a fully automated analysis package.

Inaccuracies in the elevation values of DEMs create errors in the 3–point calculation stage of the GEOSTRUC© process. Improvements in the elevation accuracy of DEMs, either by new, direct acquisition (radar interferometry) methods or advanced software generation (stereo matching) techniques, will increase the reliability of results produced by GEOSTRUC© . These improvements will hopefully effect the cost of DEM production positively, thus reducing the overall cost of the GEOSTRUC© process.

Finally, as satellite imaging technology improves, digital images with less noise and better ground cell resolution will become available. Less noise in an original image, acquired by a satellite, means that a greater amount of information is available in the original image, hence, the image will have a greater accuracy. This will make the recognition of trace patterns easier and can only improve the positioning of 3–point clusters on geological structures, and, hence, the accuracy of results in the GEOSTRUC© routine.

References

Barkakati, Nabajyoti, 1991, *X Window System Programming*, SAMS - Macmillan Computer Publishing., Carmel, Indiana., pp. 409–521.

Bates, R.L., and Jackson, J.A., 1984, *Dictionary of Geological Terms, 3rd ed.*, Anchor Press/Doubleday, New York, p. 521.

Briggs, I.C., 1974, *Machine contouring using minimum curvature*, Geophysics **39(1)**, 39–48.

Burden, R.L., & Faires, J.D., 1988, *Numerical Analysis, 3rd Edition*, Prindle, Weber & Schmidt, p. 429.

Burrough, P. A., 1986, *Principles of GIS Systems for Land Resource Assessment.*, Clarendon Press, Oxford, pp. 1–11.

CCNLIS, 1990, *National Standard for the Exchange of Digital Geo–Referenced Information*, Version 2, Standards Committee of the Co–ordinating Committee for the National Land Information System, Mowbray, South Africa, pp. 9-11, 9-23.

Chorowicz, J., Breard, J.–Y., Guillande, R., Morasse, C.–R., Prudon, D., and Rudant, J.–P., 1991, *Dip and Strike Measured Systematically on Digitized Three–Dimensional Geological Maps*, Photogrammetric Engineering & Remote Sensing **57(4)**, 431–436.

Cowen, D.J., 1988, *GIS versus CAD versus DBMS: What Are the Differences?*, Photogrammetric Engineering & Remote Sensing **54(11)**, 1551–1555.

Davis, G.H., 1984, *Structural Geology of Rocks and Regions*, John Wiley & Sons, New York, p. 389.

Day, T., and Muller, J.–P. A. L., 1988, *Quality Assessment of Digital Elevation Models Produced by Automatic Stereomatchers from SPOT Image Pairs.*, Photogrammetric Record **12(72)**, 797–808.

Drury, S.A., 1987a, *Image interpretation in geology.*, Chapman & Hall, London, pp. 136–140.

Drury, S.A., 1987b, *Image interpretation in geology.*, Chapman & Hall, London, p. 212.

Ghormley, Keith, 1993a, *Software Development Kit : Application Note*, MicroImages, Inc., Lincoln, Nebraska.

Ghormley, Keith, 1993b, *Orthoimage and DEM Creation*, Application Note for TNTmips, MicroImages Inc., Lincoln, Nebraska, pp. 1–40.

Grimaud, P., Richert, J.–P., Rolet, J., Tiercelin, J.–J., Xavier, J–P., Morley, C. K., Coussement, C., Karanja, S. W., Renaut, R. W., Guerin, G., Le Turdu, C., and M.–N., Gerard., 1994, *Geometrie des failles et mecanismes de l'extension dans le Rift du Kenya, Afrique de l'Est. Une approche 3D par teledetection.*, Bull. Centres Rech. Explor.–Prod. Elf Aquitaine **18**.

Hälbich, I.W., 1983, *A tectogenesis of the Cape Fold Belt*, Geodynamics of the Cape Fold Belt (Söhnge, A.P.G., and Hälbich, I.W., eds.), Special Publ., No. 12, Geol. Soc. of S. Afr., Johannesburg, pp. 165–176.

Hälbich, I.W., 1992, *The Cape Fold Belt Orogeny: State of the art 1970's – 1980's*, Inversion tectonics of the Cape Fold Belt, Karoo and Cretaceous Basins of Southern Africa (De Wit, M.J., Ransome, I.G.D., eds.), Balkema, Rotterdam, pp. 141–158.

Hobbs B.E., Means, W.D., & Williams, P.F., 1976, *An outline of structural geology*, John Wiley & Sons, New York, p. 294.

Horton, R.E., 1923, *Rainfall interpolation*, Mon. Wea. Rev. **51(6)**, 291–304.

Lang, Serge, 1983a, *Linear Algebra, 2nd Edition*, Addison–Wesley Publishing Company, Reading, Massachusetts, pp. 182–185.

Lang, Serge, 1983b, *Linear Algebra, 2nd Edition*, Addison–Wesley Publishing Company, Reading, Massachusetts, p. 209.

Lemmens, M.J.P.M., 1988, *A survey on stereo matching techniques*, Int. Arch. Photogramm. Rem. Sensing **27(B8)**, V11–V23.

Lillesand, T.M., and Kiefer, R.W., 1987, *Remote Sensing and Image Interpretation, 2nd ed.*, John Wiley & Sons, New York, pp. 581–586.

Maguire , D.J., 1991, *An Overview and Definition of GIS*, Geographical Information Systems Volume I : Principles (Maguire, D.J., Goodchild, M.F., Rhind, D.W., eds.), Longman, London, pp. 9–21.

Mahon, Michelle J., and North, Colin P., 1993, *Three–Dimensional Integration of Remotely Sensed Geological Data: A Methodology for Petroleum Exploration.*, Photogrammetric Engineering & Remote Sensing **59(8)**, 1251–1256.

Marble, Duane F., 1990, *Geographic Information Systems: an overview.*, Introductory Readings in GIS. (D. J. Reuquet and D. F. Marble, eds.), Taylor & Francis Ltd., pp. 8–17.

MicroImages, Inc., 1993, *TNTmips V4.40 Documentation: Volume 2 Display.*, MicroImages Press, Lincoln, Nebraska., pp. 14–15.

Millad, M., 1993, *A comparison of field– and digital elevation model–derived geometrical analysis of a test area in the Cape Fold Belt near Alicedale, South Africa*, Unpubl. Hons. Thesis, Rhodes Univ., 1–37.

Morris, Kevin, 1991, *Using Knowledge–Base Rules to Map the Three–Dimensional Nature of Geological Features*, Photogrammetric Engineering & Remote Sensing **57(9)**, 1209–1216.

Nye, Adrian, 1990, *X Protocol Reference Manual*, The Definitive Guides to the X Window System, Volume Zero, O'Reilly & Associates, Inc., Sebastopol CA, pp. 3–9.

Nye, Adrian, 1992, *Xlib Programming Manual*, The Definitive Guides to the X Window System, Volume One, O'Reilly & Associates, Inc., Sebastopol CA, pp. 3–12.

Nye, Adrian and O'Reilly, Tim, 1990, *X Toolkit Intrinsics Programming Manual (OSF/Motif edition)*, The Definitive Guides to the X Window System, Volume Four, O'Reilly & Associates, Inc., Sebastopol CA, pp. 3–60.

Reynes, P., Rolet, J., Richert, J.–P., Gruneisen, P., Palengat, J.–M., and Coquelet, D., 1993, *Apports des techniques 3D de la teledetection dans la recherche des blocs bascules du fosse Nord Tanganyika, Rift est–africain, Zaire.*, Bull. Centres Rech. Explor.–Prod. Elf Aquitaine **17**.

SACS (South African Committee for Stratigraphy), 1980, *Stratigraphy of South Africa. Part 1 (Comp. Kent, L.E.): Lithostratigraphy of the Republic of South Africa, South West Africa/Namibia, and the republics of Bophutatswana, Transkei and Venda*, Handb. Geol. surv. S. Afr. **8**, 517,521.

Sasowsky, Kathryn C., Petersen, Gary W., and Evans, Barry M., 1992, *Accuracy of SPOT Digital Elevation Model and Derivatives: Utility for Alaska's North Slope.*, Photogrammetric Engineering & Remote Sensing **58(6)**, 815–824.

Schildt, Herbert, 1987, *C: The Complete Reference*, Osborne McGraw–Hill, Berkeley CA, pp. 733–753.

Sephton, Ross, 1995, *Unfinished MSc. Thesis*, Geology Dept. University of Port Elizbeth.

Simard, R., Toutin, T., Leclerc, A., Haja, S. R., Allam, M., Boudreau, R., and Slaney, R., 1988, *Digital Terrain Modelling with SPOT Data and Geological Applications.*, Techniques Spatiales **1**, 1205–1212.

Smith, Jerry D., 1991, *Object–Oriented Programming with the X Window System Toolkits*, Wiley Professional Computing, John Wiley & Sons, Inc., New York, pp. 9–24.

SPOT Image, 1990, *The SPOT Standard CCT Format*, SPOT Image, Toulouse, France.

Tankard, A.J., Jackson, M.P.A., Erikson, K.A., Hobday, D.K., Hunter, D.R., Minter, W.E.L., 1982, *Crustal Evolution of Southern Africa*, Springer–Verlag, New York, pp. 333–363.

Tello, E.R., 1991, *Object–Oriented Programming for Windows*, Wiley Professional Computing, John Wiley & Sons, Inc., New York, p. 31.

Theron, J.N., 1969, *The Baviaanskloof Range – a South African nappé*, Trans. Geol. Soc. S. Afr. **72**, 29–40.

Truswell, J.F., 1977, *The Geological Evolution of South Africa*, Purnell, Cape Town, p. 119.

Wadge, G., Cross, A.M., Angelikaki, C., 1990, *Semi–automatic structural mapping in arid terrain from remotely sensed images.*, Remote Sensing: An Operational Technology for the Mining and Petroleum Industries, Institute of Mining and Metallurgy, London, pp. 103–110.

Ward, P.T., & Mellor, S.J., 1985, *Structured Development for Real–Time Systems, Volume One: Introduction & Tools*, Yourdon Press Computing Series, Prentice–Hall, Englewood Cliffs, New Jersey, pp. 3–39.

Watcom International Corp., 1994, *Watcom C/C++ Getting Started.*, Watcom Int. Corp., Ontario, Canada..

Watson, David F., 1992a, *Contouring: A guide to the analysis and display of spatial data.*, Computer Methods in the Geosciences Vol. 10, Pergamon Press, Oxford, pp. 101–159.

Watson, David F., 1992b, *Contouring: A guide to the analysis and display of spatial data*, Computer Methods in the Geosciences Vol. 10, Pergamon Press, Oxford, p. 122.

Weibel, R., and Heller, M., 1991, *Digital Terrain Modelling*, Geographical Information Systems Volume I : Principles (Maguire, D.J., Goodchild, M.F., Rhind, D.W., eds.), Longman, London, pp. 269–297.

Wolf, P.R., 1974, *Elements of Photogrammetry*, McGraw–Hill, Inc., New York, pp. 143–175.

Woodcock, N.H., 1977, *Specification of fabric shapes using an eigenvalue method*, Bull. Geol. Soc. Am. **88**, 1231–1236.

APPENDIX A

LINK FILE USED IN BUILDING THE PROCESS

The following options were used in the geostruc.lnk file to link the custom program GEOSTRUC© ;

(1) `file geostruc.obj` - Specifies the object file to be linked.

(2) `system win386` - Target operating environment.

(3) `name geostruc` - Name of executable file.

(4) `option mindata=256K` - Specifies the minimum number of bytes, in addition to the memory required by the executable, that must be allocated by the 386—DOS–Extender at the end of the executable.

(5) `option maxdata=512K` - Maximum of the mindata variable.

(6) `option stack=32K` - Increase the default stack size.

(7) `option c` - Language type used.

(8) `libfile cle` - Names of object files that are viewed as components of a library.

(9) `libpath m:\usr\limited\win\program\watcom\lib386\tntlib` - Path to SDK run–time libraries.

(10) `libpath m:\usr\limited\win\program\watcom\lib386\win` - Path to Windows 32-bit libraries.

(11) `library local,mixlib,xlib,xm,xt,xmu,xext,mipslib,rvclib` - SDK library names.

(12) `option map` - Compile a memory map and log it in a file.

APPENDIX B

GEOSTRUC SOURCE CODE LISTING

```
#include <mi32/stdincls.h>                                          0001
#include <mi32/rvcdefns.h>                                          0002
#include <mi32/xdefns.h>                                            0003
#include <mi32/menuitem.h>                                          0004
#include <mi32/dispwind.h>                                          0005
#include <mi32/trans2d.h>                                           0006

#include <Xm/Form.h>                                                0007
#include <Xm/Label.h>                                               0008
#include <Xm/Text.h>                                                0009
#include <Xm/Separator.h>                                           0010
#include <Xm/PushB.h>                                               0011
#include <Xm/ToggleB.h>                                             0012

#include <math.h>                                                   0013

extern XtAppContext appcontext;                                     0014

typedef struct {                                                    0015
     int IsWindowActive,HasCoords,Is2WindowActive;                  0016
     char ifilename[256];                                           0017
     char iDTMfilename[256];                                        0018
     char Vecfilename[256];                                         0019
     char ofilename[256];                                           0020
     long DTMrastinode;                                             0021
     long rastinode;                                                0022
     long Vecinode;                                                 0023
     long NumLabels;                                                0024
     long triangleNum;                                              0025
     int vecfhandle;                                                0026
     int vid;                                                       0027
     int ThickSaveActive;                                           0028
     double thickness;                                              0029
     float labelheight;                                             0030
     RVCRASTINFO rastinfo;                                          0031
     RVCRASTINFO DTMrastinfo;                                       0032
     RVCVECTINFO Vecinfo;                                           0033
     UBYTE hasinputraster;                                          0034
     UBYTE hasinputDTM;                                             0035
     UBYTE hasVoutput;                                              0036
     UBYTE hasoutput;                                               0037
     double point2[3],point3[3],point1[3];                          0038
     double THpoint2[3],THpoint1[3];                                0039
     double vpoint2[3],vpoint3[3],vpoint1[3];                       0040
     DPOINT3D vectorpt1,vectorpt2,vectorpt3;                        0041
     DPOINT3D center;                                               0042
     double planeCoeffs[5],planeOrientation[4];                     0043
```

```
        char planeClass[256];                                          0044
        RVCGEOREFINFO georefinfo;                                      0045
        RVCGEOREFINFO DTMgeorefinfo;                                   0046
        RVCGEOREFINFO vecgeorefinfo;                                   0047
        long georefinode;                                              0048
        long DTMgeorefinode;                                           0049
        long vecgeorefinode;                                           0050
        long PointNum;                                                 0051
        long LineNum;                                                  0052
        DWLAYER univectorlayer;                                        0053
        DWLAYER currentlayer;                                          0054
        DWWINDOW newwind;                                              0055
        DWGROUP newgroup;                                              0056
        DRECTXY rastextents;                                           0057
        MAPPROJPARM *rastprojparm;                                     0058
        VECTDISPPARM *vectdispparm;                                    0059
        Widget PBfastengage;                                           0060
        Widget toplevel;                                               0061
        Widget thirdLevel;                                             0062
        Widget fourthLevel;                                            0063
        Widget FifthLevel;                                             0064
        Widget SixLevel;                                               0065
        Widget TFVoutput;                                              0066
        Widget TFoutput;                                               0067
        Widget TFinput;                                                0068
        Widget TFdtmInput;                                             0069
        Widget PBinput;                                                0070
        Widget PBVoutput;                                              0071
        Widget PBoutput;                                               0072
        Widget PBdtmInput;                                             0073
        Widget ptXform[3];                                             0074
        Widget ptYform[3];                                             0075
        Widget ptZform[3];                                             0076
        Widget THptXform[2];                                           0077
        Widget THptYform[2];                                           0078
        Widget THptZform[2];                                           0079
        } DDATA;                                                       0080
/* Callback for accepting plane orientation and saving to *.geo file ****/ 0081

void static CB_PlaneDataPopupAccept(                                   0082
        Widget w,                                                      0083
        DDATA *ddp,                                                    0084
        XmAnyCallbackStruct *cbdata                                    0085
        ) {                                                            0086
        RVCVECTPOINT point;                                            0087
        RVCVECTPOINT vector[20];                                       0088
        RVCVECTLINE line;                                              0089
        RVCVECTLABEL label;                                            0090
        FILE *outfile;                                                 0091
        double minx,miny,maxx,maxy,temp[4],lblht;                      0092
        DPOINT3D basept;                                               0093
        OBJWINDOW *objwind;                                            0094
        int trianglenum;                                               0095
        char labelstring[30];                                          0096
```

```
memset(&line,0,sizeof(RVCVECTLINE));                                    0097

/* Reset input to ZERO */                                               0098
memset(ddp->point1,0,sizeof(ddp->point1));                              0099
memset(ddp->point2,0,sizeof(ddp->point2));                              0100
memset(ddp->point3,0,sizeof(ddp->point3));                              0101

XtPopdown(ddp->fourthLevel);                                            0102
if(ddp->Is2WindowActive)                                                0103
     {                                                                  0104
     XtPopdown(ddp->FifthLevel);                                        0105
     ddp->Is2WindowActive = 0;                                          0106
     }                                                                  0107

/* Write to data file */                                                0108
if(!(outfile = fopen(ddp->ofilename, "a+t")))                           0109
     { MxPopupMessageMt(ddp->toplevel,0,"geostruc","SaveGeoData",       0110
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0111
     }                                                                  0112
else                                                                    0113
     {                                                                  0114
     if(fprintf(outfile,"\n%d,%f,%f,%f,%3.0f,%3.0f,",                   0115
               ddp->vectdispparm->vectinfo.NumLabels,                   0116
               ddp->center.x,ddp->center.y,ddp->center.z,               0117
               ddp->planeOrientation[1],ddp->planeOrientation[2]) <= 0) 0118
          { MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoBytesWritten",0119
               NULL,XmDIALOG_ERROR | GET_NoCancel);                     0120
          }                                                             0121
     fprintf(outfile,"%s,",ddp->planeClass);                            0122
     if(ddp->ThickSaveActive)                                           0123
          {                                                             0124
          fprintf(outfile, "%5.0f",ddp->thickness);                     0125
          }                                                             0126
     else                                                               0127
          {                                                             0128
          fprintf(outfile,"%5.0f",0);                                   0129
          }                                                             0130
     }                                                                  0131
fclose(outfile);                                                        0132

memset(&vector,0,sizeof(vector));                                       0133

/* Point one */                                                         0134
basept.x = vector[3].x = vector[0].x = point.x = ddp->vectorpt1.x;      0135
basept.y = vector[3].y = vector[0].y = point.y = ddp->vectorpt1.y;      0136
basept.z = vector[3].z = vector[0].z = point.z = ddp->vectorpt1.z;      0137

if(point.x < ddp->vectdispparm->vectinfo.minval.x)                      0138
     ddp->vectdispparm->vectinfo.minval.x = point.x;                    0139
if(point.y < ddp->vectdispparm->vectinfo.minval.y)                      0140
     ddp->vectdispparm->vectinfo.minval.y = point.y;                    0141
if(point.z < ddp->vectdispparm->vectinfo.minval.z)                      0142
     ddp->vectdispparm->vectinfo.minval.z = point.z;                    0143
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)                      0144
     ddp->vectdispparm->vectinfo.maxval.x = point.x;                    0145
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)                      0146
     ddp->vectdispparm->vectinfo.maxval.y = point.y;                    0147
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)                      0148
```

```
     ddp->vectdispparm->vectinfo.maxval.z = point.z;                    0149

if(point.x < line.minval.x) line.minval.x = point.x;                    0150
if(point.y < line.minval.y) line.minval.y = point.y;                    0151
if(point.z < line.minval.z) line.minval.z = point.z;                    0152
if(point.x > line.maxval.x) line.maxval.x = point.x;                    0153
if(point.y > line.maxval.y) line.maxval.y = point.y;                    0154
if(point.z > line.maxval.z) line.maxval.z = point.z;                    0155

if ((MfWriteVPoint(ddp->vid,ddp->PointNum,&point)) < 0)                 0156
     {                                                                  0157
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",     0158
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0159
     }                                                                  0160

ddp->PointNum++;                                                        0161
ddp->vectdispparm->vectinfo.NumPoints++;                                0162

/* Point Two */                                                         0163
vector[1].x = point.x = ddp->vectorpt2.x;                               0164
vector[1].y = point.y = ddp->vectorpt2.y;                               0165
vector[1].z = point.z = ddp->vectorpt2.z;                               0166

if(point.x < ddp->vectdispparm->vectinfo.minval.x)                      0167
     ddp->vectdispparm->vectinfo.minval.x = point.x;                    0168
if(point.y < ddp->vectdispparm->vectinfo.minval.y)                      0169
     ddp->vectdispparm->vectinfo.minval.y = point.y;                    0170
if(point.z < ddp->vectdispparm->vectinfo.minval.z)                      0171
     ddp->vectdispparm->vectinfo.minval.z = point.z;                    0172
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)                      0173
     ddp->vectdispparm->vectinfo.maxval.x = point.x;                    0174
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)                      0175
     ddp->vectdispparm->vectinfo.maxval.y = point.y;                    0176
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)                      0177
     ddp->vectdispparm->vectinfo.maxval.z = point.z;                    0178

if(point.x < line.minval.x) line.minval.x = point.x;                    0179
if(point.y < line.minval.y) line.minval.y = point.y;                    0180
if(point.z < line.minval.z) line.minval.z = point.z;                    0181
if(point.x > line.maxval.x) line.maxval.x = point.x;                    0182
if(point.y > line.maxval.y) line.maxval.y = point.y;                    0183
if(point.z > line.maxval.z) line.maxval.z = point.z;                    0184

if ((MfWriteVPoint(ddp->vid,ddp->PointNum,&point)) < 0)                 0185
     {                                                                  0186
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",     0187
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0188
     }                                                                  0189

ddp->PointNum++;                                                        0190
ddp->vectdispparm->vectinfo.NumPoints++;                                0191

/* Point Three */                                                       0192
vector[2].x = point.x = ddp->vectorpt3.x;                               0193
vector[2].y = point.y = ddp->vectorpt3.y;                               0194
vector[2].z = point.z = ddp->vectorpt3.z;                               0195

if(point.x < ddp->vectdispparm->vectinfo.minval.x)                      0196
     ddp->vectdispparm->vectinfo.minval.x = point.x;                    0197
if(point.y < ddp->vectdispparm->vectinfo.minval.y)                      0198
```

```
     ddp->vectdispparm->vectinfo.minval.y = point.y;                    0199
if(point.z < ddp->vectdispparm->vectinfo.minval.z)                      0200
     ddp->vectdispparm->vectinfo.minval.z = point.z;                    0201
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)                      0202
     ddp->vectdispparm->vectinfo.maxval.x = point.x;                    0203
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)                      0204
     ddp->vectdispparm->vectinfo.maxval.y = point.y;                    0205
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)                      0206
     ddp->vectdispparm->vectinfo.maxval.z = point.z;                    0207

if(point.x < line.minval.x) line.minval.x = point.x;                    0208
if(point.y < line.minval.y) line.minval.y = point.y;                    0209
if(point.z < line.minval.z) line.minval.z = point.z;                    0210
if(point.x > line.maxval.x) line.maxval.x = point.x;                    0211
if(point.y > line.maxval.y) line.maxval.y = point.y;                    0212
if(point.z > line.maxval.z) line.maxval.z = point.z;                    0213

if ((MfWriteVPoint(ddp->vid,ddp->PointNum,&point)) < 0)                 0214
     {                                                                  0215
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",     0216
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0217
     }                                                                  0218

ddp->PointNum++;                                                        0219
ddp->vectdispparm->vectinfo.NumPoints++;                                0220

/* Write the triangle label */                                          0221
ltoa(ddp->vectdispparm->vectinfo.NumLabels, labelstring, 10);           0222

memset(&label,0,sizeof(label));                                         0223
label.desc.type = LABELDESC_Fixed;                                      0224
label.desc.height = (float)abs(vector[0].y - vector[2].y);              0225
label.desc.index = ddp->vectdispparm->vectinfo.NumLabels;               0226
label.LabelStr = labelstring;                                           0227
label.LabelStrSize = strlen(labelstring);                               0228
label.BasePts = &basept;                                                0229
label.NumBasePts = 1;                                                   0230
label.PointSize = sizeof(DPOINT3D);                                     0231

if(MfWriteVLabel(ddp->vid,ddp->NumLabels,&label) < 0)                   0232
     {                                                                  0233
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoLabelWritten",      0234
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0235
     }                                                                  0236

ddp->NumLabels++;                                                       0237
ddp->vectdispparm->vectinfo.NumLabels++;                                0238

/* Line */                                                              0239
line.left = line.right = line.start = line.end = -1;                    0240

if (MfWriteVLine(ddp->vid,ddp->LineNum,&line) < 0)                      0241
     {                                                                  0242
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVLinehdrWritten",   0243
          NULL,XmDIALOG_ERROR | GET_NoCancel);                          0244
     }                                                                  0245

if ((MfWriteVPoints(ddp->vid,ddp->LineNum,4,0L,4,&vector)) < 0)         0246
     {                                                                  0247
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVLineWritten",      0248
```

```
                NULL,XmDIALOG_ERROR | GET_NoCancel);                                    0249
        }                                                                               0250
    ddp->LineNum++;                                                                     0251
    ddp->vectdispparm->vectinfo.NumLines++;                                             0252
    /* Redisplay vector */                                                              0253
    objwind = MxdwGetLayerObjWind(ddp->univectorlayer);                                 0254
    objwind->ofullrect.xinit = ddp->vectdispparm->vectinfo.minval.x;                    0255
    objwind->ofullrect.yinit = ddp->vectdispparm->vectinfo.minval.y;                    0256
    objwind->ofullrect.xlast = ddp->vectdispparm->vectinfo.maxval.x;                    0257
    objwind->ofullrect.ylast = ddp->vectdispparm->vectinfo.maxval.y;                    0258
    /* Updata vector info subobject */                                                  0259
    MfWriteVectHeader(ddp->vectdispparm->fhandle,                                       0260
        ddp->vectdispparm->vectinfo.objectinode,&ddp->vectdispparm->vectinfo);          0261
    MfUpdateFile(ddp->vectdispparm->fhandle);                                           0262
    MxdwDisplayVector(ddp->univectorlayer);                                             0263
    }                                                                                   0264
/* Callback for setting "Write Thickness" toggle ****/                                 0265
static void set_thicksave(                                                              0266
    Widget w,                                                                           0267
    DDATA *ddp,                                                                         0268
    XmToggleButtonCallbackStruct *cbs                                                   0269
    ) {                                                                                 0270
    if(cbs->set) ddp->ThickSaveActive = 1;                                              0271
    else ddp->ThickSaveActive = 0;                                                      0272
    }                                                                                   0273
/* Callback for closing thickness result display ****/                                 0274
static void CB_AcceptThickness(                                                         0275
    Widget w,                                                                           0276
    DDATA *ddp,                                                                         0277
    XmAnyCallbackStruct *cbdata                                                         0278
    ) {                                                                                 0279
    XtPopdown(ddp->SixLevel);                                                           0280
    }                                                                                   0281
/* Callback for diplaying thickness calculation result ***/                            0282
static void CB_Accept2PointPopup(                                                       0283
    Widget w,                                                                           0284
    DDATA *ddp,                                                                         0285
    XmAnyCallbackStruct *cbdata                                                         0286
    ) {                                                                                 0287
    double thickness = 0;                                                               0288
    char string[100];                                                                   0289
    double temp1[2],temp2[2];                                                           0290
    int n,nb;                                                                           0291
    Arg arg[20];                                                                        0292
    Widget form,thickbutton,dataText,hsep,brow;                                         0293
    BUTTONITEM buttonitem[4];                                                           0294
    XmTextPosition position=0;                                                          0295
    MtTextSetTitle(ddp->SixLevel,"Thickness",NULL);                                     0296
    XtPopdown(ddp->FifthLevel);                                                         0297
```

```
ddp->Is2WindowActive = 0;                                                    0298
ddp->ThickSaveActive = 0;                                                    0299

n = 0;                                                                       0300
XtSetArgI(arg,n,XmNhorizontalSpacing,10);                                    0301
form = XmCreateForm(ddp->SixLevel,"form",arg,n);                             0302

/* Create Information display area */                                        0303
n = 0;                                                                       0304
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                            0305
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                           0306
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                             0307
XtSetArgI(arg,n,XmNtopOffset,5);                                             0308
XtSetArgI(arg,n,XmNcolumns,20);                                              0309
XtSetArgI(arg,n,XmNcursorPositionVisible,FALSE);                             0310
XtSetArgI(arg,n,XmNeditable,FALSE);                                          0311
XtSetArgI(arg,n,XmNeditMode,XmMULTI_LINE_EDIT);                              0312
XtSetArgI(arg,n,XmNrows,5);                                                  0313
dataText = XmCreateText(form,"text",arg,n);                                  0314

/* Create toggle button */                                                   0315
n = 0;                                                                       0316
XtSetArgI(arg,n, XmNleftAttachment, XmATTACH_FORM);                          0317
XtSetArgI(arg,n, XmNtopAttachment, XmATTACH_WIDGET);                         0318
XtSetArgI(arg,n, XmNtopWidget,dataText);                                     0319
thickbutton = XmCreateToggleButton(form,"Write Thickness",arg,n);            0320
XtAddCallback(thickbutton, XmNvalueChangedCallback, set_thicksave, ddp);     0321

/* Another horizontal separator */                                           0322

n = 0;                                                                       0323
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                            0324
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                           0325
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                           0326
XtSetArgI(arg,n,XmNtopWidget,thickbutton);                                   0327
hsep = XmCreateSeparator(form,"hsep",arg,n);                                 0328

/* Create a OK button */                                                     0329
memset(buttonitem,0,sizeof(buttonitem));                                     0330
nb=0;                                                                        0331
buttonitem[nb].label = "OK";                                                 0332
buttonitem[nb].callback = CB_AcceptThickness;                                0333
buttonitem[nb].callback_data = ddp;                                          0334
++nb;                                                                        0335

n = 0;                                                                       0336
XtSetArgI(arg,n,XmNbottomAttachment,XmATTACH_FORM);                          0337
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                            0338
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                           0339
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                           0340
XtSetArgI(arg,n,XmNtopWidget,hsep);                                          0341
brow = MxCreateButtonRow(form,buttonitem,                                    0342
     NULL,NULL,0,arg,n);                                                     0343

temp1[0] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->vpoint1[0],  0344
     ddp->vpoint1[1],ddp->THpoint1[0],ddp->vpoint1[1]);                      0345
if(ddp->THpoint1[0] < ddp->vpoint1[0]) temp1[0] = -1 * temp1[0];             0346
temp1[1] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->vpoint1[0],  0347
```

```
        ddp->vpoint1[1],ddp->vpoint1[0],ddp->THpoint1[1]);                    0348
    if(ddp->THpoint1[1] < ddp->vpoint1[1]) temp1[1] = -1 * temp1[1];          0349
    temp2[0] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->vpoint1[0],  0350
        ddp->vpoint1[1],ddp->THpoint2[0],ddp->vpoint1[1]);                    0351
    if(ddp->THpoint2[0] < ddp->vpoint1[0]) temp2[0] = -1 * temp2[0];          0352
    temp2[1] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->vpoint1[0],  0353
        ddp->vpoint1[1],ddp->vpoint1[0],ddp->THpoint2[1]);                    0354
    if(ddp->THpoint2[1] < ddp->vpoint1[1]) temp2[1] = -1 * temp2[1];          0355

    ddp->THpoint1[0] = temp1[0];                                              0356
    ddp->THpoint1[1] = temp1[1];                                              0357

    ddp->THpoint2[0] = temp2[0];                                              0358
    ddp->THpoint2[1] = temp2[1];                                              0359

    temp1[0] = ddp->THpoint1[2] - ddp->vpoint1[2];                            0360
    temp1[1] = ddp->THpoint2[2] - ddp->vpoint1[2];                            0361

    ddp->THpoint1[2] = temp1[0];                                              0362
    ddp->THpoint2[2] = temp1[1];                                              0363

    thickness = ddp->planeCoeffs[1]*(ddp->THpoint1[0] - ddp->THpoint2[0]);    0364
    thickness = thickness + ddp->planeCoeffs[2]*(ddp->THpoint1[1]-ddp->THpoint2[1]); 0365
    thickness = thickness + ddp->planeCoeffs[3]*(ddp->THpoint1[2]-ddp->THpoint2[2]); 0366

    ddp->thickness = abs(thickness);                                          0367

    sprintf(string,"Thickness >> %f\n",ddp->thickness);                       0368
    XmTextInsert(dataText,position,string);                                   0369
    position += strlen(string);                                               0370

    XtManageChild(brow);                                                      0371
    XtManageChild(hsep);                                                      0372
    XtManageChild(thickbutton);                                               0373
    XtManageChild(dataText);                                                  0374
    XtManageChild(form);                                                      0375
    XtPopup(ddp->SixLevel,XtGrabExclusive);                                   0376

    memset(ddp->THpoint1,0,sizeof(ddp->THpoint1));                            0377
    memset(ddp->THpoint2,0,sizeof(ddp->THpoint2));                            0378
    }                                                                         0379
/* Callback for closing thickness points input ***/                          0380

static void CB_Close2PointPopup(                                              0381
    Widget w,                                                                 0382
    DDATA *ddp,                                                               0383
    XmAnyCallbackStruct *cbdata                                               0384
    ) {                                                                       0385
    XtPopdown(ddp->FifthLevel);                                               0386
    ddp->Is2WindowActive = 0;                                                 0387

    /* Reset input to ZERO */                                                 0388
    memset(ddp->THpoint1,0,sizeof(ddp->THpoint1));                            0389
    memset(ddp->THpoint2,0,sizeof(ddp->THpoint2));                            0390
    }                                                                         0391
/* Callback for Two-Points for thickness selection ***/                      0392

static void CB_2PtSelection(                                                  0393
    Widget w,                                                                 0394
    DDATA *ddp,                                                               0395
```

```
XmAnyCallbackStruct *cbdata                                                    0396
) {                                                                            0397
int n,nb;                                                                      0398
char *help;                                                                    0399
Arg arg[20];                                                                   0400
Widget form,hsep,brow;                                                         0401
BUTTONITEM buttonitem[8];                                                      0402

MtTextSetTitle(ddp->FifthLevel,"2pointselect",NULL);                           0403
ddp->Is2WindowActive = 1;                                                      0404
ddp->ThickSaveActive = 0;                                                      0405

n = 0;                                                                         0406
XtSetArgI(arg,n,XmNhorizontalSpacing,10);                                      0407
form = XmCreateForm(ddp->FifthLevel,"form",arg,n);                             0408

/* Input area for point ONE */                                                 0409

n = 0;                                                                         0410
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                              0411
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                               0412
XtSetArgI(arg,n,XmNtopOffset,5);                                               0413
ddp->THptXform[1] = MxCreatePromptDouble(form,"First Point",&ddp->THpoint1[0], 0414
ddp->THpoint1[0],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);  0415

n = 0;                                                                         0416
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);                            0417
XtSetArgI(arg,n,XmNleftWidget,ddp->THptXform[1]);                              0418
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                               0419
XtSetArgI(arg,n,XmNtopOffset,5);                                               0420
ddp->THptYform[1] = MxCreatePromptDouble(form," ",&ddp->THpoint1[1],           0421
ddp->THpoint1[1],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);  0422

n = 0;                                                                         0423
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);                            0424
XtSetArgI(arg,n,XmNleftWidget,ddp->THptYform[1]);                              0425
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                             0426
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                               0427
XtSetArgI(arg,n,XmNtopOffset,5);                                               0428
ddp->THptZform[1] = MxCreatePromptDouble(form," ",&ddp->THpoint1[2],           0429
ddp->THpoint1[2],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);  0430

/* Input area for point TWO */                                                 0431

n = 0;                                                                         0432
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                              0433
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                             0434
XtSetArgI(arg,n,XmNtopWidget,ddp->THptXform[1]);                               0435
XtSetArgI(arg,n,XmNtopOffset,5);                                               0436
ddp->THptXform[2] = MxCreatePromptDouble(form,"Second Point",&ddp->THpoint2[0], 0437
ddp->THpoint2[0],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);  0438

n = 0;                                                                         0439
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);                            0440
XtSetArgI(arg,n,XmNleftWidget,ddp->THptXform[2]);                              0441
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                             0442
XtSetArgI(arg,n,XmNtopWidget,ddp->THptYform[1]);                               0443
XtSetArgI(arg,n,XmNtopOffset,5);                                               0444
ddp->THptYform[2] = MxCreatePromptDouble(form," ",&ddp->THpoint2[1],           0445
```

```
    ddp->THpoint2[1],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);     0446
    n = 0;                                                                            0447
    XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);                               0448
    XtSetArgI(arg,n,XmNleftWidget,ddp->THptYform[2]);                                 0449
    XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                0450
    XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                0451
    XtSetArgI(arg,n,XmNtopOffset,5);                                                  0452
    ddp->THptZform[2] = MxCreatePromptDouble(form," ",&ddp->THpoint2[2],              0453
    ddp->THpoint2[2],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);     0454
    /* Another horizontal separator */                                                0455
    n = 0;                                                                            0456
    XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                 0457
    XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                0458
    XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                0459
    XtSetArgI(arg,n,XmNtopWidget,ddp->THptXform[2]);                                  0460
    hsep = XmCreateSeparator(form,"hsep",arg,n);                                      0461
    /* OK and Cancel Pushbuttons */                                                   0462
    memset(buttonitem,0,sizeof(buttonitem));                                          0463
    nb=0;                                                                             0464
    buttonitem[nb].label = "OK";                                                      0465
    buttonitem[nb].callback_data = ddp;                                               0466
    ++nb;                                                                             0467
    buttonitem[nb].label = "Cancel";                                                  0468
    buttonitem[nb].callback = CB_Close2PointPopup;                                    0469
    buttonitem[nb].callback_data = ddp;                                               0470
    ++nb;                                                                             0471
    n = 0;                                                                            0472
    XtSetArgI(arg,n,XmNbottomAttachment,XmATTACH_FORM);                               0473
    XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                 0474
    XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                0475
    XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                0476
    XtSetArgI(arg,n,XmNtopWidget,hsep);                                               0477
    brow = MxCreateButtonRow(form,buttonitem,NULL,NULL,0,arg,n);                      0478
    XtManageChild(ddp->THptXform[1]);                                                 0479
    XtManageChild(ddp->THptYform[1]);                                                 0480
    XtManageChild(ddp->THptZform[1]);                                                 0481
    XtManageChild(ddp->THptXform[2]);                                                 0482
    XtManageChild(ddp->THptYform[2]);                                                 0483
    XtManageChild(ddp->THptZform[2]);                                                 0484
    XtManageChild(hsep);                                                              0485
    XtManageChild(brow);                                                              0486
    XtManageChild(form);                                                              0487
    XtPopup(ddp->FifthLevel,XtGrabNone);                                              0488
    }                                                                                 0489
/* Callback to close popup window that displays plane orientation ***/               0490
static void CB_ClosePlaneDataPopup(                                                   0491
    Widget w,                                                                         0492
    DDATA *ddp,                                                                       0493
    XmAnyCallbackStruct *cbdata                                                       0494
    ) {                                                                               0495
```

```
    XtPopdown(ddp->fourthLevel);                                    0496
    if(ddp->Is2WindowActive)                                        0497
        {                                                           0498
        XtPopdown(ddp->FifthLevel);                                 0499
        ddp->Is2WindowActive = 0;                                   0500
        }                                                           0501
    /* Reset input to ZERO */                                       0502
    memset(ddp->point1,0,sizeof(ddp->point1));                      0503
    memset(ddp->point2,0,sizeof(ddp->point2));                      0504
    memset(ddp->point3,0,sizeof(ddp->point3));                      0505
    }                                                               0506
/* Callback to popup window that displays plane orientation ***/   0507
static void CB_PlaneData(                                           0508
    Widget w,                                                       0509
    DDATA *ddp,                                                     0510
    XmAnyCallbackStruct *cbdata                                     0511
    ) {                                                             0512
    int n,nb;                                                       0513
    char string[200];                                               0514
    Arg arg[20];                                                    0515
    Widget form,brow,dataText;                                      0516
    BUTTONITEM buttonitem[8];                                       0517
    XmTextPosition position=0;                                      0518
    MtTextSetTitle(ddp->fourthLevel,"planeorient",NULL);            0519
    n = 0;                                                          0520
    XtSetArgI(arg,n,XmNhorizontalSpacing,10);                       0521
    form = XmCreateForm(ddp->fourthLevel,"form",arg,n);             0522
    /* Create Information display area */                           0523
    n = 0;                                                          0524
    XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);               0525
    XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);              0526
    XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                0527
    XtSetArgI(arg,n,XmNtopOffset,5);                                0528
    XtSetArgI(arg,n,XmNcolumns,80);                                 0529
    XtSetArgI(arg,n,XmNcursorPositionVisible,FALSE);                0530
    XtSetArgI(arg,n,XmNeditable,FALSE);                             0531
    XtSetArgI(arg,n,XmNeditMode,XmMULTI_LINE_EDIT);                 0532
    XtSetArgI(arg,n,XmNrows,12);                                    0533
    dataText = XmCreateText(form,"text",arg,n);                     0534
    /* Add and Cancel Pushbuttons */                                0535
    memset(buttonitem,0,sizeof(buttonitem));                        0536
    nb=0;                                                           0537
    buttonitem[nb].label = "Add";                                   0538
    buttonitem[nb].callback = CB_PlaneDataPopupAccept;              0539
    buttonitem[nb].callback_data = ddp;                             0540
    ++nb;                                                           0541
    buttonitem[nb].label = "Cancel";                                0542
    buttonitem[nb].callback = CB_ClosePlaneDataPopup;               0543
    buttonitem[nb].callback_data = ddp;                             0544
    ++nb;                                                           0545
    buttonitem[nb].label = "Thickness";                             0546
```

```
buttonitem[nb].callback = CB_2PtSelection;                                    0547
buttonitem[nb].callback_data = ddp;                                           0548
++nb;                                                                         0549

n = 0;                                                                        0550
XtSetArgI(arg,n,XmNbottomAttachment,XmATTACH_FORM);                           0551
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                             0552
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                            0553
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                            0554
XtSetArgI(arg,n,XmNtopWidget,dataText);                                       0555
brow = MxCreateButtonRow(form,buttonitem,NULL,NULL,0,arg,n);                  0556

XtManageChild(dataText);                                                      0557
XtManageChild(brow);                                                          0558
XtManageChild(form);                                                          0559

sprintf(string,"Plane Orientation\n\n");                                      0560
XmTextInsert(dataText,position,string);                                       0561
position += strlen(string);                                                   0562

sprintf(string,"Coord 1 --> %f %f %f\n",ddp->point1[0],ddp->point1[1],        0563
     ddp->point1[2]);                                                         0564
XmTextInsert(dataText,position,string);                                       0565
position += strlen(string);                                                   0566

sprintf(string,"Coord 2 --> %f %f %f\n",ddp->point2[0],ddp->point2[1],        0567
     ddp->point2[2]);                                                         0568
XmTextInsert(dataText,position,string);                                       0569
position += strlen(string);                                                   0570

sprintf(string,"Coord 3 --> %f %f %f\n\n",ddp->point3[0],ddp->point3[1],      0571
     ddp->point3[2]);                                                         0572
XmTextInsert(dataText,position,string);                                       0573
position += strlen(string);                                                   0574

sprintf(string,"Plane Coefficients--> %f %f %f\n\n",ddp->planeCoeffs[1],      0575
     ddp->planeCoeffs[2],ddp->planeCoeffs[3]);                                0576
XmTextInsert(dataText,position,string);                                       0577
position += strlen(string);                                                   0578

sprintf(string,"trend --> %f\n",ddp->planeOrientation[1]);                    0579
XmTextInsert(dataText,position,string);                                       0580
position += strlen(string);                                                   0581

sprintf(string,"plunge --> %f\n",ddp->planeOrientation[2]);                   0582
XmTextInsert(dataText,position,string);                                       0583
position += strlen(string);                                                   0584

memset(string,0,sizeof(string));                                              0585

sprintf(string,"Plane Class => %s\n",ddp->planeClass);                        0586
XmTextInsert(dataText,position,string);                                       0587
position += strlen(string);                                                   0588

sprintf(string,"Plane Centroid --> %f %f %f",ddp->center.x,                   0589
     ddp->center.y,ddp->center.z);                                            0590
XmTextInsert(dataText,position,string);                                       0591
position += strlen(string);                                                   0592

XtPopup(ddp->fourthLevel,XtGrabNone);                                         0593
}                                                                             0594
```

```
/* Cross-product between two vectors plus normalization ***/
void ThreePointToPlane(
     DDATA *ddp
     ) {
     double temp[5];
     double scaling;

     /* cross-product */
     temp[1] = ddp->point2[1]*ddp->point3[2] - ddp->point2[2]*ddp->point3[1];
     temp[2] = ddp->point2[2]*ddp->point3[0] - ddp->point2[0]*ddp->point3[2];
     temp[3] = ddp->point2[0]*ddp->point3[1] - ddp->point2[1]*ddp->point3[0];

     /* normalize */
     scaling = 1/(sqrt(temp[1]*temp[1] + temp[2]*temp[2] + temp[3]*temp[3]));
     if(temp[3] < 0) scaling = -1 * scaling;

     ddp->planeCoeffs[1] = temp[1] * scaling;
     ddp->planeCoeffs[2] = temp[2] * scaling;
     ddp->planeCoeffs[3] = temp[3] * scaling;
     }
/* This function will calculate the orientation ie.  trend and plunge, of a plane **/
void ThreeDCoordToGeographical(
     DDATA *ddp
     ) {
     double trend = 0;
     double plunge = 0;
     double hype,temp;

     if (ddp->planeCoeffs[1] == 0 && ddp->planeCoeffs[2] == 0)
          { plunge = 0; }
          else
          {
          hype = sqrt(ddp->planeCoeffs[1]*ddp->planeCoeffs[1]
               + ddp->planeCoeffs[2]*ddp->planeCoeffs[2]);
          temp = atan(ddp->planeCoeffs[3]/hype);
          plunge = PI/2 - temp;
          }

     if (ddp->planeCoeffs[1] == 0 && ddp->planeCoeffs[2] > 0) trend = 0;
     if (ddp->planeCoeffs[1] == 0 && ddp->planeCoeffs[2] < 0) trend = PI;
     if (ddp->planeCoeffs[1] < 0 && ddp->planeCoeffs[2] == 0) trend = 3*PI/2;
     if (ddp->planeCoeffs[1] > 0 && ddp->planeCoeffs[2] == 0) trend = PI/2;
     /* First Quadrant */
     if (ddp->planeCoeffs[1] > 0 && ddp->planeCoeffs[2] > 0)
          {
          trend = atan(ddp->planeCoeffs[1]/ddp->planeCoeffs[2]);
          }
     /* Second Quadrant */
     if (ddp->planeCoeffs[1] > 0 && ddp->planeCoeffs[2] < 0)
          {
          trend = PI - atan(ddp->planeCoeffs[1]/abs(ddp->planeCoeffs[2]));
          }
     /* Fourth Quadrant */
     if (ddp->planeCoeffs[1] < 0 && ddp->planeCoeffs[2] > 0)
          {
          trend = 2*PI - atan(abs(ddp->planeCoeffs[1])/ddp->planeCoeffs[2]);
```

```
        }                                                                    0646
    /* Third Quadrant */                                                     0647
    if (ddp->planeCoeffs[1] < 0 && ddp->planeCoeffs[2] < 0)                  0648
        {                                                                    0649
        trend = PI + atan(ddp->planeCoeffs[1]/ddp->planeCoeffs[2]);          0650
        }                                                                    0651
    ddp->planeOrientation[1] = trend * 180/PI;                               0652
    ddp->planeOrientation[2] = plunge * 180/PI;                              0653
    }                                                                        0654
/* Function to calculate the centiod of a triangle ***/                     0655
static void FbsThreePtCentroid(                                              0656
    DDATA *ddp                                                               0657
    ) {                                                                      0658
    double s,t;                                                              0659
    if(ddp->vpoint1[0] < ddp->vpoint2[0])                                    0660
        {                                                                    0661
        if(ddp->vpoint3[0] < ddp->vpoint1[0])                                0662
            {                                                                0663
            s = ddp->vpoint2[0] - ddp->vpoint3[0];                           0664
            ddp->center.x = ddp->vpoint3[0] + s/2;                           0665
            }                                                                0666
        else                                                                 0667
            {                                                                0668
            if(ddp->vpoint3[0] < ddp->vpoint2[0])                            0669
                {                                                            0670
                s = ddp->vpoint2[0] - ddp->vpoint1[0];                       0671
                ddp->center.x = ddp->vpoint1[0] + s/2;                       0672
                }                                                            0673
            else                                                             0674
                {                                                            0675
                s = ddp->vpoint3[0] - ddp->vpoint1[0];                       0676
                ddp->center.x = ddp->vpoint1[0] + s/2;                       0677
                }                                                            0678
            }                                                                0679
        }                                                                    0680
    else                                                                     0681
        {                                                                    0682
        if(ddp->vpoint3[0] < ddp->vpoint2[0])                                0683
            {                                                                0684
            s = ddp->vpoint1[0] - ddp->vpoint3[0];                           0685
            ddp->center.x = ddp->vpoint3[0] + s/2;                           0686
            }                                                                0687
        else                                                                 0688
            {                                                                0689
            if(ddp->vpoint3[0] < ddp->vpoint1[0])                            0690
                {                                                            0691
                s = ddp->vpoint1[0] - ddp->vpoint2[0];                       0692
                ddp->center.x = ddp->vpoint2[0] + s/2;                       0693
                }                                                            0694
            else                                                             0695
                {                                                            0696
                s = ddp->vpoint3[0] - ddp->vpoint2[0];                       0697
                ddp->center.x = ddp->vpoint2[0] + s/2;                       0698
```

```
                    }                                                    0699
               }                                                         0700
          }                                                              0701
     if(ddp->vpoint1[1] < ddp->vpoint2[1])                               0702
          {                                                              0703
          if(ddp->vpoint3[1] < ddp->vpoint1[1])                          0704
               {                                                         0705
               s = ddp->vpoint2[1] - ddp->vpoint3[1];                    0706
               ddp->center.y = ddp->vpoint3[1] + s/2;                    0707
               }                                                         0708
          else                                                           0709
               {                                                         0710
               if(ddp->vpoint3[1] < ddp->vpoint2[1])                     0711
                    {                                                    0712
                    s = ddp->vpoint2[1] - ddp->vpoint1[1];               0713
                    ddp->center.y = ddp->vpoint1[1] + s/2;               0714
                    }                                                    0715
               else                                                      0716
                    {                                                    0717
                    s = ddp->vpoint3[1] - ddp->vpoint1[1];               0718
                    ddp->center.y = ddp->vpoint1[1] + s/2;               0719
                    }                                                    0720
               }                                                         0721
          }                                                              0722
     else                                                                0723
          {                                                              0724
          if(ddp->vpoint3[1] < ddp->vpoint2[1])                          0725
               {                                                         0726
               s = ddp->vpoint1[1] - ddp->vpoint3[1];                    0727
               ddp->center.y = ddp->vpoint3[1] + s/2;                    0728
               }                                                         0729
          else                                                           0730
               {                                                         0731
               if(ddp->vpoint3[1] < ddp->vpoint1[1])                     0732
                    {                                                    0733
                    s = ddp->vpoint1[1] - ddp->vpoint2[1];               0734
                    ddp->center.y = ddp->vpoint2[1] + s/2;               0735
                    }                                                    0736
               else                                                      0737
                    {                                                    0738
                    s = ddp->vpoint3[1] - ddp->vpoint2[1];               0739
                    ddp->center.y = ddp->vpoint2[1] + s/2;               0740
                    }                                                    0741
               }                                                         0742
          }                                                              0743
     /* Now to find the z-value */                                       0744
     s = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->point1[0],  0745
          ddp->point1[1],ddp->center.x,ddp->point1[1]);                  0746
     if(ddp->point2[0] < ddp->point1[0]) s = -1 * s;                     0747
     t = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,ddp->point1[0],  0748
          ddp->point1[1],ddp->point1[0],ddp->center.y);                  0749
     if(ddp->point2[1] < ddp->point1[1]) t = -1 * t;                     0750
     ddp->center.z = 1 - ddp->planeCoeffs[2]*ddp->center.y               0751
```

```
            - ddp->planeCoeffs[1]*ddp->center.x;                                    0752
    ddp->center.z = abs(ddp->center.z/ddp->planeCoeffs[3]) + ddp->vpoint1[2];       0753
    }                                                                               0754

/* Transforms GLOBAL=>LOCAL, calculates normal, and converts to Trend/Plunge ***/  0755

static void Fbs3DVectorEngine(                                                      0756
    Widget w,                                                                       0757
    DDATA *ddp,                                                                     0758
    XmAnyCallbackStruct *cbdata                                                     0759
    ) {                                                                             0760
    double temp1[2],temp2[2];                                                       0761

    temp1[0] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,                     0762
            ddp->point1[0],ddp->point1[1],ddp->point2[0],ddp->point1[1]);           0763
    if(ddp->point2[0] < ddp->point1[0]) temp1[0] = -1 * temp1[0];                   0764
    temp1[1] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,                     0765
            ddp->point1[0],ddp->point1[1],ddp->point1[0],ddp->point2[1]);           0766
    if(ddp->point2[1] < ddp->point1[1]) temp1[1] = -1 * temp1[1];                   0767
    temp2[0] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,                     0768
            ddp->point1[0],ddp->point1[1],ddp->point3[0],ddp->point1[1]);           0769
    if(ddp->point3[0] < ddp->point1[0]) temp2[0] = -1 * temp2[0];                   0770
    temp2[1] = LatLonDistToMeters(&ddp->vectdispparm->ProjParm,                     0771
            ddp->point1[0],ddp->point1[1],ddp->point1[0],ddp->point3[1]);           0772
    if(ddp->point3[1] < ddp->point1[1]) temp2[1] = -1 * temp2[1];                   0773

    ddp->point2[0] = temp1[0];                                                      0774
    ddp->point2[1] = temp1[1];                                                      0775

    ddp->point3[0] = temp2[0];                                                      0776
    ddp->point3[1] = temp2[1];                                                      0777

    ddp->point1[0] = 0;                                                             0778
    ddp->point1[1] = 0;                                                             0779

    temp1[0] = ddp->point2[2] - ddp->point1[2];                                     0780
    temp1[1] = ddp->point3[2] - ddp->point1[2];                                     0781

    ddp->point2[2] = temp1[0];                                                      0782
    ddp->point3[2] = temp1[1];                                                      0783
    ddp->point1[2] = 0;                                                             0784

    /*Note that the above distance is calculated along the curvature of the Earth   0785
    but it wont make a difference I think for our scale */                          0786

    /* Now do plane orientation calculations */                                     0787

    ThreePointToPlane(ddp);                                                         0788
    ThreeDCoordToGeographical(ddp);                                                 0789
    FbsThreePtCentroid(ddp);                                                        0790
    CB_PlaneData(w,ddp,cbdata);                                                     0791
    }                                                                               0792

/* Callback to close display popup for entering 3point data ***/                   0793

static void CB_Close3PointPopup(                                                    0794
    Widget w,                                                                       0795
    DDATA *ddp,                                                                     0796
    XmAnyCallbackStruct *cbdata                                                     0797
    ) {                                                                             0798
    XtPopdown(ddp->thirdLevel);                                                     0799
    ddp->IsWindowActive = 0;                                                        0800
```

```
     /* Reset input to ZERO */                                                    0801
     memset(ddp->point1,0,sizeof(ddp->point1));                                  0802
     memset(ddp->point2,0,sizeof(ddp->point2));                                  0803
     memset(ddp->point3,0,sizeof(ddp->point3));                                  0804
     }                                                                           0805
/* Callback for accepting input in 3point popup ***/                            0806
static void CB_Accept3PointPopup(                                                0807
     Widget w,                                                                   0808
     DDATA *ddp,                                                                 0809
     XmAnyCallbackStruct *cbdata                                                 0810
     ) {                                                                         0811
     Fbs3DVectorEngine(w,ddp,cbdata);                                            0812
     }                                                                           0813
/* Callback procedure for exiting ***/                                          0814
static void CB_Close(                                                            0815
     Widget w,                                                                   0816
     DDATA *ddp,                                                                 0817
     XmAnyCallbackStruct *cbdata                                                 0818
     ) {                                                                         0819
     MfCloseVect(ddp->vid,&ddp->vectdispparm->vectinfo);                         0820
     exit(0);                                                                    0821
     }                                                                           0822
/* Callback for 3-pt selection on selected raster ***/                          0823
static void CB_3PtSelection(                                                     0824
     Widget w,                                                                   0825
     DDATA *ddp,                                                                 0826
     XmAnyCallbackStruct *cbdata                                                 0827
     ) {                                                                         0828
     int n,nb;                                                                   0829
     char *help;                                                                 0830
     Arg arg[20];                                                                0831
     Widget form,hsep,brow,hsep1,planeClassform;                                 0832
     BUTTONITEM buttonitem[8];                                                   0833
     MtTextSetTitle(ddp->thirdLevel,"3pointselect",NULL);                        0834
     ddp->IsWindowActive = 1;                                                    0835
     n = 0;                                                                      0836
     XtSetArgI(arg,n,XmNhorizontalSpacing,10);                                   0837
     form = XmCreateForm(ddp->thirdLevel,"form",arg,n);                          0838
     /* Input area for point ONE */                                              0839
     n = 0;                                                                      0840
     XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                           0841
     XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                            0842
     XtSetArgI(arg,n,XmNtopOffset,5);                                            0843
     ddp->ptXform[1] = MxCreatePromptDouble(form,"First Point",&ddp->point1[0],  0844
     ddp->point1[0],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n); 0845
     n = 0;                                                                      0846
     XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);                         0847
     XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                            0848
     XtSetArgI(arg,n,XmNtopOffset,5);                                            0849
     ddp->ptYform[1] = MxCreatePromptDouble(form," ",&ddp->point1[1],            0850
```

```
ddp->point1[1],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNleftWidget,ddp->ptYform[1]);
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);
XtSetArgI(arg,n,XmNtopOffset,5);
ddp->ptZform[1] = MxCreatePromptDouble(form," ",&ddp->point1[2],
ddp->point1[2],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
/* Input area for point TWO */
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);
XtSetArgI(arg,n,XmNtopWidget,ddp->ptXform[1]);
XtSetArgI(arg,n,XmNtopOffset,5);
ddp->ptXform[2] = MxCreatePromptDouble(form,"Second Point",&ddp->point2[0],
ddp->point2[0],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNleftWidget,ddp->ptXform[2]);
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNtopWidget,ddp->ptYform[1]);
XtSetArgI(arg,n,XmNtopOffset,5);
ddp->ptYform[2] = MxCreatePromptDouble(form," ",&ddp->point2[1],
ddp->point2[1],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNleftWidget,ddp->ptYform[2]);
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNtopWidget,ddp->ptZform[1]);
XtSetArgI(arg,n,XmNtopOffset,5);
ddp->ptZform[2] = MxCreatePromptDouble(form," ",&ddp->point2[2],
ddp->point2[2],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
/* Input area for point THREE */
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNtopWidget,ddp->ptXform[2]);
ddp->ptXform[3] = MxCreatePromptDouble(form,"Third Point",&ddp->point3[0],
ddp->point3[0],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNleftWidget,ddp->ptXform[3]);
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);
XtSetArgI(arg,n,XmNtopWidget,ddp->ptYform[2]);
XtSetArgI(arg,n,XmNtopOffset,5);
ddp->ptYform[3] = MxCreatePromptDouble(form," ",&ddp->point3[1],
ddp->point3[1],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);
n = 0;
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_WIDGET);
```

```
XtSetArgI(arg,n,XmNleftWidget,ddp->ptYform[3]);                                    0901
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                 0902
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                 0903
XtSetArgI(arg,n,XmNtopWidget,ddp->ptZform[2]);                                     0904
XtSetArgI(arg,n,XmNtopOffset,5);                                                   0905
ddp->ptZform[3] = MxCreatePromptDouble(form," ",&ddp->point3[2],                   0906
ddp->point3[2],NULL,NULL,8,help,GET_NoShowRange|GET_NoMin|GET_NoMax,arg,n);        0907

/* Another horizontal separator */                                                 0908
n = 0;                                                                             0909
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                  0910
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                 0911
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                 0912
XtSetArgI(arg,n,XmNtopWidget,ddp->ptXform[3]);                                     0913
hsep = XmCreateSeparator(form,"hsep",arg,n);                                       0914

/* Type of plane input */                                                          0915
n = 0;                                                                             0916
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                  0917
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                 0918
planeClassform = MxCreatePromptString(form,"Plane Class",ddp->planeClass,          0919
     10,help,NULL,arg,n);                                                          0920

/* Horizontal separator */                                                         0921

n = 0;                                                                             0922
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                  0923
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                 0924
XtSetArgI(arg,n,XmNtopWidget,planeClassform);                                      0925
hsep1 = XmCreateSeparator(form,"hsep1",arg,n);                                     0926

/* OK and Cancel Pushbuttons */                                                    0927

memset(buttonitem,0,sizeof(buttonitem));                                           0928
nb=0;                                                                              0929
buttonitem[nb].label = "OK";                                                       0930
buttonitem[nb].callback = CB_Accept3PointPopup;                                    0931
buttonitem[nb].callback_data = ddp;                                                0932
++nb;                                                                              0933
buttonitem[nb].label = "Close";                                                    0934
buttonitem[nb].callback = CB_Close3PointPopup;                                     0935
buttonitem[nb].callback_data = ddp;                                                0936
++nb;                                                                              0937

n = 0;                                                                             0938
XtSetArgI(arg,n,XmNbottomAttachment,XmATTACH_FORM);                                0939
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                  0940
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                                 0941
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                                 0942
XtSetArgI(arg,n,XmNtopWidget,hsep1);                                               0943
brow = MxCreateButtonRow(form,buttonitem,NULL,NULL,0,arg,n);                       0944

XtManageChild(ddp->ptXform[1]);                                                    0945
XtManageChild(ddp->ptYform[1]);                                                    0946
XtManageChild(ddp->ptZform[1]);                                                    0947
XtManageChild(ddp->ptXform[2]);                                                    0948
XtManageChild(ddp->ptYform[2]);                                                    0949
XtManageChild(ddp->ptZform[2]);                                                    0950
XtManageChild(ddp->ptXform[3]);                                                    0951
```

```
    XtManageChild(ddp->ptYform[3]);                                          0952
    XtManageChild(ddp->ptZform[3]);                                          0953
    XtManageChild(hsep);                                                     0954
    XtManageChild(hsep1);                                                    0955
    XtManageChild(planeClassform);                                           0956
    XtManageChild(brow);                                                     0957
    XtManageChild(form);                                                     0958
    XtPopup(ddp->thirdLevel,XtGrabNone);                                     0959
    }                                                                        0960
/* Callback to check if DTM covers given geographical point and retrieves value ***/  0961
static double Zvalue(                                                        0962
    DDATA *ddp,                                                              0963
    DPOINT2D point                                                           0964
    ) {                                                                      0965
    int fhandle,DTMrasterhandle;                                             0966
    Mat3x3 forward,inverse;                                                  0967
    double rastx,rasty,zvalue;                                               0968
    LONG rastxI,rastyI;                                                      0969
    int DTMaccess = 0;                                                       0970
    if ((fhandle = MxOpenFile(NULL, &ddp->iDTMfilename,RVCFILE_ReadLock)) < 0)   0971
        {                                                                    0972
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","OpenDTMraster",         0973
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         0974
        }                                                                    0975
    if(MfReadGeorefMatrix(fhandle,ddp->DTMgeorefinode,NULL,forward,          0976
            inverse,NULL) < 0)                                               0977
        {                                                                    0978
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","GetTransformMatrix",    0979
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         0980
        exit(0);                                                             0981
        }                                                                    0982
    if((DTMrasterhandle = MfOpenRast(fhandle,ddp->DTMrastinode,              0983
            &ddp->DTMrastinfo,RVCMODE_Read)) < 0)                            0984
        {                                                                    0985
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","OpenRastProcessProblem",  0986
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         0987
        exit(0);                                                             0988
        }                                                                    0989
    MfSetRastMode(DTMrasterhandle,RASTMODE_ConvToDouble);                    0990
    trans2d(point.x,point.y,inverse,&rastx,&rasty);                          0991
    rastxI = rastx;                                                          0992
    rastyI = rasty;                                                          0993
    /* We must check here if line/col is relevant to DTM raster */           0994
    if(rastyI > 0)                                                           0995
        {                                                                    0996
        if(rastyI < ddp->DTMrastinfo.numlins)                                0997
            {                                                                0998
            if(rastxI > 0)                                                   0999
                {                                                            1000
                if(rastxI < ddp->DTMrastinfo.numcols)                        1001
                    {                                                        1002
```

```
                        if(MfReadRast(DTMrasterhandle,rastyI,rastxI,1,&zvalue) < 0)1003
                        {                                                          1004
                        MxPopupMessageMt(ddp->toplevel,0,"geostruc",               1005
                        "RaedRastProcessProblem",NULL,XmDIALOG_ERROR|GET_NoCancel); 1006
                        exit(0);                                                   1007
                        }                                                          1008
    }}}}                                                                           1009
    if(DTMaccess == 0)                                                             1010
        {                                                                          1011
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","DTMoutOfRange",               1012
            NULL,XmDIALOG_ERROR | GET_NoCancel);                                   1013
        zvalue = 0;                                                                1014
        {                                                                          1015
    if(abs(zvalue) > 10000)                                                        1016
        {                                                                          1017
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","VeryHigh",                    1018
            NULL,XmDIALOG_ERROR | GET_NoCancel);                                   1019
        }                                                                          1020
    MfCloseRast(DTMrasterhandle, NULL);                                            1021
    MfCloseFile(fhandle);                                                          1022
    return(zvalue);                                                                1023
    }                                                                              1024
/* Callback for saving coords of crosshair center ***/                            1025
static int SaveFunc(                                                               1026
    CROSSHAIRINFO *crosshairinfo,                                                  1027
    DDATA *ddp,                                                                    1028
    int add                                                                        1029
    ) {                                                                            1030
    DPOINT2D point;                                                                1031
    int fhandle;                                                                   1032
    Mat3x3 forward,inverse;                                                        1033
    /* Firstly transform drawing area coords to actual raster layer coords */     1034
    MxdwTransWindowToLayer(ddp->currentlayer,&crosshairinfo->center,               1035
            &crosshairinfo->center,1);                                             1036
    if ((fhandle = MxOpenFile(NULL, &ddp->ifilename,RVCFILE_ReadLock)) < 0)        1037
        {                                                                          1038
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","OpenDTMraster",               1039
            NULL,XmDIALOG_ERROR | GET_NoCancel);                                   1040
        }                                                                          1041
    if(MfReadGeorefMatrix(fhandle,ddp->georefinode,NULL,forward,inverse,NULL) < 0) 1042
        {                                                                          1043
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","GetTransformMatrix",          1044
            NULL,XmDIALOG_ERROR | GET_NoCancel);                                   1045
        exit(0);                                                                   1046
        }                                                                          1047
    trans2d(crosshairinfo->center.x,crosshairinfo->center.y,                       1048
            forward,&point.x,&point.y);                                            1049
    /* NB NB NB NB NB                                                             1050
    The actual coordinates that we have at the moment are the                      1051
    Geographical coordinates given as a decimal not as deg/min/sec                 1052
    ----- the so called MAP Coords ------- */                                      1053
```

```
switch(ddp->HasCoords) {                                                    1054
     case 1:                                                                1055
     if(ddp->IsWindowActive == 1 && ddp->Is2WindowActive == 0)              1056
          {                                                                 1057
          ddp->vectorpt1.x = ddp->vpoint1[0] = ddp->point1[0] = point.x;    1058
          ddp->vectorpt1.y = ddp->vpoint1[1] = ddp->point1[1] = point.y;    1059
          ddp->vectorpt1.z = ddp->vpoint1[2]=ddp->point1[2]=Zvalue(ddp,point); 1060
          MxPromptUpdateValue(ddp->ptXform[1]);                             1061
          MxPromptUpdateValue(ddp->ptYform[1]);                             1062
          MxPromptUpdateValue(ddp->ptZform[1]);                             1063
          }                                                                 1064
     if(ddp->Is2WindowActive == 1)                                          1065
          {                                                                 1066
          ddp->THpoint1[0] = point.x;                                       1067
          ddp->THpoint1[1] = point.y;                                       1068
          ddp->THpoint1[2] = Zvalue(ddp,point);                             1069
          MxPromptUpdateValue(ddp->THptXform[1]);                           1070
          MxPromptUpdateValue(ddp->THptYform[1]);                           1071
          MxPromptUpdateValue(ddp->THptZform[1]);                           1072
          }                                                                 1073
     ddp->HasCoords = 2;                                                    1074
     break;                                                                 1075
     case 2:                                                                1076
     if(ddp->IsWindowActive == 1 && ddp->Is2WindowActive == 0)              1077
          {                                                                 1078
          ddp->vectorpt2.x = ddp->vpoint2[0] = ddp->point2[0] = point.x;    1079
          ddp->vectorpt2.y = ddp->vpoint2[1] = ddp->point2[1] = point.y;    1080
          ddp->vectorpt2.z = ddp->vpoint2[2]=ddp->point2[2]=Zvalue(ddp,point); 1081
          MxPromptUpdateValue(ddp->ptXform[2]);                             1082
          MxPromptUpdateValue(ddp->ptYform[2]);                             1083
          MxPromptUpdateValue(ddp->ptZform[2]);                             1084
          }                                                                 1085
     if(ddp->Is2WindowActive == 1)                                          1086
          {                                                                 1087
          ddp->THpoint2[0] = point.x;                                       1088
          ddp->THpoint2[1] = point.y;                                       1089
          ddp->THpoint2[2] = Zvalue(ddp,point);                             1090
          MxPromptUpdateValue(ddp->THptXform[2]);                           1091
          MxPromptUpdateValue(ddp->THptYform[2]);                           1092
          MxPromptUpdateValue(ddp->THptZform[2]);                           1093
          }                                                                 1094
     ddp->HasCoords = 3;                                                    1095
     break;                                                                 1096
     case 3:                                                                1097
     if(ddp->IsWindowActive == 1 && ddp->Is2WindowActive == 0)              1098
          {                                                                 1099
          ddp->vectorpt3.x = ddp->vpoint3[0] = ddp->point3[0] = point.x;    1100
          ddp->vectorpt3.y = ddp->vpoint3[1] = ddp->point3[1] = point.y;    1101
          ddp->vectorpt3.z = ddp->vpoint3[2]=ddp->point3[2]=Zvalue(ddp,point); 1102
          MxPromptUpdateValue(ddp->ptXform[3]);                             1103
          MxPromptUpdateValue(ddp->ptYform[3]);                             1104
          MxPromptUpdateValue(ddp->ptZform[3]);                             1105
```

```
                }
            if(ddp->Is2WindowActive == 1)
                {
                ddp->THpoint1[0] = point.x;
                ddp->THpoint1[1] = point.y;
                ddp->THpoint1[2] = Zvalue(ddp,point);
                MxPromptUpdateValue(ddp->THptXform[1]);
                MxPromptUpdateValue(ddp->THptYform[1]);
                MxPromptUpdateValue(ddp->THptZform[1]);
                ddp->HasCoords = 2;
                }
            else
                {
                ddp->HasCoords = 1;
                }
            break;
            }
        MfCloseFile(fhandle);
        return(1);
        }
/* Callback for enhancing raster object ***/
static void CB_RastEnhance(
        Widget w,
        DDATA *ddp,
        XmAnyCallbackStruct *cbdata
        ) {
        MxdwCtrlPanelRaster(ddp->currentlayer,0);
        }
/* Callback for enhancing vector object ***/
static void CB_VectEnhance(
        Widget w,
        DDATA *ddp,
        XmAnyCallbackStruct *cbdata
        ) {
        MxdwCtrlPanelVector(ddp->univectorlayer,0);
        }
/* Function to create vector object and RVC file if none exists ***/
static void CreateVector(
        DDATA *ddp
        ) {
        int err;
        int i;
        /* Open the file containing the vector object or create new file */
        if (fileexists(ddp->Vecfilename))
            {
            if((err = ddp->vecfhandle = MxOpenFile(NULL, &ddp->Vecfilename,
                RVCFILE_WriteLock)) < 0)
                {
                MxDisplayErrorCode(ddp->toplevel,err);
                return;
        }}
```

```
else                                                                    1157
     { RVCBASICINFO basicinfo;                                          1158
     memset(&basicinfo,0,sizeof(RVCBASICINFO));                         1159
     if ((err = ddp->vecfhandle = MfMakeFile(ddp->Vecfilename,          1160
               &basicinfo)) < 0)                                        1161
          {                                                             1162
          MxDisplayErrorCode(ddp->toplevel,err);                        1163
          return;                                                       1164
}}                                                                      1165
/* Now we make the actual vector object */                             1166
if(ddp->Vecinfo.NumPoints == 0)                                         1167
     {                                                                  1168
     ddp->Vecinfo.PointType = VPOINT_3DXYZ;                             1169
     strtouc(ddp->Vecinfo.source,"GeoStruc");                           1170
     ddp->Vecinfo.minval.x = ddp->rastextents.xinit;                    1171
     ddp->Vecinfo.minval.y = ddp->rastextents.yinit;                    1172
     ddp->Vecinfo.maxval.x = ddp->rastextents.xlast;                    1173
     ddp->Vecinfo.maxval.y = ddp->rastextents.ylast;                    1174
     ddp->Vecinfo.xscale = 10;                                          1175
     ddp->Vecinfo.yscale = 10;                                          1176
     ddp->Vecinfo.zscale = 10;                                          1177
     if((ddp->vid = MfMakeVect(ddp->vecfhandle,&ddp->Vecinfo,           1178
               RVCMODE_Write)) < 0)                                     1179
          {                                                             1180
          MxPopupMessage(ddp->toplevel,0,"geostruc","NoStrucVectObj",   1181
                    XmDIALOG_ERROR | GET_NoCancel);                     1182
          return;                                                       1183
          }                                                             1184
     memset(&ddp->vecgeorefinfo,0,sizeof(RVCGEOREFINFO));               1185
     ddp->vecgeorefinfo.parentinode = ddp->Vecinfo.objectinode;         1186
     ddp->vecgeorefinfo.georeftype = GEOREF_IMPLIED;                    1187
     CopyMAPPROJPARMtoGEOREFINFO(&ddp->vecgeorefinfo,ddp->rastprojparm);1188
     SetDftGeorefNameDesc(&ddp->vecgeorefinfo,ddp->rastprojparm);       1189
     if(MfWriteGeoref(ddp->vecfhandle,0,&ddp->vecgeorefinfo,NULL) < 0)  1190
          {                                                             1191
          MxPopupMessage(ddp->toplevel,0,"geostruc","NoStrucVectObj",   1192
                    XmDIALOG_ERROR | GET_NoCancel);                     1193
          return;                                                       1194
          }                                                             1195
     ddp->PointNum = ddp->LineNum = ddp->NumLabels = 0;                 1196
     ddp->vecgeorefinode = 0;                                           1197
     }                                                                  1198
else                                                                    1199
     {                                                                  1200
     if((ddp->vid = MfOpenVect(ddp->vecfhandle,ddp->Vecinode,           1201
               &ddp->Vecinfo,RVCMODE_Write)) < 0)                       1202
          {                                                             1203
          MxPopupMessage(ddp->toplevel,0,"geostruc","NoStrucVectObj",   1204
                    XmDIALOG_ERROR | GET_NoCancel);                     1205
          }                                                             1206
     ddp->PointNum = ddp->Vecinfo.NumPoints;                            1207
     ddp->LineNum = ddp->Vecinfo.NumLines;                              1208
```

```
        ddp->NumLabels = ddp->Vecinfo.NumLabels;                                1209
        }                                                                       1210
    }                                                                           1211
/* Callback to demarcate DTM extents on input raster ***/                      1212
static void FbsSetDTMextents(                                                   1213
    Widget w,                                                                   1214
    DDATA *ddp,                                                                 1215
    XmAnyCallbackStruct *cbdata                                                 1216
    ) {                                                                         1217
    /* Add in function to mark extents of dtm on raster */                      1218
    RVCVECTPOINT point;                                                         1219
    RVCVECTPOINT vector[20];                                                    1220
    RVCVECTLINE line;                                                           1221
    double minx,miny,maxx,maxy,tempx,tempy;                                     1222
    int fhandle,err;                                                            1223
    Mat3x3 forward,inverse;                                                     1224
    memset(&line,0,sizeof(RVCVECTLINE));                                        1225
    memset(&vector,0,sizeof(vector));                                           1226
    /* Open the DTM file */                                                     1227
    if ((err = fhandle = MxOpenFile(NULL, &ddp->iDTMfilename,                   1228
            RVCFILE_ReadLock)) < 0)                                             1229
        {                                                                       1230
        MxDisplayErrorCode(ddp->toplevel,err);                                  1231
        return;                                                                 1232
        }                                                                       1233
    if(MfReadGeorefMatrix(fhandle,ddp->DTMgeorefinode,NULL,forward,             1234
            inverse,NULL) < 0)                                                  1235
        {                                                                       1236
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","GetTransformMatrix",       1237
                NULL,XmDIALOG_ERROR | GET_NoCancel);                            1238
        exit(0);                                                                1239
        }                                                                       1240
    /* TopLeft corner of DTM */                                                 1241
    trans2d(0,0,forward,&point.x,&point.y);                                     1242
    /* Point one */                                                             1243
    vector[4].x = vector[0].x = point.x;                                        1244
    vector[4].y = vector[0].y = point.y;                                        1245
    vector[4].z = vector[0].z = 0;                                              1246
    if(point.x < ddp->vectdispparm->vectinfo.minval.x)                          1247
        ddp->vectdispparm->vectinfo.minval.x = point.x;                         1248
    if(point.y < ddp->vectdispparm->vectinfo.minval.y)                          1249
        ddp->vectdispparm->vectinfo.minval.y = point.y;                         1250
    if(point.z < ddp->vectdispparm->vectinfo.minval.z)                          1251
        ddp->vectdispparm->vectinfo.minval.z = point.z;                         1252
    if(point.x > ddp->vectdispparm->vectinfo.maxval.x)                          1253
        ddp->vectdispparm->vectinfo.maxval.x = point.x;                         1254
    if(point.y > ddp->vectdispparm->vectinfo.maxval.y)                          1255
        ddp->vectdispparm->vectinfo.maxval.y = point.y;                         1256
    if(point.z > ddp->vectdispparm->vectinfo.maxval.z)                          1257
        ddp->vectdispparm->vectinfo.maxval.z = point.z;                         1258
```

```
if(point.x < line.minval.x) line.minval.x = point.x;                    1259
if(point.y < line.minval.y) line.minval.y = point.y;                    1260
if(point.z < line.minval.z) line.minval.z = point.z;                    1261
if(point.x > line.maxval.x) line.maxval.x = point.x;                    1262
if(point.y > line.maxval.y) line.maxval.y = point.y;                    1263
if(point.z > line.maxval.z) line.maxval.z = point.z;                    1264

if (MfWriteVPoint(ddp->vid,ddp->PointNum,&point) < 0)                   1265
     {                                                                  1266
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",     1267
               NULL,XmDIALOG_ERROR | GET_NoCancel);                     1268
     }                                                                  1269
ddp->PointNum++;                                                        1270
ddp->vectdispparm->vectinfo.NumPoints++;                                1271

/* TopRight corner of DTM */                                            1272
trans2d(ddp->DTMrastinfo.numcols,0,forward,&point.x,&point.y);          1273

/* Point Two */                                                         1274
vector[1].x = point.x;                                                  1275
vector[1].y = point.y;                                                  1276
vector[1].z = 0;                                                        1277

if(point.x < ddp->vectdispparm->vectinfo.minval.x)                      1278
     ddp->vectdispparm->vectinfo.minval.x = point.x;                    1279
if(point.y < ddp->vectdispparm->vectinfo.minval.y)                      1280
     ddp->vectdispparm->vectinfo.minval.y = point.y;                    1281
if(point.z < ddp->vectdispparm->vectinfo.minval.z)                      1282
     ddp->vectdispparm->vectinfo.minval.z = point.z;                    1283
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)                      1284
     ddp->vectdispparm->vectinfo.maxval.x = point.x;                    1285
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)                      1286
     ddp->vectdispparm->vectinfo.maxval.y = point.y;                    1287
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)                      1288
     ddp->vectdispparm->vectinfo.maxval.z = point.z;                    1289

if(point.x < line.minval.x) line.minval.x = point.x;                    1290
if(point.y < line.minval.y) line.minval.y = point.y;                    1291
if(point.z < line.minval.z) line.minval.z = point.z;                    1292
if(point.x > line.maxval.x) line.maxval.x = point.x;                    1293
if(point.y > line.maxval.y) line.maxval.y = point.y;                    1294
if(point.z > line.maxval.z) line.maxval.z = point.z;                    1295

if (MfWriteVPoint(ddp->vid,ddp->PointNum,&point) < 0)                   1296
     {                                                                  1297
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",     1298
               NULL,XmDIALOG_ERROR | GET_NoCancel);                     1299
     }                                                                  1300

ddp->PointNum++;                                                        1301
ddp->vectdispparm->vectinfo.NumPoints++;                                1302

/* BottomRight corner of DTM */                                         1303
trans2d(ddp->DTMrastinfo.numcols,ddp->DTMrastinfo.numlins,forward,      1304
     &point.x,&point.y);                                                1305

/* Point Three */                                                       1306
vector[2].x = point.x;                                                  1307
vector[2].y = point.y;                                                  1308
```

```
vector[2].z = 0;

if(point.x < ddp->vectdispparm->vectinfo.minval.x)
     ddp->vectdispparm->vectinfo.minval.x = point.x;
if(point.y < ddp->vectdispparm->vectinfo.minval.y)
     ddp->vectdispparm->vectinfo.minval.y = point.y;
if(point.z < ddp->vectdispparm->vectinfo.minval.z)
     ddp->vectdispparm->vectinfo.minval.z = point.z;
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)
     ddp->vectdispparm->vectinfo.maxval.x = point.x;
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)
     ddp->vectdispparm->vectinfo.maxval.y = point.y;
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)
     ddp->vectdispparm->vectinfo.maxval.z = point.z;

if(point.x < line.minval.x) line.minval.x = point.x;
if(point.y < line.minval.y) line.minval.y = point.y;
if(point.z < line.minval.z) line.minval.z = point.z;
if(point.x > line.maxval.x) line.maxval.x = point.x;
if(point.y > line.maxval.y) line.maxval.y = point.y;
if(point.z > line.maxval.z) line.maxval.z = point.z;

if (MfWriteVPoint(ddp->vid,ddp->PointNum,&point) < 0)
     {
     MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",
               NULL,XmDIALOG_ERROR | GET_NoCancel);
     }

ddp->PointNum++;
ddp->vectdispparm->vectinfo.NumPoints++;

/* BottomLeft corner of DTM */

/* Point Four */
vector[3].x = point.x;
vector[3].y = point.y;
vector[3].z = 0;

if(point.x < ddp->vectdispparm->vectinfo.minval.x)
     ddp->vectdispparm->vectinfo.minval.x = point.x;
if(point.y < ddp->vectdispparm->vectinfo.minval.y)
     ddp->vectdispparm->vectinfo.minval.y = point.y;
if(point.z < ddp->vectdispparm->vectinfo.minval.z)
     ddp->vectdispparm->vectinfo.minval.z = point.z;
if(point.x > ddp->vectdispparm->vectinfo.maxval.x)
     ddp->vectdispparm->vectinfo.maxval.x = point.x;
if(point.y > ddp->vectdispparm->vectinfo.maxval.y)
     ddp->vectdispparm->vectinfo.maxval.y = point.y;
if(point.z > ddp->vectdispparm->vectinfo.maxval.z)
     ddp->vectdispparm->vectinfo.maxval.z = point.z;

if(point.x < line.minval.x) line.minval.x = point.x;
if(point.y < line.minval.y) line.minval.y = point.y;
if(point.z < line.minval.z) line.minval.z = point.z;
if(point.x > line.maxval.x) line.maxval.x = point.x;
if(point.y > line.maxval.y) line.maxval.y = point.y;
if(point.z > line.maxval.z) line.maxval.z = point.z;

if (MfWriteVPoint(ddp->vid,ddp->PointNum,&point) < 0)
```

```
        {                                                                    1359
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVPointWritten",       1360
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         1361
        }                                                                    1362
    ddp->PointNum++;                                                         1363
    ddp->vectdispparm->vectinfo.NumPoints++;                                 1364
    /* Line */                                                               1365
    line.left = line.right = line.start = line.end = -1;                     1366
    if (MfWriteVLine(ddp->vid,ddp->LineNum,&line) < 0)                       1367
        {                                                                    1368
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVLinehdrWritten",     1369
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         1370
        }                                                                    1371
    if (MfWriteVPoints(ddp->vid,ddp->LineNum,5,0L,5,&vector) < 0)            1372
        {                                                                    1373
        MxPopupMessageMt(ddp->toplevel,0,"geostruc","NoVLineWritten",        1374
                NULL,XmDIALOG_ERROR | GET_NoCancel);                         1375
        }                                                                    1376
    ddp->LineNum++;                                                          1377
    ddp->vectdispparm->vectinfo.NumLines++;                                  1378
    /* Redisplay vector */                                                   1379
    objwind = MxdwGetLayerObjWind(ddp->univectorlayer);                      1380
    objwind->ofullrect.xinit = ddp->vectdispparm->vectinfo.minval.x;         1381
    objwind->ofullrect.yinit = ddp->vectdispparm->vectinfo.minval.y;         1382
    objwind->ofullrect.xlast = ddp->vectdispparm->vectinfo.maxval.x;         1383
    objwind->ofullrect.ylast = ddp->vectdispparm->vectinfo.maxval.y;         1384
    MfWriteVectHeader(ddp->vectdispparm->fhandle,                            1385
        ddp->vectdispparm->vectinfo.objectinode,                             1386
        &ddp->vectdispparm->vectinfo);                                       1387
    MfUpdateFile(ddp->vectdispparm->fhandle);                                1388
    MxdwDisplayVector(ddp->univectorlayer);                                  1389
    MfCloseFile(fhandle);                                                    1390
    }                                                                        1391
/* Callback procedure display window, processing 3-point data to plane coords ***/  1392
static void FastEngage(                                                      1393
    Widget w,                                                                1394
    DDATA *ddp,                                                              1395
    XmAnyCallbackStruct *cbdata                                              1396
    ) {                                                                      1397
    int n,ni,i;                                                              1398
    Arg arg[20];                                                             1399
    Widget form,seclevel,menu;                                               1400
    DWMAIN dwmain;                                                           1401
    DWLAYER newlayer,reflayer,vectorlayer;                                   1402
    FNAMEINODE fnameinode;                                                   1403
    FNAMEINODE *fnameinodeptr;                                               1404
    MENUBARITEM menubaritems[8];                                             1405
    MENUITEM menuitems[4],menuitemsadd[4],menuitemsadd1[4];                  1406
    MXTHANDLE crosshairhandle;                                               1407
```

```
CROSSHAIRINFO *crosshairinfo;                                                    1408
VECTDISPPARM tempdispparm;                                                       1409
seclevel = ddp->toplevel;                                                        1410
dwmain = MxdwInit(seclevel,NULL);                                                1411
ddp->HasCoords = 1;                                                              1412
n = 0;                                                                           1413
XtSetArgI(arg,n,XmNhorizontalSpacing,10);                                        1414
XtSetArgI(arg,n,XmNverticalSpacing,10);                                          1415
form = XmCreateForm(seclevel,"form",arg,n);                                      1416
/* Initialize popup shell for entering three point data on raster */             1417
n = 0;                                                                           1418
ddp->thirdLevel = XtCreatePopupShell("point3",transientShellWidgetClass,         1419
          form,NULL,0);                                                          1420
++n;                                                                             1421
/* Initialize popup shell for displaying plane orientation */                    1422
ddp->fourthLevel = XtCreatePopupShell("planedisplay",transientShellWidgetClass,  1423
          form,NULL,0);                                                          1424
++n;                                                                             1425
ddp->FifthLevel = XtCreatePopupShell("point2",transientShellWidgetClass,         1426
          form,NULL,0);                                                          1427
++n;                                                                             1428
ddp->SixLevel = XtCreatePopupShell("Thickness",transientShellWidgetClass,        1429
          form,NULL,0);                                                          1430
++n;                                                                             1431
/* Create a display window */                                                    1432
n = 0;                                                                           1433
XtSetArgI(arg,n,XmNleftAttachment,XmATTACH_FORM);                                1434
XtSetArgI(arg,n,XmNbottomAttachment,XmATTACH_FORM);                              1435
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                               1436
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_FORM);                                 1437
XtSetArgI(arg,n,XmNtopOffset,27);                                                1438
MxdwCreateWindow(dwmain,form,arg,n,seclevel,600,600,"winclass","winfield",       1439
          &ddp->newwind,&ddp->newgroup,&reflayer,NULL);                          1440
/* Setup Crosshair tool for data access */                                       1441
crosshairhandle = MxdwtAddEditTool(ddp->newwind,"crosshair",NULL,0,mxtCrossHair);1442
MxdwtSetSaveCallback(crosshairhandle,SaveFunc,ddp);                              1443
/* Create a menu bar for display controls */                                     1444
n = 0;                                                                           1445
memset(menuitems,0,sizeof(menuitems));                                           1446
menuitems[n].label = "Analysis";                                                 1447
menuitems[n].callback = CB_3PtSelection;                                         1448
menuitems[n].callback_data = ddp;                                                1449
++n;                                                                             1450
menuitems[n].label = "Extents";                                                  1451
menuitems[n].callback = FbsSetDTMextents;                                        1452
menuitems[n].callback_data = ddp;                                                1453
++n;                                                                             1454
menuitems[n].label = "Exit";                                                     1455
menuitems[n].callback = CB_Close;                                                1456
```

```
++n;                                                                    1457

n = 0;                                                                  1458
memset(menuitemsadd1,0,sizeof(menuitemsadd1));                          1459
menuitemsadd1[n].label = "Vector";                                      1460
menuitemsadd1[n].callback = CB_VectEnhance;                             1461
menuitemsadd1[n].callback_data = ddp;                                   1462
++n;                                                                    1463
menuitemsadd1[n].label = "Raster";                                      1464
menuitemsadd1[n].callback = CB_RastEnhance;                             1465
menuitemsadd1[n].callback_data = ddp;                                   1466
++n;                                                                    1467

memset(menubaritems,0,sizeof(menubaritems));                            1468
ni=0;                                                                   1469
menubaritems[ni].label = "System";                                      1470
menubaritems[ni].items = menuitems;                                     1471

menubaritems[ni].label = "Enhance";                                     1472
menubaritems[ni].items = menuitemsadd1;                                 1473
++ni;                                                                   1474

menubaritems[ni].label = "View";                                        1475
menubaritems[ni].items = MxdwGetDftViewMenu(ddp->newwind);              1476
++ni;                                                                   1477

menubaritems[ni].label = "Tool";                                        1478
menubaritems[ni].items = MxdwGetDftToolMenu(ddp->newwind);              1479
++ni;                                                                   1480

menu = MxCreateMenuBar(form,menubaritems,NULL,NULL);                    1481

/* Make up FNAMEINODE struct from filename and inode and rastinfo */    1482

fnameinode.filename = ddp->ifilename;                                   1483
fnameinode.inode = ddp->rastinode;                                      1484

fnameinodeptr = &fnameinode;                                            1485

/* Display the selected raster in the display window */                 1486

MxdwCreateLayerRaster(ddp->newgroup,reflayer,&newlayer,fnameinodeptr,   1487
          NULL,DISPWIND_KeepCoordTrans);                                1488
ddp->currentlayer = newlayer;                                           1489

MxdwGetLayerProjExtents(newlayer,&ddp->rastextents);                    1490
ddp->rastprojparm = MxdwGetLayerProjParm(newlayer);                     1491
CreateVector(ddp);                                                      1492

/* Create a vector layer that will be displayed on top of the raster layer  1493
thus enabling us to highlight the following;                            1494

1.  Boundaries of the associated DTM as outlined on the raster.         1495
2.  Construct small highlighted triangles ( ahhhh cute!  )  for each    1496
3-point cluster selected for analysis.  */                              1497

/* Setup tempory structure for display parameters */                    1498
memset(&tempdispparm,0,sizeof(tempdispparm));                           1499

strcpy(tempdispparm.filename,ddp->Vecfilename);                         1500

memcpy(&tempdispparm.vectinfo,&ddp->Vecinfo,sizeof(RVCVECTINFO));       1501
tempdispparm.fhandle = ddp->vecfhandle;                                 1502
tempdispparm.ohandle = ddp->vid;                                        1503
```

```
     tempdispparm.georefinode = ddp->vecgeorefinode;                                    1504
     tempdispparm.Point.Select.Mode = SELECTMODE_All;                                   1505
     tempdispparm.Line.Select.Mode = SELECTMODE_All;                                    1506
     tempdispparm.Label.Select.Mode = SELECTMODE_All;                                   1507
     tempdispparm.dispflags = DISPFLAG_EditObject | DISPFLAG_KeepOpen;                  1508

     MxdwCreateLayerVector(ddp->newgroup,newlayer,&vectorlayer,NULL,                    1509
               &tempdispparm,DISPWIND_QuickAdd | DISPWIND_KeepOpen);                    1510

     ddp->univectorlayer = vectorlayer;                                                 1511

     XtManageChild(menu);                                                               1512
     XtManageChild(form);                                                               1513
     }                                                                                  1514
/* Callback to initiate graphics display and analysis ***/                             1515

static void CB_ExecProcess(                                                             1516
     Widget w,                                                                          1517
     DDATA *ddp,                                                                        1518
     XmAnyCallbackStruct *cbdata                                                        1519
     ) {                                                                                1520
     FastEngage(w,ddp,cbdata);                                                          1521

     if (MxdwRedrawWindow(ddp->newwind) < 0)                                            1522
          {                                                                             1523
          MxPopupMessage(ddp->toplevel,0,"Redraw Not Activated","NoHelp",               1524
                    XmDIALOG_ERROR | GET_NoCancel);                                     1525
          }                                                                             1526
     }                                                                                  1527
/* Callback procedure for selecting input raster ***/                                  1528

static void CB_RastInExec(                                                              1529
     Widget w,                                                                          1530
     DDATA *ddp,                                                                        1531
     XmAnyCallbackStruct *cbdata                                                        1532
     ) {                                                                                1533
     int err;                                                                           1534

     err = MfGetRast(ddp->toplevel,ddp->ifilename,&ddp->rastinode,                      1535
               "Select Input Raster",&ddp->rastinfo,NULL,NULL,0);                       1536
     if (err == EUserCancel) return;                                                    1537
     if (err < 0)                                                                       1538
          { MxDisplayErrorCode(ddp->toplevel,err);                                      1539
          return; }                                                                     1540

     /* Select georef object as well */                                                 1541
     err = MfGetGeoref(ddp->toplevel,ddp->ifilename,ddp->rastinode,                     1542
               &ddp->georefinode,"georef",&ddp->georefinfo,NULL,NULL,0);                1543
     if (err == EUserCancel) return;                                                    1544
     if (err < 0)                                                                       1545
          { MxDisplayErrorCode(ddp->toplevel,err);                                      1546
          return; }                                                                     1547

     MxTextSetRVCFilenameUnicode(ddp->TFinput,ddp->ifilename,ddp->rastinfo.name);       1548
     ddp->hasinputraster = 1;                                                           1549
     if (ddp->hasinputraster && ddp->hasinputDTM && ddp->hasVoutput                     1550
                    && ddp->hasoutput)                                                  1551
          XtSetSensitive(ddp->PBfastengage,TRUE);                                       1552
          }                                                                             1553
```

```
    }                                                                    1554
/* Callback procedure for selecting input DTM raster ***/               1555
static void CB_AssocDTM(                                                 1556
    Widget w,                                                            1557
    DDATA *ddp,                                                          1558
    XmAnyCallbackStruct *cbdata                                          1559
    ) {                                                                  1560
    int err;                                                             1561
    err = MfGetRast(ddp->toplevel,ddp->iDTMfilename,&ddp->DTMrastinode,  1562
                "Select Input DTM",&ddp->DTMrastinfo,NULL,NULL,0);       1563
    if (err < 0)                                                         1564
        { MxDisplayErrorCode(ddp->toplevel,err);                         1565
        return; }                                                        1566
    /* Select georef object as well */                                   1567
    err = MfGetGeoref(ddp->toplevel,ddp->iDTMfilename,ddp->DTMrastinode, 1568
            &ddp->DTMgeorefinode,"georef",&ddp->DTMgeorefinfo,NULL,NULL,0); 1569
    if (err == EUserCancel) return;                                      1570
    if (err < 0)                                                         1571
        { MxDisplayErrorCode(ddp->toplevel,err);                         1572
        return; }                                                        1573
    MxTextSetRVCFilenameUnicode(ddp->TFdtmInput,ddp->iDTMfilename,       1574
            ddp->DTMrastinfo.name);                                      1575
    ddp->hasinputDTM = 1;                                                1576
    if (ddp->hasinputraster && ddp->hasinputDTM && ddp->hasVoutput       1577
            && ddp->hasoutput)                                           1578
        {                                                                1579
        XtSetSensitive(ddp->PBfastengage,TRUE);                          1580
        }                                                                1581
    }                                                                    1582
/* Callback to select vector object if want or create new ***/          1583
static void CB_VecObjectIn(                                              1584
    Widget w,                                                            1585
    DDATA *ddp,                                                          1586
    XmAnyCallbackStruct *cbdata                                          1587
    ) {                                                                  1588
    int err;                                                             1589
    strtouc(ddp->Vecinfo.name,"VectorOverlay");                          1590
    strtouc(ddp->Vecinfo.desc,"3-Cluster Selection");                    1591
    err = MfGetVect(ddp->toplevel,ddp->Vecfilename,&ddp->Vecinode,       1592
            "Select Vector",&(ddp->Vecinfo),NULL,(void*)ddp,GETRVC_NewOK); 1593
    if (err == EUserCancel)                                              1594
        { return; }                                                      1595
    if(ddp->Vecinfo.NumPoints > 0)                                       1596
        {                                                                1597
        err = MfGetGeoref(ddp->toplevel,ddp->Vecfilename,ddp->Vecinode,  1598
                &ddp->vecgeorefinode,"georef",&ddp->vecgeorefinfo,NULL,NULL,0); 1599
        if (err == EUserCancel) return;                                  1600
        if (err < 0)                                                     1601
            { MxDisplayErrorCode(ddp->toplevel,err);                     1602
            return; }                                                    1603
```

```
        }                                                                    1604
    MxTextSetRVCFilenameUnicode(ddp->TFVoutput,ddp->Vecfilename,             1605
            ddp->Vecinfo.name);                                              1606
    ddp->hasVoutput = 1;                                                     1607
    if (ddp->hasinputraster && ddp->hasinputDTM && ddp->hasVoutput           1608
                && ddp->hasoutput)                                           1609
        {                                                                    1610
        XtSetSensitive(ddp->PBfastengage,TRUE);                              1611
        }                                                                    1612
    }                                                                        1613
/* Callback procedure for selecting an output Data file ***/                1614
static void CB_DataFileExec(                                                 1615
    Widget w,                                                                1616
    DDATA *ddp,                                                              1617
    XmAnyCallbackStruct *cbdata                                              1618
    ) {                                                                      1619
    int err;                                                                 1620
    err = MxGetFile(ddp->toplevel,ddp->ofilename,"geo",NULL,NULL,NULL,       1621
            GETFILE_NewOK);                                                  1622
    if (err < 0)                                                             1623
        {                                                                    1624
        return;                                                              1625
        }                                                                    1626
    MxTextSetRVCFilename(ddp->TFoutput,"file",ddp->ofilename);               1627
    ddp->hasoutput = 1;                                                      1628
    if (ddp->hasinputraster && ddp->hasoutput && ddp->hasinputDTM            1629
                && ddp->hasVoutput)                                          1630
        {                                                                    1631
        XtSetSensitive(ddp->PBfastengage,TRUE);                              1632
        }                                                                    1633
    }                                                                        1634
/* Main Program Control Loop ***/                                           1635
int main(                                                                    1636
int argc,                                                                    1637
char *argv[]                                                                 1638
) {                                                                          1639
int n,ni;                                                                    1640
Widget toplevel,form,menu,iform,dtmform,vecform,oform;                       1641
Arg arg[20];                                                                 1642
DDATA ddata;                                                                 1643
MENUBARITEM menubaritems[8];                                                 1644
MENUITEM menuitems[4];                                                       1645
memset(&ddata,0,sizeof(DDATA));                                              1646
ddata.toplevel = MxInit(&argc,argv,NULL);                                    1647
/* Routines for initializing */                                              1648
VMInit(256*1024);                                                            1649
OFontInit();                                                                 1650
MtTextSetTitle(ddata.toplevel,"ssgc",NULL);                                  1651
n = 0;                                                                       1652
XtSetArgI(arg,n,XmNhorizontalSpacing,10);                                    1653
```

```
XtSetArgI(arg,n,XmNverticalSpacing,10);                                    1654
form = XmCreateForm(ddata.toplevel,"form",arg,n);                          1655

/* Create the menubar at the top of the form */                            1656

n = 0;                                                                     1657
memset(menuitems,0,sizeof(menuitems));                                     1658
menuitems[n].label = "Close";                                              1659
menuitems[n].callback = CB_Close;                                          1660
++n;                                                                       1661
menuitems[n].label = "Run...";                                             1662
menuitems[n].callback = CB_ExecProcess;                                    1663
menuitems[n].flags = MxMENU_DISABLED;                                      1664
menuitems[n].callback_data = &ddata;                                       1665
++n;                                                                       1666

memset(menubaritems,0,sizeof(menubaritems));                               1667
ni=0;                                                                      1668
menubaritems[ni].label = "File";                                           1669
menubaritems[ni].items = menuitems;                                        1670
++ni;                                                                      1671

menu = MxCreateMenuBar(form,menubaritems,NULL,NULL);                       1672
ddata.PBfastengage = menuitems[1].widget;                                  1673
XtSetSensitive(ddata.PBfastengage,FALSE);                                  1674

/* select raster file for input */                                         1675
n = 0;                                                                     1676
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                         1677
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                         1678
XtSetArgI(arg,n,XmNtopWidget,menu);                                        1679
iform = MxCreateSelectField(form,"Input Raster",NULL,&ddata.TFinput,       1680
                &ddata.PBinput,arg,n);                                     1681
XtAddCallback(ddata.PBinput,XmNactivateCallback,CB_RastInExec,&ddata);     1682

/* select corresponding dtm for above raster */                            1683
n = 0;                                                                     1684
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                         1685
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                         1686
XtSetArgI(arg,n,XmNtopWidget,iform);                                       1687
dtmform = MxCreateSelectField(form,"Associated DTM",NULL,&ddata.TFdtmInput,1688
                &ddata.PBdtmInput,arg,n);                                  1689
XtAddCallback(ddata.PBdtmInput,XmNactivateCallback,CB_AssocDTM,&ddata);    1690

/* select vector object to save 3cluster points */                         1691
n = 0;                                                                     1692
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                         1693
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                         1694
XtSetArgI(arg,n,XmNtopWidget,dtmform);                                     1695
vecform = MxCreateSelectField(form,"Vector Object",NULL,&ddata.TFVoutput,  1696
                &ddata.PBVoutput,arg,n);                                   1697
XtAddCallback(ddata.PBVoutput,XmNactivateCallback,CB_VecObjectIn,&ddata);  1698

/* select output file for storing 3-point data */                          1699
n = 0;                                                                     1700
XtSetArgI(arg,n,XmNrightAttachment,XmATTACH_FORM);                         1701
XtSetArgI(arg,n,XmNtopAttachment,XmATTACH_WIDGET);                         1702
XtSetArgI(arg,n,XmNtopWidget,vecform);                                     1703
oform = MxCreateSelectField(form,"Data File",NULL,&ddata.TFoutput,         1704
```

```
                &ddata.PBoutput,arg,n);                                          1705
XtAddCallback(ddata.PBoutput,XmNactivateCallback,CB_DataFileExec,&ddata);        1706

XtManageChild(iform);                                                            1707
XtManageChild(dtmform);                                                          1708
XtManageChild(vecform);                                                          1709
XtManageChild(oform);                                                            1710
XtManageChild(form);                                                             1711
XtManageChild(menu);                                                             1712

XtRealizeWidget(ddata.toplevel);                                                 1713

XtAppMainLoop(appcontext);                                                       1714
return(0);                                                                       1715
}                                                                                1716
```

APPENDIX C

SOURCE CODE FOR PROGRAM UTILIZED IN PREPARING DATA

The program presented in this Appendix is written in C compiled with the Watcom Compiler (Section 2.4.1.). The code should be portable to most other compilers that target the DOS 16–bit environment.

The program converts DEM data files in the CINLIS format to simple arrayed binary format. The program can also be used to read information from SPOT Leader Files by not utilizing the converter part of the program.

To convert DEM data files the user is required to discard the file header for each file by searching for the beginning of the image data. Once the position of the initial image data in the file has been found, a straight forward conversion from ASCII to binary takes place.

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

FILE *in, *out;
char buffer[500];
char inbuf[30];
long discard,cursor;
int y,k=1,elev;
char name[30];

main()
{
printf("\n Input file name :  ");
gets(inbuf);
sscanf(inbuf,"%s",&name);
if ((in = fopen(name,"rt")) == NULL)
     { fprintf(stderr,"Cannot open input file.\n");
     return 1;}
if ((out = fopen("test.dat","wb")) == NULL)
     { fprintf(stderr,"Cannot open output file.\n");
     return 1;}
```

```
printf("\n Search file for start?  y/n");
while ((y = getch()) != 'n')
     {
     printf("\n enter chars to skip ");
     gets(inbuf);
     sscanf(inbuf,"%dl",&discard);
     fseek(in,discard,SEEK_SET);
     fread(buffer,20,1,in);
     printf("char= %s",buffer);
     printf("\n Search again?  y/n");
     }
printf("\n\n\n creating structure...");
fseek(in,discard,SEEK_SET);
while (k != 0)
     {
     if (fgets(buffer,5,in) == NULL)
     { break; }
     cursor = ftell(in);
     sscanf(buffer,"%d",&elev);
     fwrite(&elev,sizeof(int),1,out);
     if (fseek(in,cursor+1,SEEK_SET) != 0)
          {printf("\nfile pointer not located properly"); break;}
     }
printf("\n\n ...Structure completed");
sleep(3);
flushall();
fclose(in);
fclose(out);
return 1;
}
```

APPENDIX D

GEOSTRUC V3.14 USER MANUAL

D.1. Introduction

This application note describes the use of the GEOSTRUC© customized software routine, written as an additional routine to the existing TNTmips image processing package distributed by MicroImages Inc. The creation of the GEOSTRUC© routine was motivated by the need for a semi–automated process that analizes subsurface geological structures on remotely sensed images. It provides a user–friendly, interactive platform for fast assesment of an area's subsurface geology. The advantages of using such a technique are obvious in cases where terrain access by a mapping team is difficult due to rugged terrain, and where the area that is to be covered is very large. Used in conjunction with limited fieldwork, GEOSTRUC© provides a quick method for structural reconnaissance mapping covering large regions.

The process interactively extracts dip and dip direction of planar structures that have a surface expresssion and calculates true thickness, if required. The data is written as a TNTmips compatible vector object and in an ASCII file format. This allows the user to further process the data via spreadsheets, databases and stereonet programs.

D.2. Preparation

The process makes use of a remotely sensed image of the required locality as well as a Digital Elevation Model (DEM) of the same area. Both the image and the DTM must be in the RVC–raster format supported by TNTmips. Examples of images are;

(1) SPOT Panchromatic images processed to level S.

(2) Digital aerial photographs (corrected for distortion).

(3) Orthophotos.

(4) Landsat TM images.

Examples of DEMs are;

(1) DEMs derived by stereo–correlation of a SPOT stereo–pair.

(2) National Digital Elevation Model (NDEM) data purchased from Government Survey.

(3) DEMs derived from topographical sheets by digitizing contours, resampling and interpolation.

(4) DEMs derived from Radar Altimetry.

(5) DEMs produced by photogrammetric methods.

TNTmips offers all the necessary routines for the construction of DEMs and the importing of images from source formats. It is critically important that the image and the DEM overlap in the desired area of investigation. Furthermore, both rasters must be georeferenced in the Latitude/Longitude projection system. If the rasters are not initially in this system, then there are processes in TNTmips that can be used to convert them into the required projection system.

D.3. Operation

The process is initiated by activating **custom-geostruc** from the main TNTmips menu. The first window to appear is an input window, **Fig. D1.**, requesting the user to select the image that will be used for the interpretation process (Input Raster), the DEM partially or fully overlapping the image (Associated DEM), a vector object that is mainly used in the graphics display of selected structures (Vector Object), and an ASCII file for the output data (Data File).

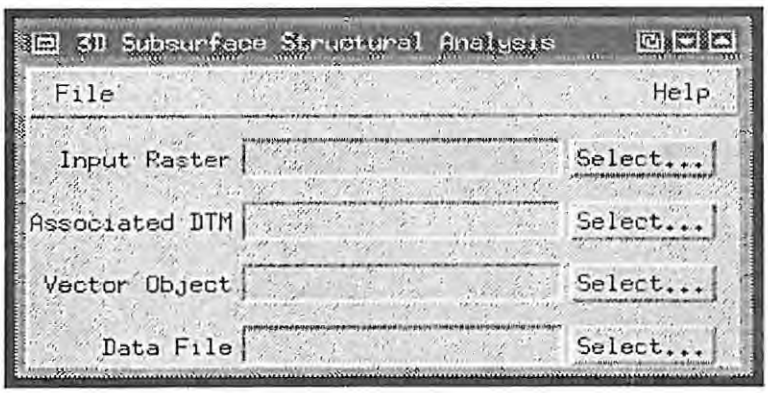


Fig. D1. The Input window.

Except for the ASCII file the user will be prompted to select a georeference subobject that has to be in the Latitude/Longitude projection format. If a new job is started the vector object selected must be a new one, and the user will not be prompted to select a georeference subobject as GEOSTRUC© will automatically supply the vector object with the same georeference subobject as the input raster. On successful completion of the input objects the **file-run** text will become active enabling the start of the process.

When the user clicks on the **run** option in the **file** pulldown menu, the current input window is closed and replaced by a graphics display window displaying the selected input raster overlaid with the selected vector object. Note that the DEM exists only in the background of this process, and, although accessible by the process, is not displayed in this window. That is, access to the DEM is transparent to the user. If a new process has been initiated the vector object will contain no elements.

The display window has the following options in its menubar; **system** for exiting the process, initializing the structural examination and displaying the extents of the DEM, **enhance** for standard display controls of the raster and the vector object, **view** for changing the display area size, and, finally, **tool** for activating the crosshair and zoombox tools.

Before beginning the analysis the user may want to outline the area covered by the DEM on the input raster. This can be done by activating the **system-extents** option on the menubar. The outline of the DEM should appear on the screen, overlaying the input raster.

To start the analysis the user is required to select an area where subsurface structures have suitable surface expression. Use the **tool-zoombox** option to zoom into such areas. Now select the **system-analysis** option from the menubar. This activates a window, **Fig. D2.**, that requires the user to input three points on the structural plane under examination.

The three points are selected by means of the crosshair activated by selecting the **tool-crosshair** option. Position the crosshair with the pointing device. Once the crosshair is in the desired position, click the righthand button on the pointing device. This action will retrieve the Latitude/Longitude coordinates for the specified pixel. GEOSTRUC© automatically accesses the DEM at the specified lat/long coordinates in orfer to obtain

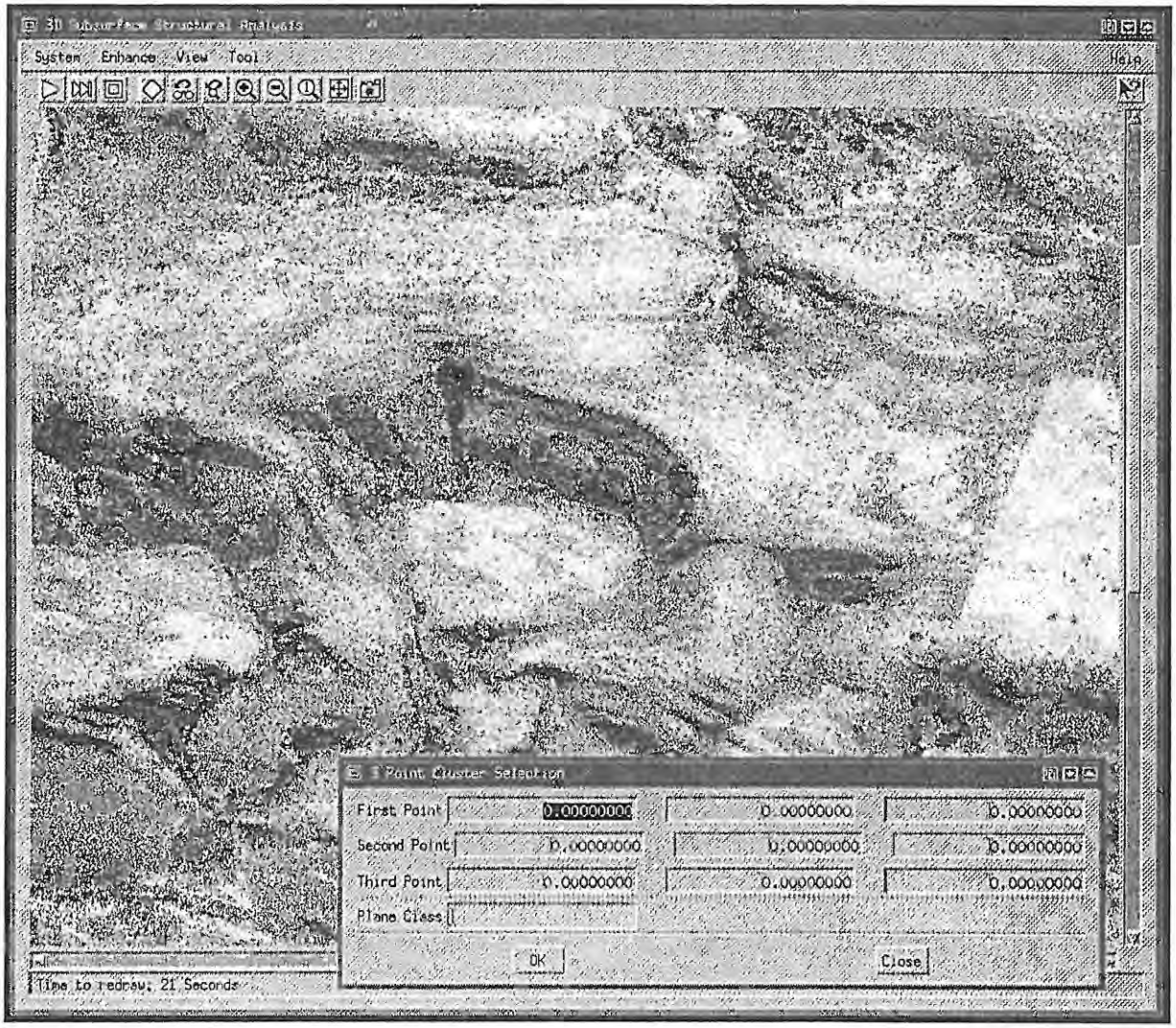


Fig. D2. Input raster and 3–Point selection window.

the true elevation for that geographical location. If no DEM coverage is available for a specific Lat/Long coordinate, the program will notify the user and write a zero as the third coordinate. Proceed to select a further two points on the input raster and optionally fill in the classification text input area with the class of the structure being measured, for example BED, FAULT or DYKE. After completion of the above the user is now ready to perform the plane orientation calculation.

Click on the **OK** button in the 3–point cluster selection window. GEOSTRUC© firstly transforms the three points into a local coordinate system. It then calculates the orientation of a plane that uniquely fits the three points. The result of this calculation is displayed in a text window which details the dip direction (trend) and dip (plunge) of the

plane. For purposes of later identification the coordinates of a point within the bounds of the triangle, defined by the three points, are also determined in the original coordinate system. This point is called the centroid.

The user now has the option to calculate the true thickness of a unit under investigation. Click on the **thickness** button in the plane orientation text window. This action opens another input window prompting the user to select two points on the input image. Position one of these points at the top of the unit and one at the base of the unit, using the same selection procedure as for the 3–point input. Click on the **OK** button. A popup window will display the true thickness. The thickness of the structure can be determined at several locations along the strike of the unit, but, for best results, these should be confined to the close vicinity of the associated plane orientation calculation. A toggle button in this popup window provides the user with the option of writing the last thickness measured to the ASCII data file (**Fig. D3.**).

Finally, if the user is satisfied that the results seem reasonable the **add** button is clicked on. This action firstly writes the triangle coordinates (the three selected points) to a vector object and actually draws a triangle on the input image. Secondly, the relevant information is saved to the comma delimited ASCII data file in the following format; centroid-longitude, centroid-latitude, centroid-elevation, dip direction; dip, classification and, optionally, thickness. Note that if the thickness is not written to the output file, a zero will be printed instead. The user can now proceed with the next measurement.

The output data is written to the output files as the process continues, that is, there is no need to save data at any time. If more information needs to be added to a data file during a later session, it is simply selected at the input stage of the GEOSTRUC© process. In other words, the complete analysis does not have to be completed in one session.

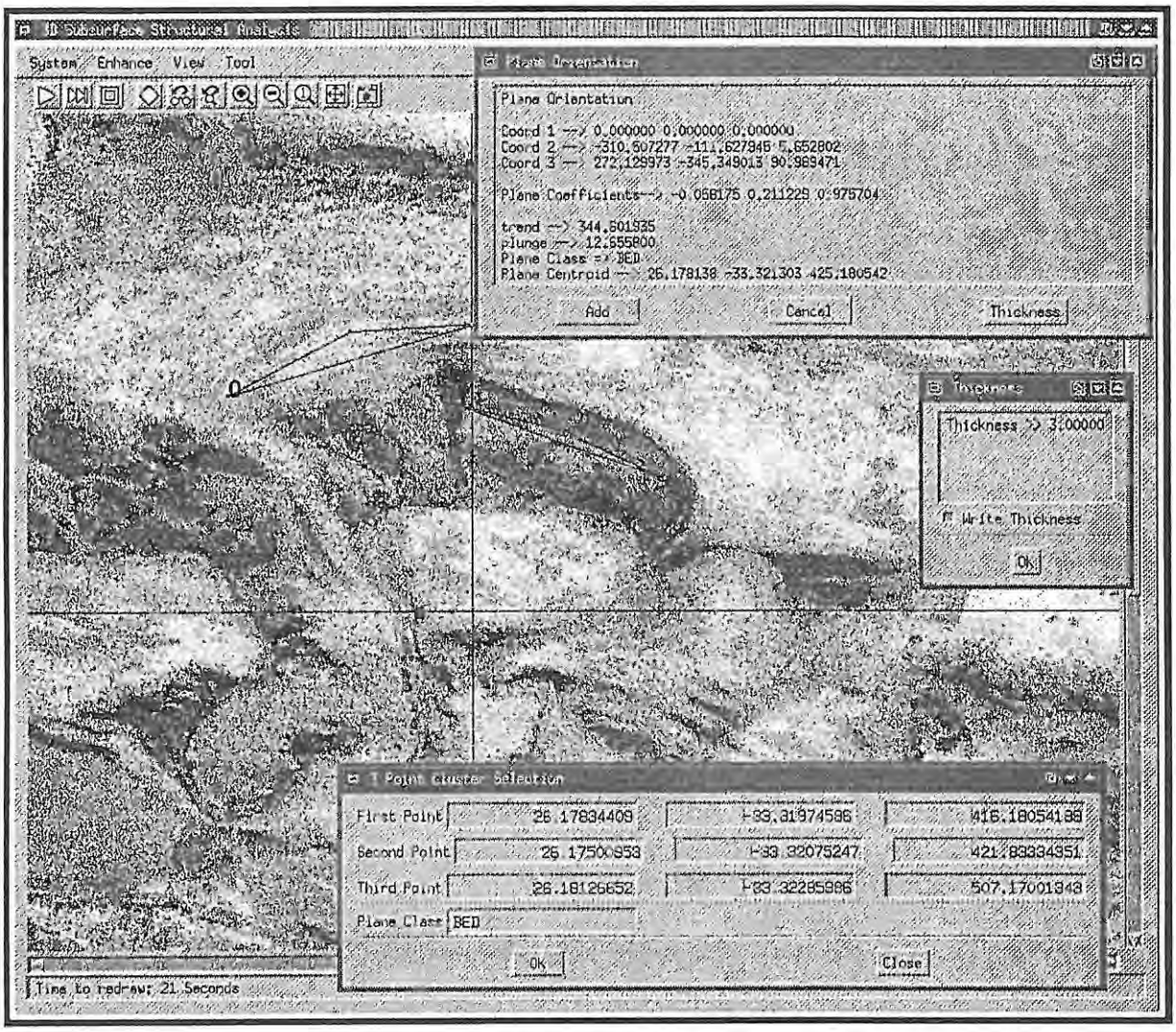


Fig. D3. Selected geological structure with text window and thickness popup.

APPENDIX E

ASCII FILES GENERATED BY GEOSTRUC

E.1 The Alicedale Case Study : Program Application One

125,26.162214,-33.304956,581.333313, 5, 30,BED, 0
126,26.162185,-33.305968,594.700684, 1, 27,BED, 0
127,26.163067,-33.306659,562.295532, 7, 15,BED, 0
128,26.172180,-33.305845,531.444458, 14, 14,BED, 0
129,26.174033,-33.306487,539.312622, 33, 9,BED, 0
130,26.178031,-33.307326,523.888916, 28, 16,BED, 0
131,26.186351,-33.306635,481.666656,356, 7,BED, 0
132,26.193377,-33.311052,558.082397,341, 3,BED, 0
133,26.160039,-33.309498,534.666656,169, 38,BED, 0
134,26.158187,-33.314286,533.333313,192, 22,BED, 0
135,26.157893,-33.314730,503.222229,197, 28,BED, 0
136,26.157657,-33.315569,459.444458,193, 13,BED, 0
137,26.162067,-33.316087,507.444458,192, 43,BED, 0
138,26.160362,-33.315297,502.666656,199, 37,BED, 0
139,26.160450,-33.316605,459.777771,192, 41,BED, 0
140,26.159715,-33.317765,377.666656, 3, 32,BED, 0
141,26.182735,-33.312484,563.741333, 8, 22,BED, 0
142,26.184881,-33.311990,516.000000,332, 7,BED, 0
143,26.190261,-33.312435,567.780396, 8, 38,BED, 0
144,26.172827,-33.313002,572.306152, 68, 3,BED, 0
145,26.172739,-33.313520,563.387634,129, 4,BED, 0
146,26.175238,-33.313915,546.440308,351, 9,BED, 0
147,26.164243,-33.320875,479.662689, 14, 8,BED, 0
148,26.164419,-33.320653,473.402771, 15, 12,BED, 0
149,26.171798,-33.318975,454.000000,180, 38,BED, 0
150,26.171739,-33.319567,469.000000,193, 57,BED, 0
151,26.172768,-33.320776,452.666656,195, 48,BED, 0
152,26.172827,-33.321023,453.000000,195, 54,BED, 0
153,26.173004,-33.321344,459.333344,192, 46,BED, 0
154,26.177237,-33.319493,437.000000,186, 16,BED, 0
155,26.176473,-33.320628,459.333344,188, 31,BED, 0
156,26.183617,-33.318136,532.666687,209, 23,BED, 0
157,26.184557,-33.318901,538.111084,190, 17,BED, 0
158,26.190319,-33.321369,606.000000,205, 21,BED, 0

159,26.192407,-33.321838,608.333313,180, 10,BED, 0
160,26.185498,-33.322430,526.000000,207, 24,BED, 0
161,26.181911,-33.322701,505.111115,199, 49,BED, 0
162,26.181559,-33.322899,499.000000,199, 53,BED, 0
163,26.187203,-33.323664,547.000000,193, 35,BED, 0
164,26.188438,-33.323244,551.383423,201, 26,BED, 0
165,26.189555,-33.324898,473.000000,220, 29,BED, 0
166,26.190143,-33.324281,528.333313,216, 30,BED, 0
167,26.194288,-33.329859,492.444458,198, 40,BED, 0
168,26.194406,-33.330278,498.666656,184, 23,BED, 0
169,26.181823,-33.324923,427.666656,209, 10,BED, 0
170,26.170005,-33.324404,386.958405,246, 7,BED, 0
171,26.160597,-33.322455,450.535095,207, 12,BED, 0
172,26.188026,-33.321147,537.070984,257, 11,BED, 0
173,26.170564,-33.317420,495.666656,211, 36,BED, 0
174,26.159686,-33.333131,448.103271,317, 10,BED, 0
175,26.164272,-33.334958,508.247681,190, 34,BED, 0
176,26.165536,-33.334217,527.822449,192, 23,BED, 0
177,26.176737,-33.336167,508.109222, 12, 8,BED, 0
178,26.171681,-33.334982,494.197601,298, 10,BED, 0
179,26.173592,-33.336439,524.495239,268, 7,BED, 0
180,26.175826,-33.338536,529.666687,201, 35,BED, 0
181,26.183999,-33.346952,516.648193,352, 13,BED, 0
182,26.174973,-33.343596,554.468750, 25, 20,BED, 0
183,26.176296,-33.343966,547.145386, 24, 21,BED, 0
184,26.168594,-33.344731,494.943451,247, 20,BED, 0
185,26.161303,-33.343152,469.776611, 6, 40,BED, 0
186,26.162038,-33.343843,493.239929, 7, 32,BED, 0
187,26.163184,-33.344509,505.079742, 12, 41,BED, 0
188,26.167330,-33.346434,501.458832, 2, 13,BED, 0
189,26.163449,-33.345891,480.074707, 9, 16,BED, 0
190,26.164096,-33.346483,506.622711, 15, 15,BED, 0
191,26.164390,-33.347323,520.469116, 14, 11,BED, 0
192,26.157510,-33.344855,453.217926, 7, 16,BED, 0
193,26.157305,-33.344386,446.435364, 3, 13,BED, 0
194,26.184910,-33.324716,485.333344,205, 52,BED, 0
195,26.157334,-33.321186,444.803619,209, 25,CONTAct, 0
196,26.158304,-33.325012,372.974976,278, 1,CONTAct, 0
197,26.162861,-33.325579,374.947937,232, 3,CONTAct, 0
198,26.165860,-33.324395,464.691620,189, 20,CONTAct, 0
199,26.167506,-33.327998,387.406677,332, 4,CONTAct, 0
200,26.165860,-33.324395,464.691620,187, 19,CONTAct, 0
201,26.169711,-33.325036,462.802155,206, 40,CONTAct, 0
202,26.170652,-33.337056,459.074158,221, 8,CONTAct, 0
203,26.175767,-33.316393,607.000000, 14, 54,AXIAL, 0
204,26.176855,-33.316517,589.555542, 14, 59,AXIAL, 0

205,26.178472,-33.316665,501.000000, 13, 48,AXIAL, 0
206,26.154512,-33.304646,548.896851,124, 43,FRAC, 0
207,26.160009,-33.309557,529.333344,170, 36,FRAC, 0
208,26.163508,-33.305189,446.333344, 31, 11,FRAC, 0
209,26.164625,-33.313481,557.513184, 99, 29,FRAC, 0
210,26.164625,-33.316739,557.333313,106, 24,FRAC, 0
211,26.174856,-33.314715,568.000000,139, 13,FRAC, 0
212,26.173415,-33.319898,529.333313, 70, 32,FRAC, 0
213,26.180765,-33.324415,730.222351,159, 78,FRAC, 0
214,26.171534,-33.326463,584.612213,172, 79,FRAC, 0
215,26.190613,-33.331374,509.656097,164, 48,FRAC, 0
216,26.169652,-33.334538,467.382538,280, 25,FRAC, 0
217,26.152777,-33.334267,512.124420,344, 34,FRAC, 0
218,26.155188,-33.331503,405.781891,267, 27,FRAC, 0
219,26.184969,-33.341572,500.071289,203, 15,FRAC, 0
220,26.187644,-33.343078,527.345581,236, 19,FRAC, 0

E.2. The Kareedouw Case Study : Program Application Two

E.2.1. Bedding Orientation for 3323DB.

0,23.795213,-33.596378,1015.000000, 31, 12,BED, 0
1,23.830597,-33.602651,2168.000000, 9, 88,BED, 0
2,23.865851,-33.609139,1238.000000, 13, 64,BED, 0
3,23.909142,-33.617250,1212.000000, 11, 39,BED, 0
4,23.954895,-33.624821,1109.000000,197, 32,BED, 0
5,23.783548,-33.600055,1519.000000, 9, 72,BED, 0
6,23.816988,-33.607085,1256.000000,342, 13,BED, 0
7,23.853020,-33.612060,1460.000000, 13, 72,BED, 0
8,23.914067,-33.619846,1435.000000,191, 85,BED, 0
9,23.931824,-33.618115,1139.000000,186, 56,BED, 0
10,23.956191,-33.615844,1762.000000,188, 88,BED, 0
11,23.952692,-33.629796,1415.000000,193, 76,BED, 0
12,23.807396,-33.622335,1402.000000, 7, 45,BED, 0
13,23.838892,-33.621794,1216.000000,188, 35,BED, 0
14,23.876479,-33.627417,1049.000000,187, 69,BED, 0
15,23.784454,-33.622227,1273.000000,189, 55,BED, 0
16,23.920159,-33.635636,1154.000000,226, 13,BED, 0
17,23.908105,-33.641044,1051.000000,196, 47,BED, 0
18,23.788473,-33.607302,1770.000000,188, 84,BED, 0
19,23.804415,-33.630014,1336.000000,160, 19,BED, 0
20,23.812451,-33.609897,1432.000000,201, 37,BED, 0
21,23.852890,-33.636502,1043.000000, 14, 23,BED, 0
22,23.840317,-33.626336,1151.000000,186, 68,BED, 0
23,23.790565,-33.647696,1263.000000,182, 14,BED, 0
24,23.790375,-33.646505,1211.000000,190, 32,BED, 0

25,23.779629,-33.632222,1889.000000,170, 86,BED, 0
26,23.776871,-33.637221,1269.000000,200, 32,BED, 0
27,23.847340,-33.654440,1420.000000,181, 25,BED, 0
28,23.826418,-33.650473,1198.000000,358, 71,BED, 0
29,23.861890,-33.667058,1522.000000,185, 81,BED, 0
30,23.859037,-33.654361,1503.000000,192, 15,BED, 0
31,23.785524,-33.656584,1347.000000, 2, 71,BED, 0
32,23.774778,-33.669677,1089.000000,195, 38,BED, 0
33,23.790659,-33.679914,975.000000, 6, 23,BED, 0
34,23.756709,-33.674677,953.000000, 9, 64,BED, 0
35,23.778962,-33.658568,1267.000000,179, 33,BED, 0
36,23.827844,-33.661979,1061.000000, 1, 44,BED, 0
37,23.827178,-33.672930,979.000000, 11, 69,BED, 0
38,23.796270,-33.673804,905.000000,186, 36,BED, 0
39,23.765458,-33.646744,1093.000000, 20, 44,BED, 0
40,23.776966,-33.618890,2019.000000,193, 88,BED, 0
41,23.763367,-33.604130,7897.000000, 8, 90,BED, 0
42,23.805211,-33.634364,1482.000000, 14, 40,BED, 0
43,23.933520,-33.674414,996.000000,326, 7,BED, 0
44,23.922230,-33.672988,1489.000000,354, 86,BED, 0
45,23.937042,-33.682884,903.000000,188, 53,BED, 0
46,23.905087,-33.661795,1162.000000,185, 68,BED, 0
47,23.899545,-33.690361,887.000000,179, 69,BED, 0
48,23.894418,-33.692133,746.000000, 21, 57,BED, 0
49,23.937145,-33.691657,1021.000000,188, 71,BED, 0
50,23.932121,-33.694769,819.000000, 2, 51,BED, 0
51,23.920210,-33.677266,928.000000,176, 54,BED, 0
52,23.948332,-33.681890,1105.000000,169, 23,BED, 0
53,23.882351,-33.663005,1209.000000,186, 29,BED, 0
54,23.894936,-33.669920,928.000000,191, 77,BED, 0
55,23.933986,-33.668580,1522.000000,185, 85,BED, 0
56,23.949524,-33.655658,1391.000000,203, 81,BED, 0
57,23.917776,-33.651337,1221.000000, 26, 82,BED, 0
58,23.934970,-33.663350,1338.000000,174, 11,BED, 0
59,23.938596,-33.651293,1288.000000,189, 59,BED, 0
60,23.946313,-33.646799,2101.000000, 7, 88,BED, 0
61,23.886856,-33.684095,1107.000000, 8, 64,BED, 0
62,23.909645,-33.676316,780.000000,349, 20,BED, 0
63,23.884163,-33.696109,855.000000,189, 74,BED, 0
64,23.877948,-33.684441,1107.000000,194, 71,BED, 0
65,23.923213,-33.703282,807.000000,191, 74,BED, 0
66,23.872562,-33.695763,1330.000000, 13, 84,BED, 0
67,23.950559,-33.706048,892.000000, 1, 59,BED, 0
68,23.959467,-33.705097,970.000000,180, 56,BED, 0
69,23.984015,-33.706393,1074.000000,183, 75,BED, 0
70,23.989298,-33.702244,930.000000,190, 43,BED, 0

71,23.980286,-33.723161,918.000000,192, 64,BED, 0
72,23.942583,-33.712530,960.000000,198, 60,BED, 0
73,23.884784,-33.708382,1223.000000,192, 85,BED, 0
74,23.841798,-33.698529,875.000000, 11, 27,BED, 0
75,23.839519,-33.720656,656.000000,106, 41,BED, 0
76,23.848221,-33.689973,1044.000000, 9, 57,BED, 0
77,23.851950,-33.687120,855.000000,169, 1,BED, 0
78,23.997585,-33.687119,1453.000000,355, 62,BED, 0
79,23.924090,-33.738570,627.000000,205, 40,BED, 0
80,23.862205,-33.736859,618.000000, 21, 49,BED, 0
81,23.877506,-33.722880,601.000000, 75, 21,BED, 0
82,23.896481,-33.745561,521.000000, 15, 57,BED, 0
83,23.918192,-33.730154,626.000000,212, 27,BED, 0
84,23.831731,-33.721031,805.000000, 26, 60,BED, 0
85,23.757697,-33.708523,840.000000,206, 83,BED, 0
86,23.756609,-33.698250,678.000000,186, 45,BED, 0
87,23.748904,-33.719005,762.000000,205, 73,BED, 0
88,23.804262,-33.690353,855.000000,187, 53,BED, 0
89,23.758117,-33.678683,980.000000,196, 31,BED, 0
90,23.831228,-33.729067,682.000000, 8, 44,BED, 0

E.2.2. Fracture Orientation for 3323DB.

0,23.797461,-33.646026,1202.000000,273, 60,FRAC, 0
1,23.785119,-33.646027,1057.000000,282, 68,FRAC, 0
2,23.774689,-33.645301,947.000000,281, 62,FRAC, 0
3,23.847525,-33.656180,1112.000000,111, 62,FRAC, 0
4,23.819190,-33.650378,906.000000,255, 32,FRAC, 0
5,23.905239,-33.611212,972.000000, 75, 46,FRAC, 0
6,23.770866,-33.596999,1000.000000,288, 90,FRAC, 0
7,23.963646,-33.636451,1191.000000,257, 33,FRAC, 0
8,23.770865,-33.641965,1114.000000,116, 70,FRAC, 0
9,23.757827,-33.637614,919.000000,288, 48,FRAC, 0
10,23.792073,-33.617451,1416.000000,108, 80,FRAC, 0
11,23.856757,-33.700649,846.000000,129, 46,FRAC, 0
12,23.853174,-33.700921,812.000000,159, 17,FRAC, 0
13,23.792013,-33.687534,1161.000000,111, 86,FRAC, 0
14,23.785498,-33.688350,809.000000, 90, 4,FRAC, 0
15,23.761881,-33.679244,1007.000000,116, 80,FRAC, 0
16,23.755610,-33.680875,759.000000,127, 53,FRAC, 0
17,23.837456,-33.698951,774.000000,108, 60,FRAC, 0
18,23.839980,-33.722259,685.000000,106, 37,FRAC, 0
19,23.837863,-33.720900,728.000000,117, 59,FRAC, 0
20,23.836153,-33.719541,813.000000,106, 73,FRAC, 0
21,23.831185,-33.719201,697.000000,119, 24,FRAC, 0
22,23.842179,-33.736122,553.000000,109, 34,FRAC, 0

23,23.825565,-33.730142,633.000000,282, 34,FRAC, 0
24,23.823204,-33.728579,737.000000,287, 76,FRAC, 0
25,23.745918,-33.725930,543.000000,105, 64,FRAC, 0
26,23.810499,-33.710503,697.000000,321, 44,FRAC, 0
27,23.812128,-33.703776,765.000000,325, 63,FRAC, 0
28,23.830941,-33.695553,936.000000,154, 80,FRAC, 0
29,23.815141,-33.722667,797.000000, 97, 65,FRAC, 0
30,23.764730,-33.725998,690.000000,233, 16,FRAC, 0
31,23.749746,-33.698884,752.000000,280, 23,FRAC, 0
32,23.845193,-33.704999,788.000000,280, 48,FRAC, 0
33,23.876710,-33.705882,2259.000000,114, 88,FRAC, 0
34,23.976300,-33.717569,1005.000000,270, 63,FRAC, 0
35,23.958546,-33.716754,841.000000,257, 37,FRAC, 0
36,23.999592,-33.694600,1492.000000,250, 28,FRAC, 0
37,23.886717,-33.716075,864.000000,244, 20,FRAC, 0
38,23.955615,-33.689979,900.000000,265, 11,FRAC, 0
39,23.986399,-33.675844,1341.000000,270, 73,FRAC, 0
40,23.951218,-33.674621,1268.000000,109, 39,FRAC, 0
41,23.929228,-33.707648,745.000000,264, 75,FRAC, 0
42,23.890301,-33.682776,808.000000,106, 35,FRAC, 0
43,23.916200,-33.636022,1259.000000,247, 16,FRAC, 0
44,23.909847,-33.669729,1100.000000,287, 69,FRAC, 0
45,23.990634,-33.674892,1257.000000,258, 54,FRAC, 0
46,23.835900,-33.652605,1462.000000,270, 76,FRAC, 0
47,23.935093,-33.680465,1082.000000,164, 29,FRAC, 0
48,23.995521,-33.637652,1055.000000, 4, 21,FRAC, 0

E.2.3. Axial Plane Orientation for 3323DB.

0,23.818862,-33.658511,1237.000000, 8, 81,AXIAL, 0
1,23.947329,-33.689736,910.000000,185, 84,AXIAL, 0
2,23.860528,-33.635172,1028.000000,190, 38,AXIAL, 0

E.2.3. Bedding Orientation for 3323DD.

0,23.810802,-33.912320,705.000000, 24, 70,BED_Ps, 0
1,23.805391,-33.915792,739.000000,206, 59,BED_Ps, 0
2,23.808166,-33.913362,798.000000, 23, 76,BED_Ps, 0
3,23.892391,-33.940390,994.000000, 8, 79,BED_Ps, 0
4,23.887447,-33.941672,914.000000,196, 60,BED_Ps, 0
5,23.842213,-33.931697,702.000000,215, 33,BED_Ps, 0
6,23.835866,-33.923895,623.000000, 28, 45,BED_Ps, 0
7,23.872480,-33.915424,1088.000000, 32, 61,BED_Ps, 0
8,23.887781,-33.923783,739.000000, 28, 61,BED_Ps, 0
9,23.880431,-33.918433,809.000000, 38, 68,BED_Ps, 0
10,23.885041,-33.939331,947.000000, 9, 83,BED_Ps, 0
11,23.899006,-33.891684,1267.000000,177, 82,BED_Ps, 0

12,23.888516,-33.896644,1224.000000,360, 52,BED_Ps, 0
13,23.854908,-33.913084,843.000000,207, 38,BED_Ps, 0
14,23.850031,-33.897313,1119.000000,195, 64,BED_Ps, 0
15,23.870075,-33.902551,1345.000000,185, 86,BED_Ps, 0
16,23.880230,-33.940446,835.000000,343, 23,BED_Ps, 0
17,23.888716,-33.942953,951.000000,306, 17,BED_Ps, 0
18,23.869674,-33.941895,655.000000,205, 38,BED_Ps, 0
19,23.848293,-33.937270,878.000000,339, 33,BED_Ps, 0
20,23.838872,-33.945462,713.000000,196, 52,BED_Ps, 0
21,23.839207,-33.899988,1020.000000, 36, 62,BED_Ps, 0
22,23.833394,-33.893022,1098.000000,189, 65,BED_Ps, 0
23,23.801723,-33.919270,739.000000,215, 25,BED_Ps, 0
24,23.875154,-33.889957,864.000000, 27, 65,BED_Ps, 0
25,23.868806,-33.888396,1045.000000,360, 30,BED_Ps, 0
26,23.863595,-33.889344,2282.000000, 11, 89,BED_Ps, 0
27,23.859052,-33.855127,961.000000,196, 84,BED_Ps, 0
28,23.877559,-33.858025,839.000000,189, 61,BED_Ps, 0
29,23.880032,-33.860421,661.000000,180, 49,BED_Ps, 0
30,23.862994,-33.858805,1288.000000,194, 87,BED_Ps, 0
31,23.873216,-33.863430,949.000000,186, 74,BED_Ps, 0
32,23.863729,-33.865994,1200.000000, 7, 80,BED_Ps, 0
33,23.864798,-33.870842,957.000000,175, 63,BED_Ps, 0
34,23.864597,-33.871957,1077.000000, 24, 52,BED_Ps, 0
35,23.856446,-33.866941,1100.000000,193, 79,BED_Ps, 0
36,23.850432,-33.865715,1045.000000,191, 81,BED_Ps, 0
37,23.863929,-33.872626,1215.000000,180, 13,BED_Ps, 0
38,23.864998,-33.874019,1115.000000, 15, 67,BED_Ps, 0
39,23.891991,-33.882099,1074.000000,196, 81,BED_Ps, 0
40,23.853171,-33.885276,1300.000000,212, 90,BED_Ps, 0
41,23.899942,-33.862204,1567.000000, 13, 88,BED_Ps, 0
42,23.897002,-33.890012,1248.000000,360, 72,BED_Ps, 0
43,23.845221,-33.870174,1109.000000, 6, 79,BED_Ps, 0
44,23.916713,-33.865213,1379.000000,193, 88,BED_Ps, 0
45,23.905422,-33.864990,1561.000000, 13, 88,BED_Ps, 0
46,23.917782,-33.874575,1357.000000, 8, 87,BED_Ps, 0
47,23.926802,-33.873461,2135.000000, 7, 88,BED_Ps, 0
48,23.910700,-33.880873,1162.000000, 17, 62,BED_Ps, 0
49,23.924129,-33.882377,1225.000000,188, 73,BED_Ps, 0
50,23.899141,-33.865603,662.000000,202, 56,BED_Ps, 0
51,23.928473,-33.849888,639.000000,194, 38,BED_KB, 0
52,23.937092,-33.851671,667.000000,194, 39,BED_KB, 0
53,23.947515,-33.853789,674.000000,195, 52,BED_KB, 0
54,23.952661,-33.931444,816.000000, 19, 47,BED_Ps, 0
55,23.950132,-33.946370,900.000000,207, 90,BED_Ps, 0
56,23.986410,-33.889425,1071.000000,199, 73,BED_Ps, 0
57,23.967056,-33.886505,843.000000, 20, 51,BED_Ps, 0

58,23.995844,-33.902891,942.000000, 18, 83,BED_Ps, 0
59,23.999539,-33.919276,1308.000000,359, 59,BED_Ps, 0
60,23.949160,-33.893076,1547.000000,188, 87,BED_Ps, 0
61,23.966277,-33.912625,694.000000,177, 76,BED_Ps, 0
62,23.913466,-33.931039,650.000000, 18, 48,BED_Ps, 0
63,23.965048,-33.960481,470.000000,178, 50,BED_Ps, 0
64,23.934198,-33.954764,506.000000,347, 27,BED_Ps, 0
65,23.987614,-33.961493,514.000000,193, 44,BED_Ps, 0
66,23.998113,-33.923016,1337.000000,173, 73,BED_Ps, 0
67,23.943054,-33.927008,595.000000,210, 21,BED_Ps, 0
68,23.924058,-33.933202,558.000000, 31, 72,BED_Ps, 0
69,23.911132,-33.932011,710.000000,185, 47,BED_Ps, 0
70,23.933841,-33.922779,515.000000,358, 38,BED_Ps, 0
71,23.992329,-33.911104,786.000000, 4, 45,BED_Ps, 0
72,23.985187,-33.938562,878.000000,360, 79,BED_Ps, 0
73,23.920487,-33.954525,308.000000,176, 40,BED_Ps, 0
74,23.916488,-33.903779,776.000000,196, 84,BED_Ps, 0
75,23.766678,-33.905216,498.000000, 32, 47,BED_Ps, 0
76,23.765919,-33.904302,501.000000, 28, 43,BED_Ps, 0
77,23.767353,-33.916189,755.000000,185, 85,BED_Ps, 0
78,23.751751,-33.900292,847.000000,206, 60,BED_Ps, 0
79,23.827147,-33.862027,870.000000, 12, 83,BED_Ps, 0
80,23.821159,-33.869061,838.000000,192, 38,BED_Ps, 0
81,23.798136,-33.863223,1133.000000, 8, 76,BED_Ps, 0
82,23.777390,-33.859565,1294.000000,186, 67,BED_Ps, 0
83,23.763727,-33.875181,1090.000000, 5, 70,BED_Ps, 0
84,23.756390,-33.871383,1000.000000,180, 90,BED_Ps, 0
85,23.751921,-33.852180,998.000000,204, 29,BED_Ps, 0
86,23.749728,-33.868499,847.000000, 86, 20,BED_Ps, 0
87,23.752425,-33.933774,520.000000,197, 8,BED_Ps, 0
88,23.772834,-33.935181,469.000000,187, 78,BED_Ps, 0
89,23.775449,-33.928498,328.000000, 31, 41,BED_Ps, 0
90,23.789534,-33.878065,508.000000, 31, 69,BED_Ps, 0
91,23.797461,-33.893188,663.000000,342, 32,BED_Ps, 0
92,23.759271,-33.843131,788.000000,353, 10,BED_Ps, 0
93,23.782290,-33.845820,1022.000000,192, 82,BED_Ps, 0
94,23.791508,-33.848603,880.000000, 17, 66,BED_Ps, 0
95,23.776238,-33.851245,985.000000,173, 20,BED_Ps, 0
96,23.772958,-33.844546,869.000000,186, 62,BED_Ps, 0
97,23.811077,-33.854594,964.000000, 14, 54,BED_Ps, 0
98,23.799144,-33.855301,1329.000000,192, 86,BED_Ps, 0
99,23.786135,-33.854877,1332.000000, 6, 83,BED_Ps, 0
100,23.770469,-33.850396,1261.000000,187, 82,BED_Ps, 0
101,23.752314,-33.844594,1077.000000, 17, 76,BED_Ps, 0
102,23.753276,-33.848273,1106.000000,198, 64,BED_Ps, 0
103,23.811813,-33.848461,736.000000,187, 19,BED_Ps, 0

104,23.878351,-33.838975,714.000000,198, 73,BED_Ps, 0
105,23.867940,-33.815572,698.000000, 13, 56,BED_Tch, 0
106,23.877716,-33.822137,842.000000, 15, 79,BED_Ko, 0
107,23.916695,-33.831244,909.000000,199, 84,BED_Ko, 0
108,23.805852,-33.807630,719.000000, 14, 35,BED_Ko, 0
109,23.881272,-33.795345,654.000000, 12, 4,BED_Ko, 0
110,23.872511,-33.773636,668.000000,192, 14,BED_Ko, 0
111,23.900825,-33.786026,431.000000,193, 33,BED_Ko, 0
112,23.901333,-33.784861,427.000000,190, 35,BED_Ko, 0
113,23.930790,-33.790791,470.000000,196, 62,BED_Ko, 0
114,23.860576,-33.776072,656.000000,181, 68,BED_Ko, 0
115,23.899936,-33.805299,511.000000,184, 51,BED_Ko, 0
116,23.867178,-33.806782,703.000000, 15, 73,BED_Ko, 0
117,23.820200,-33.797252,650.000000,186, 18,BED_Ko, 0
118,23.804456,-33.794816,789.000000, 9, 64,BED_Ko, 0
120,23.779315,-33.826904,687.000000,189, 60,BED_Ko, 0
121,23.805054,-33.770405,872.000000,352, 59,BED_Ko, 0
122,23.816626,-33.761360,644.000000,166, 10,BED_Ko, 0
123,23.793151,-33.748289,718.000000,188, 71,BED_Ko, 0
124,23.953045,-33.760145,637.000000,194, 36,BED_Ko, 0
125,23.937571,-33.748453,511.000000,212, 26,BED_Ko, 0
126,23.978039,-33.806473,449.000000,198, 60,BED_Ko, 0
127,23.992720,-33.764667,705.000000,186, 80,BED_Ko, 0
128,23.960449,-33.841330,733.000000, 20, 69,BED_Ko, 0
129,23.974468,-33.818938,672.000000,195, 76,BED_Ko, 0

E.2.4. Fracture Orientation for 3323DD.

0,23.759933,-33.858413,858.000000,110, 47,FRAC, 0
1,23.786585,-33.864161,2935.000000,108, 90,FRAC, 0
2,23.783038,-33.854018,964.000000,288, 53,FRAC, 0
3,23.793577,-33.885544,408.000000,189, 5,FRAC, 0
4,23.746759,-33.887827,742.000000,280, 66,FRAC, 0
5,23.809081,-33.888587,1203.000000, 79, 86,FRAC, 0
6,23.805737,-33.886728,823.000000,335, 21,FRAC, 0
7,23.884172,-33.920535,1101.000000,102, 83,FRAC, 0
8,23.881537,-33.919690,877.000000, 90, 59,FRAC, 0
9,23.875964,-33.915802,1066.000000, 90, 75,FRAC, 0
10,23.872518,-33.913943,1006.000000, 51, 45,FRAC, 0
11,23.887617,-33.941581,904.000000, 80, 59,FRAC, 0
12,23.884273,-33.939891,1000.000000, 90, 90,FRAC, 0
13,23.882753,-33.938707,880.000000, 80, 75,FRAC, 0
14,23.830059,-33.851482,741.000000, 95, 53,FRAC, 0
15,23.856913,-33.885459,1034.000000, 52, 54,FRAC, 0
16,23.851036,-33.867794,1042.000000,261, 50,FRAC, 0
17,23.783949,-33.930848,403.000000, 75, 82,FRAC, 0

18,23.805940,-33.899237,512.000000,119, 21,FRAC, 0
19,23.805433,-33.917916,882.000000,169, 83,FRAC_flt, 0
20,23.842015,-33.939215,818.000000,319, 12,FRAC, 0
21,23.914323,-33.881581,1144.000000,352, 19,FRAC, 0
22,23.904387,-33.879400,1194.000000,263, 71,FRAC, 0
23,23.985863,-33.889169,1138.000000,276, 73,FRAC, 0
24,23.983353,-33.889257,1180.000000,282, 67,FRAC, 0
25,23.977810,-33.890216,2480.000000, 90, 90,FRAC, 0
26,23.973208,-33.889955,1111.000000,298, 44,FRAC, 0
27,23.965782,-33.883674,949.000000,304, 41,FRAC, 0
28,23.988477,-33.943168,1063.000000, 26, 41,FRAC, 0
29,23.978541,-33.944389,990.000000, 30, 74,FRAC, 0
30,23.970174,-33.945785,876.000000,360, 26,FRAC, 0
31,23.950615,-33.946658,976.000000, 25, 79,FRAC, 0
32,23.935240,-33.936102,422.000000,111, 66,FRAC, 0
33,23.951138,-33.928251,735.000000, 79, 51,FRAC, 0
34,23.943817,-33.926070,3701.000000,130, 89,FRAC, 0
35,23.937437,-33.924064,2113.000000,325, 89,FRAC, 0
36,23.898843,-33.889781,1196.000000,246, 81,FRAC, 0
37,23.896752,-33.889607,1295.000000,253, 81,FRAC, 0
38,23.893928,-33.889694,1201.000000, 38, 56,FRAC, 0
39,23.945910,-33.869280,695.000000,104, 59,FRAC, 0
40,23.951229,-33.755542,618.000000,116, 45,FRAC, 0
41,23.965248,-33.771452,656.000000,131, 22,FRAC, 0
42,23.998114,-33.761005,468.000000, 83, 35,FRAC, 0
43,23.983634,-33.771835,782.000000, 84, 80,FRAC, 0
44,23.997194,-33.787170,385.000000,231, 37,FRAC, 0
45,23.985358,-33.781899,695.000000,257, 40,FRAC, 0
46,23.939278,-33.756022,719.000000,293, 76,FRAC, 0
47,23.932268,-33.767044,502.000000,113, 27,FRAC, 0
48,23.971913,-33.791004,498.000000,245, 42,FRAC, 0
49,23.966627,-33.791388,503.000000,260, 48,FRAC, 0
50,23.949849,-33.792059,483.000000,141, 49,FRAC, 0
51,23.945942,-33.788896,524.000000,301, 59,FRAC, 0
52,23.938703,-33.786979,428.000000,252, 15,FRAC, 0
53,23.928705,-33.792442,472.000000,110, 58,FRAC, 0
54,23.927441,-33.785446,458.000000,275, 59,FRAC, 0
55,23.909860,-33.779887,473.000000,129, 15,FRAC, 0
56,23.889864,-33.790718,990.000000,270, 90,FRAC, 0
57,23.876764,-33.788705,641.000000, 64, 46,FRAC, 0
58,23.887452,-33.760143,684.000000,296, 80,FRAC, 0
59,23.982140,-33.810461,659.000000,256, 77,FRAC, 0
60,23.980876,-33.813336,605.000000, 66, 41,FRAC, 0
61,23.912502,-33.799439,598.000000,135, 65,FRAC, 0
62,23.970993,-33.819374,481.000000,214, 10,FRAC, 0
63,23.935140,-33.809982,695.000000,290, 80,FRAC, 0

64,23.956514,-33.812569,106.000000, 90, 90,FRAC, 0
65,23.883429,-33.824455,661.000000,123, 69,FRAC, 0
66,23.873776,-33.821388,614.000000,131, 31,FRAC, 0
67,23.759087,-33.777722,976.000000, 95, 75,FRAC, 0
68,23.871753,-33.774907,514.000000,245, 37,FRAC, 0
69,23.853520,-33.775083,578.000000, 87, 44,FRAC, 0
70,23.828647,-33.766821,740.000000, 36, 26,FRAC, 0
71,23.814313,-33.770601,700.000000,249, 69,FRAC, 0
72,23.824958,-33.799697,629.000000, 92, 64,FRAC, 0
73,23.820215,-33.798466,592.000000, 96, 39,FRAC, 0
74,23.808622,-33.792137,781.000000,285, 80,FRAC, 0
75,23.786489,-33.804884,600.000000,242, 24,FRAC, 0
76,23.834865,-33.795302,674.000000,114, 11,FRAC, 0
77,23.834443,-33.792049,693.000000,332, 39,FRAC, 0
78,23.800506,-33.794423,680.000000,112, 52,FRAC, 0
79,23.775520,-33.800851,1954.000000,288, 89,FRAC, 0
80,23.932292,-33.829721,707.000000,360, 45,FRAC, 0
81,23.772523,-33.848600,1197.000000, 84, 43,FRAC, 0
82,23.780179,-33.854569,1402.000000, 70, 46,FRAC, 0
83,23.869882,-33.863591,813.000000, 79, 77,FRAC, 0
84,23.954925,-33.831664,105.000000, 81, 90,FRAC, 0

E.2.3. Axial Plane Orientation for 3323DD.

0,23.777847,-33.910774,710.000000,196, 82,AXIAL, 0
1,23.981712,-33.808581,2809.000000, 11, 89,AXIAL, 0
2,23.872023,-33.783975,1244.000000, 11, 86,AXIAL, 0

E.2.5. Bedding Orientation for 3324CA.

0,24.013360,-33.641157,1151.000000,202, 85,BED, 0
1,24.010841,-33.709925,992.000000,184, 73,BED, 0
2,24.013358,-33.705023,822.000000,170, 20,BED, 0
3,24.046256,-33.715387,1092.000000,195, 44,BED, 0
4,24.089727,-33.724910,976.000000,197, 69,BED, 0
5,24.032325,-33.681073,1400.000000,358, 28,BED, 0
6,24.042732,-33.658244,1090.000000,200, 79,BED, 0
7,24.061867,-33.623790,816.000000, 9, 78,BED, 0
8,24.026788,-33.624911,1277.000000, 12, 16,BED, 0
9,24.115240,-33.693957,1081.000000, 24, 72,BED, 0
10,24.067573,-33.652641,1199.000000,189, 83,BED, 0
11,24.142096,-33.672808,951.000000,358, 32,BED, 0
12,24.157705,-33.680371,1085.000000, 3, 54,BED, 0
13,24.195974,-33.665105,772.000000, 4, 18,BED, 0
14,24.109702,-33.657122,1122.000000,199, 84,BED, 0
15,24.136389,-33.688915,1076.000000,217, 28,BED, 0
16,24.187414,-33.681211,899.000000, 25, 64,BED, 0

17,24.159689,-33.723151,7779.000000, 20, 90,BED, 0
18,24.249137,-33.683655,975.000000,198, 73,BED, 0
19,24.224023,-33.674950,814.000000,202, 77,BED, 0
20,24.219965,-33.688330,952.000000,195, 56,BED, 0
21,24.199390,-33.708320,954.000000,200, 36,BED, 0
22,24.210788,-33.725005,1339.000000,191, 86,BED, 0
23,24.148967,-33.737660,916.000000, 2, 26,BED, 0
24,24.244693,-33.741689,1658.000000, 15, 88,BED, 0
25,24.157081,-33.744833,980.000000,201, 49,BED, 0
26,24.150126,-33.743544,877.000000,199, 27,BED, 0

E.2.6. Fracture Orientation for 3324CA.

0,24.011457,-33.671547,1273.000000, 81, 60,FRAC, 0
1,24.076228,-33.683979,1382.000000,251, 85,FRAC, 0
2,24.065883,-33.684624,1433.000000,252, 77,FRAC, 0
3,24.055384,-33.686041,1303.000000,259, 74,FRAC, 0
4,24.043418,-33.676443,1336.000000, 62, 46,FRAC, 0
5,24.095528,-33.673608,971.000000,260, 67,FRAC, 0
6,24.066038,-33.643460,972.000000,261, 61,FRAC, 0
7,24.048514,-33.649902,1079.000000, 69, 68,FRAC, 0
8,24.030757,-33.678698,1405.000000, 35, 23,FRAC, 0
9,24.061708,-33.706946,1075.000000,298, 69,FRAC, 0
10,24.055940,-33.704891,1063.000000,297, 77,FRAC, 0
11,24.048357,-33.704620,864.000000,279, 43,FRAC, 0
12,24.012454,-33.697645,1076.000000,292, 61,FRAC, 0
13,24.000011,-33.702512,1386.000000,297, 84,FRAC, 0
14,24.000918,-33.696455,1100.000000,153, 35,FRAC, 0
15,24.046349,-33.687315,1196.000000,111, 66,FRAC, 0
16,24.042979,-33.686342,1145.000000,127, 53,FRAC, 0
17,24.043367,-33.714355,1096.000000, 31, 35,FRAC, 0
18,24.035914,-33.709055,874.000000,285, 69,FRAC, 0
19,24.071947,-33.719383,1157.000000, 98, 39,FRAC, 0
20,24.027230,-33.680448,1411.000000,248, 52,FRAC, 0
21,24.044468,-33.723494,1203.000000, 72, 76,FRAC, 0
22,24.005454,-33.710137,1292.000000,119, 77,FRAC, 0
23,24.029433,-33.737933,700.000000,282, 49,FRAC, 0
24,24.046088,-33.736148,746.000000,280, 50,FRAC, 0
25,24.066632,-33.737392,744.000000,144, 15,FRAC, 0
26,24.075252,-33.720249,1172.000000,327, 61,FRAC, 0
27,24.130465,-33.730360,1014.000000,108, 84,FRAC, 0
28,24.155310,-33.677257,1129.000000,279, 63,FRAC, 0
29,24.192863,-33.699024,1069.000000,272, 51,FRAC, 0
30,24.198214,-33.709628,913.000000,259, 47,FRAC, 0
31,24.151353,-33.727359,919.000000,113, 47,FRAC, 0
32,24.229648,-33.707741,1198.000000,284, 83,FRAC, 0

33,24.229244,-33.730725,783.000000,275, 60,FRAC, 0
34,24.214313,-33.646451,771.000000,295, 12,FRAC, 0
35,24.126332,-33.668846,1285.000000,254, 87,FRAC, 0
36,24.109382,-33.668847,1169.000000,291, 6,FRAC, 0
37,24.130570,-33.635844,2708.000000, 90, 90,FRAC, 0
38,24.202205,-33.660090,957.000000,280, 49,FRAC, 0
39,24.168103,-33.662784,956.000000,117, 56,FRAC, 0
40,24.191914,-33.675918,889.000000,271, 64,FRAC, 0

E.2.3. Axial Plane Orientation for 3324CA.

0,24.108558,-33.695873,1053.000000, 26, 54,AXIAL, 0
1,24.044212,-33.676357,35435.000000,180, 90,AXIAL, 0
2,24.205257,-33.725799,875.000000,196, 68,AXIAL, 0

E.2.7. Bedding Orientation for 3324CC.

0,24.093788,-33.908646,600.000000, 25, 32,BED_Ps, 0
1,24.091759,-33.912271,683.000000,193, 57,BED_Ps, 0
2,24.100065,-33.910499,650.000000,196, 37,BED_Ps, 0
3,24.087317,-33.910177,745.000000,198, 69,BED_Ps, 0
4,24.091373,-33.915574,886.000000, 20, 80,BED_Ps, 0
5,24.076307,-33.912674,943.000000,360, 24,BED_Ps, 0
6,24.076983,-33.908244,888.000000,196, 78,BED_Ps, 0
7,24.076597,-33.905425,726.000000,201, 39,BED_Ps, 0
8,24.057764,-33.906391,757.000000,199, 72,BED_Ps, 0
9,24.046272,-33.898578,1419.000000, 18, 88,BED_Ps, 0
10,24.045403,-33.897289,1314.000000,196, 87,BED_Ps, 0
11,24.049555,-33.929590,701.000000,167, 23,BED_Ps, 0
12,24.056799,-33.903733,764.000000,190, 53,BED_Ps, 0
13,24.066649,-33.911224,1490.000000, 14, 87,BED_Ps, 0
14,24.102866,-33.919037,850.000000,212, 77,BED_Ps, 0
15,24.081811,-33.965918,831.000000,193, 64,BED_Ps, 0
16,24.133190,-33.937161,3851.000000, 17, 90,BED_Ps, 0
17,24.027922,-33.901478,926.000000,190, 60,BED_Ps, 0
18,24.017588,-33.898901,716.000000,213, 38,BED_Ps, 0
19,24.114938,-33.914768,721.000000,199, 70,BED_Ps, 0
20,24.108564,-33.922420,1001.000000,178, 46,BED_Ps, 0
21,24.095622,-33.949405,853.000000,198, 49,BED_Ps, 0
22,24.077562,-33.948841,639.000000, 18, 27,BED_Ps, 0
23,24.109722,-33.971234,749.000000, 11, 15,BED_Ps, 0
24,24.116579,-33.958426,681.000000, 17, 81,BED_Ps, 0
25,24.108756,-33.938692,875.000000,213, 10,BED_Ps, 0
26,24.013822,-33.919200,995.000000,116, 24,BED_Ps, 0
27,24.038545,-33.951742,944.000000,170, 70,BED_Ps, 0
28,24.012276,-33.945056,969.000000, 45, 59,BED_Ps, 0
29,24.059695,-33.933376,656.000000,192, 32,BED_Ps, 0

30,24.078818,-33.914930,938.000000, 10, 15,BED_Ps, 0
31,24.038931,-33.924918,958.000000,358, 52,BED_Ps, 0
32,24.018844,-33.886093,658.000000,204, 65,BED_Ps, 0
33,24.070512,-33.948197,441.000000,215, 74,BED_Ps, 0
34,24.177878,-33.930698,526.000000,198, 9,BED_Ps, 0
35,24.177100,-33.931428,4110.000000, 16, 90,BED_Ps, 0
36,24.231174,-33.959088,700.000000,188, 19,BED_Ps, 0
37,24.165137,-33.951464,910.000000,185, 30,BED_Ps, 0
38,24.194216,-33.962009,773.000000,186, 57,BED_Ps, 0
39,24.228257,-33.930859,379.000000, 33, 5,BED_Ps, 0
40,24.221157,-33.930535,450.000000, 17, 74,BED_Ps, 0
41,24.192952,-33.975475,900.000000,207, 90,BED_Ps, 0
42,24.222581,-33.966311,2335.000000,189, 89,BED_Ps, 0
43,24.209849,-33.946316,626.000000,211, 82,BED_Ps, 0
44,24.175364,-33.940645,779.000000,190, 53,BED_Ps, 0
45,24.180915,-33.951004,580.000000, 26, 55,BED_Ps, 0
46,24.170437,-33.954187,719.000000, 31, 28,BED_Ps, 0
47,24.192918,-33.963157,729.000000,171, 21,BED_Ps, 0
48,24.222478,-33.940992,452.000000,161, 18,BED_Ps, 0
49,24.197152,-33.929012,382.000000,192, 18,BED_Ps, 0
50,24.171755,-33.960553,839.000000, 12, 68,BED_Ps, 0
51,24.243316,-33.775230,946.000000,199, 85,Bed_Gd, 0
52,24.226310,-33.759259,811.000000,186, 29,Bed_Gd, 0
53,24.248971,-33.761260,975.000000, 13, 77,Bed_Gd, 0
54,24.240703,-33.767727,769.000000,325, 13,Bed_Gd, 0
55,24.222968,-33.755544,809.000000,213, 20,Bed_Gd, 0
56,24.208703,-33.754436,692.000000, 41, 57,Bed_Gd, 0
57,24.246529,-33.779018,811.000000,191, 81,Bed_Gd, 0
58,24.241089,-33.783377,1781.000000,208, 89,Bed_Gd, 0
59,24.229522,-33.769692,744.000000, 22, 69,Bed_Gd, 0
60,24.240360,-33.785807,501.000000,194, 25,Bed_Gd, 0
61,24.219104,-33.794370,552.000000, 21, 22,Bed_Gd, 0
62,24.069821,-33.815244,707.000000,199, 65,Bed, 0
63,24.062036,-33.830101,747.000000, 9, 46,Bed, 0
64,24.026543,-33.814915,704.000000, 15, 82,Bed, 0
65,24.056757,-33.865536,694.000000, 19, 76,Bed, 0
66,24.015063,-33.856403,806.000000,199, 80,Bed, 0
67,24.095813,-33.838244,580.000000,185, 17,Bed, 0
68,24.099375,-33.842756,549.000000, 12, 57,Bed, 0
69,24.081431,-33.866967,639.000000,198, 60,Bed, 0
70,24.163499,-33.888975,582.000000,198, 76,Bed, 0
71,24.149909,-33.867956,1181.000000, 16, 86,Bed, 0
72,24.158222,-33.853760,521.000000, 10, 16,Bed, 0
73,24.170361,-33.843966,427.000000, 21, 22,Bed, 0
74,24.133945,-33.825698,369.000000,177, 4,Bed, 0
75,24.118111,-33.860364,702.000000, 15, 48,Bed, 0

76,24.125631,-33.878301,519.000000,197, 41,Bed, 0
77,24.163631,-33.894698,437.000000,201, 52,Bed, 0
78,24.160993,-33.857282,642.000000, 17, 62,Bed, 0
79,24.036703,-33.825699,769.000000,191, 77,Bed, 0
80,24.044751,-33.813814,746.000000,199, 78,Bed, 0
81,24.004885,-33.777855,624.000000,183, 37,Bed, 0
82,24.015675,-33.779276,585.000000, 6, 25,Bed, 0
83,24.049522,-33.747066,729.000000, 1, 73,Bed, 0
84,24.029532,-33.747446,564.000000, 8, 40,Bed, 0
85,24.045319,-33.802580,658.000000, 9, 46,Bed, 0
86,24.017720,-33.763929,672.000000,348, 73,Bed, 0
87,24.021127,-33.753793,538.000000,237, 45,Bed, 0
88,24.034643,-33.754929,667.000000,214, 83,Bed, 0
89,24.039413,-33.762603,1182.000000,176, 87,Bed, 0
90,24.081438,-33.757866,598.000000,200, 65,Bed, 0
91,24.066218,-33.781549,554.000000,179, 74,Bed, 0
92,24.102109,-33.779748,454.000000,209, 31,Bed, 0
93,24.099724,-33.784580,492.000000,181, 32,Bed, 0
94,24.128460,-33.769801,621.000000,189, 62,Bed, 0
95,24.146973,-33.777095,647.000000,189, 58,Bed, 0
96,24.142089,-33.789126,601.000000,238, 5,Bed, 0
97,24.139136,-33.801726,627.000000, 23, 48,Bed, 0
98,24.077917,-33.769139,660.000000,356, 71,Bed, 0
99,24.130504,-33.813567,547.000000,199, 33,Bed, 0
100,24.213420,-33.880791,611.000000, 18, 66,BED, 0
101,24.212403,-33.865622,438.000000,201, 14,BED, 0
102,24.193402,-33.812355,608.000000,207, 54,BED, 0
103,24.234631,-33.827487,527.000000, 4, 6,BED, 0
104,24.188764,-33.815840,605.000000,202, 19,BED, 0

E.2.8. Fracture Orientation for 3324CC.

0,24.104277,-33.915032,1336.000000,289, 85,, 0
1,24.097997,-33.912122,1329.000000,102, 86,, 0
2,24.116954,-33.912316,897.000000,289, 73,FRAC, 0
3,24.094275,-33.906399,771.000000,296, 49,FRAC, 0
4,24.087995,-33.910958,824.000000, 84, 34,FRAC, 0
5,24.083576,-33.908242,824.000000, 91, 49,FRAC, 0
6,24.124978,-33.919009,3331.000000,112, 89,FRAC, 0
7,24.039382,-33.899028,866.000000,308, 57,FRAC, 0
8,24.053803,-33.903586,1271.000000,288, 73,FRAC, 0
9,24.090553,-33.941416,534.000000,170, 36,FRAC, 0
10,24.082993,-33.962466,466.000000, 61, 75,FRAC, 0
11,24.077411,-33.961205,433.000000, 59, 71,FRAC, 0
12,24.092530,-33.963339,1096.000000,255, 88,FRAC, 0
13,24.000071,-33.948693,1137.000000, 90, 85,FRAC, 0

14,24.169521,-33.925701,396.000000, 98, 43,FRAC, 0
15,24.030310,-33.906594,1066.000000,117, 84,FRAC, 0
16,24.002282,-33.896118,611.000000,329, 6,FRAC, 0
17,24.173327,-33.927334,792.000000,288, 82,FRAC, 0
18,24.129678,-33.948770,530.000000,277, 65,FRAC, 0
19,24.132118,-33.950805,560.000000,116, 70,FRAC, 0
20,24.137171,-33.930822,464.000000,291, 46,FRAC, 0
21,24.144750,-33.949569,752.000000,288, 80,FRAC, 0
22,24.152765,-33.953420,670.000000,126, 70,FRAC, 0
23,24.180296,-33.961558,858.000000, 96, 78,FRAC, 0
24,24.163481,-33.974711,597.000000,308, 19,FRAC, 0
25,24.205299,-33.980669,544.000000,124, 55,FRAC, 0
26,24.176376,-33.928569,556.000000, 85, 24,FRAC, 0
27,24.245114,-33.952475,539.000000,108, 31,FRAC, 0
28,24.243024,-33.938305,524.000000,126, 73,FRAC, 0
29,24.228213,-33.949859,641.000000,310, 66,FRAC, 0
30,24.203732,-33.927842,395.000000,111, 43,FRAC, 0
31,24.244678,-33.970931,424.000000,270, 40,FRAC, 0
32,24.232656,-33.974710,575.000000,107, 79,FRAC, 0
33,24.163569,-33.924064,585.000000, 80, 27,FRAC, 0
34,24.124394,-33.854593,624.000000, 95, 23,FRAC, 0
35,24.156302,-33.861079,756.000000, 72, 18,FRAC, 0
36,24.111013,-33.867852,662.000000,147, 71,FRAC, 0
37,24.125080,-33.873575,735.000000,148, 79,FRAC, 0
38,24.155044,-33.878726,572.000000,314, 44,FRAC, 0
39,24.113072,-33.838091,503.000000,111, 57,FRAC, 0
40,24.166710,-33.846771,496.000000,292, 22,FRAC, 0
41,24.185237,-33.870236,616.000000,323, 13,FRAC, 0
42,24.208910,-33.908009,399.000000,107, 64,FRAC, 0
43,24.244935,-33.906960,465.000000, 97, 69,FRAC, 0
44,24.201706,-33.840570,393.000000,287, 54,FRAC, 0
45,24.114443,-33.874529,523.000000,293, 69,FRAC, 0
46,24.179632,-33.894178,463.000000,288, 65,FRAC, 0
0,24.194843,-33.765331,604.000000,116, 16,FRAC, 0
1,24.139352,-33.754633,681.000000,143, 18,FRAC, 0
2,24.145208,-33.753470,941.000000,115, 82,FRAC, 0
3,24.069082,-33.752773,589.000000,285, 53,FRAC, 0
4,24.088183,-33.768355,573.000000,286, 55,FRAC, 0
5,24.230118,-33.768819,588.000000,107, 23,FRAC, 0
6,24.164587,-33.813009,657.000000,150, 75,FRAC, 0
7,24.091250,-33.801614,399.000000,275, 31,FRAC, 0
8,24.150505,-33.829406,409.000000,315, 42,FRAC, 0
9,24.210458,-33.805333,518.000000,334, 23,FRAC, 0
10,24.137954,-33.795101,555.000000,269, 37,FRAC, 0

E.2.3. Axial Plane Orientation for 3324CC.

0,24.128758,-33.954482,972.000000, 17, 76,AXIAL, 0
1,24.134774,-33.871510,986.000000, 15, 52,AXIAL, 0
2,24.042608,-33.918149,767.000000, 20, 75,AXIAL, 0
3,24.160293,-33.938213,659.000000,200, 52,AXIAL, 0
4,24.167119,-33.944856,493.000000,187, 48,AXIAL, 0
5,24.083407,-33.911912,919.000000, 18, 82,AXIAL, 0

E.2.3. Thickness Measurements on 3323DD.

0,23.886580,-33.824560,2009.000000,196, 89,, 450
1,23.870879,-33.839760,798.000000,194, 74,, 199
2,23.930365,-33.852262,959.000000, 14, 85,, 144

APPENDIX F

THE FORMAT FILES FOR TNTMIPS

The IMPGEO.fmt File

This file is used to import ASCII files generated by GEOSTRUC© into TNTmps database facilities.

```
Header Lines:  13
Desc:  Import geostruc.geo ASCII files

Fields:  8
Name:  RECORDNO          Type:  Int   Start:  4   Length:  3
Name:  LONGITUDE         Type:  Float Start:  7   Length:  10 Dec:  6
Name:  LATITUDE          Type:  Float Start:  17  Length:  10 Dec:  6
Name:  ELEVATION         Type:  Float Start:  27  Length:  11 Dec:  6
Name:  AZIMUTH           Type:  Int   Start:  38  Length:  3
Name:  DIP_PLUNGE        Type:  Int   Start:  41  Length:  3
Name:  STRUCTURE         Type:  Char  Start:  44  Length:  20
Name:  THICKNESS         Type:  Int   Start:  64  Length:  4
```

The FBSSTRUCSYM.qry File

This file is used to display the structure tick marks.

```
if (IMPGEO.DIP_PLUNGE < 5) then
{Style$ = "HORIZONTAL";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE >= 5 and IMPGEO.DIP_PLUNGE <= 15) then
{Style$ = "DIP5_15";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE > 15 and IMPGEO.DIP_PLUNGE <= 30) then
{Style$ = "DIP15_30";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE > 30 and IMPGEO.DIP_PLUNGE <= 45) then
{Style$ = "DIP30_45";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}
```

```
else if (IMPGEO.DIP_PLUNGE > 45 and IMPGEO.DIP_PLUNGE <= 60) then
{Style$ = "DIP45_60";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE > 60 and IMPGEO.DIP_PLUNGE <= 75) then
{Style$ = "DIP60_75";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE > 75 and IMPGEO.DIP_PLUNGE <= 85) then
{Style$ = "DIP75_85";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

else if (IMPGEO.DIP_PLUNGE > 85 and IMPGEO.DIP_PLUNGE <= 90) then
{Style$ = "VERTICAL";
DrawColor$ = "WHITE";
Angle = IMPGEO.AZIMUTH * pi/180 * -1}

UseStyle = 1
XScale = 2.5
YScale = 2.5
MapScale = 160000
```

APPENDIX G

AXIAL PLANE ORIENTATION MEASUREMENT

This appendix describes the measurement of the orientation of axial planes on folds with a visible surface trace.

Surface traces of folded beds must be clearly distinguishable on digital images for the GEOSTRUC© routine to be employed. The user selects three points with a pointing device on the surface trace of a fold, estimating where the axial plane lies (**Fig. G1.**).

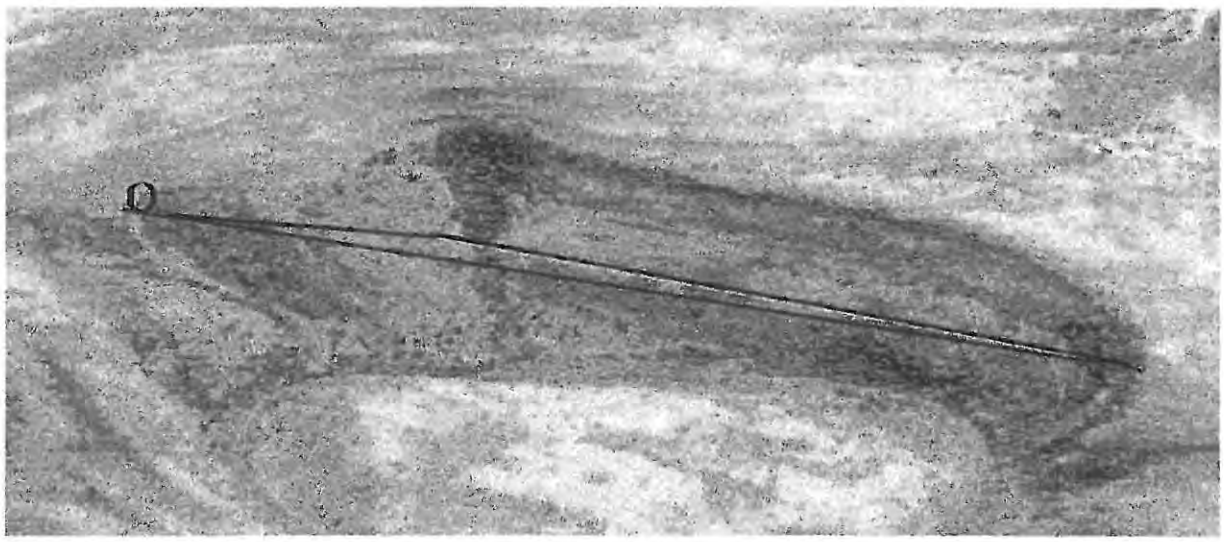


Fig. G1. Axial plane orientation measurement on bedding surface trace.

The user has to ensure that the three selected points differ in elevation by more than 50m.

The resultant measurement, after the GEOSTRUC© calculation is completed, gives the orientation of the axial plane of the folded structures.

APPENDIX H

GEOSTRUC MEASUREMENTS IN STUDY AREAS

Fig. H1. GEOSTRUC© measurements for the Kareedouw Case Study, 3323DD.

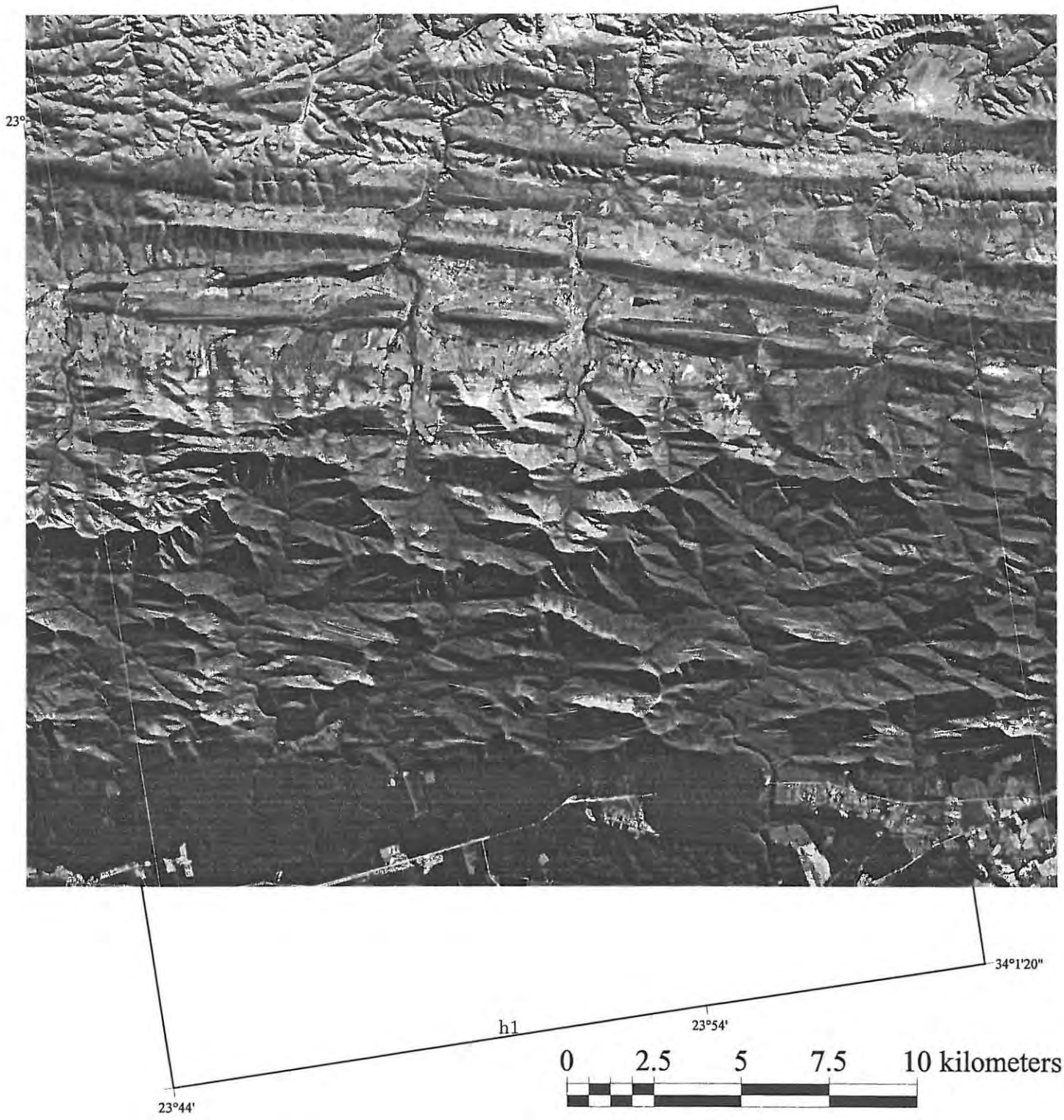


Fig. H2. GEOSTRUC© measurements for the Alicedale Case Study.