JJM
SYSTEMS INC

C931033-U-2R00

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - FINAL REPORT 1993

November 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
1225 Jefferson Davis Hwy, Suite 412
Arlington, VA 22202

TABLE OF CONTENTS

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

INTRODUCTION

The NASA VEGetation Workbench (VEG) is a knowledge based system that infers vegetation characteristics from reflectance data. VEG is described in detail in several references (e.g., 1, 2). The first generation version of VEG has been extended.

In the first year of this contract, an interface to a file of unknown cover type data was constructed. An interface that allowed the results of VEG to be written to a file was also implemented. A learning system that learned class descriptions from a data base of historical cover type data and then used the learned class descriptions to classify an unknown sample was built. This system had an interface that integrated it into the rest of VEG. The VEG subgoal PROPORTION.GROUND.COVER was completed and a number of additional techniques that inferred the proportion ground cover of a sample were implemented. This work was described in reference (3).

This report describes the work carried out in the second year of the contract. The historical cover type database has been removed from VEG and stored as a series of flat files that are external to VEG. An interface to the files has been provided. The framework and interface for two new VEG subgoals that estimate the atmospheric effect on reflectance data have been built. A new interface that allows the scientist to add techniques to VEG without assistance from the developer has been designed and implemented. A prototype Help System that allows the user to get more information about each screen in the VEG interface has also been added to VEG.

VEG is written using the Knowledge Engineering Environment (KEE) by Intellicorp. Data and methods are contained in KEE units in the file veg4.u. VEG also uses Lisp methods contained in Lisp files. The file veg-methods.lisp contained the Lisp methods for the first generation version of VEG. Additional Lisp files were created to hold the Lisp code for each of the extensions to VEG. However, each extension of VEG also required some minor changes to the file veg-methods.lisp. The current versions of all the VEG files have been delivered to the NASA GSFC technical representative on a Sun cartridge tape.

The changes to the historical cover type database were described in detail in the JJM Systems report C931020-U-2R05. This report is included as Appendix A. All the subgoals in VEG make use of a database of historical cover types. This database contains results from experiments by scientists on a wide variety of different cover types. The learning system uses the database to provide positive and negative training examples of classes that enable it to learn distinguishing features between classes of vegetation. Other VEG subgoals use the database to estimate the error bounds involved in the results obtained when various analysis techniques are applied to the sample of cover type data that is being studied.

In the previous version of VEG, the historical cover type database was stored as part of the VEG knowledge base. This database has been removed from the knowledge base. It is now stored as a series of flat files that are external to VEG. An interface between VEG and these files has been provided. The interface allows the user to select which files of historical data to use. The files are then read, and the data are stored in KEE units using the same organization of units as in the previous version of VEG. The interface also allows the user to delete some or all of the historical database units from VEG and load new historical data from a file. The database of historical cover types in the previous version of VEG occupied 123 units. Removing this database to external files and only loading a subset of the data, has reduced the memory requirements for VEG.

Allowing the user to select the historical data to load has made VEG more flexible. VEG is now useful to a wider group of scientists. Scientists with different areas of interest can use different sets of historical data with VEG and restrict the use of VEG to the data that are of interest to them.

The structure of the subgoals in VEG has been modified. Subgoals are now divided into categories. Two new subgoals in the category ATMOSPHERIC.TECHNIQUES, have been added to VEG. The basic framework and interfaces for these subgoals have been implemented. The subgoal Atmospheric Passes allows the scientist to take reflectance data measured at ground level and predict what the reflectance values would be if the data were measured at a different atmospheric height. The subgoal Atmospheric Corrections allows atmospheric corrections to be made to data collected from an aircraft or by a satellite to determine what the equivalent reflectance values would be if the data were measured at ground level. The new subgoals were described in JJM Systems report C931031-U-2R06 which is included as Appendix B.

VEG provides the scientist with several different analysis techniques which are stored in the knowledge base. When VEG is run, rules assist the scientist in selecting the best of the available techniques to apply to the sample of cover type data being studied.

In the previous version of VEG, the addition of a new technique was a complex process. For each new technique, extra units were added manually to the VEG knowledge base and additional Common Lisp code was added to the methods file. Changes were also made manually to the interface that allow the scientist to select which techniques to use.

A new interface that enables the scientist to add techniques to VEG without assistance from the developer has been designed and implemented. This interface does not require the scientist to have a thorough knowledge of KEE or a detailed knowledge of the structure of VEG. The interface prompts the scientist to enter the required information about the new technique. It prompts the scientist to enter the required Common Lisp functions for executing the technique and the left hand side of the rule that causes the technique to be selected. A template for each function and rule and detailed instructions about the arguments of the functions, the values they should return, and the format of the rule are displayed. Checks are made to ensure that the required data have been entered, the functions compiled correctly and the rule parsed correctly before the new technique is stored. The additional techniques are stored separately from the VEG knowledge base.

When the VEG knowledge base is loaded, the additional techniques are not normally loaded. The interface allows the scientist the option of adding all the previously defined new techniques before running VEG. When the techniques are added, the required units to store the additional techniques are created automatically in the correct places in the VEG knowledge base. The methods file containing the functions required by the additional techniques is loaded. New rule units are created to store the new rules. The interface that allows the scientist to select which techniques to use is updated automatically to include the new techniques. The interface that allows the scientist to add new techniques to VEG was described in JJM Systems report C931021-U-2R07 which is included in Appendix C.

A prototype Help System has been designed and implemented. The Help System allows the scientist to get more information about each screen in the VEG interface. It was designed to help the new user of VEG to learn how to operate the system. An interface that allows the scientist to add and modify help messages has also been integrated into the "Administration" part of the VEG system. This enables the scientist to evolve the Help System over time.

Since the Help System may not be needed by an experienced user, it had been configured so that it is not loaded until the first time the user clicks on the Help System option in the Tool Box Menu. This minimizes the overhead for the VEG environment. JJM Systems report C931032-U-2R08, which described the prototype Help System, forms Appendix D of this report.

REFERENCES

1. Kimes, D. S., Harrison, P. R. and Ratcliffe, P. A. A Knowledge-Based Expert System for Inferring Vegetation Characteristics, International Journal of Remote Sensing, Vol 12, 10, pp. 1987-2020, 1991.

2. Kimes, D. S., Harrison, P. A. and Harrison, P. R. New Developments of a Knowledge Based System (VEG) for Inferring Vegetation Characteristics, Proceedings of International Geoscience and Remote Sensing Symposium, Houston, Texas, May 1992.

3. JJM Systems Inc., October 1992. An Expert System for Inferring Vegetation Characteristics - Final Report. Ivyland, PA B921020-U-2R00.

JJM SYSTEMS INC

APPENDIX A

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - CHANGES TO THE HISTORICAL COVER TYPE DATABASE (TASK F)

JJM SYSTEMS INC

N93-27072

C931020-U-2R05

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - CHANGES TO THE HISTORICAL COVER TYPE DATABASE (TASK F)

26 May 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
1225 Jefferson Davis Hwy., Suite 412
Arlington, VA 22202

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

SECTION 1.0

INTRODUCTION

All the options in the NASA VEGetation Workbench (VEG) make use of a database of historical cover types. This database contains results from experiments by scientists on a wide variety of different cover types. The learning system uses the database to provide positive and negative training examples of classes that enable it to learn distinguishing features between classes of vegetation. All the other VEG options use the database to estimate the error bounds involved in the results obtained when various analysis techniques are applied to the sample of cover type data that is being studied.

In the previous version of VEG, the historical cover type database was stored as part of the VEG knowledge base. This database has been removed from the knowledge base. It is now stored as a series of flat files that are external to VEG. An interface between VEG and these files has been provided. The interface allows the user to select which files of historical data to use. The files are then read, and the data are stored in Knowledge Engineering Environment (KEE) units using the same organization of units as in the previous version of VEG. The interface also allows the user to delete some or all of the historical database units from VEG and load new historical data from a file.

This report summarizes the use of the historical cover type database in VEG. It then describes the new interface to the files containing the historical data. It describes minor changes that were made to VEG to enable the externally stored database to be used. Test runs to test the operation of the new interface and also to test the operation of VEG using historical data loaded from external files are also described.

Task F has been completed. A Sun cartridge tape containing the KEE and Common Lisp code for the new interface and the modified version of the VEG knowledge base has been delivered to the NASA GSFC technical representative.

SECTION 2.0

BACKGROUND

The need for the historical cover type database in VEG is summarized in this section. Then, the organization of units in the historical cover type database in the VEG knowledge base is described.

2.1 THE USE OF THE HISTORICAL COVER TYPE DATABASE

All the options in VEG make use of the historical cover type database. This database contains results from experiments by scientist on a wide variety of different cover types.

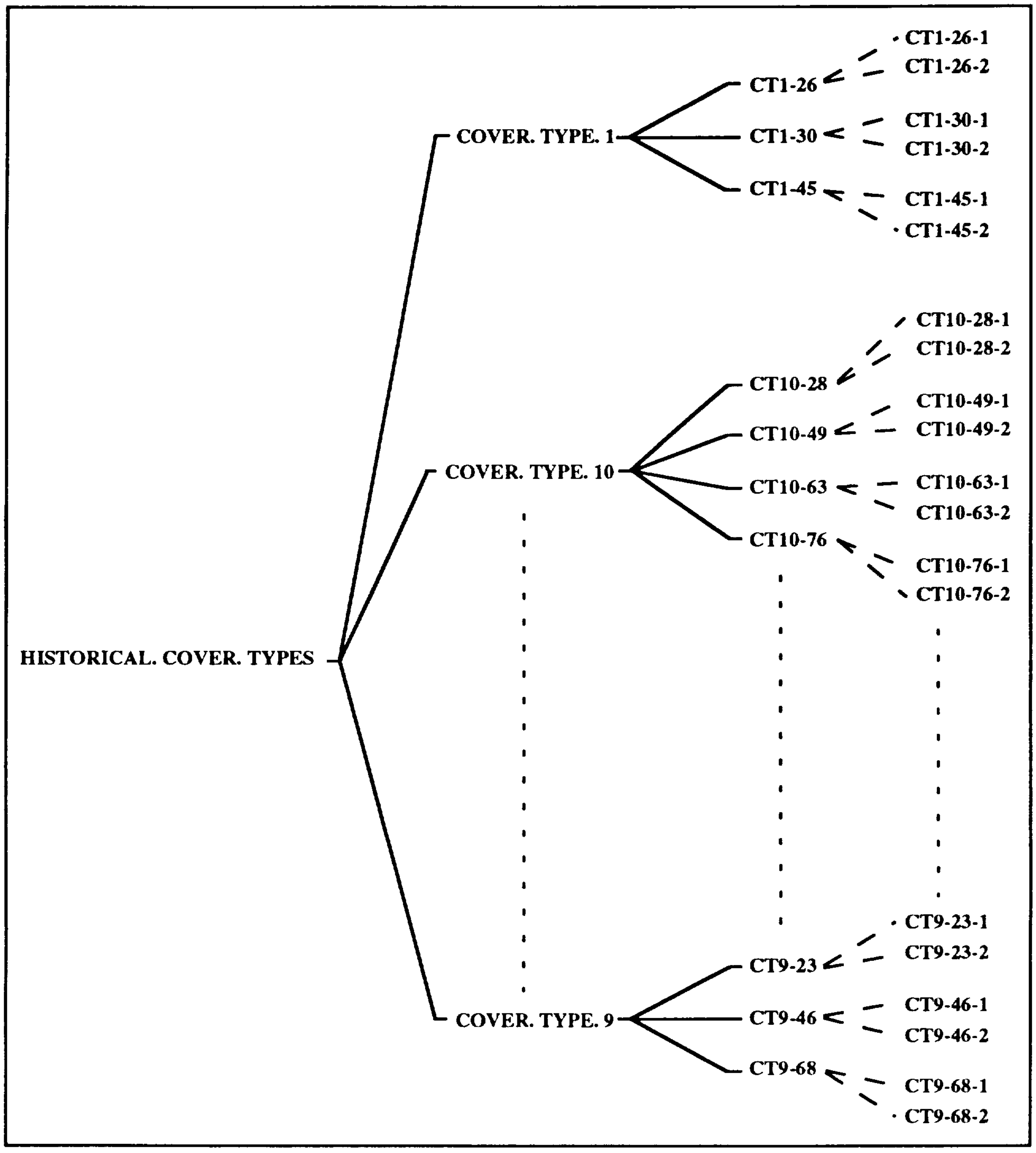
When the learning system is in use, VEG selects positive and negative training examples from the historical cover type database. From these training examples, VEG determines relationships that discriminate between classes of vegetation. These classes can then used to classify an unknown sample. The learning system was described in detail in Kimes, Harrison and Harrison (1992) and JJM Systems report B921014-U-2R03. The historical cover type database is a fundamental part of the learning system.

All the other options in VEG use the historical cover type database in order to estimate the error terms when various analysis techniques are applied to the sample of cover type data being studied. A subset of historical data, referred to as the "restricted data set," is selected for each run. VEG can automatically select the restricted data set that best matches the sample. Alternatively, the user can indicate the bounds on each parameter of interest in and instruct VEG to select the subset of the historical cover type data that falls within the set bounds. Once the restricted data set has been selected, the reflectance data in each cover type are interpolated and extrapolated so that they match the exact view angles of the input spectral data. The restricted data set contains the true results, for example the spectral hemispherical reflectance, for each cover type. Each technique that is applied to the sample that is being studied is also applied to each cover type in the restricted data set. A difference score based on errors (true results minus calculated result) for the restricted data set is calculated for each technique. This score provides an estimate of the error involved in applying the technique to the sample being studied. The use of the historical cover type database was described in detail in Kimes, Harrison and Ratcliffe (1991) and JJM Systems report B921019-U-2R04.

2.2 ORGANIZATION OF HISTORICAL COVER TYPE DATABASE UNITS

The historical cover type database in the previous version of VEG was stored permanently in units in the VEG knowledge base. These units were subclasses and instances of the unit HISTORICAL.COVER.TYPES. Figure 2-1 shows part of this hierarchy of units. All the units that were children of the unit HISTORICAL.COVER.TYPES inherited slots from the parent unit. Figure 2-2 shows the slots in the unit HISTORICAL.COVER.TYPES.

Units such as COVER.TYPE.1 held data that were common for all the data sets for the particular cover type. These data were inherited by all the units such as CT1-26 and CT1-26-1 that were descendants of the unit COVER.TYPE.1. Data shared by all data sets for a particular cover type included the description, cover type, solar azimuth, solar zenith angles and the zenith interval. The DESCRIPTION slot of COVER.TYPE.1 contained the description "PLOWED FIELD - TUNISIA AFRICA - KIMES DATA SET." This description referred to all data sets collected at



C931020G1

Figure 2-1
Some of the Units in the Historical Cover Type Database in the Previous Version of VEG

```
AZIMUTH.INTERVAL
COVER.TYPE
DATE
DESCRIPTION
DRY.BIOMASS.KG.HC
GROUND.COVER
HEIGHT.CM
LEAF.AREA.INDEX
LEAF.ORIENTATION.DISTRIBUTION
LOCAL.STANDARD.TIME
MAX.ZENITH.DATA
PROPORTION.GREEN
RAW.DATA
SOLAR.AZIMUTH
SOLAR.ZENITH.ANGLE
SOLAR.ZENITH.ANGLES
SPECTRAL.HEM.REFLECTANCE
STRUCTURE
TOTAL.HEM.REFLECTANCE
WAVELENGTH.MAX
WAVELENGTH.MIN
WAVELENGTHS
WET.BIOMASS.KG.HC
ZENITH.INTERVAL
```

Figure 2-2
Slots in the Unit HISTORICAL.COVER.TYPES

the particular location. The SOLAR.AZIMUTH slot contained the value 180. The slot SOLAR.ZENITH.ANGLES held the list of solar zenith angles for which data were available for the cover type. The ZENITH.INTERVAL slot contained the value 15 indicating that measurements of reflectance for COVER.TYPE.1 were collected at 15 degree zenith intervals in each set of data.

The units such as CT1-26 and CT1-39 held data for COVER.TYPE.1 that were collected at solar zenith angles 27° and 39°, respectively. The data stored in these units were collected at the same location but at different times of the day when the sun was at different positions in the sky. The slot SOLAR.ZENITH.ANGLE in the units, such as CT1-26 and CT1-39, held the value of the solar zenith angle at the time that the data were collected. At each solar zenith angle, data were collected in one or more different wavebands.

The units CT1-26-1 and CT1-26-2 were instances of the unit CT1-26. These units held data collected at solar zenith angle 26° and in the wavebands 0.58 - 0.68 μm and 0.73 - 1.1 μm, respectively. The maximum and minimum wavelength in the wavebands were stored in the slots WAVELENGTH.MAX and WAVELENGTH.MIN. The spectral hemispherical reflectance measured in a particular waveband was stored in the slot SPECTRAL.HEM.REFLECTANCE. Reflectance data were stored in the slot RAW.DATA. The reflectance data consisted of reflectance measurements taken at different zenith and azimuth combinations. Each data point was specified by a zenith, azimuth and reflectance value. The data were stored in the slot RAW.DATA as a list of lists; e.g. ((0 0 0.231) (15 0 0.1968) (15 45 0.2094)). In this example, data at the nadir (zenith and azimuth both zero) and at two other points were recorded. The reflectance values at the three points were 0.231, 0.1968 and 0.2094, respectively.

SECTION 3.0

CHANGES TO VEG

This section describes the interface that allows the user to load historical cover type data from external files and subsequently delete the loaded data from VEG. The format of the files that hold the historical data, and the structure of the units that hold the database when it is loaded into VEG are also described. The removal of the historical database from within the VEG hierarchy of units to a series of external files necessitated various minor changes to the operation of VEG and to the VEG user interface. These changes are described in this section.

3.1 CHANGING THE HISTORICAL COVER TYPE DATABASE

In the previous version of VEG when the user left-clicked on RUN.VEG, VEG was run and the Processing Mode Screen was displayed. This screen enabled the user to specify whether VEG should be run in the Automatic or Research mode. An additional option, "ADMINISTRATION," has been included in the menu as shown in Figure 3-1. This option allows the user to make various changes to VEG before processing the data.

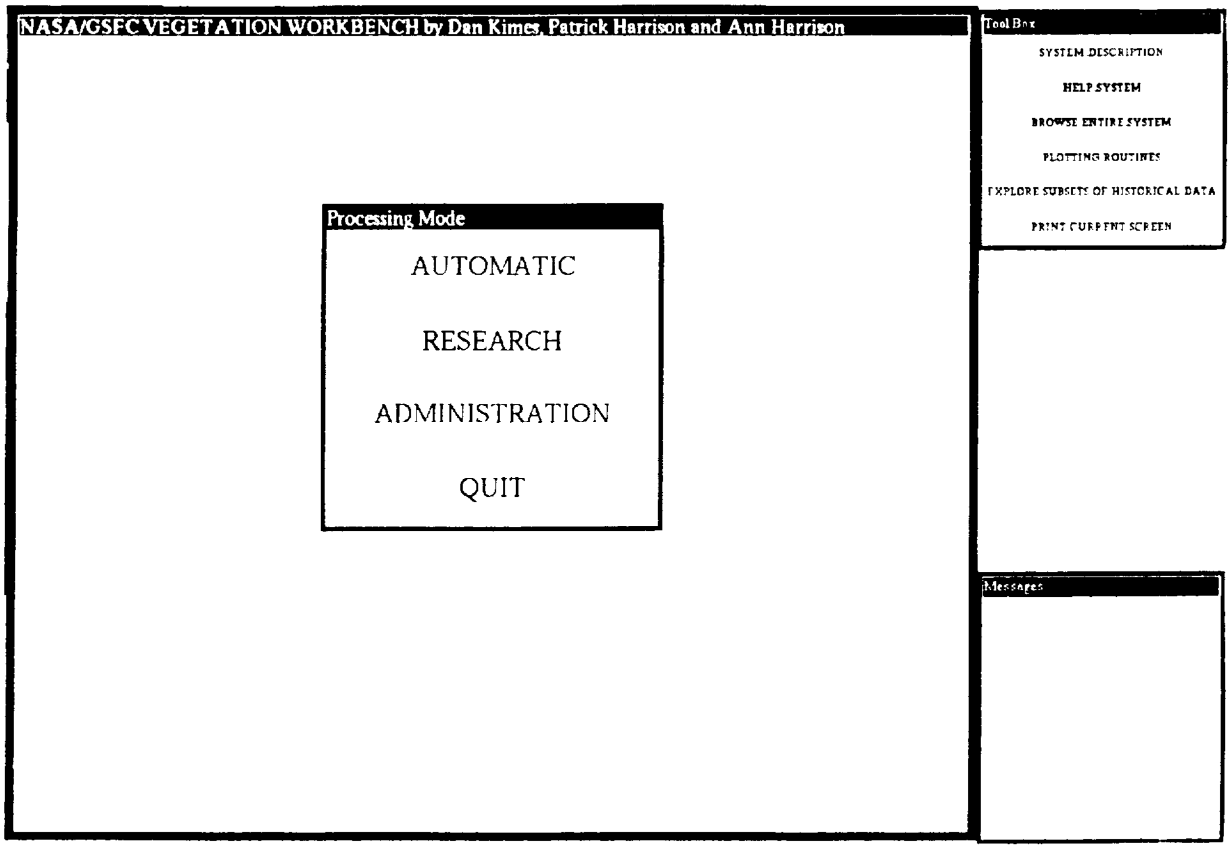


Figure 3-1
The Processing Mode Screen in the Current Version of VEG

If the user selects the ADMINISTRATION option from the Processing Mode menu, the screen shown in Figure 3-2 is opened. This screen enables the user to either change the historical database or add techniques. The CHANGE.HISTORICAL.DATABASE option is described in this section of this report. The ADD.TECHNIQUES option was implemented as Task H of this contract, and it was described in JJM Systems Report C931021-U-2R07. More options may be added to this menu at a later date.

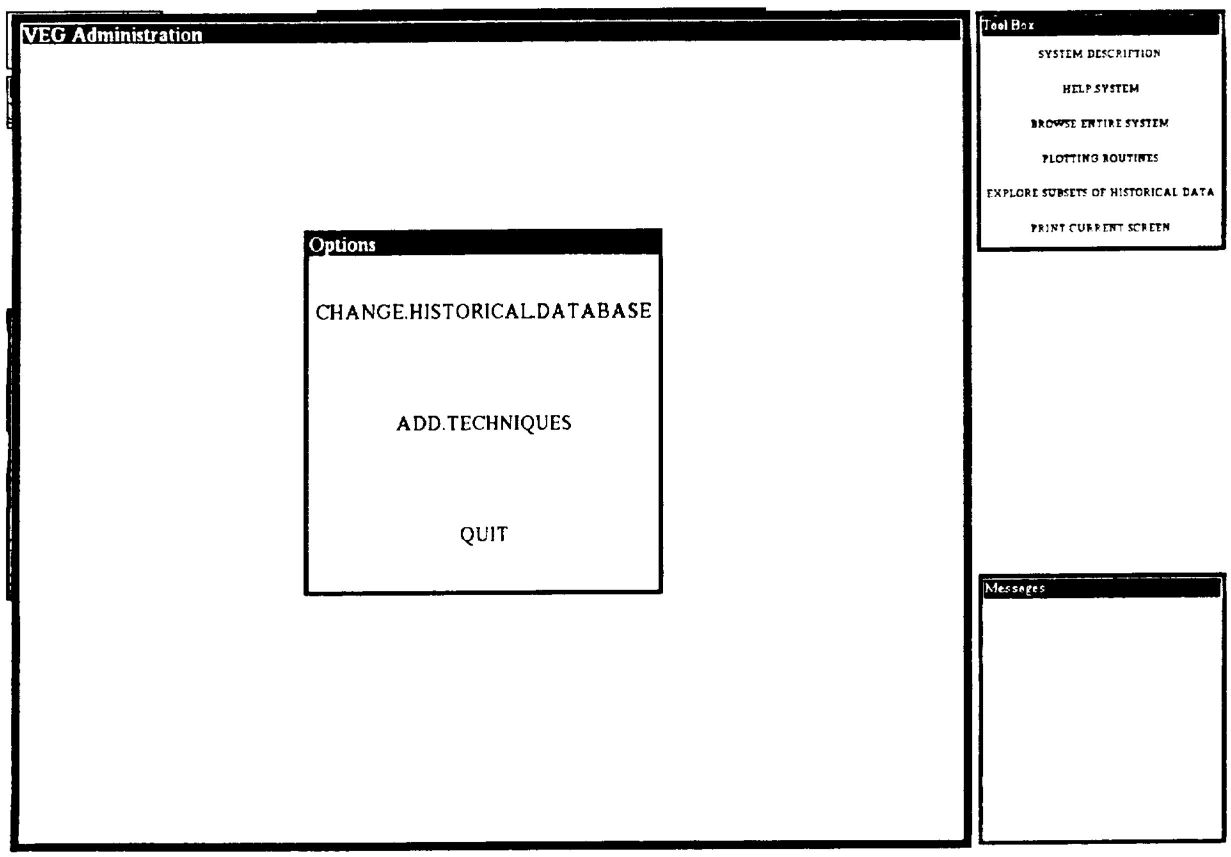


Figure 3-2
The Administration Screen

Left-clicking on the CHANGE.HISTORICAL.DATABASE option of the ADMINISTRATION menu, reveals the Change Historical Database Screen as shown in Figure 3-3. This screen allows the user to load historical data from an external file or delete previously loaded historical database units.

The unit CHANGE.HISTORICAL.DATABASE has been created in VEG. The subwindows (KEE ActiveImages) in the Change Historical Database Screen are attached to slots in this unit. The unit also holds the slots required by the methods that are used to add and remove cover types. For example, the names of the loaded databases are stored in the slot LOADED.DATABASES in this unit.

The database files are stored in the "historical-data" subdirectory of the directory containing the VEG files. All the database file names are in upper case letters. Each time the Change Historical Database Screen is opened, the names of the available databases are stored in the VALUECLASS facet of the AVAILABLE.DATABASES slot of the unit CHANGE. HISTORICAL.DATABASE. This enables the names of the available databases to be displayed using a "Vertical Pushbutton" KEE ActiveImage attached to the slot so the user can select a database by simply left-clicking on the database name.

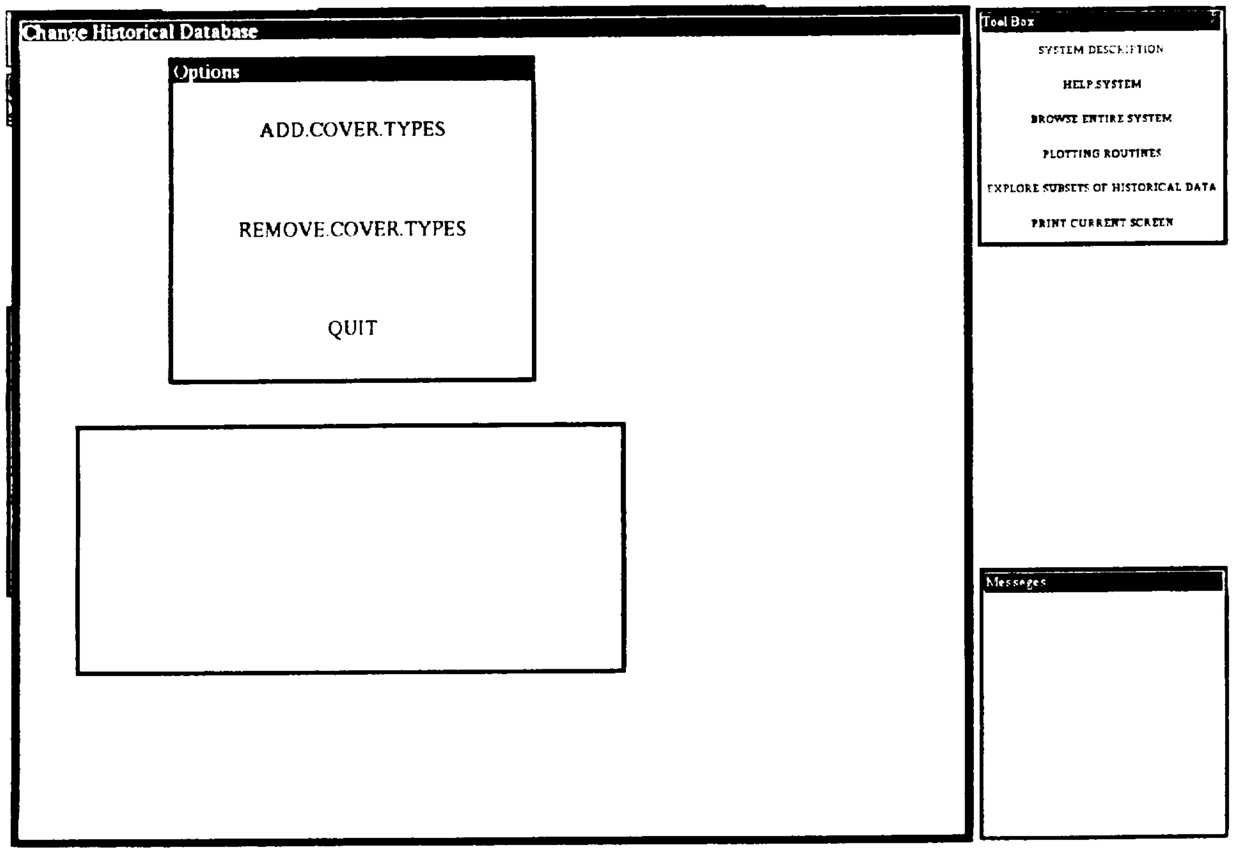
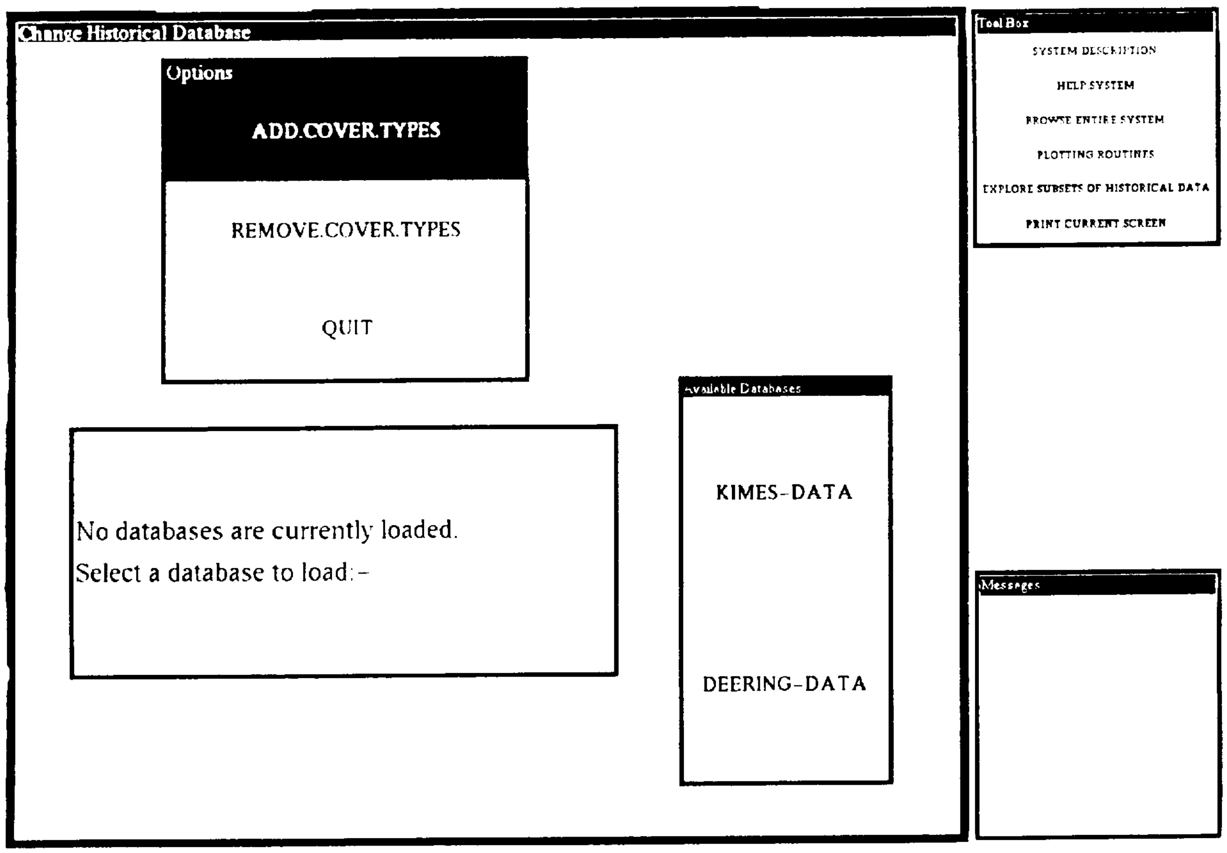


Figure 3-3
The Change Historical Database Screen

If the user selects the ADD.COVER.TYPES option (as shown in Figure 3-4) from the Change Historical Database Screen, the names of any databases that have already been loaded are displayed on the screen. A subwindow that is a Vertical Pushbutton ActiveImage attached to the AVAILABLE.DATABASES slot is opened. This subwindow contains the names of the available databases. The user is prompted to select the name of the database to load by left-clicking on the appropriate option in this subwindow.

VEG cannot run without a historical database. If the user attempts to run VEG before loading historical data, the Change Historical Database Screen is automatically opened. In this case, the ADD.COVER.TYPES option is automatically selected, indicating that the user should load historical data. The screen depicted in Figure 3-4 shows this situation.

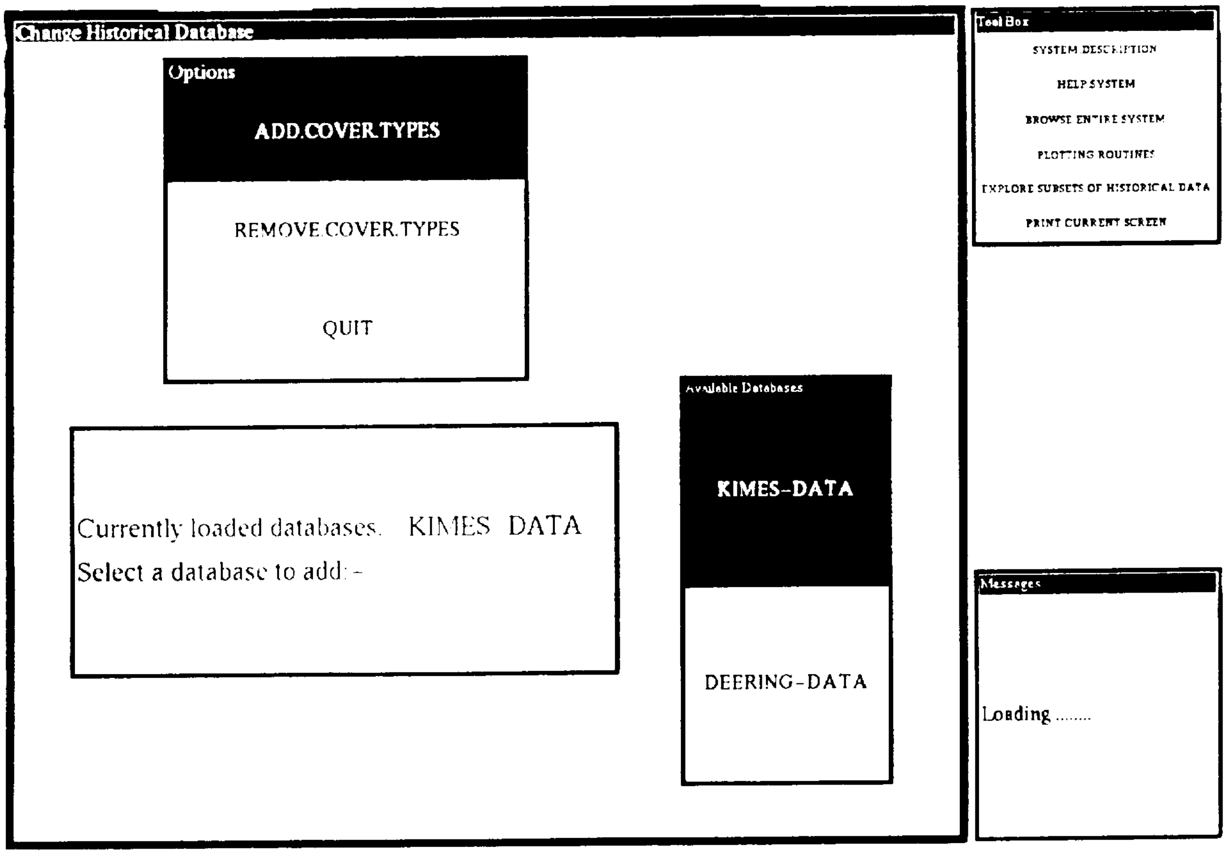


No databases are currently loaded.
Select a database to load:-

Figure 3-4
The Add Cover Types Option Before Any Cover Types have been Added

In the example shown in Figure 3-5, the database KIMES-DATA has already been loaded, and the user has selected the database "DEERING-DATA." The message "Loading ...." in the "Messages" box indicates that the data are being loaded from the file. When the loading has been completed, this message is removed. The newly loaded database is added to the list of loaded databases stored in the unit CHANGE.HISTORICAL.DATABASE. Its name is also displayed on the screen. If the user attempts to load data from a database that has already been loaded, a message is displayed and the database is not re-loaded. A message is also displayed if the user attempts to load data from an empty or missing database.

The format of the cover type data files, and an example of typical values are shown in Table 3-1. Data that apply to all data sets for the same cover type are stored first. These are followed by the data at each wavelength for the first solar zenith angle. The data for each wavelength for the remaining solar zenith angles are then listed. For clarity, only part of each set of reflectance data is shown in the table. In this example, data are available at solar zenith angles 26°, 30° and 45°. At each solar zenith angle, data are available in the wavebands 0.58 - 0.68 µm and 0.73 - 1.1 µm. In this example, the reflectance data beginning (0 0 0.231) and the spectral hemispherical reflectance value 0.1892 correspond to the solar zenith angle 26° and the waveband 0.58 - 0.68 µm. The reflectance data beginning (0 0 0.2733), and the spectral hemispherical reflectance value 0.2268 correspond to the solar zenith angle 26° and the waveband 0.73 - 1.1 µm



Currently loaded databases. KIMES DATA
Select a database to add: -

Loading ....

Figure 3-5
Adding Cover Types

Table 3-1
Historical Data File Format and an Example of Typical Values

TYPICAL VALUES	DESCRIPTION
"PLOWED-FIELD - TUNISIA AFRICA"	Description
45	Azimuth Interval
SOIL	Cover Type
(4 28 1983)	Date
0.0	Dry Biomass
0.0	Ground Cover
0.0	Height
0.0	Leaf Area Index
UNKNOWN	Leaf Orientation Distribution
(0909 1045 1217)	Local Standard Times
75	Maximum Zenith Data
0.0	Proportion Green
180	Solar Azimuth
HOMOGENEOUS	Structure
0.0	Total Hem. Reflectance
((0.58 0.68) (0.73 1.1))	Wavelengths Available
0.0	Wet Biomass
15	Zenith Interval
(26 30 45)	Solar Zenith Angles
((0 0 0.231) (15 0 0.1968) ... (75 315 0.1115))	Reflectance Data
0.1892	Spectral Hem. Reflectance
((0 0 0.2733) (15 0 0.2362) ... (75 315 0.137))	Reflectance Data
0.2268	Spectral Hem. Reflectance
((0 0 0.204) (15 0 0.1599) ... (75 315 0.101))	Reflectance Data
0.1813	Spectral Hem. Reflectance
((0 0 0.244) (15 0 0.1921) ... (75 315 0.1262))	Reflectance Data
0.2173	Spectral Hem. Reflectance
((0 0 0.1738) (15 0 0.1451) ... (75 315 0.0923))	Reflectance Data
0.1868	Spectral Hem. Reflectance
((0 0 0.2033) (15 0 0.1691) ... (75 315 0.1103))	Reflectance Data
0.2219	Spectral Hem. Reflectance

When data for a cover type are read from a file, a hierarchy of KEE units is created to hold the data. Data which are common to all sets of data for the same cover type are stored in a unit which is created as a subclass of the unit HISTORICAL.COVER.TYPES. The name of this unit begins with "COVER.TYPE." This name is automatically generated. The name of the data file from which the cover type data has been read is also stored in the cover type unit. The purpose of this is to identify the data file that was the source of the data so that all data from a selected data file can later be identified prior to deletion. A subclass of the cover type unit is created for each solar zenith angle for which data are available. Instances of these units are created to hold the reflectance data, spectral hemispherical reflectance value and the maximum and minimum wavelengths in each waveband. This is the same organization of units that was used in the previous version of VEG. As the data are read from the file, the required units are created and data are stored in the units. Each data value is checked before it is stored. If the value is of the wrong type, or it is out of range for the slot in which it is to be stored, an error message is displayed on the screen. If an error is detected, processing of the data file is aborted and all the data for the cover type that is currently being stored are deleted from VEG. Any cover types that have been previously stored correctly are not deleted. Figure 3-6 shows a hierarchy of KEE units that might be created to hold the data shown in Table 3-1.

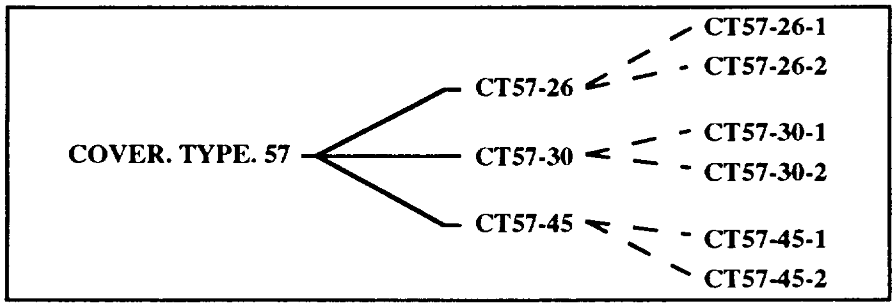
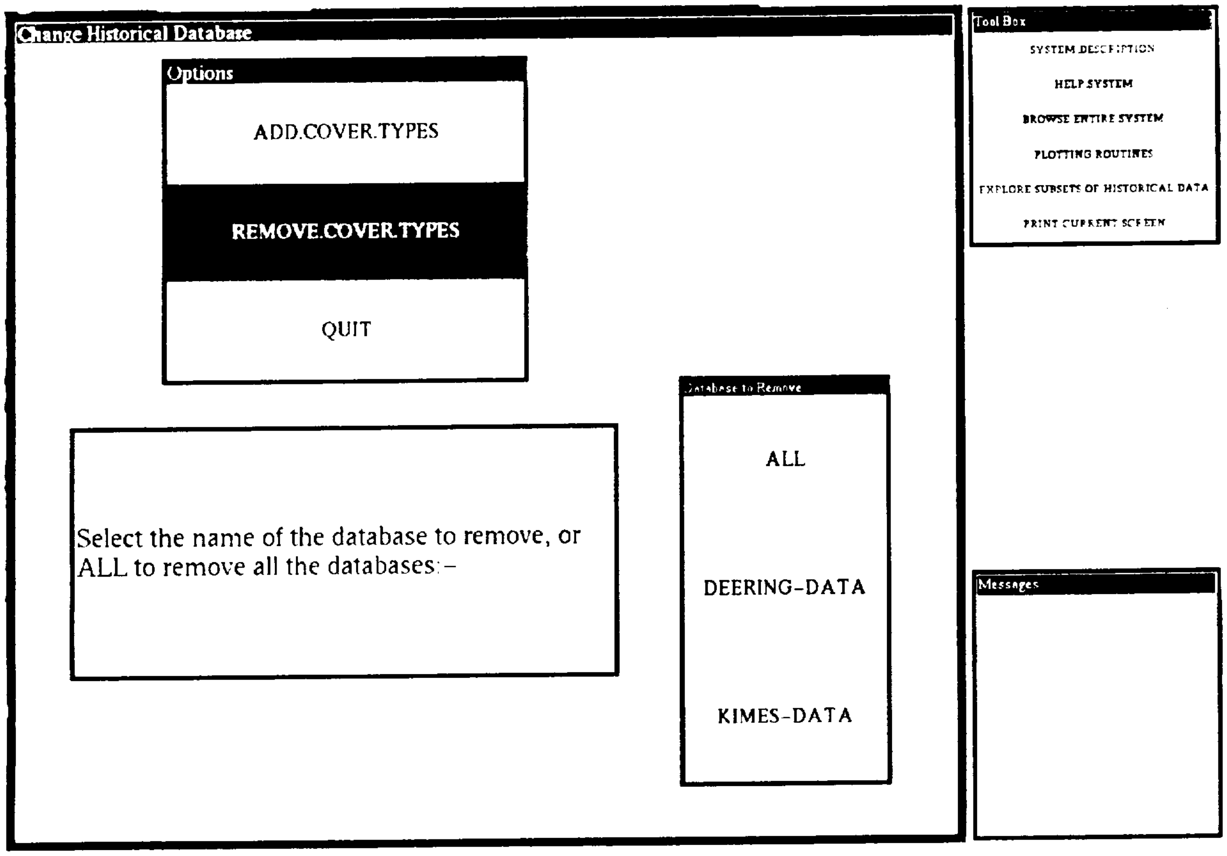


Figure 3-6
KEE Units that Hold the Data from Table 3-1

If the user selects the REMOVE.COVER.TYPES option from the Change Historical Database menu, a different subwindow is opened and the names of the currently loaded databases are displayed on the screen, as shown in Figure 3-7. The user is prompted to select the database to be removed or to select "ALL" if all the databases are to be removed. If the user selects "ALL," all the units that are descendants of the unit HISTORICAL.COVER.TYPES are deleted from VEG, and all the values are removed from the LOADED.DATABASES slot of the unit CHANGE.HISTORICAL.DATA. The message on the screen is then updated to indicate that no databases are currently loaded, and the subwindow showing the loaded databases is removed from the screen. Alternatively, if the user selects a database to be removed, VEG searches through all the cover type units that are subclasses of the unit HISTORICAL.COVER.TYPES and identifies the cover types that originated from the selected database file. These cover type units and all their subclass and member units are then deleted. The name of the deleted database is removed from the LOADED.DATABASES slot of the unit CHANGE.HISTORICAL.DATA and the subwindow containing the names of the loaded databases is updated on the screen. If no databases are loaded, the subwindow containing the names of the loaded databases is removed from the screen. Note that when databases are removed from VEG, the underlying files are not modified or destroyed.

Selecting QUIT from the Change Historical Database menu closes this screen. If the user previously selected the Change Historical Database option from the Administration menu, the

Administration menu is again displayed. If the Change Historical Database Screen was opened automatically by VEG when either the automatic or the research mode was selected because no historical data was present, the Processing Mode Screen (Figure 3-1) is reopened.



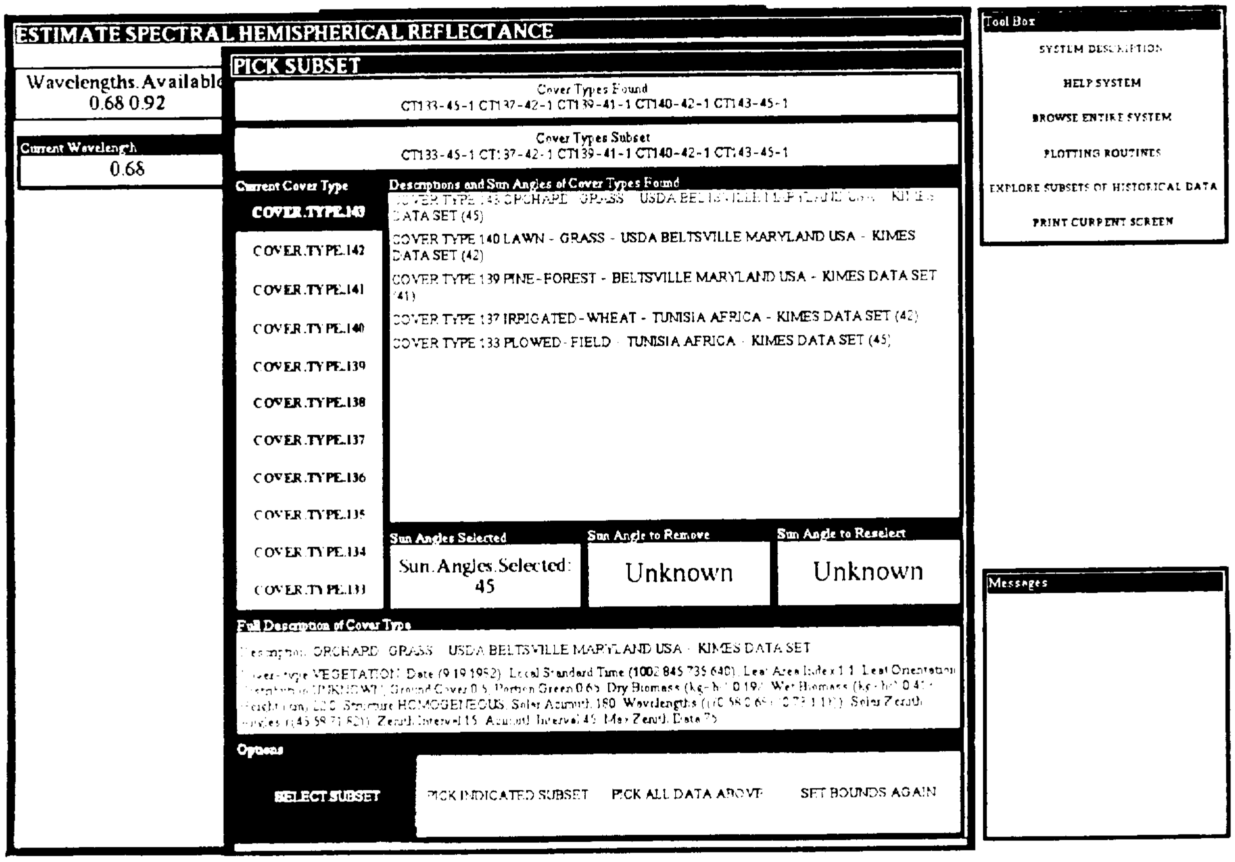
Select the name of the database to remove, or ALL to remove all the databases:-

Figure 3-7
Removing Cover Types

3.2 CHANGES TO THE PICK SUBSET SCREEN

The names of the historical cover type database units were hard-wired into various slots in the previous version of VEG. Changes have been made to these slots and the screens that use them to provide the greater flexibility required in the current version of VEG.

In all the VEG options except the learning system, the user can choose to select the restricted data set manually. If the manual Pick Restricted Data Set option is selected, the Pick Restricted Data Set Screen is opened. This screen allows the user to enter the maximum and minimum values to be considered for parameters such as height and solar zenith angle. The historical cover type database is then searched to find the cover types that match the criteria entered by the user. The user can select a subset of the matched data using the Pick Subset Screen shown in Figure 3-8. In the previous version of this screen, the user selected a cover type by left-clicking on a choice in a menu of available cover types that were hard-wired into a slot in VEG. In order to provide the additional flexibility required by the current version of VEG, the subwindow (ActiveImage) labeled "Current Cover Type" is updated to show the names of currently available cover types each time the Pick Subset Screen is opened. This is done by changing the VALUECLASS facet of the slot to which the ActiveImage is attached. The operation of the Pick Subset Screen is the same as in the previous version of VEG.


ESTIMATE SPECTRAL HEMISPHERICAL REFLECTANCE

Wavelengths.Availabl
0.68 0.92

Current Wavelength
0.68

PICK SUBSET

Cover Types Found
CT133-45-1 CT137-42-1 CT139-41-1 CT140-42-1 CT143-45-1

Cover Types Subset
CT133-45-1 CT137-42-1 CT139-41-1 CT140-42-1 CT143-45-1

Current Cover Type
COVER.TYPE.143
COVER.TYPE.142
COVER.TYPE.141
COVER.TYPE.140
COVER.TYPE.139
COVER.TYPE.138
COVER.TYPE.137
COVER.TYPE.136
COVER.TYPE.135
COVER.TYPE.134
COVER.TYPE.133

Descriptions and Sun Angles of Cover Types Found
COVER TYPE 143 ORCHARD - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (45)
COVER TYPE 140 LAWN - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (42)
COVER TYPE 139 PINE-FOREST - BELTSVILLE MARYLAND USA - KIMES DATA SET (41)
COVER TYPE 137 IRRIGATED-WHEAT - TUNISIA AFRICA - KIMES DATA SET (42)
COVER TYPE 133 PLOWED-FIELD - TUNISIA AFRICA - KIMES DATA SET (45)

Sun Angles Selected
Sun.Angles.Selected: 45

Sun Angle to Remove
Unknown

Sun Angle to Reselect
Unknown

Full Description of Cover Type
Description: ORCHARD - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET
Cover-type VEGETATION Date (9 19 1952) Local Standard Time (1002 845 735 640) Leaf Area Index 1.1 Leaf Orientation Distribution UNKNOWN Ground Cover 0.5 Portion Green 0.65 Dry Biomass (kg-ha) 0.19 Wet Biomass (kg-ha) 0.41 Height (cm) 22.0 Structure HOMOGENEOUS Solar Azimuth 180 Wavelengths ((0.58 0.68) (0.73 1.10)) Solar Zenith Angles ((45 58 71 82)) Zenith Interval 15 Azimuth Interval 45 Max Zenith Data 75

Options
SELECT SUBSET
PICK INDICATED SUBSET PICK ALL DATA ABOVE SET BOUNDS AGAIN

Tool Box
SYSTEM DESCRIPTION
HELP SYSTEM
BROWSE ENTIRE SYSTEM
PLOTTING ROUTINES
EXPLORE SUBSETS OF HISTORICAL DATA
PRINT CURRENT SCREEN

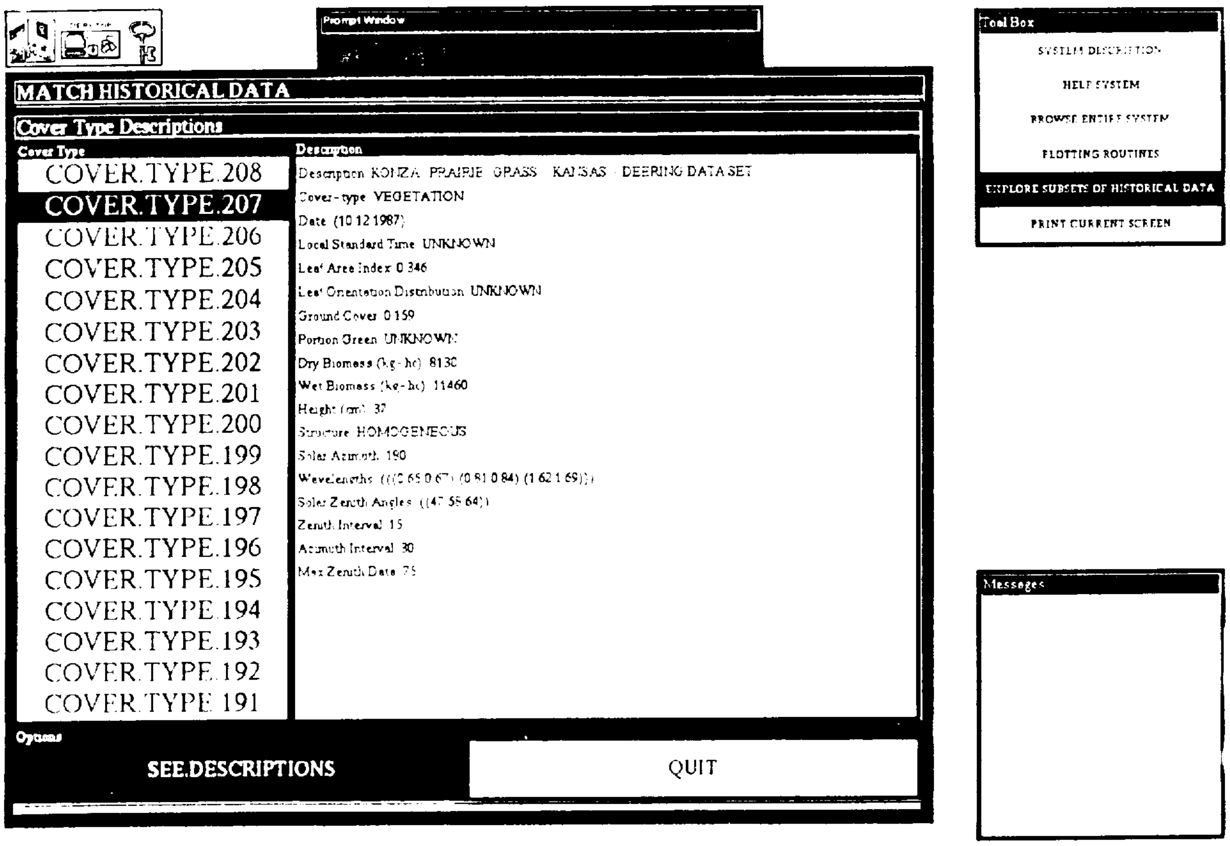
Messages

Figure 3-8
The Current Version of the Pick Subset Screen

3.3 CHANGES TO THE BROWSER

In the previous version of VEG, the names of the historical cover type units were hard-wired into a slot that was displayed on the Cover Type Descriptions Screen of the Browser (shown in Figure 3-9). This screen can be selected by choosing the EXPLORE.SUBSETS.OF.HISTORICAL.DATA browser option and then picking the SEE.DESCRIPTIONS option. The Lisp code controlling the Cover Type Descriptions Screen has been modified. Each time the Cover Type Descriptions Screen is opened, the names of the currently available cover types are stored in the VALUECLASS facet of the slot CT.TO.USE of the unit DATA.MATCHER. This causes the updating of the Vertical Pushbutton ActiveImage (subwindow) labeled "Cover Type" in Figure 3-9 which is attached to the slot. As in the previous version of VEG, when the user left-clicks on a cover type in the "Cover Type" subwindow, the description of the cover type is displayed in the "Description" subwindow. If no cover types are available, the message "No cover types are currently available" is displayed in the "Description" subwindow and the "Cover Type" subwindow is removed from the screen. The screen is closed by selecting the QUIT option at the foot of the screen.

The Cover Type Descriptions Screen may be open at the same time as the Change Historical Database Screen. In this case, the Cover Type Descriptions Screen is updated every time cover types are added to or removed from the historical cover type database.



Description KONZA PRAIRIE GRASS KANSAS DEERING DATA SET
Cover-type VEGETATION
Date (10 12 1987)
Local Standard Time UNKNOWN
Leaf Area Index 0 346
Leaf Orientation Distribution UNKNOWN
Ground Cover 0 159
Portion Green UNKNOWN
Dry Biomass (kg-ha) 8130
Wet Biomass (kg-ha) 11460
Height (cm) 37
Structure HOMOGENEOUS
Solar Azimuth 190
Wavelengths (((0 65 0 67) (0 81 0 84) (1 62 1 69)))
Solar Zenith Angles ((47 55 64))
Zenith Interval 15
Azimuth Interval 30
Max Zenith Data 75

Figure 3-9
The Cover Type Description Screen in the Browser

SECTION 4.0

TESTING AND RESULTS

All the Change Historical Database options were tested using both valid and invalid inputs. All the VEG goals, including the learning system, were tested using historical cover type data loaded from external files. After some minor changes necessitated because the available wavelength data were stored in a different format than in the previous version of VEG, all the options were found to be operating correctly. The test runs are described in detail in this section.

3.1 TEST 1

The purpose of Test 1 was to test the navigation back and forth through the various menu levels from the VEG top level to the Change Historical Database Screen. The user left-clicked on RUN.VEG, ADMINISTRATION and CHANGE.HISTORICAL.DATABASE on successive screens. As expected, the Change Historical Database Screen was opened. The user then selected QUIT in each successive menu to navigate back to the top level of VEG. This test showed that the screens between the VEG top level and the Change Historical Database Screen were opened and closed in the correct sequence.

3.2 TEST 2

This test was designed to test the ADD.COVER.TYPES option of the Change Historical Database menu. This option was selected, and the user was prompted to select the database to add to the historical cover type database. According to the display, the databases KIMES-DATA and DEERING-DATA were available. The user selected "KIMES-DATA." The message "Loading ...." appeared in the "Messages" box while the database was being loaded. Inspection of the KEE knowledge base confirmed that the database had been correctly loaded.

The user then attempted to load the same database again by again selecting "KIMES-DATA" as the database to be loaded. The message "Database KIMES-DATA has already been loaded" was displayed in the "Messages" box. The database was not reloaded.

The next part of this test, was designed to confirm that a second database could be loaded. The database "DEERING-DATA" was selected. The DEERING-DATA database was successfully loaded in addition to the KIMES-DATA database.

The operation of the ADD.COVER.TYPES option with invalid databases was tested in the final part of this test. Three new files were added to the historical-data subdirectory. These were named "EMPTY," "MISSING" and "FAULTY-DATA." The file "EMPTY" was empty. The files "MISSING" and "FAULTY-DATA" were copies of the file "KIMES-DATA." Several data items that were part of the second cover type in the file "FAULTY-DATA" were deleted so that the file had an incorrect format. The ADD.COVER.TYPES option was deselected and selected again. The names of the newly created files were then shown in the "Available Databases" subwindow. The user selected the "EMPTY" database. The message "Database file is empty" was displayed in the "Messages" box. The user then deleted the file "MISSING" from the historical-data subdirectory. When the user selected the "MISSING" file in the Change Historical Database screen, the message "Database file not found" was displayed in the "Messages" box. Finally, the user selected the "FAULTY-DATA" file. The first cover type was successfully loaded from this file. The message "File reading aborted - the data (843 1053 1254) is invalid for the slot

LEAF.AREA.INDEX." was then displayed in the "Messages" box and the reading of the file was aborted.

Test 1 showed that the ADD.COVER.TYPES option was working correctly with both valid and invalid databases.

3.3 TEST 3

The REMOVE.COVER.TYPES option was thoroughly tested in Test 3. At the beginning of this test, both databases were loaded, as in the previous test. Then the REMOVE.COVER.TYPES option was selected. The prompt confirmed that both databases were loaded and asked the user to select the database to remove, or "ALL" if all databases were to be removed. The user selected "KIMES-DATA." The database KIMES-DATA was removed. Inspection of the VEG knowledge base confirmed this. The content of the "Database to Remove" subwindow changed to report that the only database now loaded was DEERING-DATA.

The user then specified that the DEERING-DATA database should be deleted. After the deletion, the prompt on the screen indicated that no databases were currently loaded and the "Database to Remove" subwindow was removed from the screen.

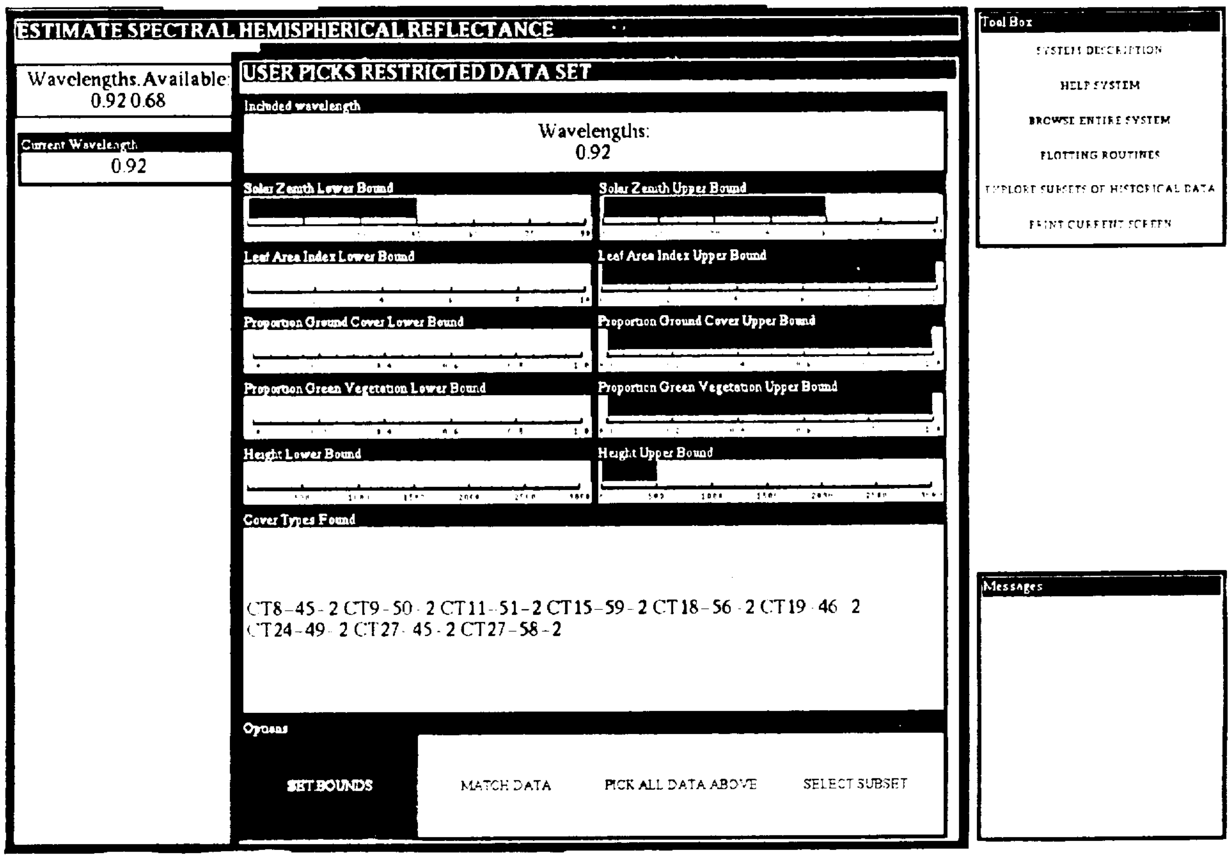
The final part of this test was designed to test the option that causes all loaded databases to be removed. First, both databases were again loaded. The option REMOVE.COVER.TYPES was then selected again, and the user selected "ALL." As expected, all the databases were deleted, the prompt on the screen then reported that no databases were loaded, and the "Database to Remove" subwindow was removed from the screen.

Test 3 confirmed that all parts of the REMOVE.COVER.TYPES option were operating correctly.

3.4 TEST 4

Test 4 was designed to test the operation of VEG using historical data loaded from external files. At the beginning of this test, the historical cover type database was empty. The user left-clicked on RUN.VEG and then RESEARCH. The Change Historical Database Screen was automatically opened and the ADD.COVER.TYPES option was automatically selected. The databases KIMES-DATA and DEERING-DATA were then loaded. When the user selected QUIT, the Processing Mode menu was again revealed. The user left-clicked on RESEARCH again. This time, the Goals menu was displayed and the user selected the goal SPECTRAL.HEMISPHERICAL.REFLECTANCE. This option was run using SAMPLE3 from the VEG knowledge base.

In this test run, the user chose to manually select the restricted data set. Using the User Pick Restricted Data Set Screen, the user indicated that cover types with the same wavelengths as SAMPLE3, solar zenith angles between 45° and 60° and height less than 500 cm should be placed in the restricted data set. Figure 4-1 shows this screen after the cover types that match the chosen parameters had been identified. The user then decided to pick a subset of the matched data using the Pick Subset Screen. The sun angle 58° for the COVER.TYPE.27 was removed using this screen as shown in Figure 4-2.


ESTIMATE SPECTRAL HEMISPHERICAL REFLECTANCE

Wavelengths Available
0.92 0.68

Current Wavelength
0.92

USER PICKS RESTRICTED DATA SET

Included wavelength

Wavelengths:
0.92

Solar Zenith Lower Bound

Solar Zenith Upper Bound

Leaf Area Index Lower Bound

Leaf Area Index Upper Bound

Proportion Ground Cover Lower Bound

Proportion Ground Cover Upper Bound

Proportion Green Vegetation Lower Bound

Proportion Green Vegetation Upper Bound

Height Lower Bound

Height Upper Bound

Cover Types Found

CT8-45-2 CT9-50-2 CT11-51-2 CT15-59-2 CT18-56-2 CT19-46-2
CT24-49-2 CT27-45-2 CT27-58-2

Options

SET BOUNDS

MATCH DATA

PICK ALL DATA ABOVE

SELECT SUBSET

Tool Box

SYSTEM DESCRIPTION

HELP SYSTEM

BROWSE ENTIRE SYSTEM

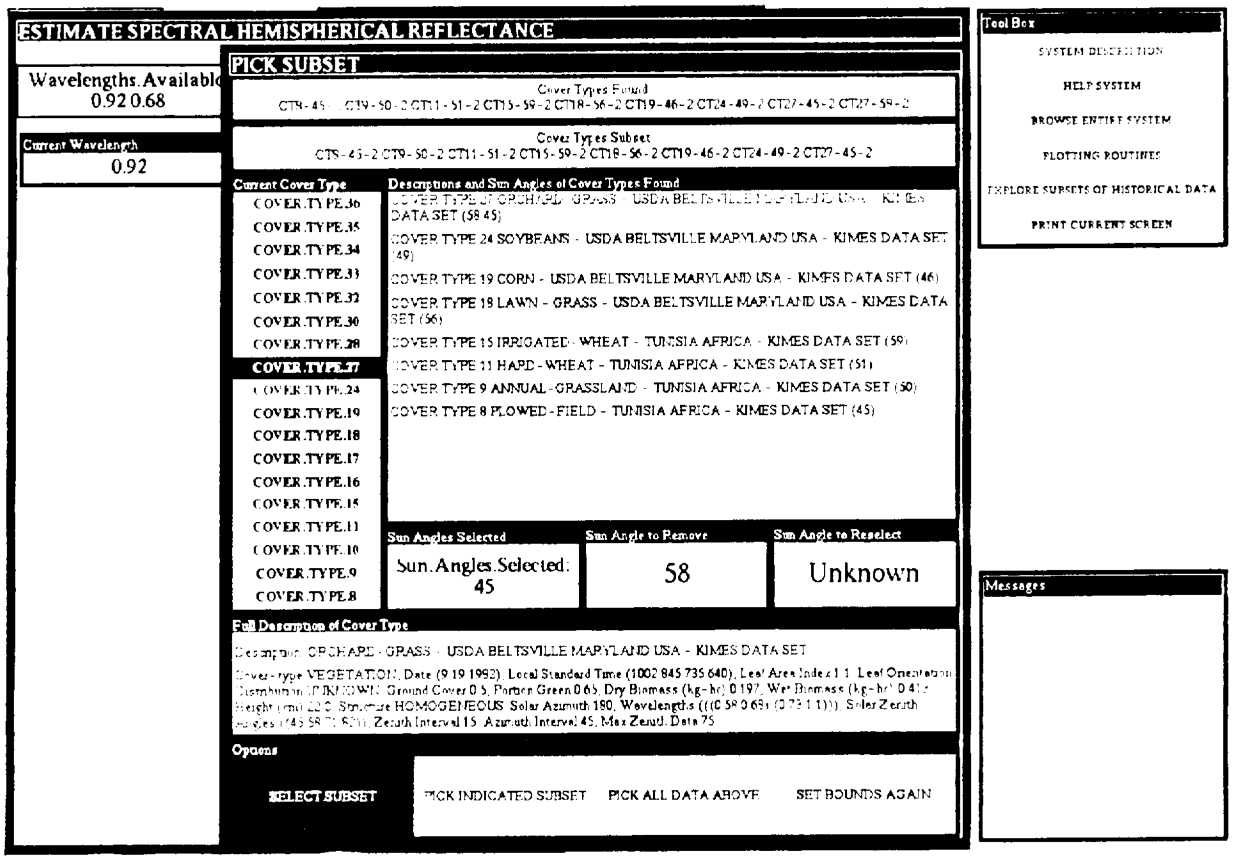
PLOTTING ROUTINES

EXPLORE SUBSETS OF HISTORICAL DATA

PRINT CURRENT SCREEN

Messages

Figure 4-1
The User Picks Restricted Data Set Screen After the Cover Types that Match the Chosen Parameters have been Identified



ESTIMATE SPECTRAL HEMISPHERICAL REFLECTANCE

Wavelengths.Availabl
0.92 0.68

Current Wavelength
0.92

PICK SUBSET

Cover Types Found
CT8-45-2 CT9-50-2 CT11-51-2 CT15-59-2 CT18-56-2 CT19-46-2 CT24-49-2 CT27-45-2 CT27-58-2

Cover Types Subset
CT8-45-2 CT9-50-2 CT11-51-2 CT15-59-2 CT18-56-2 CT19-46-2 CT24-49-2 CT27-45-2

Current Cover Type
COVER.TYPE.36
COVER.TYPE.35
COVER.TYPE.34
COVER.TYPE.33
COVER.TYPE.32
COVER.TYPE.30
COVER.TYPE.28
COVER.TYPE.27
COVER.TYPE.24
COVER.TYPE.19
COVER.TYPE.18
COVER.TYPE.17
COVER.TYPE.16
COVER.TYPE.15
COVER.TYPE.11
COVER.TYPE.10
COVER.TYPE.9
COVER.TYPE.8

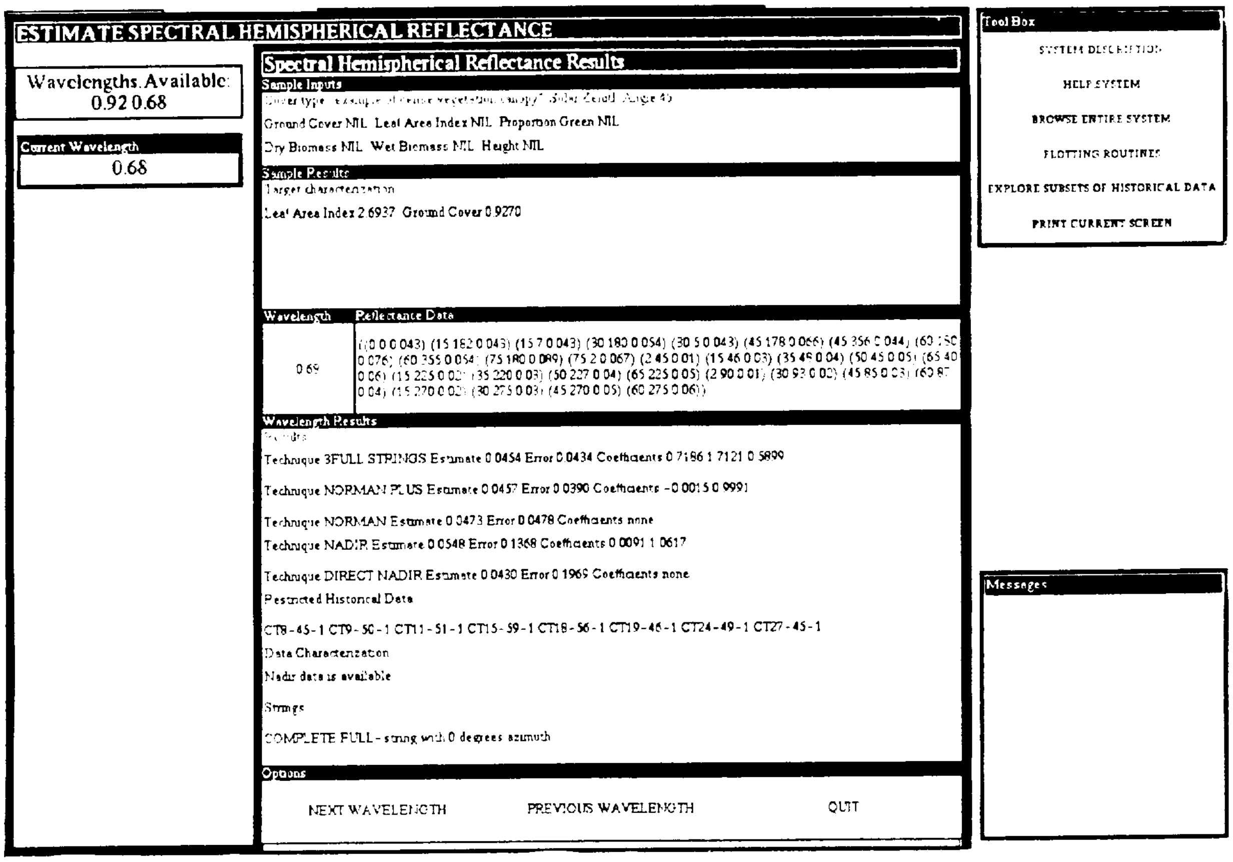
Descriptions and Sun Angles of Cover Types Found
COVER TYPE 27 ORCHARD - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (58 45)
COVER TYPE 24 SOYBEANS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (49)
COVER TYPE 19 CORN - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (46)
COVER TYPE 18 LAWN - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET (56)
COVER TYPE 15 IRRIGATED - WHEAT - TUNISIA AFRICA - KIMES DATA SET (59)
COVER TYPE 11 HARD - WHEAT - TUNISIA AFRICA - KIMES DATA SET (51)
COVER TYPE 9 ANNUAL - GRASSLAND - TUNISIA AFRICA - KIMES DATA SET (50)
COVER TYPE 8 PLOWED - FIELD - TUNISIA AFRICA - KIMES DATA SET (45)

Sun Angles Selected: Sun.Angles.Selected: 45
Sun Angle to Remove: 58
Sun Angle to Reselect: Unknown

Full Description of Cover Type
Description ORCHARD - GRASS - USDA BELTSVILLE MARYLAND USA - KIMES DATA SET
Cover-type VEGETATION, Date (9 19 1982), Local Standard Time (1002 845 735 640), Leaf Area Index 1 1, Leaf Orientation Distribution UNKNOWN, Ground Cover 0 5, Portion Green 0 65, Dry Biomass (kg-ha) 0 197, Wet Biomass (kg-ha) 0 41, Height (cm) 22 0, Structure HOMOGENEOUS, Solar Azimuth 180, Wavelengths (((0 58 0 68) (0 73 1 1))), Solar Zenith Angles ((45 58 71 82)), Zenith Interval 15, Azimuth Interval 45, Max Zenith Data 75

Figure 4-2
The Pick Subset Screen After a Cover Type has been Removed

The processing of the sample was then completed and the results were displayed as shown in Figure 4-3. This test showed that SPECTRAL.HEMISPHERICAL. REFLECTANCE operated correctly using historical cover type data loaded from external files. The test also showed that the Pick Subset Screen that had been modified to deal with the historical data loaded from files was operating correctly.



ESTIMATE SPECTRAL HEMISPHERICAL REFLECTANCE

Wavelengths Available:
0.92 0.68

Current Wavelength
0.68

Spectral Hemispherical Reflectance Results

Sample Inputs

Cover type (example of dense vegetation canopy) Solar Zenith Angle 45

Ground Cover NIL Leaf Area Index NIL Proportion Green NIL

Dry Biomass NIL Wet Biomass NIL Height NIL

Sample Results

Target characterization

Leaf Area Index 2.6937 Ground Cover 0.9270

Wavelength	Reflectance Data
0.69	((0 0 0.043) (15 182 0.043) (15 7 0.043) (30 180 0.054) (30 5 0.043) (45 178 0.066) (45 356 0.044) (60 180 0.076) (60 355 0.054) (75 180 0.089) (75 2 0.067) (2 45 0.01) (15 46 0.03) (35 49 0.04) (50 45 0.05) (65 40 0.06) (15 225 0.02) (35 220 0.03) (50 227 0.04) (65 225 0.05) (2 90 0.01) (30 93 0.02) (45 85 0.03) (60 87 0.04) (15 270 0.02) (30 275 0.03) (45 270 0.05) (60 275 0.06))

Wavelength Results

Results

Technique 3FULL STRINGS Estimate 0.0454 Error 0.0434 Coefficients 0.7186 1.7121 0.5899

Technique NORMAN PLUS Estimate 0.0457 Error 0.0390 Coefficients -0.0015 0.9991

Technique NORMAN Estimate 0.0473 Error 0.0478 Coefficients none

Technique NADIR Estimate 0.0548 Error 0.1368 Coefficients 0.0091 1.0617

Technique DIRECT NADIR Estimate 0.0430 Error 0.1969 Coefficients none

Restricted Historical Data

CT8-45-1 CT9-50-1 CT11-51-1 CT15-59-1 CT18-56-1 CT19-46-1 CT24-49-1 CT27-45-1

Data Characterization

Nadir data is available

Strings

COMPLETE FULL - string with 0 degrees azimuth

Options

Figure 4-3
The Output Screen at the End of Test 4

4.5 TEST 5

In Test 5, VEG was run in Automatic Mode and the operation of all the VEG goals except the Learning System was tested. In Automatic mode, the restricted data set is selected automatically by the system. Several of the test runs from Task B used the VEG Automatic mode. When Task B was performed, the historical cover type database was stored in the VEG knowledge base. The test runs were reported in JJM Systems Report B921016-U-2R02. In Test 5 of the current task, databases KIMES-DATA and DEERING-DATA were loaded into VEG. The selected Task B tests were then repeated. The results of the current runs were compared with the Task B results. The results from the two sets of runs were found to be very similar but not identical. In particular, there were small but insignificant differences in the error terms and coefficients. The differences can be explained by the fact that the restricted data set is organized in a different order in the current version of VEG compared with the previous version. The algorithm that selects the restricted historical data picks the first ten statistical matches. For example, if the DEERING-DATA database is loaded before the KIMES-DATA database, the first ten matches may not be the same as in the case in which the KIMES-DATA database is loaded after the DEERING-DATA database. If the first ten matches are different, a different set of cover types are selected for calculating the coefficients and the error terms. This leads to slightly different results.

Test 5 showed that all the VEG goals tested were operating correctly using historical cover type data loaded from external files.

4.6 TEST 6

The Learning System was tested in Test 6 to confirm that it would operate correctly using historical cover type data loaded from external files. The Learning System was developed in Tasks C and D. When these tasks were tested, the historical database was part of the VEG knowledge base. In Test 6 of the current task, test runs 1, 2 and 3 from the Task C and D report, JJM Systems Report B921014-U-2R03, were repeated using the current version of VEG, including historical data loaded from external files. The results of Test 6 were identical to the results from the Tasks C and D in the earlier report. This provided further confirmation that the historical data loaded from external files had been correctly incorporated in VEG.

4.7 TEST 7

In this test the Cover Type Description Screen of the Browser was tested. The Lisp code controlling this screen had been modified so that it would work with historical data loaded from external files. This screen was selected by choosing the EXPLORE.SUBSETS.OF. HISTORICAL.DATA browser option and then picking the SEE.DESCRIPTIONS option. The display indicated that cover types 564 through 581 were available. The user selected COVER.TYPE.570. A description of this cover type was displayed.

The second part of this test was designed to test whether the Cover Type Description Screen would be correctly updated if cover types were added to or removed from the historical cover type database while the screen was open. Without closing the Cover Type Description Screen, the user opened the Change Historical Database Screen. The user then removed all the historical cover type data from VEG. Next, the user reloaded the KIMES-DATA database and then the DEERING-DATA database. Finally, the user removed the DEERING-DATA database. Inspection of the Cover Type Description Screen after each step in this test confirmed that it was being correctly updated.

Test 7 showed that the Cover Type Description Screen of the Browser was operating correctly.

SECTION 5.0

CONCLUSIONS

The historical cover type database has been removed from VEG and stored as a series of external files. An interface has been implemented. This interface allows the user to load historical cover type data from the files into VEG and subsequently delete the loaded data from VEG. Some minor changes were made to VEG to enable it to operate with the loaded data.

All the options provided by the new interface were tested. Data were loaded from external files into VEG and the operation of all the VEG goals was then tested. The test runs showed that the interface was working correctly and that the data had been loaded successfully from the external files into VEG. The tests also confirmed that the historical cover type data were correctly integrated into VEG.

The database of historical cover types in the previous version of VEG occupied 123 units. Removing this database to external files and only loading a subset of the data has reduced the memory requirements for VEG.

Allowing the user to select the historical data to load has made VEG more flexible. VEG is now useful to a wider group of scientists. Scientists with different areas of interest can use different sets of historical data with VEG and restrict the use of VEG to the particular types of data that are of interest to them.

Relational database environments could easily be used to hold the historical cover type data in place of the flat files. However, the cost of the KEE interface to a relational database is presently prohibitive.

REFERENCES

JJM Systems, Inc. April 1993. An Expert System for Inferring Vegetation Characteristics - Interface for the Addition of Techniques (Task H). Ivyland, PA. C931021-U-2R07.

JJM Systems, Inc. October 1992. An Expert System for Inferring Vegetation Characteristics - Implementation of Additional Techniques (Task E). Ivyland, PA. B921019-U-2R04.

JJM Systems, Inc. September 1992. An Expert System for Inferring Vegetation Characteristics - The Learning System (Tasks C and D). Ivyland, PA. B921014-U-2R03.

JJM Systems, Inc. September 1992. An Expert System for Inferring Vegetation Characteristics - Output of Results to a File (Task B). Ivyland, PA. B921016-U-2R02.

Kimes, D.S., Harrison, P.R. and Harrison, P.A.. March 1992. Learning Class Descriptions from a Data Base of Spectral Reflectance with Multiple View Angles. In IEEE Transactions on Geoscience and Remote Sensing, Vol. 30, No 2, pp. 315-325.

Kimes, D.S., Harrison, P.R. and Ratcliffe, P.A.. October 1991. A Knowledge-Based Expert System for Inferring Vegetation Characteristics. International Journal of Remote Sensing, Vol 12, no 10: pp. 1987-2020.

APPENDIX A

LISTING OF CODE FOR CHANGING THE HISTORICAL DATABASE

```
;;; veg-methods4.lisp
;;;
;;; Task F
;;; Methods for making historical cover type database external to VEG
;;;
;;; Written by Ann & Patrick Harrison
;;; Created March 16 1993
;;; Last Modified May 21 1993

(in-package `kee)

(defun input-historical-data-from-file (file db-name)
"Controls the input of historical cover type data from a file."
  (catch 'invalid-historical-data
    (my-documentation-print "Loading ........")
   (with-open-file (str file :direction :input)
     (store-historical-data db-name str)))
  (update-current-cover-types))

(defvar *historical-data-ct-slot-list*
   '(azimuth.interval cover.type date dry.biomass.kg.hc ground.cover
height.cm leaf.area.index leaf.orientation.distribution local.standard.time
max.zenith.data proportion.green solar.azimuth structure total.hem.reflectance
wavelengths wet.biomass.kg.hc zenith.interval solar.zenith.angles)
"Slots in which to store historical data at the cover type level.")

(defvar *historical-data-wavelength-slot-list*
     '(raw.data spectral.hem.reflectance)
"Slots in which to store historical data at the wavelength level.")

;;; Note that the function read-file is included in the methods file
;;; veg-methods1.lisp.

(defun store-historical-data (db-name str)
"Stores the data for any number of cover-types."
  (add.value 'change.historical.database 'loaded.databases db-name)
  (do ((first-slot (read-file str)(read-file str)))
     ((null first-slot)(clear-prompt))
    (let ((new-cover-type (create.unit
                              (gentemp "COVER.TYPE.")
                              'veg 'historical.cover.types ()))
           (wavelengths nil)
           (solar-zenith-angles nil))
      (put.value new-cover-type 'description first-slot)
      (put.value new-cover-type 'database db-name)
      (dolist (slot *historical-data-ct-slot-list*)
          (let ((data (read-file str)))
            (cond ((null data)
                    (abort-historical-data-reading new-cover-type 'eof))
                                         ;Eof in wrong place
                   ((not (valid-historical-ct-data data slot))
                    (abort-historical-data-reading new-cover-type data slot))
                   (t (put.value new-cover-type slot data)))
            (when (eq slot 'wavelengths)
            (setf wavelengths data))
           (when (eq slot 'solar.zenith.angles)
            (setf solar-zenith-angles data))))
       (process-sun-ang-data str new-cover-type solar-zenith-angles
                                   wavelengths))))

(defun process-sun-ang-data (str new-cover-type solar-zenith-angles
                                        wavelengths)
"Creates the required units and stores the sun angle data."
 (dolist (sun-ang solar-zenith-angles)
   (let ((new-sun-ang (create.unit
                                   (get-sun-angle-unit-name new-cover-type sun-ang)
                                   'veg new-cover-type)))
     (put.value new-sun-ang 'solar.zenith.angle sun-ang)
     (process-wave-data str new-cover-type new-sun-ang wavelengths))))

(defun process-wave-data (str new-cover-type new-sun-ang wavelengths)
"Creates the required units and stores the data at the wavelength level.  These
data consist of the maximum and minimum wavelengths, raw data and spectral
hemispherical reflectance."
 (let ((n 1))
   (dolist (wave wavelengths)
     (let ((new-wave (create.unit
                                  (get-wave-unit-name new-sun-ang n)
                                  'veg nil new-sun-ang)))
          (put.value new-wave 'wavelength.min (first wave))
          (put.value new-wave 'wavelength.max (second wave))
          (dolist (slot *historical-data-wavelength-slot-list*)
            (let ((data (read-file str)))
              (cond ((null data)
                       (abort-historical-data-reading new-cover-type 'eof))
                                               ;Eof in wrong place
                      ((not (valid-historical-wave-data data slot))
                       (abort-historical-data-reading new-cover-type data slot ))
                      (t (put.value new-wave slot data)))))
          (incf n)))))

(defun get-sun-angle-unit-name (new-cover-type new-sun-ang)
"Returns the name of a unit composed of the combination of the cover-type name
and the sun angle."
  (intern (format () "CT~A-~A"
                   (string-trim "COVER.TYPE."
                                (string (unit.name new-cover-type)))
                   new-sun-ang)))

(defun get-wave-unit-name (new-sun-ang n)
"Returns the name of a unit composed of the combination of the sun-angle unit
 name and a number."
  (intern (format () "~A-~A"
                   (unit.name new-sun-ang)
                   n)))
```

```
(defun valid-historical-ct-data (data slot)
"Returns t if the data are valid for the slot and nil otherwise."
 (case slot
  (azimuth.interval (and (integerp data)
                         (>= data 0)
                         (<= data 45)))
  (cover.type (member data `(soil vegetation)))
  (date t)
  (dry.biomass.kg.hc (or (eq data 'unknown)
                         (and (numberp data)
                              (>= data 0)
                              (<= data 25000))))
  (ground.cover (or (eq data 'unknown)
                    (and (numberp data)
                         (>= data 0)
                         (<= data 1))))
  (leaf.orientation.distribution t)
  (local.standard.time t)
  (max.zenith.data (and (integerp data)
                        (>= data 0)
                        (<= data 90)))
  (proportion.green (or (eq data 'unknown)
                        (and (numberp data)
                            (>= data 0)
                            (<= data 1))))
  (solar.azimuth (and (integerp data)
                      (>= data 0)
                      (<= data 360)))
  (structure (member data '(homogeneous heterogeneous)))
  (total.hem.reflectance (and (numberp data)
                              (>= data 0)
                              (<= data 1)))
  (wavelengths (and (consp data)
                    (dolist (waves data t)
                       (unless (and (consp waves)
                                    (= (length waves) 2)
                                    (dolist (wave waves t)
                                     (unless (and (numberp wave)
                                                  (>= wave 0)
                                                  (<= wave 10))
                                         (return nil))))
                        (return-from valid-historical-ct-data nil)))))
  (wet.biomass.kg.hc (or (eq data 'unknown)
                         (and (numberp data)
                              (>= data 0)
                              (<= data 25000)))) ;Dan confirm
  (zenith.interval (and (integerp data)
                        (>= data 0)
                        (<= data 45)))
  (description (stringp data))
  (leaf.area.index (or (eq data 'unknown)
                       (and (numberp data)
                           (>= data 0)
                           (<= data 10))))
  (height.cm (and (numberp data)
                  (>= data 0)
                  (<= data 3000)))
  (number.wavelengths (and (integerp data)
                           (>= data 0)
                           (<= data 10)))
  (solar.zenith.angles (and (consp data)
                            (dolist (dat data t)
                               (unless (and (integerp dat)
                                            (>= dat 0)
                                            (<= dat 90))
                                 (return-from valid-historical-ct-data nil)))))))

(defun valid-historical-wave-data (data slot)
"Returns t if the data are valid for the slot and nil otherwise."
  (case slot
   (raw.data (valid-reflectance-data data))
   (t (and (numberp data) ;Must be spectral hemispherical reflectance
           (>= data 0)
           (<= data 1)))))

(defun abort-historical-data-reading (new-cover-type data &optional slot)
"Displays an error message and aborts the reading of the file if invalid data
are encountered or the end of file is encountered in the wrong place."
  (put.value 'methods 'general.message
     (if (eq data 'eof)
         "File reading aborted - end of file encountered prematurely"
         (format ()
           "File reading aborted - the data ~S is invalid for the slot ~S"
           data slot)))
  ;;; Remove from the historical data base all data for this cover type."
  (dolist (sun-ang (unit.children new-cover-type 'subclass))
   (dolist (wave (unit.children sun-ang 'member))
     (delete.unit wave))
   (delete.unit sun-ang))
   (delete.unit new-cover-type)
  (throw 'invalid-historical-data nil))

(defun open-admin-menu ()
"Opens and initializes the Admin menu."
 (remove.all.values 'admin 'options)
 (unitmsg 'viewport-admin.1 'open-panel!))

(defun open-change-historical-database-menu ()
"Opens and initializes the Change Historical Database menu."
 (unitmsg 'viewport-historical.database.1 'open-panel!)
 (remove.all.values 'change.historical.database 'options)
 (put.value 'change.historical.database 'entry.box "")
 (put.value 'change.historical.database 'message ""))
```

```
(defun empty-db-file (db)
"Returns t if the file is empty and nil otherwise."
  (with-open-file (str db :direction :input)
    (let ((len (file-length str)))
      (or (null len)
          (zerop len)))))

(defun correct-empty-db ()
"This function is called when the historical cover type database is found to be
empty. The Change Historical Database screen is opened and set to prompt the
user to add cover types."
  (unitmsg 'viewport-historical.database.1 'open-panel!)
  (put.value 'change.historical.database 'entry.box "")
  (put.value 'change.historical.database 'options
     'add.cover.types)
  (put.value 'change.historical.database 'message
     (format ()
"No databases are currently loaded.  Select a database to load:-"
        (show-available-databases))))

(defun show-available-databases ()
"Updates the interface to show the available databases."
  (let ((dbs (list-available-databases)))
    (cond ((null dbs) (put.value 'change.historical.database 'message
                              "No databases are available"))
          (t (remove.all.values 'change.historical.database
                   'available.databases)
            (put.facet.value 'change.historical.database 'available.databases
                  'valueclass (cons 'one.of dbs))
            (unitmsg
             'windowpane-available.databases-of-change.historical.database.1
             'update!)
            (unitmsg
             'windowpane-available.databases-of-change.historical.database.1
             'open!)))))

(defun list-available-databases ()
"Returns a list of all the files in the subdirectory historical-data."
  (let ((dbs nil))
    (dolist (item (directory "historical-data/*") dbs)
      (let ((file (file-namestring item)))
          (unless (equal file "")
            (push file dbs))))))

(defun show-loaded-databases ()
"Updates the interface to show the available databases"
  (let ((dbs (get.values 'change.historical.database 'loaded.databases)))
    (cond ((null dbs) (put.value 'change.historical.database 'message
                              "No databases are currently loaded"))
          (t (remove.all.values 'change.historical.database
                   'database.to.remove)
            (put.facet.value 'change.historical.database 'database.to.remove
                  'valueclass (cons 'one.of (cons 'all dbs)))
            (unitmsg
              'windowpane-database.to.remove-of-change.historical.database.2
              'update!)
            (unitmsg
             'windowpane-database.to.remove-of-change.historical.database.2
             'open!)))))

(defun remove-historical-database (db)
"Remove all the historical database units originating from the named database
from VEG."
  (remove.value 'change.historical.database 'loaded.databases db)
  (dolist (cover-type (unit.children 'historical.cover.types 'subclass))
    (when (equal db (get.value cover-type 'database))
      (dolist (sun-ang (unit.children cover-type 'subclass))
          (dolist (wave (unit.children sun-ang 'member))
            (delete.unit wave))
          (delete.unit sun-ang))
      (delete.unit cover-type)))
  (update-current-cover-types))

(defun remove-all-historical-database ()
"Remove all the historical database units from VEG."
  (remove.all.values 'change.historical.database 'loaded.databases)
  (dolist (cover-type (unit.children 'historical.cover.types 'subclass))
    (dolist (sun-ang (unit.children cover-type 'subclass))
      (dolist (wave (unit.children sun-ang 'member))
          (delete.unit wave))
      (delete.unit sun-ang))
    (delete.unit cover-type))
  (update-current-cover-types))

(defun update-data-matcher-if-necessary ()
"If the see descriptions screen of the data matcher in the browser is currently
open, update this screen."
  (when (eq (get.value 'viewport-data.matcher.2 'openp) 'open)
    (update-browser-see-descriptions-screen)))

(defun update-current-cover-types ()
"Updates the current cover types slot of the unit historical cover types."
  (put.value 'historical.cover.types 'current.cover.types
      (get-unit-names (unit.children 'historical.cover.types 'subclass))))

(defun update-pick-subset-screen ()
"Updates the pick subset screen for selecting restricted historical data."
  (remove.all.values '3.create.restricted.data 'ct.to.use)
  (put.facet.value '3.create.restricted.data 'ct.to.use 'valueclass
      (cons 'one.of (get.value 'historical.cover.types
                               'current.cover.types)))
  (unitmsg
   'windowpane-ct.to.use-of-3.create.restricted.data.3
   'update!))
```

```
(defun update-browser-see-descriptions-screen ()
"Updates the see descriptions screen from the browser."
(let ((cts (get.value 'historical.cover.types
                                   'current.cover.types))
     (current-ct (get.value 'data.matcher 'ct.to.use)))
  (remove.all.values 'data.matcher 'ct.to.use)
  (put.value 'data.matcher 'ct.full.description "")
  (cond (cts
            (put.facet.value 'data.matcher 'ct.to.use 'valueclass
               (cons 'one.of cts))
            (unitmsg
             'windowpane-ct.to.use-of-data.matcher.4
             'update!)
            (unitmsg
             'windowpane-ct.to.use-of-data.matcher.4 'open!)
            (when (unit.exists.p current-ct)
              (put.value 'data.matcher 'ct.to.use current-ct)))
          (t (put.value 'data.matcher 'ct.full.description
                "No cover types are currently available")
            (unitmsg
             'windowpane-ct.to.use-of-data.matcher.4 'close!)))))
```

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Expert System Shell for Inferring Vegetation Characteristics - Changes to the Historical Cover Type Database (Task F)		5. Report Date May 1993
		6. Performing Organization Code
7. Author(s) P. Ann Harrison and Patrick R. Harrison		8. Performing Organization Report No. C931020-U-2R05
		10. Work Unit No. 462-61-14
9. Performing Organization Name and Address JJM Systems, Inc. One Ivybrook Blvd., Suite 190 Ivyland, PA 18974		11. Contract or Grant No. NAS5-30127
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 NASA/Goddard Space Flight Center Greenbelt, MD 20771		13. Type of Report and Period Covered Task Report for Task F March - May 1993
		14. Sponsoring Agency Code

15. Supplementary Notes

The Lisp and KEE code for this work is available on a Sun Cartridge Tape.

16. Abstract

The NASA VEGetation Workbench (VEG) is a knowledge based system that infers vegetation characteristics from reflectance data. All the options in VEG make use of a database of historical cover types. This database contains results from experiments by scientists on a wide variety of different cover types. In the previous version of VEG, the historical cover type database was stored as part of the VEG knowledge base. This database has been removed from the knowledge base. It is now stored as a series of flat files that are external to VEG. The report summarizes the use of the historical cover type database in VEG. It then describes the new interface to the files containing the historical data. Runs to test the operation of the new interface and to test the operation of VEG using historical data loaded from external files are also described.

17. Key Words (Suggested by Author(s)) EXPERT SYSTEM, ARTIFICIAL INTELLIGENCE, REMOTE SENSING, LEARNING, DISCRIMINATION		18. Distribution Statement UNCLASSIFIED - UNLIMITED	
19. Security Classif. (of this report) UNCLASSIFIED	20. Security Classif. (of this page) UNCLASSIFIED	21. No. of pages 37	22. Price

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, VA 22161-2171

APPENDIX B

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - ATMOSPHERIC TECHNIQUES (TASK G)

N94-15331
NASA-CR-194570

JJM
SYSTEMS INC

C931031-U-2R06

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - ATMOSPHERIC TECHNIQUES (TASK G)

October 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
1225 Jefferson Davis Hwy., Suite 190
Arlington, VA 22202

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

SECTION 1.0

INTRODUCTION

The NASA VEGetation Workbench (VEG) infers vegetation characteristics from reflectance data. For a detailed description of VEG, see references 1 and 2. A number of subgoals are available in VEG. In the previous version of VEG, the subgoals SPECTRAL.HEMISPHERICAL.REFLECTANCE, TOTAL.AND.SPECTRAL.HEMISPHERICAL.REFLECTANCE, PROPORTION.GROUND.COVER, VIEW.ANGLE.EXTENSION and LEARN.CLASS.DESCRIPTIONS were implemented.

The structure of the subgoals in VEG has been modified. Subgoals are now divided into categories. Two new subgoals in the category ATMOSPHERIC.TECHNIQUES have been added to VEG. The basic framework and interfaces for these subgoals have been implemented. No techniques for these subgoals were yet available so dummy techniques for each subgoal were included in VEG. Replacement of the dummy techniques with the real techniques when they become available should require little additional work.

This report describes the reorganization of VEG subgoals into categories and the new subgoals ATMOSPHERIC.PASSES and ATMOSPHERIC.CORRECTIONS. The code for the Lisp methods involved is included in Appendix A. A Sun cartridge tape containing these Lisp methods and the current version of VEG including the subgoal category ATMOSPHERIC.TECHNIQUES has been delivered to the NASA GSFC technical representative.

SECTION 2.0

THE SUBGOAL CATEGORY ATMOSPHERIC TECHNIQUES IN THE VEG RESEARCH MODE

The structure of the VEG subgoals has been reorganized into four categories as shown in Figure 2-1. When the user runs VEG and selects Research Mode, the Categories menu is displayed. This menu allows the user to select the required subgoal category. Selecting the category VEGETATION.PARAMETER.TECHNIQUES allows the user to select the VEG subgoals TOTAL.AND.SPECTRAL.HEMISPHERICAL.REFLECTANCE, SPECTRAL. HEMISPHERICAL.REFLECTANCE, PROPORTION.GROUND.COVER and VIEW.ANGLE. EXTENSION. Selecting the category LEARNING.SYSTEM invokes the learning system. The option NEURAL.NETWORK is included in the categories submenu although this category has not yet been implemented.

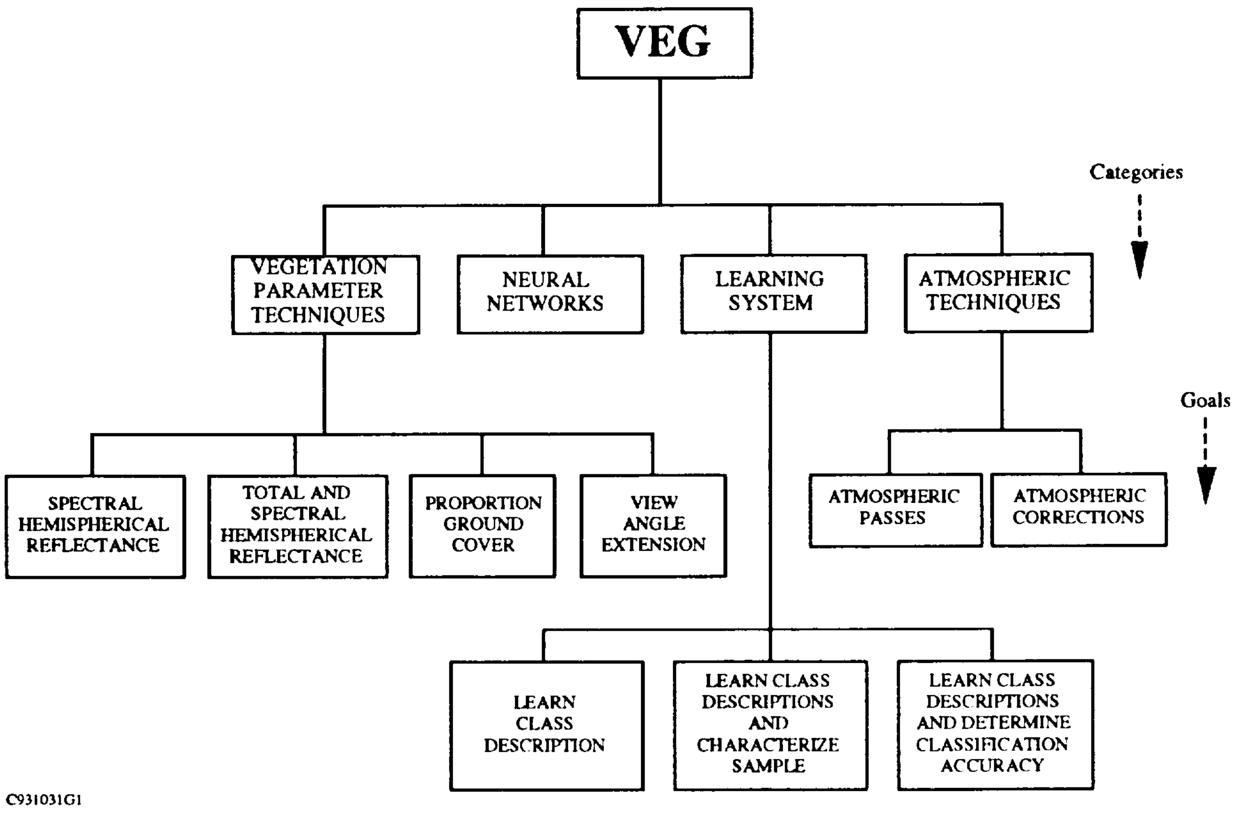

VEG
Categories
Goals
VEGETATION PARAMETER TECHNIQUES
NEURAL NETWORKS
LEARNING SYSTEM
ATMOSPHERIC TECHNIQUES
SPECTRAL HEMISPHERICAL REFLECTANCE
TOTAL AND SPECTRAL HEMISPHERICAL REFLECTANCE
PROPORTION GROUND COVER
VIEW ANGLE EXTENSION
ATMOSPHERIC PASSES
ATMOSPHERIC CORRECTIONS
LEARN CLASS DESCRIPTION
LEARN CLASS DESCRIPTIONS AND CHARACTERIZE SAMPLE
LEARN CLASS DESCRIPTIONS AND DETERMINE CLASSIFICATION ACCURACY
C931031G1

Figure 2-1
Categories of Subgoals in VEG

A new category, ATMOSPHERIC.TECHNIQUES, has been added to VEG. When this category is selected, the menu shown in Figure 2-2 is displayed. The subgoal ATMOSPHERIC.PASSES allows the scientist to take reflectance data measured at ground level and predict what the reflectance values would be if the data were measured at a different atmospheric height. The subgoal ATMOSPHERIC.CORRECTIONS allows atmospheric corrections to be made to data collected from an aircraft or by a satellite to determine what the equivalent reflectance values would be if the data were measured at ground level.

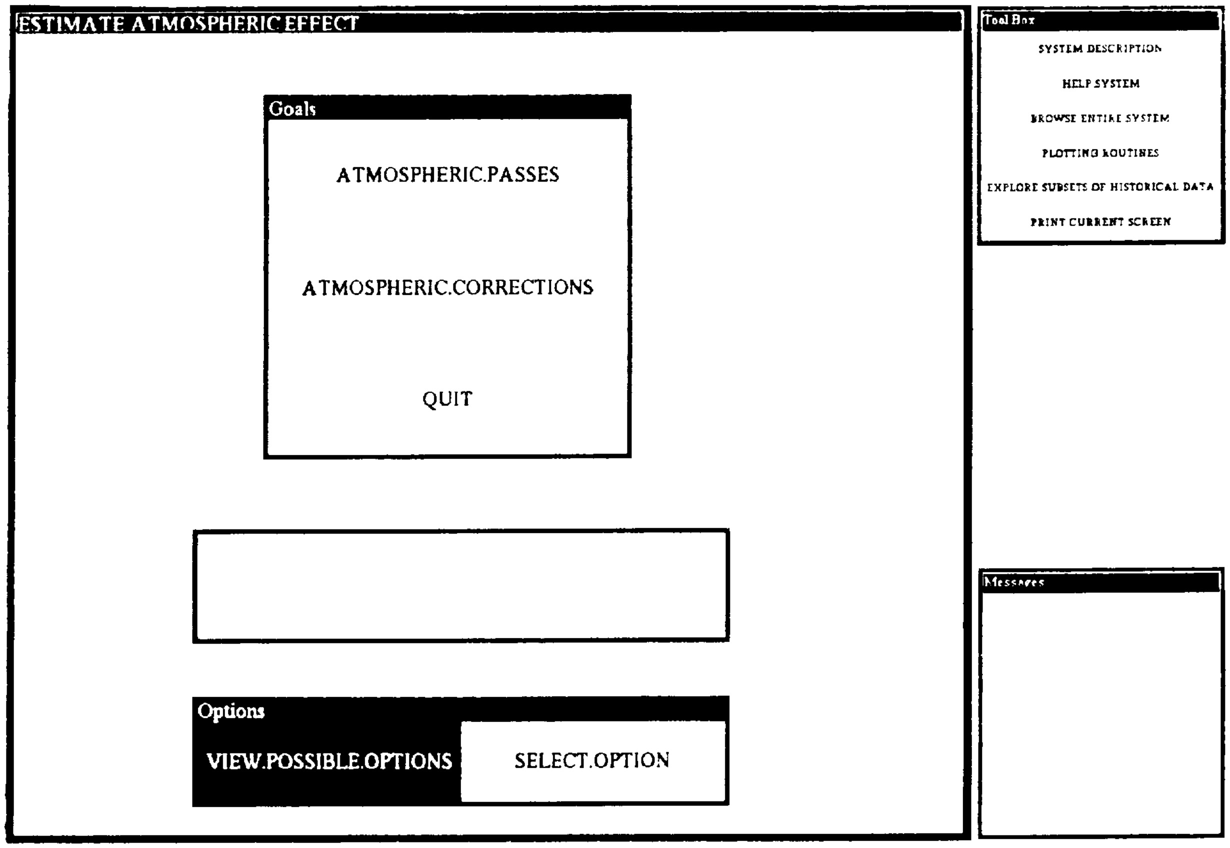


Figure 2-2
The Atmospheric Techniques Menu

When the user selects the subgoal ATMOSPHERIC.PASSES from the menu shown in Figure 2-2, the menu shown in Figure 2-3 is displayed. Selecting the subgoal ATMOSPHERIC.CORRECTIONS reveals the menu shown in Figure 2-4. The steps involved in the subgoals ATMOSPHERIC.PASSES and ATMOSPHERIC.CORRECTIONS are similar. The subgoal ATMOSPHERIC.PASSES will be described in detail in this section. Any variations for the subgoal ATMOSPHERIC.CORRECTIONS will be mentioned in the description.

The menu shown in Figure 2-3 enables the user to invoke the steps involved in processing reflectance data to estimate the reflectance values at different atmospheric heights. Before each step is carried out, a check is made to make sure that the necessary prerequisite steps have been carried out. For example, the results cannot be output before the techniques have been executed. If any prerequisite steps have not been carried out, a message is displayed and the user is prompted to complete the necessary prerequisite steps.

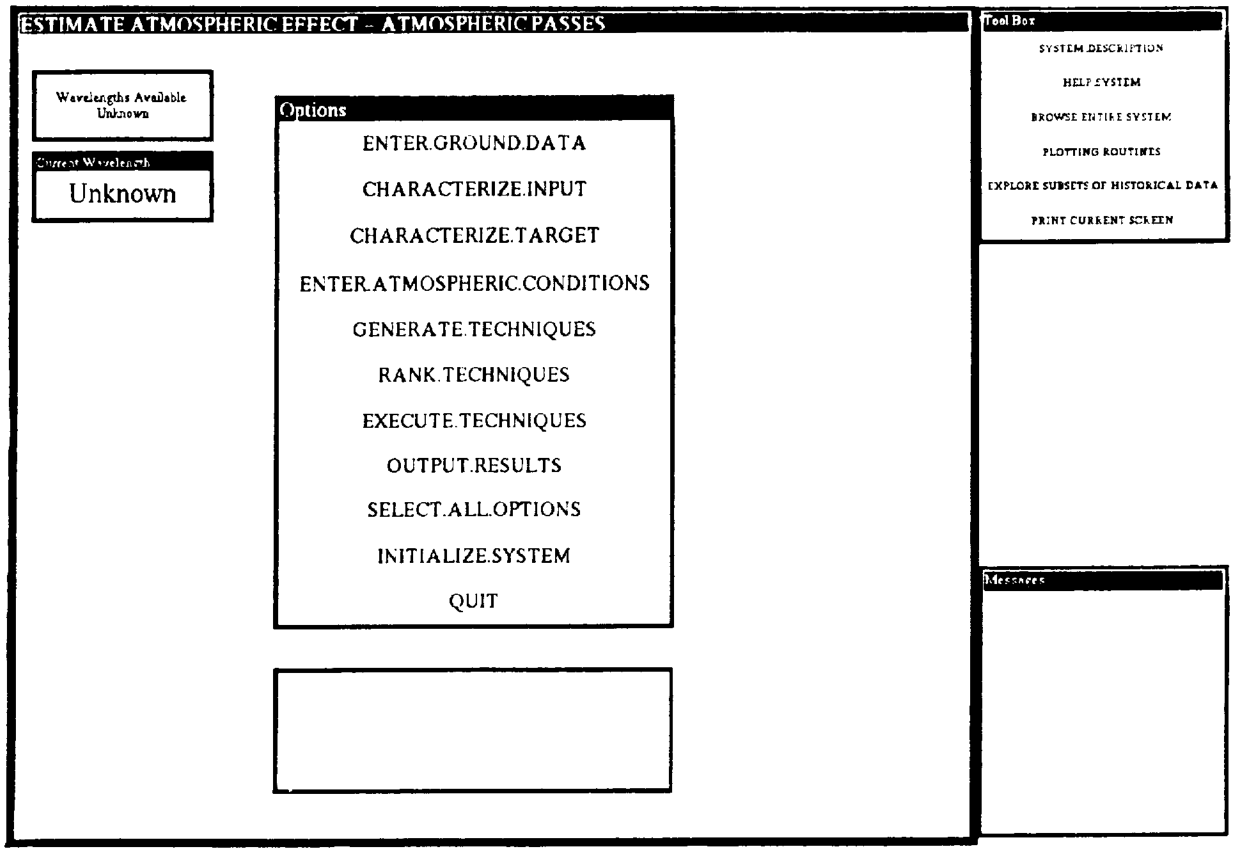


Figure 2-3
Menu for the Subgoal ATMOSPHERIC.PASSES

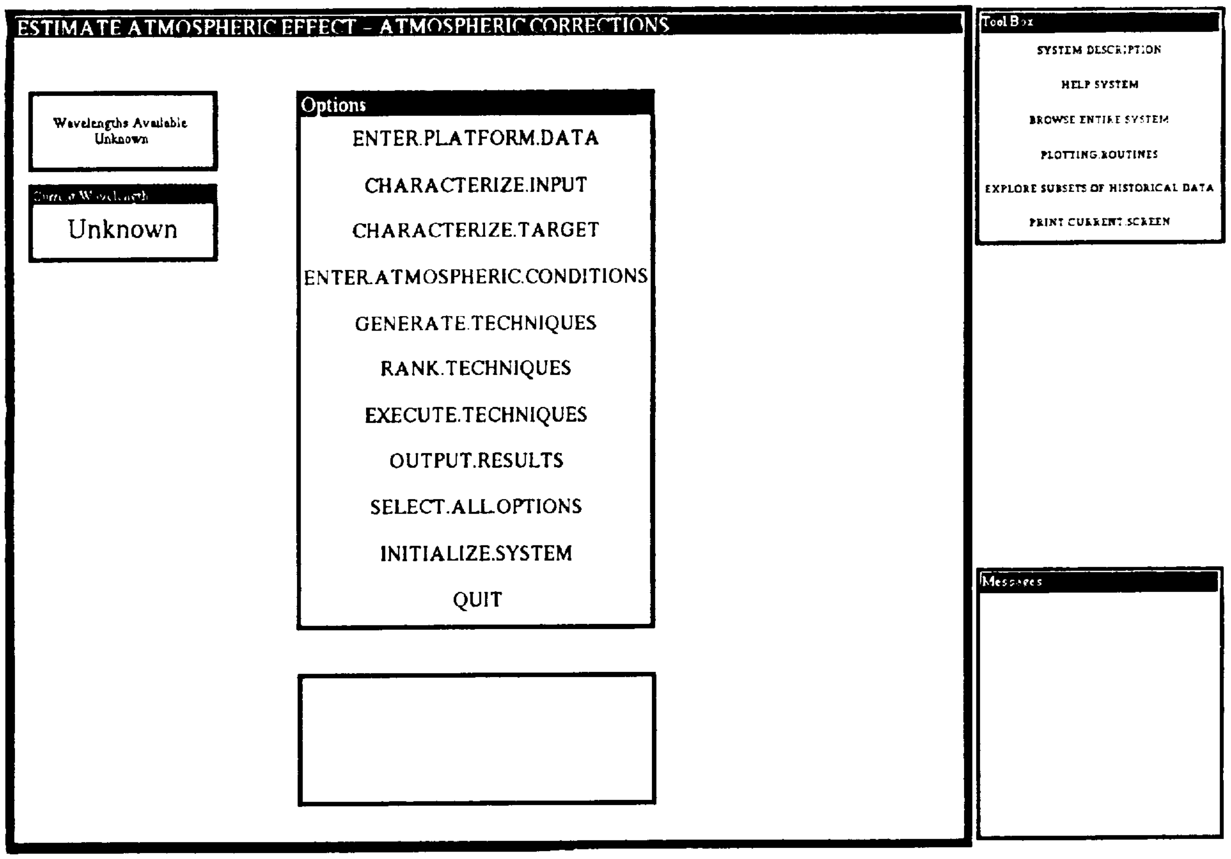
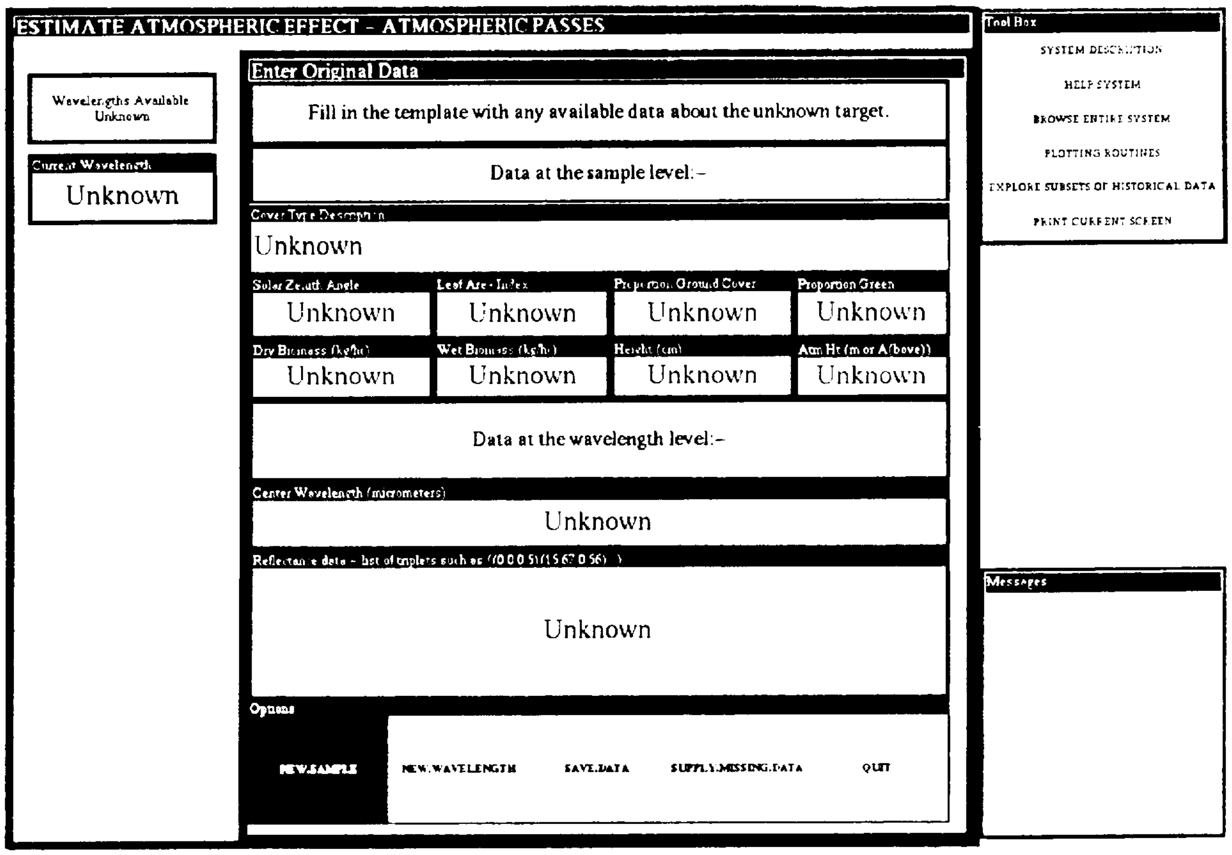


Figure 2-4
Menu for the Subgoal ATMOSPHERIC.CORRECTIONS

2.1 ENTER PLATFORM DATA

The code and interface that were originally developed for the step ENTER.DATA for the VEG subgoal SPECTRAL.HEMISPHERICAL.REFLECTANCE have been modified for re-use in this step. When the user selects the step ENTER.DATA, an interface opens. This interface allows the user to either enter a new original set of data for an unknown cover type or select one of a number of samples of cover type data stored in the VEG historical database. If the user chooses to enter original data, another interface opens as shown in Figure 2-5. This interface allows the user to enter data for the new sample. In addition to the data required for the subgoal SPECTRAL.HEMISPHERICAL.REFLECTANCE, the subgoal ATMOSPHERIC.PASSES requires the entry of the atmospheric height to which the reflectance data will be projected. A subwindow labeled "Atm Ht (m or (A)bove))" has been added to the screen. This subwindow enables the user to enter the number of meters to which the data should be projected or "A" if the data are to be projected to above the atmosphere. Each data value is checked as soon as it has been entered to make sure that it is of the correct type and is in the valid range for the data item it represents. The user can left click on the menu button "SAVE.DATA" at the bottom of the screen in Figure 2-5 to store the data. Before a set of cover type data is stored, the system checks that at a minimum the solar zenith angle, wavelength, reflectance data and atmospheric height have been entered. If any of these items is missing, the user is prompted to supply the missing items before the data are stored.


ESTIMATE ATMOSPHERIC EFFECT - ATMOSPHERIC PASSES

Wavelengths Available
Unknown

Current Wavelength
Unknown

Enter Original Data

Fill in the template with any available data about the unknown target.

Data at the sample level:-

Cover Type Description
Unknown

Solar Zenith Angle	Leaf Area Index	Proportion Ground Cover	Proportion Green
Unknown	Unknown	Unknown	Unknown

Dry Biomass (kg/ha)	Wet Biomass (kg/ha)	Height (cm)	Atm Ht (m or A(bove))
Unknown	Unknown	Unknown	Unknown

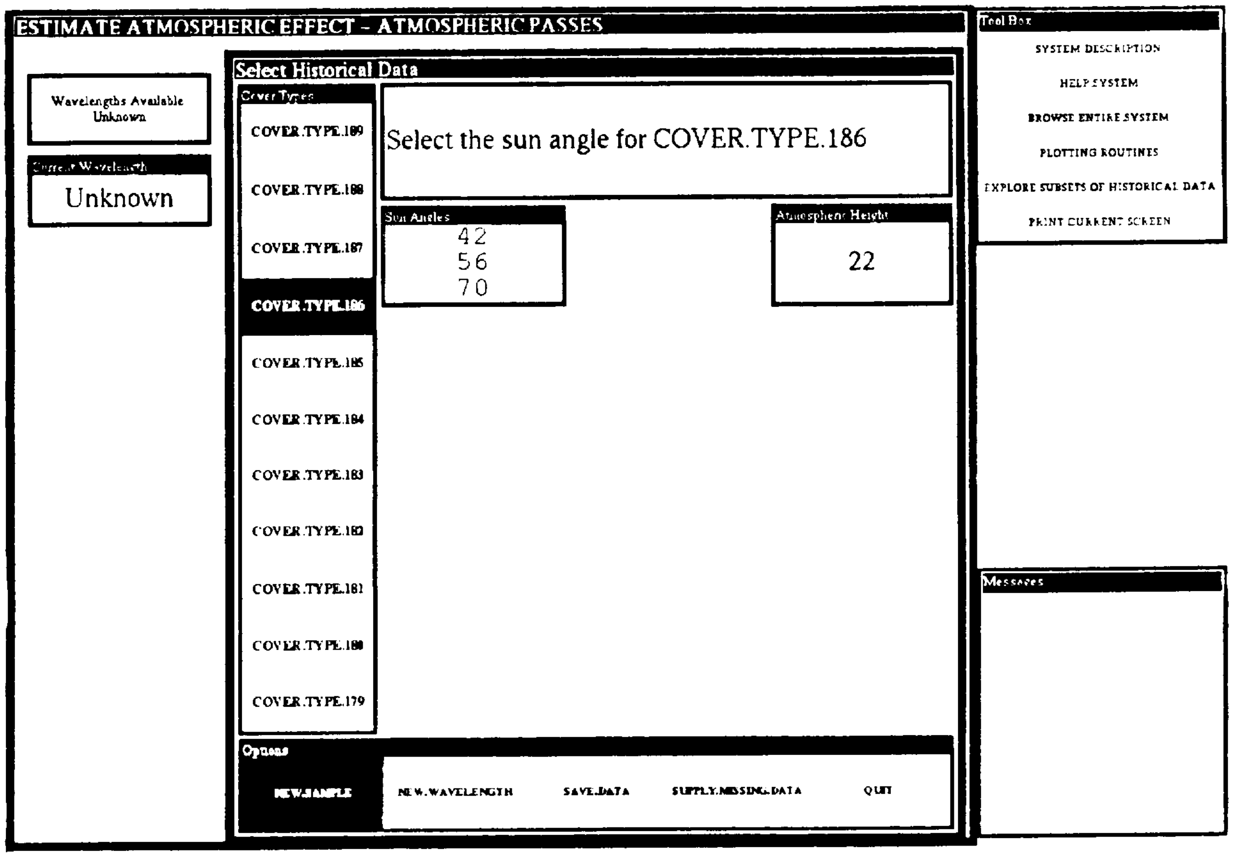
Data at the wavelength level:-

Center Wavelength (micrometers)
Unknown

Reflectance data - list of triplets such as ((0 0 0 5)(15 67 0 56))
Unknown

Figure 2-5
The Screen for Entering Original Platform Data

If the user elects to process a sample of cover type data from the historical cover type database, the screen shown in Figure 2-6 is opened. It is important that the user enter data into this screen in the correct order. When this screen is first opened, only the message and atmospheric height subwindows are opened. The other subwindows in this screen are automatically opened in turn, after the user has entered the necessary prerequisite data. If the user enters a data value and then subsequently changes it, the appropriate subwindows are closed to backtrack the data entry process to where the changed data value was first entered. The user is required to first enter the atmospheric height. Next the user must select the required cover type from the historical cover type database, followed by the sun angle. Figure 2-6 shows the "Select Historical Data" screen at this stage of data entry. Once the sun angle has been selected, the user is prompted to select the waveband of interest. The user is then prompted to enter the directional data. If invalid directional data is entered, the user is prompted to reenter the data in the correct format. Once valid directional data have been entered, VEG automatically interpolates and extrapolates the reflectance data for the selected historical database cover type to the specified view angles. The complete directional reflectance data, including the interpolated and extrapolated reflectance values, are displayed in the subwindow labeled "Directional Reflectance Data," as shown in Figure 2-7.



Wavelengths Available
Unknown

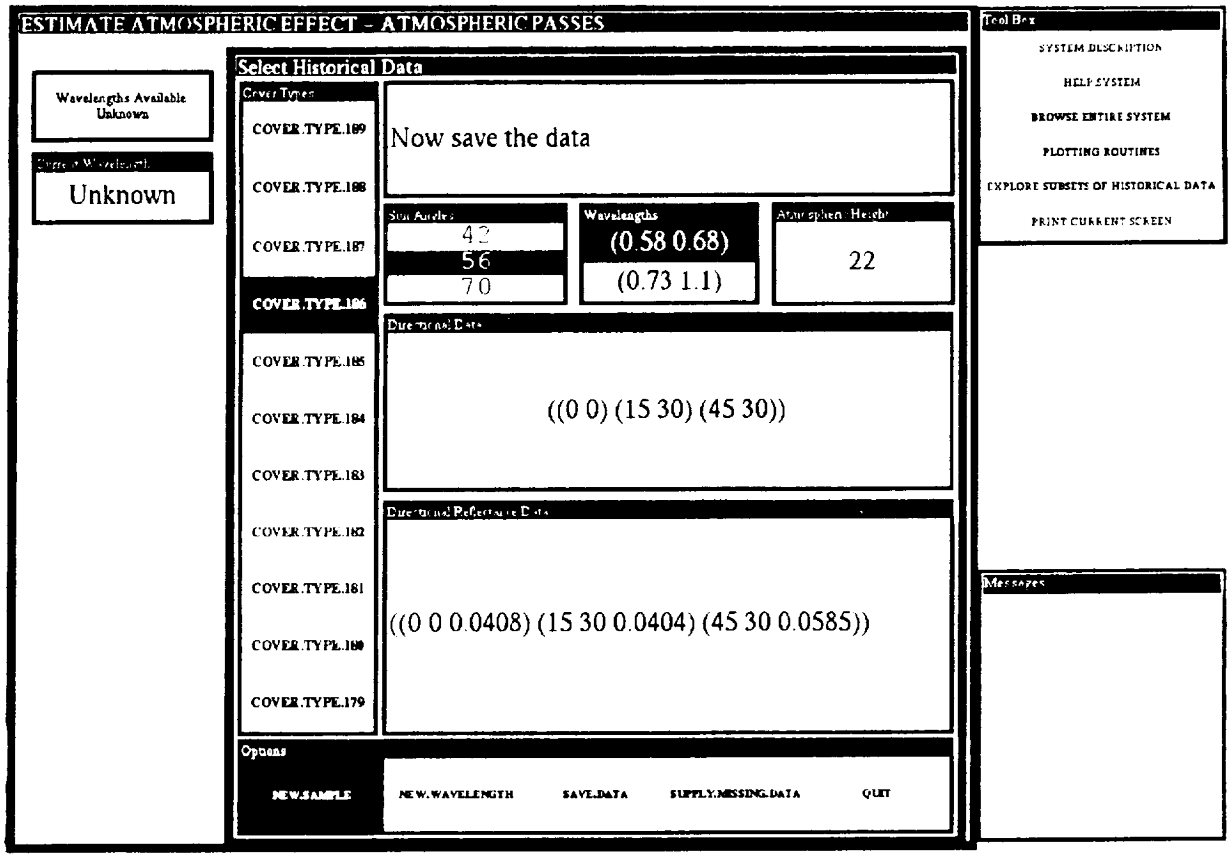
Current Wavelength
Unknown

Select the sun angle for COVER.TYPE.186

Sun Angles
42
56
70

Atmospheric Height
22

Figure 2-6
Selecting Historical Data



Unknown

Now save the data

42
56
70

(0.58 0.68)
(0.73 1.1)

22

((0 0) (15 30) (45 30))

((0 0 0.0408) (15 30 0.0404) (45 30 0.0585))

Figure 2-7
A Complete Set of Selected Historical Data

Once the data has been entered, they can be saved. If the user attempts to save an incomplete data set, the user is prompted to supply the missing data before saving them. The interface allows the user to select multiple wavebands of the same historical data sample. If the user selects "NEW.WAVELENGTH" from the "Options" at the bottom of the "Select Historical Data" screen, the "Directional Reflectance Data" subwindow is cleared and highlighting of the previously selected waveband is removed. When the user selects a different waveband, the historical cover type data for the new waveband is automatically interpolated and extrapolated to the required view angles. The set of directional reflectance data for the new waveband is automatically displayed in the "Directional Reflectance Data" subwindow. The user also has the option of entering a different set of directional view angles for the new waveband. Selecting "QUIT" returns the user to the screen shown in Figure 2-3.

When the VEG subgoal ATMOSPHERIC.CORRECTIONS is in use, platform data rather than ground data must be entered. The interface for entering platform data is the same as the interface shown in Figure 2-5, except that the atmospheric height subwindow is replaced by a subwindow for entering the platform height. All the cover type data sets in the current historical database were collected at ground level. Thus, the option of selecting data from the historical database for the ATMOSPHERIC.CORRECTIONS subgoal is not yet available.

2.2 CHARACTERIZE INPUT

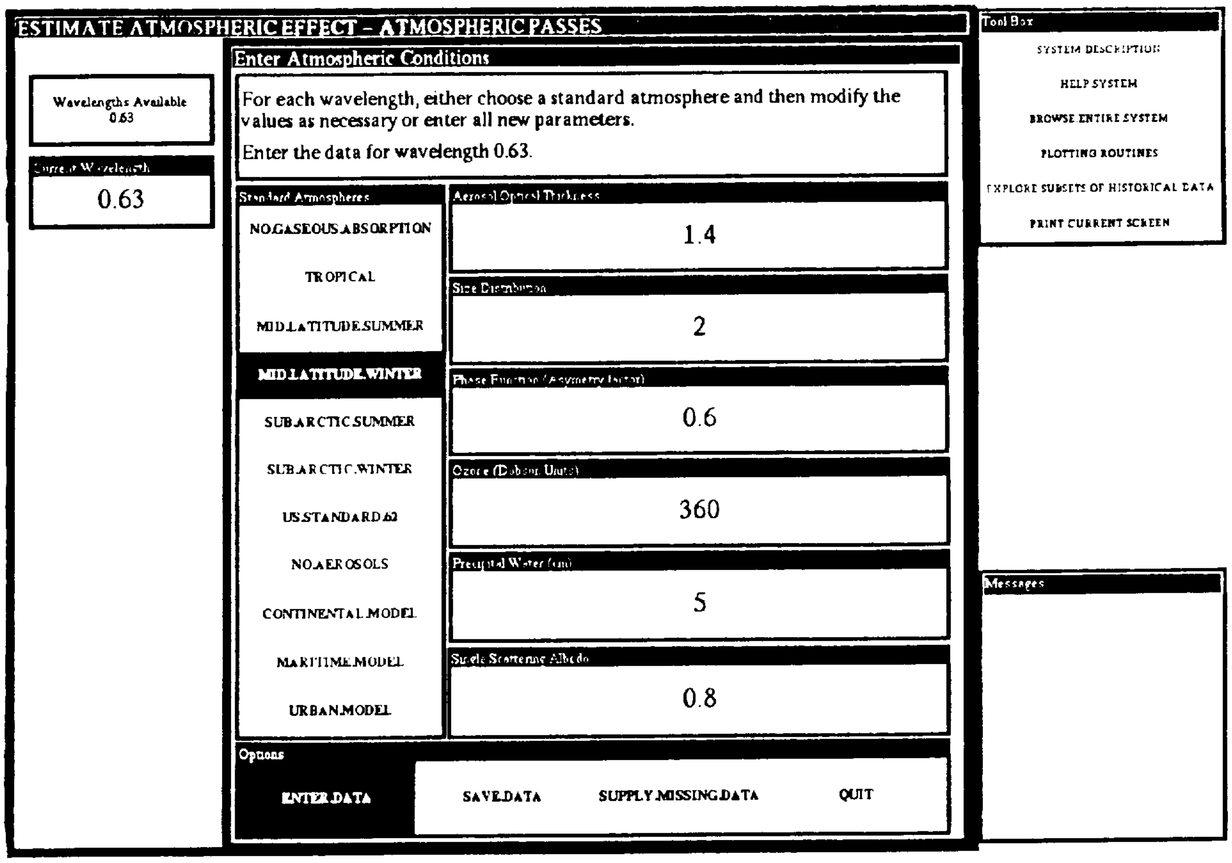
The unknown cover type data at each wavelength are characterized using code that was developed for the VEG subgoal SPECTRAL.HEMISPHERICAL.REFLECTANCE. Sets of view angles in the same azimuthal plane are identified as "strings." Strings are characterized as full-strings if they contain both forwardscatter and backscatter data and half-strings if they contain either backscatter or forwardscatter data.

2.3 CHARACTERIZE TARGET

If the sample data do not contain a value for ground cover or leaf area index, an estimation of these values is made. The code developed for the VEG subgoal SPECTRAL. HEMISPHERICAL.REFLECTANCE is re-used for this purpose.

2.4 ENTER ATMOSPHERIC CONDITIONS

When the user selects the option ENTER.ATMOSPHERIC.CONDITIONS from the menu shown in Figure 2-3, the screen shown in Figure 2-8 is opened. This screen allows the user to define the atmospheric conditions for the target data in each waveband. VEG automatically selects the first waveband and prompts the user to define the atmospheric conditions for that waveband. The user can select a standard atmosphere, such as "sub-arctic winter," by left clicking on the name of the standard atmosphere. The NASA GSFC technical representative was unable to provide the specifications of the standard atmospheres so dummy values were used for the development of this option. The dummy values will be replaced with the correct values when they become available. When a standard atmosphere has been selected, the user has the option to change the value of any of the atmospheric parameters as required. Alternatively, instead of selecting a standard atmosphere, the user can enter the value of each parameter independently. The type and range of each parameter value is checked after it is entered. If the user enters an invalid value, an error message is displayed, prompting the user to enter another value. Once a complete set of data has been entered, the data can be saved. If the user attempts to save an incomplete set of data he/she is prompted to supply the missing data before the data can be saved. After the data have been saved, if the target data contains more than one waveband, the prompt at the top of the screen is changed to include the next waveband. The values for the other parameters are not changed. The user has the option of changing the parameter values before saving the data for the next waveband. When atmospheric conditions have been saved for all the selected wavebands, the Enter Atmospheric Conditions Screen is automatically closed.


ESTIMATE ATMOSPHERIC EFFECT – ATMOSPHERIC PASSES

Enter Atmospheric Conditions

For each wavelength, either choose a standard atmosphere and then modify the values as necessary or enter all new parameters.

Enter the data for wavelength 0.63.

Wavelengths Available
0.63

Current Wavelength
0.63

Aerosol Optical Thickness
1.4

Size Distribution
2

Phase Function (Asymmetry factor)
0.6

Ozone (Dobson Units)
360

Precipital Water (cm)
5

Single Scattering Albedo
0.8

Figure 2-8
The Enter Atmospheric Conditions Screen

2.5 GENERATE TECHNIQUES

Techniques can be generated automatically by the system or selected by the user. The code for generating techniques for the VEG subgoal SPECTRAL.HEMISPHERICAL. REFLECTANCE was copied and modified for this step. A new screen was created to allow the user to select the atmospheric passes techniques, but many existing functions were used to operate this screen. The NASA GSFC technical representative advised that the atmospheric techniques were not yet available in the appropriate format for incorporation into VEG. Thus, dummy rules and technique functions were incorporated in VEG at this stage. The dummy rules and functions should be replaced with the actual atmospheric technique rules and functions when they become available.

If the user elects to have the system generate the techniques, the rules in the rulebase ATMOSPHERIC.PASSES.RULES are run. The rules operate on the unknown sample data at the wavelength level and determine the techniques that are suitable for estimating the reflectance data of a sample at a particular height. The names of the selected techniques are stored in the TECHNIQUES slot of the wavelength level unit.

If the user elects to choose the techniques manually, the Pick Techniques screen is opened. When the user left clicks on the name of a dummy technique, a brief description of the technique is displayed. A function is called to check whether the technique is suitable for the sample. If the technique is suitable for the sample, the message "Technique is suitable for this sample" is displayed, and the technique is selected. Otherwise, an error message is displayed in the same subwindow and the technique is not selected. When the user left clicks on PICK.SELECTED.TECHNIQUES at the bottom of the screen, the selected techniques are stored in the TECHNIQUES slot of the unknown cover type unit.

Dummy rules for selecting atmospheric correction techniques were constructed in the ATMOSPHERIC.CORRECTIONS.RULES rulebase. An additional screen that allows the user to select atmospheric corrections techniques was also constructed.

Minor changes were made to the Add Techniques interface and the code for adding techniques. These changes enabled the scientist to add new techniques for the subgoals Atmospheric Passes and Atmospheric Corrections without the assistance of the developer. The Add Techniques option is described in detail in Reference 3.

2.6 RANK TECHNIQUES

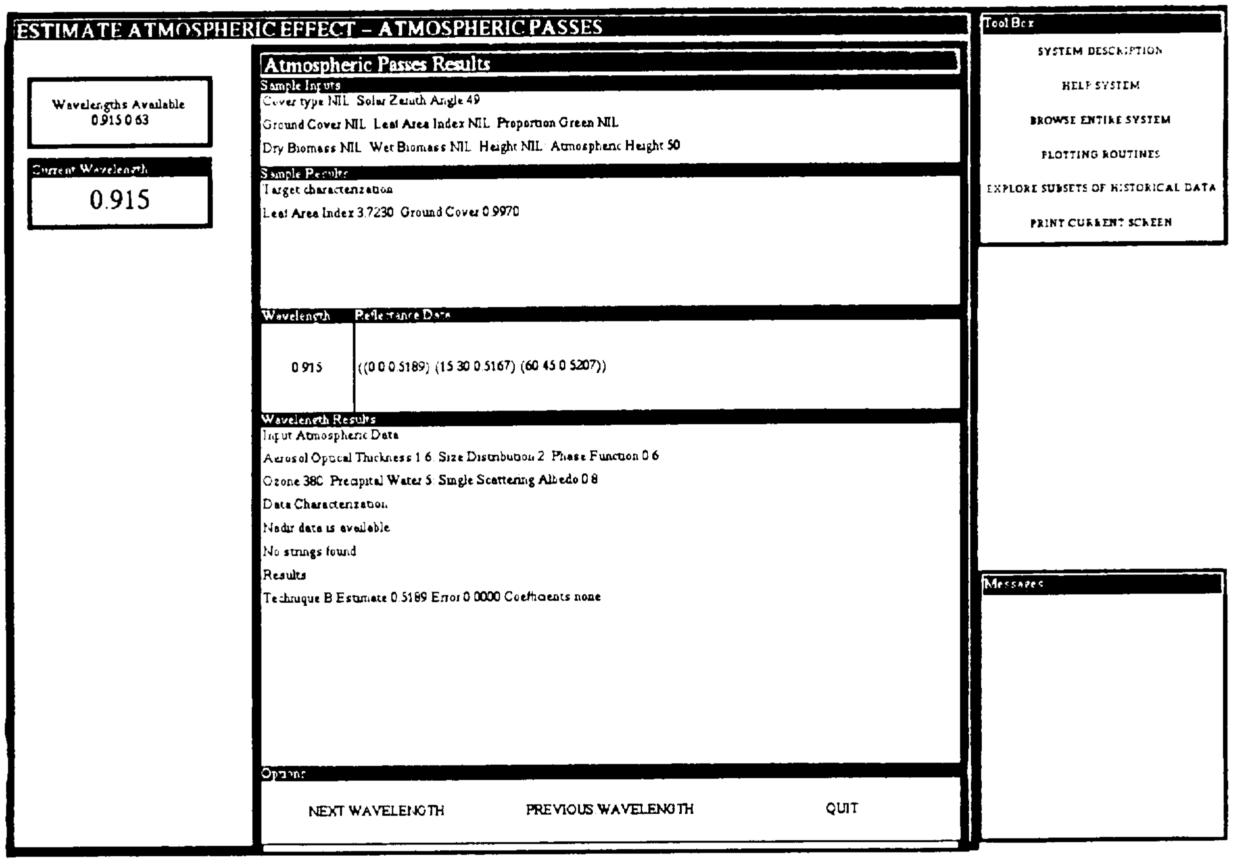
The code from the same step for the subgoal SPECTRAL.HEMISPHERICAL. REFLECTANCE was re-used for this step. The techniques are ranked according to a simple weighting scheme and the ranked techniques at each wavelength are displayed on the screen. The user can select the best one, two or three techniques for each wavelength, pick all the selected techniques, or repeat the previous step and generate the techniques again.

2.7 EXECUTE TECHNIQUES

The code providing the framework for this step from the VEG subgoal SPECTRAL.HEMISPHERICAL.REFLECTANCE was re-used for this step. Dummy functions for generating the coefficients and calculating the projected reflectance data for each technique were written. When the step EXECUTE.TECHNIQUES is selected, the techniques are applied to the data in the unknown cover type sample. If a technique requires coefficients, the user is asked whether all or half the restricted data set should be used for generating the coefficients and estimating the error. The appropriate coefficient methods are applied as necessary. A hierarchy of units is set up to hold the calculated projected reflectance data for each technique.

2.8 OUTPUT RESULTS

The results are displayed on the screen shown in Figure 2-9. This screen was originally constructed for the VEG subgoal SPECTRAL.HEMISPHERICAL.REFLECTANCE. The title has been changed to "Atmospheric Passes Results." The results are displayed one wavelength at a time. The atmospheric conditions specified for the wavelength are displayed in the subwindow labeled "Wavelength Results." For each technique, the name of the technique is displayed together with the results from applying that technique to the sample of cover type data. In Figure 2-9, the dummy technique B has been applied to the sample of cover type data. The results displayed for this technique are meaningless since technique B is a dummy technique that returns the reflectance value at the first view angle. When atmospheric techniques have been added to VEG, the correct results will be displayed on the "Output Results" screen. The user can view the results for different wavebands by left clicking on "NEXT.WAVELENGTH" or "PREVIOUS. WAVELENGTH."



Wavelengths Available
0.915 0.63

Current Wavelength
0.915

Atmospheric Passes Results

Sample Inputs
Cover type NIL Solar Zenith Angle 49
Ground Cover NIL Leaf Area Index NIL Proportion Green NIL
Dry Biomass NIL Wet Biomass NIL Height NIL Atmospheric Height 50

Sample Results
Target characterization
Leaf Area Index 3.7230 Ground Cover 0.9970

Wavelength	Reflectance Data
0.915	((0 0 0.5189) (15 30 0.5167) (60 45 0.5207))

Wavelength Results
Input Atmospheric Data
Aerosol Optical Thickness 1.6 Size Distribution 2 Phase Function 0.6
Ozone 380 Precipital Water 5 Single Scattering Albedo 0.8
Data Characterization
Nadir data is available
No strings found
Results
Technique B Estimate 0.5189 Error 0.0000 Coefficients none

Figure 2-9
The Output Screen for the Subgoal ATMOSPHERIC.PASSES

SECTION 3.0

THE SUBGOAL CATEGORY ATMOSPHERIC TECHNIQUES IN THE VEG AUTOMATIC MODE

The menus for the VEG Automatic Mode were modified to accommodate the restructuring of VEG subgoals into the four categories described in Section 2.0. When the user selects the Automatic Mode from the Processing Mode menu, the screen shown in Figure 3-1 is opened. This screen enables the user to select the category of subgoal for automatic processing.

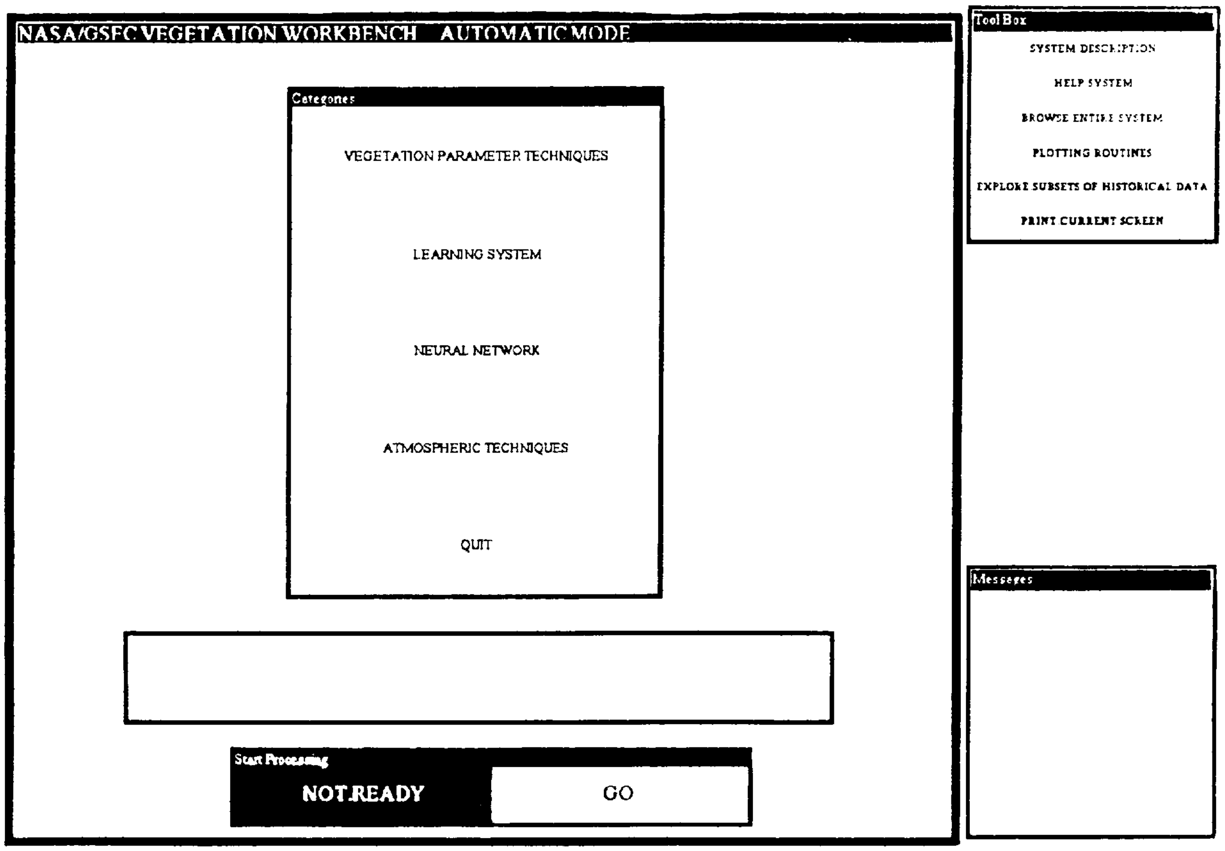
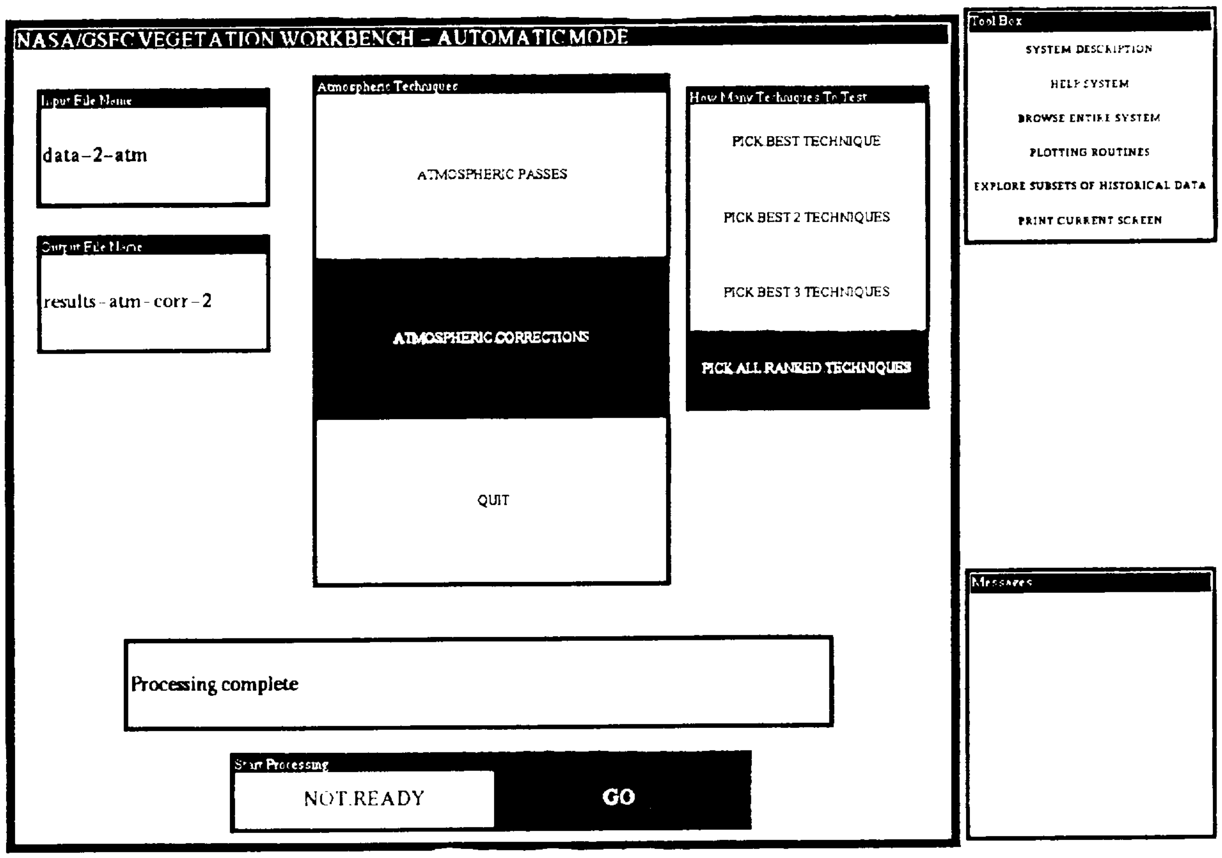


Figure 3-1
The VEG Automatic Mode Categories Menu

When the user selects the ATMOSPHERIC.TECHNIQUES option from the screen shown in Figure 3-1, the "Categories" subwindow is replaced by the "Atmospheric Techniques" subwindow. Selecting the option ATMOSPHERIC.PASSES or ATMOSPHERIC. CORRECTIONS causes additional subwindows to be opened, as shown in Figure 3-2. These subwindows enable the user to name the input and output files, select the output file format and specify how many techniques should be applied to each unknown cover type data sample.



Input File Name: data-2-atm

Output File Name: results-atm-corr-2

Processing complete

Figure 3-2
Running the Subgoal ATMOSPHERIC.CORRECTIONS in the VEG Automatic Mode

The input file for an atmospheric technique must contain atmospheric conditions data and the data platform elevation or atmospheric height as well as the cover type data that are required for subgoals in the other VEG categories. A standard file format for input files to atmospheric techniques has been defined. The file format and an example of typical values are shown in Table 3-3. Global variables *STANDARD-ATM-PASS-SAMPLE-FORMAT*, *STANDARD-ATM-CORR-SAMPLE-FORMAT* and *STANDARD-ATM-WAVELENGTH-FORMAT* have been created in the file "veg-methods1.lisp." This file contains the methods used for inputting data from a file into VEG. The new global variables hold the list of fields in the standard format for an atmospheric techniques input file. The field names correspond to the names of slots in which the data should be stored. When the input file is named by the user, the values from the appropriate field list global variables are put in the slots AUTO.INPUT.SAMPLE.FORMAT and AUTO.INPUT.WAVELENGTH.FORMAT of the unit AUTOMATIC.PROCESS. When the file is read, these slots are referenced to determine the file format.

Table 3-1
Input File Format for Atmospheric Techniques and an Example of Typical Values

FIELD NAMES	TYPICAL VALUES
COVER.TYPE.DESCRIPTION	"Dense Vegetation Canopy"
SOLAR.ZENITH LEAF.AREA.INDEX	45 3.5
GROUND.COVER PROPORTION.GREEN	0.7 0.3
DRY.BIOMASS WET.BIOMASS HEIGHT	0.2 0.5 1000
DATA.PLATFORM.ELEVATION	30
NUMBER.WAVELENGTHS	2
WAVELENGTH	0.68
NUMBER.VIEW.ANGLES	1
REFLECTANCE.DATA	((0 0 0.043))
AEROSOL.OPTICAL.THICKNESS SIZE.DISTRIBUTION	1.6 2
PHASE.FUNCTION OZONE	0.6 360
PRECIPITAL.WATER SINGLE.SCATTERING.ALBEDO	5 0.8
WAVELENGTH	0.68
NUMBER.VIEW.ANGLES	1
REFLECTANCE.DATA	((0 0 0.043))
AEROSOL.OPTICAL.THICKNESS SIZE.DISTRIBUTION	1.8 2
PHASE.FUNCTION OZONE	0.6 400
PRECIPITAL.WATER SINGLE.SCATTERING.ALBEDO	5 0.8

When the user left clicks on "GO," the unknown cover type data are read from the file using the correct format. The input data and the target are then characterized as in the Research Mode. The rules are run, and the best techniques for the sample are selected. The selected techniques are then executed and the results are written to the named file using the specified format. The code in the file "veg-methods.lisp" that was originally written for the automatic processing of data for the vegetation parameter techniques was modified for the processing of data for atmospheric techniques.

SECTION 4.0

TESTING AND RESULTS

The following capabilities of the VEG Atmospheric Techniques subgoal category were tested:

- Test 1 - Navigate through the category and subgoal menus in the Research Mode.
- Test 2 - Atmospheric Passes Subgoal using data entered by the user
- Test 3 - Atmospheric Passes Subgoal using historical data
- Test 4 - Atmospheric Corrections Subgoal
- Test 5 - Automatic Mode, Atmospheric Passes Subgoal
- Test 6 - Automatic Mode, Atmospheric Corrections Subgoal
- Test 7 - Add Techniques for Atmospheric Passes and Atmospheric Corrections Subgoals

All the tests were successful, showing that the system was working correctly. The tests are described in detail in this section.

4.1 TEST 1

This test was designed to test the new sequence of menus in the VEG Research Mode that was implemented as a result of the subgoals being divided into categories. After left clicking on "RUN.VEG," the user selected the Research Mode. Because no historical data were loaded, the Change Historical Database screen automatically opened. The user indicated that both the Kimes and the Deering databases should be added. After quitting the Change Historical Database screen, the user again selected the Research Mode from the Processing Mode screen. The categories screen was then opened. The user selected the VEGETATION.PARAMETER. TECHNIQUES category. The Vegetation Parameters Techniques Goals screen was opened. The user then selected and successfully ran the Estimate Spectral Hemispherical Reflectance option before navigating back to the Categories menu. Subsequently, the user successfully opened and quit the Learning System and Atmospheric Techniques goal screens. When the user selected the NEURAL.NETWORK option, a message indicating that the option was not available was displayed. This message was removed when another option was selected. The user then navigated out of the VEG system back to the KEE interface. This test showed that the new menus were operating correctly.

4.2 TEST 2

Test 2 was designed to test the Atmospheric Passes Subgoal using original data entered by the user. The user navigated to the Atmospheric Passes main menu. The data were processed by carrying out the steps in the ATMOSPHERIC.PASSES menu, as shown in Figure 2-3.

The user selected the ENTER.GROUND.DATA option, and elected to enter original data. The user entered various invalid values for the atmospheric height and the message "Atmospheric Height out of range error" was displayed in the "Messages" box. The user attempted to save an incomplete data set. VEG prompted the user to supply the missing data before the data set could be saved. Valid data for the solar zenith angle, atmospheric height, wavelength and reflectance data were then entered and the data were saved. Next, the input data and target were characterized.

The interface for entering atmospheric conditions was thoroughly tested. The user entered invalid values for each parameter. In every case, an error message was displayed in the "Messages" box and the user was prompted to reenter the data. When attempting to save an incomplete set of atmospheric conditions data, the user was prompted to supply the missing data before the data could be saved.

The user elected to select the techniques manually. The Pick Techniques screen was opened. When the user left clicked on a technique to select it, a description of the technique was displayed. If the technique was suitable for the sample, the message "Technique is suitable for this sample" was displayed and the technique was selected. Otherwise, an error message was displayed and the technique was not selected. The test showed that the Pick Techniques screen was operating correctly.

The techniques were ranked and all the ranked techniques were selected. The techniques were then executed. The results were displayed on the screen. The atmospheric conditions were included in the results displayed in the "Wavelength Results" window. The results of applying the dummy techniques to the sample were also displayed.

Test 2 confirmed that the subgoal "Atmospheric Passes" was operating correctly when original data were entered and the techniques were generated manually.

4.3 TEST 3

In Test 3, the Atmospheric Passes Subgoal was tested using cover type data from the historical cover type database. The user selected the step ENTER.GROUND.DATA and the option SELECT.HISTORICAL.DATA. The Select Historical Data screen was opened. At this stage, only the "Message," "Option," and "Atmospheric Height" subwindows were opened. The user entered the value "300" into the "Atmospheric Height" subwindow. The "Cover Types" subwindow then opened. The user selected COVER.TYPE.7. The user then selected sun angle 59 and waveband (0.58 0.68) in successive subwindows. The user then changed the cover type selection to COVER.TYPE.11. The "Wavelengths" and "Directional Reflectance" subwindows were automatically closed and the sun angle deselected. This part of the test showed that the data entry process was correctly backtracked when previously entered data were changed. The user reselected the previous values for cover type, sun angle and waveband. The user entered the directional view angles ((0 0)(15 30)(30 45)). Directional reflectance data for these view angles were displayed in the "Directional Reflectance Data" subwindow. The user saved the data by left clicking on the "SAVE.DATA" Option at the bottom of the screen. The user then selected the waveband (0.73 1.1) with the same view angles and saved the data. This part of the test showed that the Select Historical Data Screen was operating correctly.

The data and target were then characterized. The user selected the NO.AEROSOLS standard atmosphere in both wavebands for the data. The techniques were generated automatically by running the rules. The techniques were then ranked and the best technique for each wavelength was selected. After the techniques had been executed, the results were displayed on the screen. Since the techniques were dummy, the results were not meaningful. This test showed that the select historical data step and the technique generation rules of the Atmospheric Passes Subgoal were operating correctly.

4.4 TEST 4

This test was designed to test the steps in the subgoal Atmospheric Corrections that were different from the steps in the subgoal Atmospheric Passes. The subgoal ATMOSPHERIC.CORRECTIONS was selected from the Atmospheric Techniques menu. The user selected the step ENTER.PLATFORM.DATA and then the option SELECT.HISTORICAL.DATA. The message "This option is not yet available" was displayed. The user then selected ENTER.ORIGINAL.DATA. The Enter Original Data screen opened. As expected, the screen included a subwindow for entering the Data Platform Elevation. The user entered and saved a complete set of data.

The steps Characterize Input, Characterize Target, and Enter Atmospheric Conditions were then executed. Initially, the user chose to pick the techniques manually. The Pick Techniques screen worked correctly. The user then activated the rules to generate the techniques automatically. The correct techniques were selected. The techniques were then ranked and the best technique for each wavelength was selected. The techniques were executed and the results displayed. The output screen included the Data Platform Elevation in the data at the sample level. Since the techniques were dummies, the actual values of the results were not relevant.

Test 4 confirmed that all the options in the subgoal Atmospheric Corrections were operating correctly.

4.5 TEST 5

This test was designed to test the new sequence of menus and the operation of the subgoal Atmospheric Passes in the VEG Automatic Mode. The user selected the Automatic Mode from the Processing Mode menu. The Automatic Mode screen with the Categories subwindow was opened. Tests confirmed that subgoals in the Vegetation Parameters and Learning System categories could be successfully operated via the new menu structure. When the user selected the category NEURAL.NETWORK, the message "This option is not yet available" was displayed.

The user selected the ATMOSPHERIC.TECHNIQUES Subgoal category and the ATMOSPHERIC.PASSES Subgoal. The "Atmospheric Techniques," "Input File Name," "Output File Name," and "Number of Techniques" subwindows opened. The user entered the file name DATA-2-ATM as the input file and RESULTS-ATM-PASS as the output file. Standard template number 1 was selected as the output file format. The option to test all techniques was selected. When the user left clicked on "GO," the data were processed. Inspection of the output file indicated that the data had been processed correctly.

This test showed that the new sequence of menus and the subgoal Atmospheric Passes were operating correctly in the VEG Automatic Mode.

4.6 TEST 6

In this test, Test 5 was repeated using the subgoal Atmospheric Corrections. Inspection of the output file confirmed that this subgoal was operating correctly.

4.7 TEST 7

The Add Techniques Option allows the scientist to define new techniques and add them to VEG without the assistance of the developer. Test 7 was designed to test the operation of the Add Techniques Option when the user attempted to add new techniques for the subgoals Atmospheric Passes and Atmospheric Corrections.

Using the DEFINE.NEW.TECHNIQUE option from the Add Techniques menu, the user defined a new technique for each of the Atmospheric Passes and Atmospheric Corrections Subgoals. The user then add the new techniques to VEG using the ADD.PREVIOUSLY. DEFINED.TECHNIQUES option form the Add Techniques menu. Subsequently, the user ran both the Atmospheric Passes and the Atmospheric Corrections Subgoals. It was confirmed that the new techniques had been correctly incorporated in VEG.

SECTION 5.0

CONCLUSIONS

The report described the implementation of the VEG subgoal category ATMOSPHERIC.TECHNIQUES in both the Research and Automatic Modes of VEG. It then described the testing of the new components of VEG to demonstrate their basic functionality.

The addition of Atmospheric Techniques to VEG illustrated that additional functionality can easily be added to the system without any major problems being encountered. The new subgoals Atmospheric Passes and Atmospheric Corrections were integrated into the overall VEG interface so that they worked smoothly as part of the overall system. The additional functionality provided by these new subgoals allows the scientist to take data measured at ground level and predict what the reflectance values would be if the data were measured at a different atmospheric height. It also allows atmospheric corrections to be made to data collected from an aircraft or by a satellite to determine what the equivalent reflectance values would be if the data were measured at ground level.

REFERENCES

1. Kimes, D. S., Harrison, P. R. and Ratcliffe, P. A. 1991. A Knowledge-Based Expert System for Inferring Vegetation Characteristics. International Journal of Remote Sensing: Vol 12, 10, pp. 1987-2020.

2. Kimes, D. S., Harrison, P. A. and Harrison, P. R. 1992. New Developments of a Knowledge Based System (VEG) for Inferring Vegetation Characteristics. International Geoscience and Remote Sensing Symposium, Houston, Texas, May 1992.

3. JJM Systems Inc. April 1993. An Expert System Shell For Inferring Vegetation Characteristics - Interface for the Addition of Techniques (Task H). Arlington, VA. C931021-U-2R07.

APPENDIX A

LISP CODE FOR THE VEG SUBGOAL CATEGORY ATMOSPHERIC TECHNIQUES

```
;;; veg-methods6.lisp
;;;
;;; Code for VEG Atmospheric Techniques
;;;
;;; Created April 27, 1993
;;; Last Modified October 18, 1993

(in-package 'kee)

(defun open-atmospheric-screen ()
"Opens the screen that allows the user to select the atmospheric technique
goal."
  (remove.all.values 'atmospheric 'goals)
  (put.value 'atmospheric 'message "")
  (put.value 'atmospheric 'options 'view.possible.options)
  (unitmsg 'viewport-atmospheric.1 'open-panel!))

;;;-----------------------------------------------------------------------
;;; Methods for Atmospheric Passes
;;;-----------------------------------------------------------------------

(defun atm.pass.p ()
"Returns t if the current goal is atmospheric passes and nil otherwise."
  (eq (get.value 'atmospheric 'goals) 'atmospheric.passes))

(defun initialize-atmospheric-screen-research ()
"Initializes the main atmospheric category screen in the VEG research mode."
 (remove.all.values 'atmospheric 'goals)
 (put.value 'atmospheric 'error.message ""))

(defun open-atmospheric-passes-interface ()
"Opens the interface for the atmospheric passes main menu."
 (remove.all.values 'atmospheric.passes 'ap.menu)
 (unitmsg 'viewport-atmospheric.passes.1 'open-panel!))

;;;-----------------------------------------------------------------------
;;; Methods for Atmospheric Correction
;;;-----------------------------------------------------------------------

(defun atm.corr.p ()
"Returns t if the current goal is atmospheric corrections and nil otherwise."
  (eq (get.value 'atmospheric 'goals) 'atmospheric.corrections))

(defun open-atmospheric-corrections-interface ()
"Opens the interface for the atmospheric corrections main menu."
 (remove.all.values 'atmospheric.corrections 'ac.menu)
 (unitmsg 'viewport-atmospheric.corrections.1 'open-panel!))
```

```
;;;------------------------------------------------------------------------
;;; Methods for Entering Original Ground Data
;;;------------------------------------------------------------------------

(defun open-enter-ground-data-interface ()
"Opens the interface that allows the user to select between entering original
ground data and selecting historical data."
 (put.value 'atmospheric 'error.message
    "Reinitializing the system")
 (unitmsg 'initialize.system 'initialize.system)
 (put.value 'atmospheric 'error.message "")
 (remove.all.values 'atmospheric.passes 'options)
 (remove.all.values 'estimate.hemispherical.reflectance 'current.sample)
 (remove.all.values 'estimate.hemispherical.reflectance
          'current.sample.wavelengths)
 (unitmsg 'viewport-atmospheric.passes.2 'open-panel!))

(defun enter-original-ground-data ()
"Opens the enter data interface, including the atmospheric height subwindow."
 (enter-original-data)
 (unitmsg 'windowpane-atmospheric.height-of-atmospheric.passes.1 'open!))

;;;------------------------------------------------------------------------
;;; Methods for Entering Original Platform Data
;;;------------------------------------------------------------------------

(defun open-enter-platform-data-interface ()
"Opens the interface that allows the user to select between entering original
platform data and selecting historical data."
 (put.value 'atmospheric 'error.message
    "Reinitializing the system")
 (unitmsg 'initialize.system 'initialize.system)
 (put.value 'atmospheric 'error.message "")
 (remove.all.values 'atmospheric.corrections 'options)
 (remove.all.values 'estimate.hemispherical.reflectance 'current.sample)
 (remove.all.values 'estimate.hemispherical.reflectance
          'current.sample.wavelengths)
 (unitmsg 'viewport-atmospheric.corrections.2 'open-panel!))

(defun enter-original-platform-data ()
"Opens the enter data interface, including the platform elevation subwindow."
 (enter-original-data)
 (unitmsg 'windowpane-data.platform.elevation-of-atmospheric.corrections.2
            'open!))
```

```
;;;-----------------------------------------------------------------------
;;; Methods for Selecting Historical Ground Data
;;;-----------------------------------------------------------------------

(defun update-cover-type-window ()
"Update the cover types subwindow of the select historical ground data screen."
  (remove.all.values 'atmospheric.passes 'historical.cover.types)
  (put.facet.value 'atmospheric.passes 'historical.cover.types 'valueclass
     (cons 'one.of (get.value 'historical.cover.types 'current.cover.types)))
  (slot-image-toggle-enable
   (unit
    'windowpane-historical.cover.types-of-atmospheric.passes.2))
  (slot-image-toggle-enable
   (unit
    'windowpane-historical.cover.types-of-atmospheric.passes.2)))

(defun select-historical-ground-data ()
"Opens the select historical ground data screen."
  (remove.all.values 'atmospheric.passes 'directional.data)
  (put.value 'atmospheric.passes 'historical.data.options 'new.sample)
  (put.value 'atmospheric.passes 'error.message
     "Enter the atmospheric height")
  (unitmsg 'viewport-atmospheric.passes.3 `open-panel!))

(defun valid-directional-data (data)
"Returns t if the directional data is valid and nil otherwise."
   (and (consp data)
        (dolist (point data t)
              (unless (and (listp point)
                           (= (length point) 2))
                    (return-from valid-directional-data nil))
              (let ((z (zenith point))
                    (a (azimuth-360 point)))
                (unless (and (numberp z)(>= z 0)(<= z 90)
                             (numberp a)(>= a 0)(< a 360))
                  (return-from valid-directional-data nil))))))

(defun get-appropriate-cover-type ()
"Returns the name of the cover type unit at the wavelength level that has been
selected. i.e. The descendant of the selected cover type with the selected sun
angle and waveband."
  (let* ((this-sun (get.value 'atmospheric.passes 'sun.angles))
         (this-waves (get.value 'atmospheric.passes 'wavelengths))
         (this-wave-max (second this-waves))
         (this-wave-min (first this-waves)))
    (dolist (sun (unit.children
                   (get.value 'atmospheric.passes 'historical.cover.types)
                   'subclass))
      (when (= (get.value sun 'solar.zenith.angle) this-sun)
          (dolist (wave (unit.children sun 'member))
            (when (and (= (get.value wave 'wavelength.max) this-wave-max)
                       (= (get.value wave 'wavelength.min) this-wave-min))
              (return-from get-appropriate-cover-type wave)))))))
```

```
(defun find-matching-reflectance-values (view-angles)
"Interpolates and extrapolates the cover type data to match the entered
directional view angles and returns a list of points, each having zenith,
azimuth and reflectance values."
  (put.value 'atmospheric.passes 'directional.reflectance
     (match-unaltered-target-data view-angles
                                   (get-appropriate-cover-type))))

(defun reset-hct-sample-data ()
"Initializes all the select historical data screen."
   (remove.all.values 'atmospheric.passes 'cover.types)
   (remove.all.values 'atmospheric.passes 'sun.angles)
   (remove.all.values 'atmospheric.passes 'wavelengths)
   (remove.all.values 'atmospheric.passes 'directional.reflectance)
   (remove.all.values 'atmospheric.passes 'atmospheric.height))

(defun reset-hct-wavelength-data ()
"Initializes the wavelength and directional reflectance data in preparation for
 selection of a different wavelength in the select historical data screen."
   (remove.all.values 'atmospheric.passes 'wavelengths)
   (remove.all.values 'atmospheric.passes 'directional.reflectance)
   (put.value '1.enter.data 'successful.save t))

(defun insufficient-data-hct-sample ()
"Displays an error message because the data at the sample level is incomplete
and hence cannot be saved."
  (my-documentation-print
   "DATA NOT SAVED - Insufficient data - minimum data required is solar zenith, wavelength,
directional data and atmospheric height")
   (put.value '1.enter.data 'successful.save nil)
   (put.value '1.enter.data 'sample.flag 'sample))

(defun insufficient-data-hct-wavelength ()
"Displays an error message because the data at the wavelength level is
incomplete and hence cannot be saved."
  (my-documentation-print "DATA NOT SAVED - Insufficient data - minimum data required is
wavelength and directional data")
   (put.value '1.enter.data 'successful.save nil)
   (put.value '1.enter.data 'sample.flag 'wave))

(defun save-hct-sample-data ()
"If sufficient data is present, calls a function to save the data at the sample
level."
   (let ((solar-zenith (get.value 'atmospheric.passes 'sun.angles))
        (wavelength (get.value 'atmospheric.passes 'wavelengths))
        (reflectance-data (get.value 'atmospheric.passes
                                     'directional.reflectance))
           (atmospheric-height (get.value 'atmospheric.passes
                                          'atmospheric.height)))
      (if (and solar-zenith wavelength reflectance-data atmospheric-height)
         (save-hct-sample-data-aux solar-zenith wavelength reflectance-data
                                   atmospheric-height)
         (insufficient-data-hct-sample))))
```

```
(defun save-hct-sample-data-aux (solar-zenith wavelength reflectance-data
                                              atmospheric-height)
"Saves the data at the sample level."
  (let ((new-sample
          (create.unit (gentemp "SAMPLE-UNKNOWN-TARGET")
                  'veg 'target.data nil)))
    (put.value new-sample 'solar.zenith solar-zenith)
    (put.value new-sample 'atmospheric.height atmospheric-height)
    (put.value 'estimate.hemispherical.reflectance 'current.sample
         new-sample)
    (save-hct-wavelength-data-aux wavelength reflectance-data)))

(defun save-hct-wavelength-data ()
"If sufficient data is present, calls a function to save the data at the
wavelength level."
  (let ((wavelengths (get.value 'atmospheric.passes 'wavelengths))
       (reflectance-data (get.value 'atmospheric.passes
                                    'directional.reflectance)))
     (if (and wavelengths reflectance-data)
        (save-hct-wavelength-data-aux wavelengths reflectance-data)
        (insufficient-data-hct-wavelength))))

(defun save-hct-wavelength-data-aux (wavelengths reflectance-data)
"Saves the data at the wavelength level."
  (let* ((parent-sample
           (get.value 'estimate.hemispherical.reflectance 'current.sample))
        (new-wavelength (create.unit (gentemp "W") 'veg nil parent-sample)))
       (put.value new-wavelength 'wavelength
            (/ (+ (first wavelengths)(second wavelengths)) 2))
       (put.value new-wavelength 'reflectance.data reflectance-data)
          (put.value 'atmospheric.passes 'error.message "Data saved")
        (put.value '1.enter.data 'successful.save t)))

;;;-----------------------------------------------------------------------
;;; Methods for Entering Atmospheric Data
;;;-----------------------------------------------------------------------

(defun open-enter-atmospheric-conditions-interface ()
"Opens the interface for entering the atmospheric conditions."
 (initialize-enter-atmospheric-data)
 (put.values 'atmospheric 'wavelengths.left
    (get.values 'estimate.hemispherical.reflectance
          'current.sample.wavelengths))
 (next-wavelength)
 (unitmsg 'viewport-atmospheric.2 'open-panel!))
```

```
(defun initialize-enter-atmospheric-data ()
"Initializes the enter atmospheric conditions interface."
  (remove.all.values 'atmospheric 'standard.atmospheres)
  (remove.all.values 'atmospheric 'aerosol.optical.thickness)
  (remove.all.values 'atmospheric 'size.distribution)
  (remove.all.values 'atmospheric 'phase.function)
  (remove.all.values 'atmospheric 'ozone)
  (remove.all.values 'atmospheric 'precipital.water)
  (remove.all.values 'atmospheric 'single.scattering.albedo)
  (put.value 'atmospheric 'enter.atmospheric.data.options 'enter.data))

(defun save-atmospheric-data ()
"If sufficient data has been entered, calls a function to save the atmospheric
data.  Otherwise displays an error message."
  (let ((current-wavelength
          (get.value 'estimate.hemispherical.reflectance
             'current.wavelength))
         (aero (get.value 'atmospheric 'aerosol.optical.thickness))
         (size-dist (get.value 'atmospheric 'size.distribution))
         (phase-function (get.value 'atmospheric 'phase.function))
         (ozone (get.value 'atmospheric 'ozone))
         (precipital-water (get.value 'atmospheric 'precipital.water))
         (single-scattering-albedo (get.value 'atmospheric
                                    'single.scattering.albedo)))
    (if (and aero size-dist phase-function ozone precipital-water
                  single-scattering-albedo)
         (save-atmospheric-data-aux current-wavelength aero size-dist
                                    phase-function ozone
                                    precipital-water single-scattering-albedo)
         (insufficient-atmospheric-data))))

(defun save-atmospheric-data-aux (current-wavelength aero size-dist
      phase-function ozone
      precipital-water single-scattering-albedo)
"Save the atmospheric data in the currently selected wavelength level unit."
   (put.value current-wavelength 'aerosol.optical.thickness aero)
   (put.value current-wavelength 'size.distribution size-dist)
   (put.value current-wavelength 'phase.function phase-function)
   (put.value current-wavelength 'ozone ozone)
   (put.value current-wavelength 'precipital.water precipital-water)
   (put.value current-wavelength 'single.scattering.albedo
         single-scattering-albedo)
   (put.value 'atmospheric 'successful.save t)
   (next-wavelength))
```

```
(defun next-wavelength ()
"Prompts the user to enter atmospheric data at the next wavelength or closes
the screen is all wavelengths have been processed."
  (let ((new-wavelength (get.value 'atmospheric 'wavelengths.left)))
    (cond ((null new-wavelength)
           (quit-enter-atmospheric-conditions-interface))
          (t (remove.value 'atmospheric 'wavelengths.left new-wavelength)
             (put.value 'estimate.hemispherical.reflectance 'current.wavelength
                    new-wavelength)
             (put.value 'atmospheric 'error.message (format ()
"For each wavelength, either choose a standard atmosphere and then modify the values as
necessary or enter all new parameters. Enter the data for wavelength ~S."
             (get.value new-wavelength 'wavelength)))))))

(defun quit-enter-atmospheric-conditions-interface ()
"Closes the enter atmospheric conditions interface."
  (put.value 'atmospheric 'error.message "")
  (unitmsg 'viewport-atmospheric.2 'close-panel!)
  (put.value 'atmospheric 'done.enter.atmospheric.conditions.p t)
  (when (get.value 'estimate.hemispherical.reflectance 'select.all)
    (open-generate-techniques-interface)))

(defun insufficient-atmospheric-data ()
"Displays an error message if insufficient atmospheric data has been entered."
  (my-documentation-print
   "DATA NOT SAVED - Insufficient data - all boxes must be filled before data can be saved")
   (put.value 'atmospheric 'successful.save nil))

(defun set-up-standard-atmosphere (name)
"Sets up the correct arguments and calls a function to assign the correct slot
values for a standard atmosphere. This function is a dummy at present. It should
be replaced by the correct descriptions of standard atmospheres when they are
available."
  (case name
    (no.gaseous.absorption (set-up-standard-atmosphere-aux 1 2 0.5 300 5 0.8))
    (tropical (set-up-standard-atmosphere-aux 1.2 2 0.6 350 5 0.8))
    (mid.latitude.summer (set-up-standard-atmosphere-aux 1.3 2 0.6 350 5 0.8))
    (mid.latitude.winter (set-up-standard-atmosphere-aux 1.4 2 0.6 360 5 0.8))
    (sub.arctic.summer (set-up-standard-atmosphere-aux 1.5 2 0.6 370 5 0.8))
    (sub.arctic.winter (set-up-standard-atmosphere-aux 1.6 2 0.6 380 5 0.8))
    (us.standard.62 (set-up-standard-atmosphere-aux 1.7 2 0.6 390 5 0.8))
    (no.aerosols (set-up-standard-atmosphere-aux 1.8 2 0.6 400 5 0.8))
    (continental.model (set-up-standard-atmosphere-aux 1.9 2 0.6 250 5 0.8))
    (maritime.model (set-up-standard-atmosphere-aux 2 2 0.6 260 5 0.8))
    (urban.model (set-up-standard-atmosphere-aux 2.1 2 0.6 270 5 0.8))))
```

```
(defun set-up-standard-atmosphere-aux (aero size-dist phase-function ozone
                                       precipital-water
                                       single-scattering-albedo)
"Assigns the appropriate slot values for a standard atmosphere."
  (put.value 'atmospheric 'aerosol.optical.thickness aero)
  (put.value 'atmospheric 'size.distribution size-dist)
  (put.value 'atmospheric 'phase.function phase-function)
  (put.value 'atmospheric 'ozone ozone)
  (put.value 'atmospheric 'precipital.water precipital-water)
  (put.value 'atmospheric 'single.scattering.albedo single-scattering-albedo))

;;;----------------------------------------------------------------------
;;; Methods for Generating Atmospheric Passes Techniques
;;;----------------------------------------------------------------------

(defun user-pick-atm-pass-techniques ()
"Opens the interface that selects each wavelength in turn to allow the user to
select atmospheric passes techniques."
  (unitmsg 'viewport-6.generate.techniques.3 'open-panel!)
  (dolist (thisunit (get.values 'estimate.hemispherical.reflectance
                               'current.sample.wavelengths)
          (all-generate-techniques-finished-message))
    (put.value 'estimate.hemispherical.reflectance 'current.wavelength
        thisunit)
    (user-pick-atm-pass-techniques-aux)
    (remove.all.values '6.generate.techniques 'push.button)
    (wait-for-mouse-gt)))

(defun user-pick-atm-pass-techniques-aux ()
"Opens the interface to allow the user to select atmospheric passes
techniques."
  (reset-initial-values-pick-atm-pass-techniques)
  (unitmsg 'viewport-atmospheric.passes.4 'open-panel!))

(defun reset-initial-values-pick-atm-pass-techniques ()
"Initializes the user pick atmospheric passes techniques screen."
  (remove.all.values 'atmospheric.passes 'selected.techniques)
  (put.value '6.generate.techniques 'error.message "")
  (put.value '6.generate.techniques 'description.of.technique "")
  (put.value 'atmospheric.passes 'action.on.selecting.techniques
     'select.techniques))

(defun pick-selected-values-atm-pass ()
"Stores the selected atmospheric passes techniques in the correct wavelength
level unit and displays a list of the selected techniques."
  (let ((techs (get.values 'atmospheric.passes 'selected.techniques))
        (current-wave (get.value 'estimate.hemispherical.reflectance
                                 'current.wavelength)))
    (unless (null current-wave)
      (put.values current-wave 'techniques techs))
    (tech-message (format ()
          "Techniques selected for the sample at wavelength ~S are:~~{ ~S~}"
                       (wav current-wave) (get-unit-names techs)))))
```

```
;;;-----------------------------------------------------------------------
;;; Methods for Generating Atmospheric Corrections Techniques
;;;-----------------------------------------------------------------------

(defun user-pick-atm-corr-techniques ()
"Opens the interface that selects each wavelength in turn to allow the user to
select atmospheric corrections techniques."
  (unitmsg 'viewport-6.generate.techniques.3 'open-panel!)
  (dolist (thisunit (get.values 'estimate.hemispherical.reflectance
                                'current.sample.wavelengths)
             (all-generate-techniques-finished-message))
    (put.value 'estimate.hemispherical.reflectance 'current.wavelength
         thisunit)
    (user-pick-atm-corr-techniques-aux)
    (remove.all.values '6.generate.techniques 'push.button)
    (wait-for-mouse-gt)))

(defun user-pick-atm-corr-techniques-aux ()
"Opens the interface to allow the user to select atmospheric corrections
techniques."
  (reset-initial-values-pick-atm-corr-techniques)
  (unitmsg 'viewport-atmospheric.corrections.3 'open-panel!))

(defun reset-initial-values-pick-atm-corr-techniques ()
"Initializes the user pick atmospheric corrections techniques screen."
  (remove.all.values 'atmospheric.corrections 'selected.techniques)
  (put.value '6.generate.techniques 'error.message "")
  (put.value '6.generate.techniques 'description.of.technique "")
  (put.value 'atmospheric.corrections 'action.on.selecting.techniques
     'select.techniques))

(defun pick-selected-values-atm-corr ()
"Stores the selected atmospheric corrections techniques in the correct
wavelength level unit and displays a list of the selected techniques."
  (let ((techs (get.values 'atmospheric.corrections 'selected.techniques))
        (current-wave (get.value 'estimate.hemispherical.reflectance
                                 'current.wavelength)))
    (unless (null current-wave)
      (put.values current-wave 'techniques techs))
    (tech-message (format ()
           "Techniques selected for the sample at wavelength ~S are:~{ ~S~}"
                          (wav current-wave) (get-unit-names techs)))))
;;;-----------------------------------------------------------------------
;;; Select all options for atmospheric techniques
;;;-----------------------------------------------------------------------

(defun select-all-atm-options ()
  (determine-atm-starting-point-and-start)
  (put.value 'estimate.hemispherical.reflectance 'select.all t))
```

```
(defun determine-atm-starting-point-and-start ()
  (cond ((not (get.value 'estimate.hemispherical.reflectance
                         'done.enter.data.p))
         (if (atm.pass.p)
             (open-enter-ground-data-interface)
             (open-enter-platform-data-interface)))
      ((not (get.value 'estimate.hemispherical.reflectance
                         'done.characterize.input.p))
        (open-characterize-input-interface))
      ((not (get.value 'estimate.hemispherical.reflectance
                         'done.characterize.target.p))
        (open-characterize-target-interface))
         ((not (get.value 'atmospheric 'done.enter.atmospheric.conditions.p))
            (open-enter-atmospheric-conditions-interface))
      ((not (get.value 'estimate.hemispherical.reflectance
                         'done.generate.techniques.p))
        (open-generate-techniques-interface))
      ((not (get.value 'estimate.hemispherical.reflectance
                         'done.rank.techniques.p))
        (open-rank-techniques-interface))
      ((not (get.value 'estimate.hemispherical.reflectance
                         'done.execute.techniques.p))
        (open-execute-techniques-interface))
      (t (open-output-results-interface))))
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Expert System Shell for Inferring Vegetation Characteristics - Atmospheric Techniques (Task G)		5. Report Date October 1993
		6. Performing Organization Code
7. Author(s) P. Ann Harrison and Patrick R. Harrison		8. Performing Organization Report No. C931031-U-2R06
		10. Work Unit No. 462-61-14
9. Performing Organization Name and Address JJM Systems, Inc. One Ivybrook Blvd., Suite 190 Ivyland, PA 18974		11. Contract or Grant No. NAS5-30127
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 NASA/Goddard Space Flight Center Greenbelt, MD 20771		13. Type of Report and Period Covered Task Report for Task G April - October 1993
		14. Sponsoring Agency Code

15. Supplementary Notes

The Lisp and KEE code for this work is available on a Sun Cartridge Tape.

16. Abstract

The NASA VEGetation Workbench (VEG) is a knowledge based system that infers vegetation characteristics from reflectance data. The VEG Subgoals have been reorganized into categories. A new subgoal category "Atmospheric Techniques" containing two new subgoals has been implemented. The subgoal Atmospheric Passes allows the scientist to take reflectance data measured at ground level and predict what the reflectance values would be if the data were measured at a different atmospheric height. The subgoal Atmospheric Corrections allows atmospheric corrections to be made to data collected from an aircraft or by a satellite to determine what the equivalent reflectance values would be if the data were measured at ground level. The report describes the implementation and testing of the basic framework and interface for the Atmospheric Techniques Subgoals.

17. Key Words (Suggested by Author(s)) EXPERT SYSTEM, ARTIFICIAL INTELLIGENCE, REMOTE SENSING		18. Distribution Statement UNCLASSIFIED - UNLIMITED	
19. Security Classif. (of this report) UNCLASSIFIED	20. Security Classif. (of this page) UNCLASSIFIED	21. No. of pages 35	22. Price

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, VA 22161-2171

APPENDIX C

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - INTERFACE FOR THE ADDITION OF TECHNIQUES (TASK H)

JJM
SYSTEMS INC

N93-25154

C931021-U-2R07

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - INTERFACE FOR THE ADDITION OF TECHNIQUES (TASK H)

22 April 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
One Ivybrook Boulevard, Suite 190
Ivyland, PA 18974

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

SECTION 1.0

INTRODUCTION

All the NASA VEGetation Workbench (VEG) goals except the Learning System provide the scientist with several different techniques. When VEG is run, rules assist the scientist in selecting the best of the available techniques to apply to the sample of cover type data being studied. The techniques are stored in the VEG knowledge base. The design and implementation of an interface that allows the scientist to add new techniques to VEG without assistance from the developer have been completed.

In the previous version of VEG, the addition of a new technique was a complex process. For each new technique, extra units were added manually to the VEG knowledge base and additional Common Lisp code was added to the methods file. Changes were also made manually to the interface that allow the scientist to select which techniques to use.

A new interface that enables the scientist to add techniques to VEG without assistance from the developer has been designed and implemented. This interface does not require the scientist to have a thorough knowledge of Knowledge Engineering Environment (KEE) by Intellicorp or a detailed knowledge of the structure of VEG. The interface prompts the scientist to enter the required information about the new technique. It prompts the scientist to enter the required Common Lisp functions for executing the technique and the left hand side of the rule that causes the technique to be selected. A template for each function and rule and detailed instructions about the arguments of the functions, the values they should return, and the format of the rule are displayed. Checks are made to ensure that the required data have been entered, the functions compiled correctly and the rule parsed correctly before the new technique is stored. The additional techniques are stored separately from the VEG knowledge base.

When the VEG knowledge base is loaded, the additional techniques are not normally loaded. The interface allows the scientist the option of adding all the previously defined new techniques before running VEG. When the techniques are added, the required units to store the additional techniques are created automatically in the correct places in the VEG knowledge base. The methods file containing the functions required by the additional techniques is loaded. New rule units are created to store the new rules. The interface that allow the scientist to select which techniques to use is updated automatically to include the new techniques.

Task H has been completed. The interface that allows the scientist to add techniques to VEG has been implemented and comprehensively tested. The Common Lisp code for the Add Techniques system is listed in Appendix A. A Sun cartridge tape containing KEE and Common Lisp code for the new version of VEG, including the new interface, has been delivered to the NASA GSFC technical representative.

SECTION 2.0

THE ADD TECHNIQUES INTERFACE

When the ADD.TECHNIQUES option is selected from the VEG Administration screen, the Add Techniques interface, shown in Figure 2-1, is opened. The option DEFINE.NEW. TECHNIQUE allows the user to define a new technique and store it ready for subsequent loading into VEG. Selecting the menu option ADD. PREVIOUSLY.DEFINED.TECHNIQUES causes the data, functions and rules for previously defined new techniques that have been defined using DEFINE.NEW.TECHNIQUE to be read from files and added to VEG. The option PURGE.PREVIOUSLY.DEFINED.TECHNIQUES is used to delete all the techniques defined using DEFINE.NEW.TECHNIQUE from the files so they are no longer available to VEG. All three options are described in detail in this section.

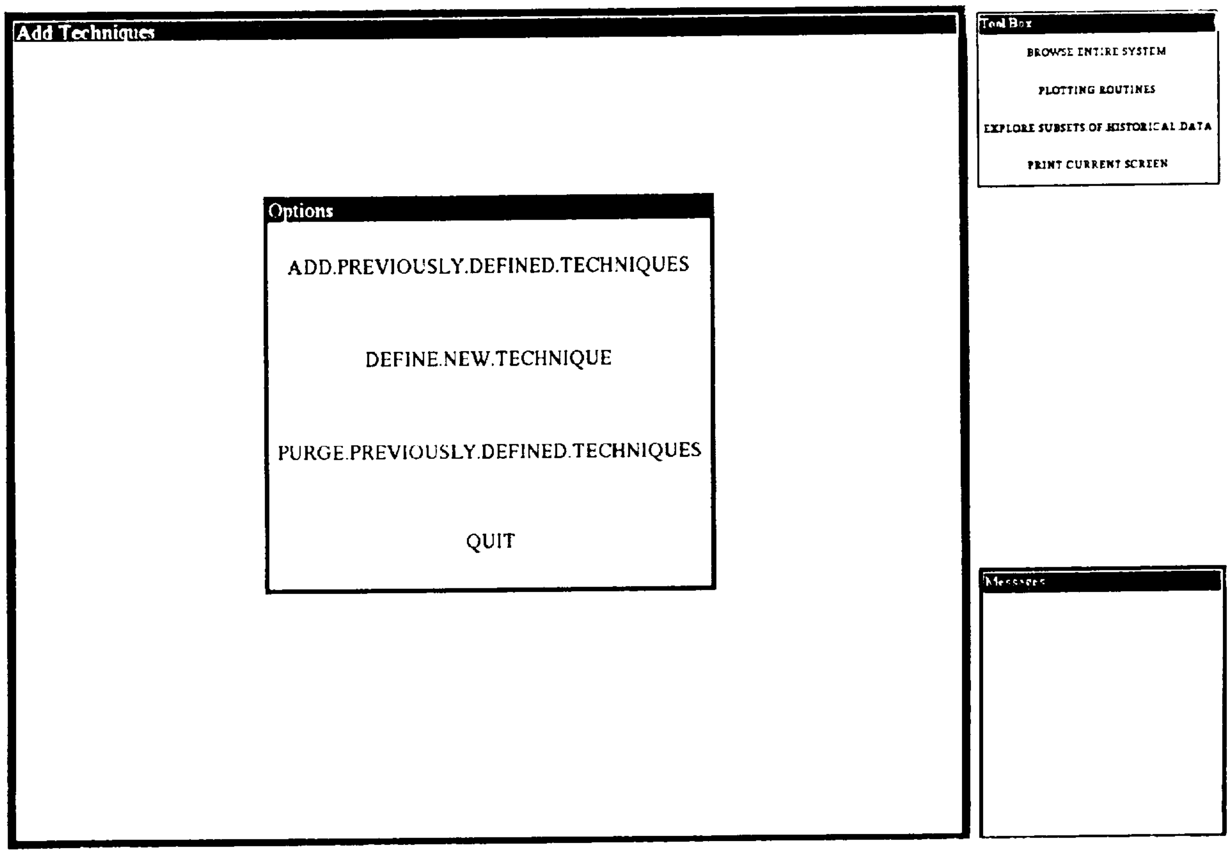
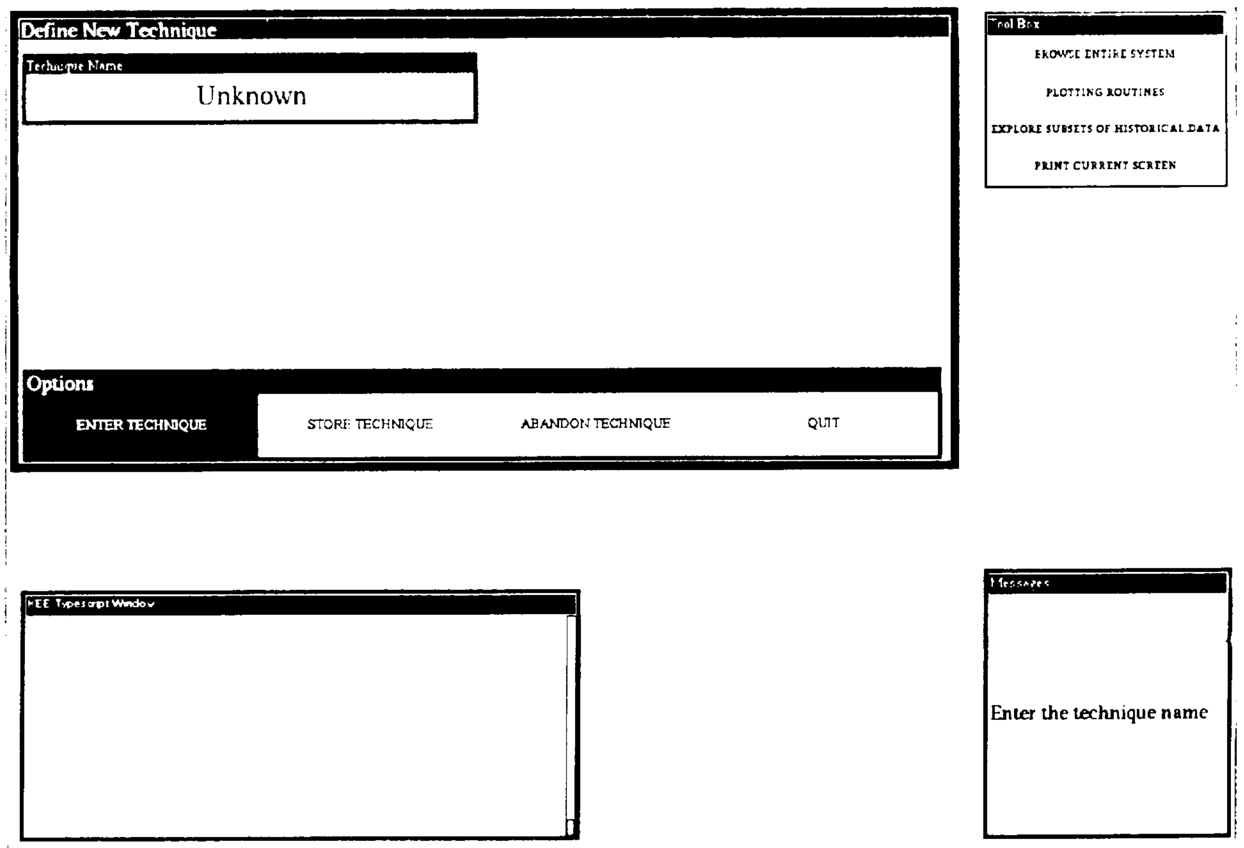


Figure 2-1
The Add Techniques Interface

2.1 DEFINING A NEW TECHNIQUE

When the user selects the option DEFINE.NEW.TECHNIQUE from the Add Techniques Interface (Figure 2-1), the Define New Technique Screen is opened. This screen allows the user to enter the data, functions and rule for the new technique, store the new technique, abandon the new technique or quit the screen. When the screen is first opened, only the "Technique Name" and "Options" subwindows are opened. The user is prompted to enter the name of the new technique. Figure 2-2 shows this. When the Define New Technique Screen is open, the KEE Typescript Window is visible. This allows the user to see any error messages that are displayed in the Typescript window when the functions for the new techniques are compiled or the rules are parsed.



Enter the technique name

Figure 2-2
The Define New Technique Screen When it is First Opened

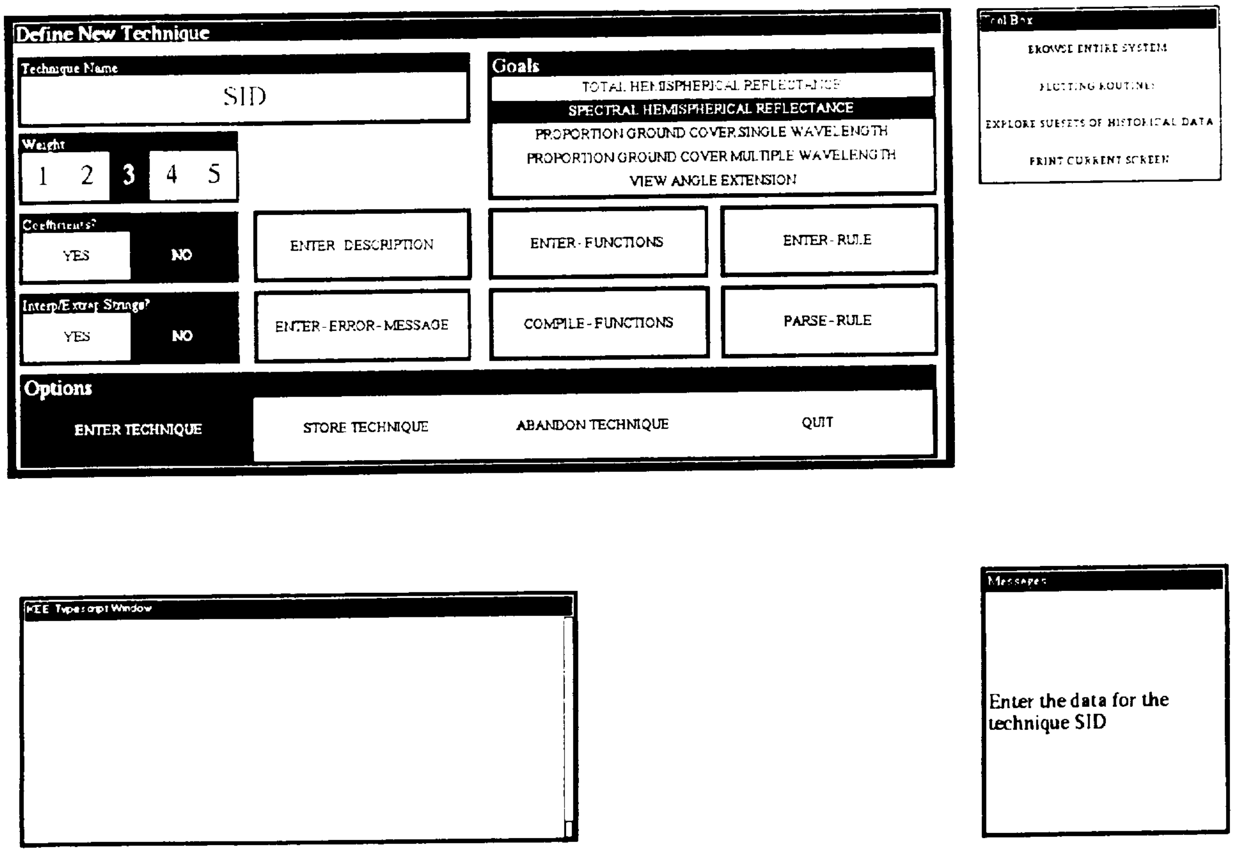
A new unit called ADD.TECHNIQUES has been created in the VEG knowledge base. Figure 2-3 shows the slots in this unit. Each subwindow in the Define New Techniques Screen is a KEE ActiveImage connected to a slot in the unit ADD.TECHNIQUES. Data for the new technique are entered via the interface and stored in slots such as DESCRIPTION and GOALS of the ADD.TECHNIQUES unit. The slots ENTER.DESCRIPTION, ENTER.ERROR.MESSAGE, ENTER.FUNCTIONS, COMPILE.FUNCTIONS, ENTER.RULE and COMPILE.RULE are methods slots. They contain methods which are executed when the user left-clicks on the method-actuator ActiveImage attached to the slot.

COEFFS.METHOD
COEFFS.P
COMPILE.FUNCTIONS
DESCRIPTION
ENTER.DESCRIPTION
ENTER.ERROR.MESSAGE
ENTER.FUNCTIONS
ENTER.RULE
ERROR.MESSAGE
GOALS
INITIALIZED.FUNCTION
INITIALIZED.RULE
INTERPOLATE.EXTRAPOLATE?
MESSAGE
NEW.TECH.OPTIONS
OK.TO.USE
OPTIONS
PARSE.RULE
PREVIOUS.TECHS
RULE.PARSED
TECH.NAME
TECHNIQUE.METHOD
WEIGHT
YES.NO

Figure 2-3
Slots in the Unit ADD.TECHNIQUES

The first step in defining a new technique is to enter the name of the new technique into the subwindow labelled "Technique Name" (Figure 2-1). When the Define New Technique screen is opened, the names of any previously defined new techniques are read from the file and stored in the slot PREVIOUS.TECHS of the unit ADD.TECHNIQUES. If the name of the new technique matches a value in the PREVIOUS.TECHS slot or an existing VEG unit, a message is displayed. This message indicates that the technique has already been defined. In this case, the technique name is not stored. Otherwise, the technique name is stored in the TECH.NAME slot of the

ADD.TECHNIQUES unit. For a function named "SID," for example, the function names "tech-SID," "coeffs-SID" and "SID.ok" are then constructed and stored in the slots TECHNIQUE.METHOD, COEFFS.METHOD and OK.TO.USE, respectively. After the function name has been stored, the rest of the subwindows of the Define New Technique interface are automatically opened. The user must enter or select data or activate methods in all the subwindows of this interface before the technique can be saved. Figure 2-4 shows the Define New Technique screen after all the data for a new technique has been entered. If the user enters a technique name and then subsequently enters another technique name before storing the previously named technique, the interface is re-initialized and any data, functions, or rule entered for the previously named technique are lost.



Technique Name: SID

Weight: 1 2 [✓ 3] 4 5

Coefficients? YES [✓ NO]

Interp/Extrap Strings? YES [✓ NO]

Goals: [✓ SPECTRAL HEMISPHERICAL REFLECTANCE]

Enter the data for the technique SID

Figure 2-4
The Define New Technique Screen After All the Data for a New Technique have been Entered

The "Weight" subwindow (Figure 2-4) holds a number between 1 and 5 which indicates the priority to be given to the technique when the techniques are ranked. The highest priority is 5. Selecting "YES" in the subwindow labelled "Coefficients" indicates that the function used for executing the technique requires coefficients. The default selection for the "Interp/Extrap Strings?" subwindow is "NO." If the technique requires the strings in the reflectance data to be interpolated and extrapolated before the technique is applied, "YES" must be selected in this subwindow. The string techniques for the goal SPECTRAL.HEMISPHERICAL.REFLECTANCE require this extrapolation.

The subwindow labelled "Goals" holds the VEG goal to which the technique applies. New techniques can be added for the VEG goals SPECTRAL.HEMISPHERICAL.REFLECTANCE, PORTION.GROUND.COVER (either single or multiple wavelength) and VIEW.ANGLE. EXTENSION. Only one total hemispherical reflectance technique is currently available in VEG. The interface to select or rank total hemispherical reflectance techniques has not yet been implemented. If the user selects the goal TOTAL.HEMISPHERICAL.REFLECTANCE, a message is displayed. The message indicates that the techniques for this goal have not been implemented and the new technique will not be stored.

When the user is running VEG and chooses to select the techniques manually, the User Pick Techniques screen is opened. Each time a technique is selected, a description of the technique is displayed on the screen. If the selected technique is suitable for the sample being studied, the message "This technique is suitable for this sample" is displayed. Otherwise an error message is displayed. Left-clicking on ENTER-DESCRIPTION and ENTER-ERROR-MESSAGE in the Define New Technique screen enables the user to enter the description and error message that will be displayed in the User Pick Techniques screen when the technique is added to VEG .

When the user left-clicks on ENTER-FUNCTIONS, the temporary file "temp.lisp" is opened. Templates for the functions required by the new technique are written to this file. The names for the technique functions that were constructed when the technique name was entered, are automatically incorporated in the templates. Then the editor is opened and the user is prompted to enter the new functions. Functions are required to execute the technique, calculate the coefficients (if any) required by the technique, and to determine whether the technique is suitable for a particular sample. A prompt in the "Message" window tells the user how to save the temporary file and exit the editor. The method actuator COMPILE-FUNCTIONS is used to compile the functions for the new technique. The method first checks that functions have been entered. If the file "temp.lisp" is not found, an error message is displayed and no attempt at compilation is made. Otherwise, the function file is compiled and any compilation errors or warnings are displayed in the KEE Typescript window. If the compilation is successful, the compiled functions are stored in the binary file "temp.sbin." If any errors occurred during the compilation, this file remains empty. Note that the functions are compiled both for efficiency and also to create additional error checking. If the user reselects ENTER-FUNCTIONS while still entering data for the same technique, the previously edited temporary file is opened once again. The changes made in the previous edit session are not lost. This allows the user to edit a file repeatedly until the functions are correct and the file compiles successfully.

When the user selects ENTER-RULE, another temporary file is opened and a template for the rule is written to the file. The name of the new technique is incorporated in the template. The editor is then opened, and the user is prompted to modify the template to create the rule required for the new technique. If the user left-clicks on PARSE-RULE, checks are made to confirm that a rule has been entered, and that the rule contains the same number of left and right parentheses. The failure of either of these checks causes an error message to be displayed. Otherwise, an attempt is made to parse the rule using a user-defined function named TEST-RULE-PARSES. This function sets up the structure so that the KEE parser can parse the new rule. The function creates a temporary rule unit as an instance of the KEE unit VEG.RULES. The newly entered rule is stored in the EXTERNAL.FORM slot of the temporary rule unit. The KEE PARSE function is then applied to the rule unit. If no parse errors occur, the function returns T. Otherwise the function returns NIL. Before the value is returned, the temporary rule unit is deleted from VEG. If the rule did not parse correctly, an error message is displayed in the "Message" box. The slot RULE.PARSED in the ADD.TECHNIQUES unit is used as a flag to indicate whether or not a correctly parsed rule has been entered. Note that even though a rule contains the same number of left and right parentheses and parses correctly, it might still be incorrect.

When the user selects the option STORE-TECHNIQUES, checks are made to make sure that all the required data has been entered, the functions have been defined and compiled successfully, and a correctly parsed rule has been entered. If the checks are unsuccessful, nothing is stored and the user is prompted to complete entry of the required data. Otherwise, the data and rule are appended to the file "new-tech-data," and the functions are appended to the file "new-tech.lisp" which is immediately compiled. Table 2-1 shows the format in which the data taken from the ActiveImages shown in Figure 2-4 would be stored in the file "new-tech-data." Next, the new technique name is added to the PREVIOUS.TECHS slot of the unit ADD.TECHNIQUES. The interface is then re-initialized, and the subwindows, except the "Technique Name" and "Options" subwindows, are closed, as in Figure 2-2.

Table 2-1
The Format in which the Data taken from the ActiveImages in Figure 2-4 would be Stored in the File "new-tech-data"

DESCRIPTION	VALUE
Goal	SPECTRAL.HEMISPHERICAL.REFLECTANCE
Technique Name	SID
Description	"TECHNIQUE SID - A NEW TECHNIQUE FOR CALCULATING THE SPECTRAL HEMISPHERICAL REFLECTANCE OF A SAMPLE THAT HAS DATA AT 4 VIEW ANGLES"
Error Message	"TECHNIQUE SID IS UNSUITABLE FOR THIS SAMPLE BECAUSE IT DOES NOT HAVE DATA AT 4 VIEW ANGLES"
Technique function	tech-SID
Interp/extrap Strings?	NO
Function uses Coefficients?	YES
Coefficients function	coeffs-SID
Suitability Function	SID.ok
Weight	3
Rule	(IF (THE CURRENT.SAMPLE.WAVELENGTHS OF ESTIMATE.HEMISPHERICAL.REFLECTANCE IS ?X) (THE NUMBER.VIEW.ANGLES OF ?X IS 4) THEN (LISP (ADD.VALUE ?X (QUOTE TECHNIQUES) (QUOTE SID))))

Selecting the option ABANDON-TECHNIQUE causes the deletion of any data, functions, and rules that have been entered but not stored. The interface is then re-initialized. This is, in effect, a panic button that the user can activate to stop the process at any point.

The Define New Technique interface can be exited by selecting the QUIT option. Any partially entered technique is deleted when this option is selected. It is important to note that at this stage any newly defined and stored techniques have been saved in files, but they have not yet been added to VEG.

2.2 ADDING PREVIOUSLY DEFINED NEW TECHNIQUES

Adding a new technique to VEG involves several steps. A new unit must be created in the VEG knowledge base to hold the data required by the technique. Another unit is required to hold the rule that will enable the technique to be selected when it is appropriate for the cover type sample being studied. The functions required by the technique must be loaded. The interface must be updated so that the displays that list the available techniques for a VEG goal include the additional techniques. It is important to note that this system automatically places newly created units in their proper location in the system.

When the user selects the option ADD.PREVIOUSLY.DEFINED.TECHNIQUES from the Add Techniques screen (Figure 2-1), VEG first checks that the files "new-tech-data" and "new-tech.lisp" are present. These files hold the data, functions and rules for the new techniques. If either of these files is missing, the message "No techniques available" is displayed in the "Messages" box and processing stops. If the required files are present, the file "new-tech.sbin" is loaded. This file is the compiled version of the file "new-tech.sbin" that contains the functions required by the new techniques. Processing of the data then begins. The data are read from the file "new-tech-data." The format of this file was shown in Table 2-1.

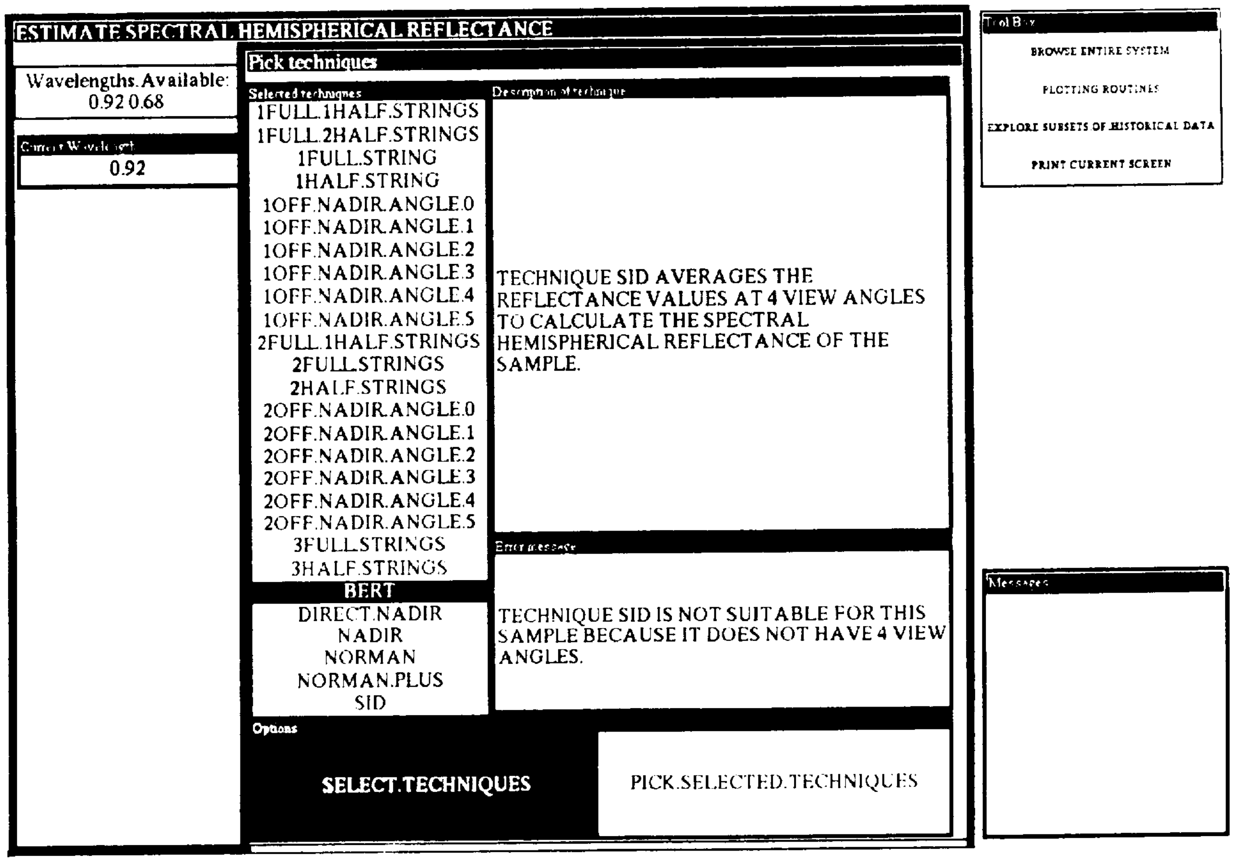
The VEG goal to which the new technique applies is read first. The techniques for each VEG goal are stored in instances of different subclasses of the unit TECHNIQUES. For example, techniques for the goal SPECTRAL.HEMISPHERICAL.REFLECTANCE are stored in instances of the subclass unit SPECTRAL.HEMISPHERICAL.REFLECTANCE.TECHNIQUES. The rules for each goal have names that reflect the goal to which they apply, and they are stored in ruleclasses that are subclasses of the unit VEG.RULES. For example, rules for the goal SPECTRAL.HEMISPHERICAL. REFLECTANCE have names that are prefixed by "HRTR," and they are stored in instances of the unit HEMISPHERICAL.REFLECTANCE.TECHNIQUES. The techniques for different goals are displayed in different windows in the interface. After the goal has been read from the file, the names of the technique subclass, rule prefix, ruleclass, and interface window that apply to the goal are identified.

The technique name is next read from the file. The system will not allow the same technique to be added more than once to VEG. If the technique has already been added to VEG, the remainder of the data for this technique is skipped in the file. Otherwise, a new unit is created as an instance of the correct VEG subclass to which the technique applies. Information about the technique, such as its description, the name of the technique function, and whether the technique function requires coefficients, are read from the file and stored in this unit. A rule unit is then created in the correct ruleclass. The name of the rule unit is constructed using the appropriate prefix. The rule is read from the file and stored in the "External Form" slot of the rule unit.

When VEG is running and a cover type sample is being processed, the user can select the techniques to apply to the sample using the Pick Techniques Screen. The names of all the available VEG techniques for the appropriate goal are displayed on this screen. The final step in adding a new technique to VEG is to update this screen to include the new technique. Figure 2-5 shows the

Pick Techniques Screen for the goal SPECTRAL.HEMISPHERICAL.REFLECTANCE after two new techniques called SID and BERT have been added. In the example in the figure, BERT is in dark because it has already been selected. The user has attempted to select SID. However, an error message is being displayed because SID is not suitable for the sample being studied.

Adding new techniques continues until the end of the file "new-tech-data" is reached. The message "Loading ...." is then removed from the screen.



Wavelengths.Available: 0.92 0.68

Current Wavelength: 0.92

1FULL.1HALF.STRINGS
1FULL.2HALF.STRINGS
1FULL.STRING
1HALF.STRING
1OFF.NADIR.ANGLE.0
1OFF.NADIR.ANGLE.1
1OFF.NADIR.ANGLE.2
1OFF.NADIR.ANGLE.3
1OFF.NADIR.ANGLE.4
1OFF.NADIR.ANGLE.5
2FULL.1HALF.STRINGS
2FULL.STRINGS
2HALF.STRINGS
2OFF.NADIR.ANGLE.0
2OFF.NADIR.ANGLE.1
2OFF.NADIR.ANGLE.2
2OFF.NADIR.ANGLE.3
2OFF.NADIR.ANGLE.4
2OFF.NADIR.ANGLE.5
3FULL.STRINGS
3HALF.STRINGS
BERT
DIRECT.NADIR
NADIR
NORMAN
NORMAN.PLUS
SID

TECHNIQUE SID AVERAGES THE REFLECTANCE VALUES AT 4 VIEW ANGLES TO CALCULATE THE SPECTRAL HEMISPHERICAL REFLECTANCE OF THE SAMPLE.

TECHNIQUE SID IS NOT SUITABLE FOR THIS SAMPLE BECAUSE IT DOES NOT HAVE 4 VIEW ANGLES.

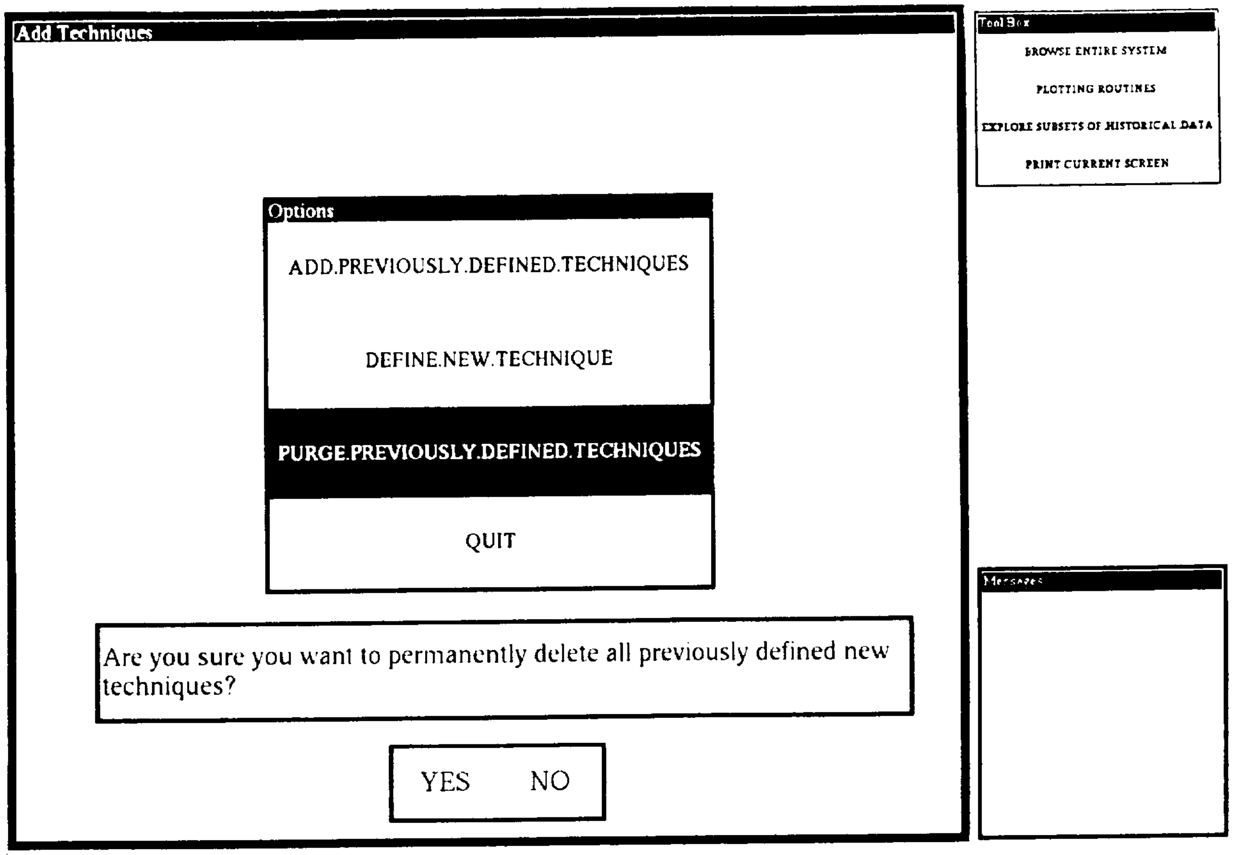
Figure 2-5
The Pick Techniques Screen for the Goal SPECTRAL.HEMISPHERICAL.REFLECTANCE

2.3 PURGING PREVIOUSLY DEFINED NEW TECHNIQUES

This option allows the user to delete from the files any techniques defined using the DEFINE.NEW.TECHNIQUE option. When this option is activated the techniques are permanently deleted from the files so they are no longer available to VEG.

When the user selects the option PURGE.PREVIOUSLY.DEFINED.TECHNIQUES from the Add Techniques Interface (Figure 2-1), additional subwindows are opened, as shown in Figure 2-6. The user is asked to confirm that the techniques should be deleted. If the user left-clicks on "YES," the file "new-tech-data," that contained the data and rules for the new techniques, is deleted. The technique functions are then removed form the file "new-tech.lisp." If the user selects "NO," the techniques are not deleted. Finally, the additional subwindows are removed from the screen.

Selecting QUIT from the Add Techniques screen (Figure 2-1) returns the user to the Administration screen.



Are you sure you want to permanently delete all previously defined new techniques?

Figure 2-6
The Add Techniques Interface with the Option PURGE.PREVIOUSLY.DEFINED.TECHNIQUES Selected

SECTION 3.0

TESTING AND RESULTS

The Add Techniques options were tested using both valid and invalid data. When errors were detected, they were corrected and the test runs were repeated to ensure that the corrections were successful. The tests were designed to test the typical range of user behavior. The test runs and results are described in this section.

3.1 TEST 1

The purpose of Test 1 was to test the navigation back and forth through the various menu levels from the VEG top level to the Add Techniques Screen. The user left-clicked on RUN.VEG, ADMINISTRATION and ADD.TECHNIQUES on successive screens. As expected, the Add Techniques Screen was opened. The user then selected QUIT in each successive menu to navigate back to the top level of VEG. This test showed that the screens between the VEG top level and the Add Techniques Screen were opened and closed in the correct sequence.

3.2 TEST 2

This test was designed to test the operation of the Add Previously Defined New Techniques option before any new techniques had been defined. At this time, no additional techniques had been defined so the file "new-tech-data" had not been created and the file "new-tech.lisp" contained only comments. The option ADD.PREVIOUSLY. DEFINED.NEW.TECHNIQUES was selected from the Add Techniques menu. The message "No techniques available" was displayed in the "Messages" window. This test showed that the system could deal correctly with an attempt to add previously defined new techniques before any new techniques had been defined.

3.3 TEST 3

Test 3 was intended to test the entry of valid data to define a new technique. The DEFINE.NEW.TECHNIQUE option was selected from the Add Techniques menu. The Define New Technique Screen was opened and the user was prompted to enter the name of the new technique. At this time, most of the subwindows in the Define New Technique screen were closed. The user entered the name "SID" for the new technique. Then the rest of the subwindows were automatically opened and the user was prompted to enter the rest of the data for the new technique. Note that the technique SID was invented by the developer for testing the system. It is not a real technique. The user selected "NO" in both the "Coefficients" and the "Interp/Extrap?" boxes. A weight of three was specified. The goal SPECTRAL.HEMISPHERICAL. REFLECTANCE was selected. A description and an error message were entered. The user left clicked on ENTER-FUNCTIONS. The function SID.ok was edited so that it would return T if the sample had four view angles and nil otherwise. The function tech-SID was edited to return the average reflectance value of the four view angles. Because the function to execute the technique SID does not require coefficients, the function coeffs-SID was deleted. The file was then saved and compiled. Compilation was successful. The user selected ENTER-RULE and edited the template so that the rule would fire if the cover type sample had four view angles. The rule parsed correctly. Finally, the STORE.TECHNIQUE option was selected from the menu at the bottom of the screen. No error messages were displayed. Inspection of the files "new-tech-data" and "new-tech.lisp" showed that the new technique had been successfully saved. The user was then prompted to enter the name of the next new technique.

This test showed that the Define New Technique interface was working correctly when valid data were entered. The data, functions, and rule for the new technique were entered via the interface and successfully stored in the appropriate files. Inspection of the created files confirmed this.

3.4 TEST 4

The Define New Technique Interface should not allow the same technique name to be used more than once. In Test 4, attempts were made to use the same technique name twice.

In the first part of Test 4, the technique name "SID" was entered again. The message "Technique SID has already been defined" was displayed in the messages box.

In the second part of Test 4, the technique name "NORMAN" was entered. The message "Technique NORMAN has already been defined" was displayed in the messages box. Technique NORMAN is a technique for estimating spectral hemispherical reflectance that is part of the VEG knowledge base.

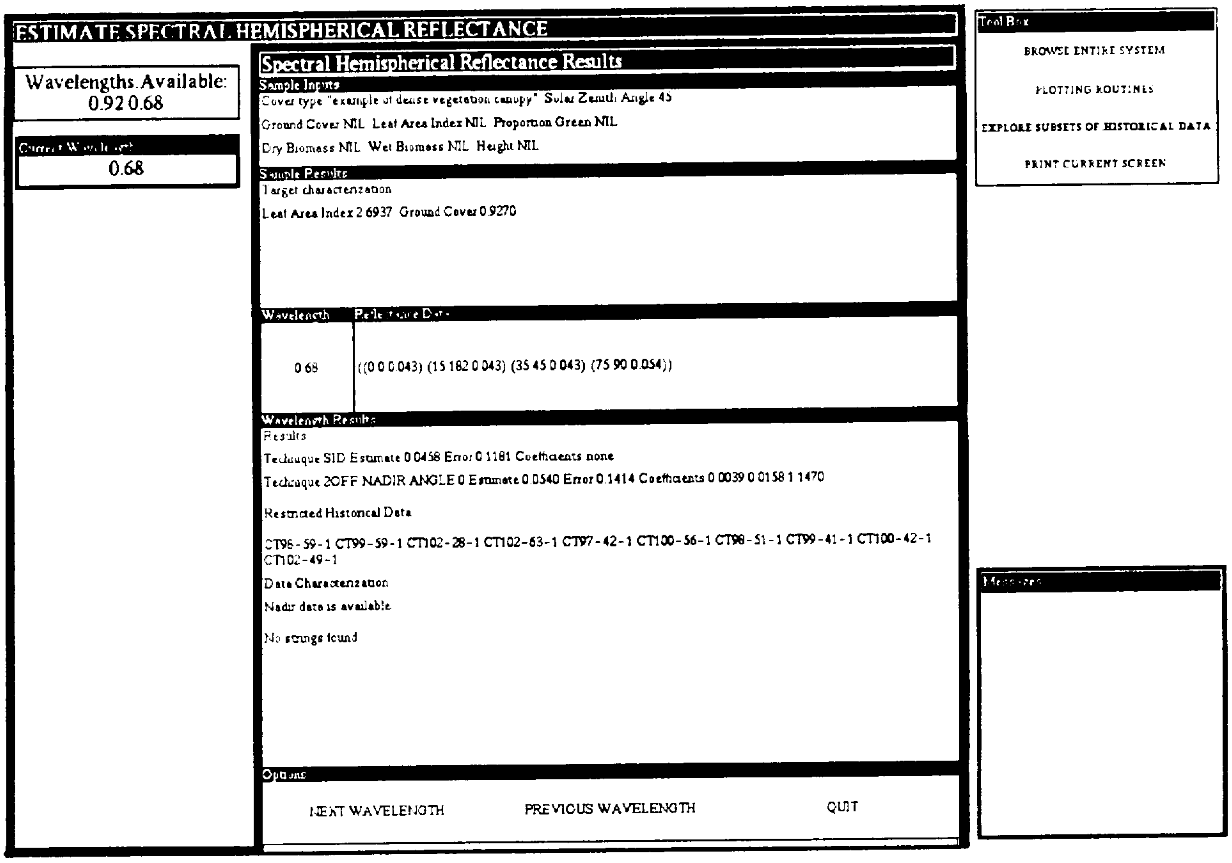
Test 4 showed that the Add Techniques interface will not allow the same technique name to be used twice. It can detect attempts to re-use a technique name either already stored in VEG, or saved in the file of previously defined new techniques.

3.5 TEST 5

This test was designed to test the addition of multiple, previously defined new techniques to VEG from files and the operation of VEG using the newly added techniques. At the beginning of this test, another new technique called BERT was defined for the goal SPECTRAL. HEMISPHERICAL.REFLECTANCE. This technique was also invented by the developer for testing purposes. The technique was effectively the technique DIRECT.NADIR, applied only to samples with one view angle. Technique BERT was saved in the files "new-tech-data" and "new-tech.lisp." Then the ADD. PREVIOUSLY.DEFINED.NEW.TECHNIQUES option was selected for the Add Techniques interface. The message "Loading ...." was displayed. After loading had been completed, this message was removed. Inspection of the VEG knowledge base showed that new units had been created in the correct places in the VEG knowledge base. These units held the data and the rules for the new techniques.

The user ran VEG in Research Mode using the goal SPECTRAL.HEMISPHERICAL. REFLECTANCE. Techniques SID and BERT were designed to operate on samples with data at four and one view angles, respectively. Sample 4 was selected as the sample to be studied so that both new techniques could be tested. This sample has four view angles at wavelength 0.68 μm and one view angle at wavelength 0.92 μm.

Sample 4 was processed. The manual method of selecting techniques was chosen for the data at both wavelengths. As shown in Figure 2-5, both new techniques were displayed on the User Pick Techniques screen. The user attempted to select both new techniques for the data at both wavelengths. The functions that check the suitability of each technique for each sample worked correctly. The user was prevented from selecting technique SID for the wavelength 0.92 μm since data at only one view angle were available at this wavelength. Similarly, the user was prevented from selecting technique BERT for the wavelength 0.68 μm. After selecting the techniques manually, the user also chose to have VEG choose the techniques automatically. The purpose of this was to test the rules for selecting the techniques. Technique SID was among the techniques chosen for the wavelength 0.68 μm and technique BERT was among the techniques chosen for the wavelength 0.92 µm. This proved that the new rules were operating correctly. Then the techniques were ranked and the user chose to use the best two techniques for each wavelength so that both new techniques were used. The techniques were then executed and the results were displayed. Figure 3-1 shows some of the results obtained. The results were correct. Test 5 showed that newly defined techniques could be added to VEG and used correctly within VEG.


ESTIMATE SPECTRAL HEMISPHERICAL REFLECTANCE
Spectral Hemispherical Reflectance Results
Tool Box
BROWSE ENTIRE SYSTEM
PLOTTING ROUTINES
EXPLORE SUBSETS OF HISTORICAL DATA
PRINT CURRENT SCREEN
Messages
Options
NEXT WAVELENGTH
PREVIOUS WAVELENGTH
QUIT

Wavelengths.Available:
0.92 0.68

Current Wavelength
0.68

Sample Inputs
Cover type "example of dense vegetation canopy" Solar Zenith Angle 45
Ground Cover NIL Leaf Area Index NIL Proportion Green NIL
Dry Biomass NIL Wet Biomass NIL Height NIL

Sample Results
Target characterization
Leaf Area Index 2 6937 Ground Cover 0.9270

Wavelength	Reflectance Data
0 68	((0 0 0.043) (15 182 0 043) (35 45 0 043) (75 90 0.054))

Wavelength Results
Results
Technique SID Estimate 0 0458 Error 0 1181 Coefficients none
Technique 2OFF NADIR ANGLE 0 Estimate 0.0540 Error 0.1414 Coefficients 0 0039 0 0158 1 1470

Restricted Historical Data

CT96-59-1 CT99-59-1 CT102-28-1 CT102-63-1 CT97-42-1 CT100-56-1 CT98-51-1 CT99-41-1 CT100-42-1 CT102-49-1

Data Characterization
Nadir data is available

No strings found

Figure 3-1
The Output Screen at the End of Test 5

3.6 TEST 6

In Test 6, new techniques were defined and added to VEG for the goals VIEW.ANGLE. EXTENSION and PORTION.GROUND.COVER. Then VEG, along with the new techniques, was run. The test showed that the addition of new techniques for these goals was successful.

3.7 TEST 7

In Test 7, a new technique was defined for the VEG goal TOTAL.HEMISPHERICAL. REFLECTANCE. This option is not yet available. When the user selected this goal from the Define New Technique Screen, an error message was displayed. The user ignored this error message and continued to enter data for the new technique. When the user attempted to store the data for the new technique, another error message was displayed and the data were not stored. This test showed that the system was correctly blocking attempts to store new techniques for the goal TOTAL.HEMISPHERICAL.REFLECTANCE.

3.8 TEST 8

Test 8 was designed to ensure that an incomplete set of data and functions for a new technique would not be stored. It was also designed to ensure that attempting to compile functions before they were defined, or to parse a rule before it was entered, would produce appropriate error messages.

The user opened the Define New Technique screen and immediately attempted to store a new technique, even though none had yet been defined. The message "Technique name not found - data not stored" was displayed and nothing was stored. After entering a technique name, the user again attempted to store the new technique. This time the user was prompted to select the goal. The user continued to enter the data items, one at a time, in response to the prompts, each time attempting to store the new technique. As expected, every attempt to save the incomplete technique data was unsuccessful.

After all the data items had been entered, the message "Functions not found - data not stored" was displayed when the user attempted to store the technique. The user then attempted to compile the functions before entering them. This time the error message "Functions not found - enter them before compiling" was displayed. The user entered the required functions and then once again attempted to store the technique. This time the error message informed the user that the functions must be compiled. The functions were then successfully compiled. The next attempt to store the technique produced the error message "Rule not found - data not stored." The user attempted to parse the rule before entering it. Again an error message was displayed. After entering a rule, the user tried again to store the data. This time the user was prompted to parse the rule. After the rule had been successfully parsed, an attempt to store the new technique succeeded.

This test showed that incomplete data for a technique could not be stored. It also showed that an appropriate error message is displayed if the user attempts to compile functions before defining them or to parse a rule before entering it.

3.9 TEST 9

The user may enter invalid functions for a new technique. The Add Techniques interface tests whether new functions will compile and it does not store a new technique unless the new functions compile correctly. Test 9 was designed to test the behavior of the system with invalid technique functions.

Various errors such as unmatched right parentheses, undefined functions, incorrect arguments to functions, and missing arguments to functions were introduced into the function file. These produced warnings which were reported in the KEE Typescript window when the function file was compiled. However, a compiled function file was created in each of these cases and attempts to store the function were successful.

An unmatched left parenthesis error was also introduced into the function file. When this file was compiled, the "End of file reading in a list" error was signaled and a list of debugging action options was shown in the KEE Typescript Window. The user entered the debugging action number 1 to kill the process in this case. No compiled file was created and the user was prevented from storing the new technique.

This test showed that many errors in the technique functions produce warnings rather than error messages. Although the interface prevents the user from storing a technique function that produces a compiler error, it does not prevent the user from storing a function that produces a compiler warning. The user should correct warnings before storing a technique, even though the interface does not insist on this. Test 9 showed that there are limitations on the detection of invalid functions by the Define New Function system.

3.10 TEST 10

This test was designed to test the parsing of a new rule and to determine the limitations of the system in preventing invalid rules from being stored. Various errors were introduced into a rule to determine how the system would respond. For example, an extra term was added to a rule clause. It was interpreted by the rule compiler as a literal so the rule parsed successfully even though it was incorrect. In this case, the invalid rule was stored.

In separate tests, extra left and right parentheses were added to the rule. These were detected before the rule was parsed, and in each case, the technique with the invalid rule was not stored.

In separate tests, the IF and THEN clauses of the rule were omitted. Despite these omissions, the rules parsed successfully.

Test 10 showed that the use of the KEE rule parser to detect errors in a rule is limited to some syntactic errors. Note that adding new rules is the most difficult part of the Add Techniques system to control. It is quite possible to add rules that are nonsense. The user is cautioned, therefore, to be careful when adding rules.

3.11 TEST 11

Test 11 was designed to test the ABANDON.TECHNIQUE option from the Define New Technique screen. Several new techniques were entered. Each time, the entry of the new technique was abandoned at a different point. In every case, the interface was initialized correctly and all the data, functions and rule for the abandoned technique were correctly deleted. This test showed that the ABANDON.TECHNIQUE option was operating correctly.

3.12 TEST 12

This test was designed to test the operation of the PURGE.PREVIOUSLY.DEFINED. TECHNIQUES option from the Add Techniques interface (Figure 2-1). When this option was selected, additional subwindows were opened. The user was prompted to confirm that the techniques should be deleted. The user left-clicked on "NO." The message "Techniques not deleted" was displayed in the "Messages" box and the subwindows were then closed. Inspection of the files "new-tech-data" and "new-tech.lisp" confirmed that the techniques had not been deleted. The user then selected the PURGE.PREVIOUSLY.DEFINED.TECHNIQUES option again. This time the user left-clicked on "YES" to confirm that the techniques should be deleted. The message "Techniques deleted" was displayed in the "Messages" box and the subwindows were closed once again. Inspection of the files confirmed that the file "new-tech-data" had been deleted and the file "new-tech.lisp" contained only headings. Test 12 showed that the PURGE.PREVIOUSLY. DEFINED.TECHNIQUES option was operating correctly.

SECTION 4.0

CONCLUSIONS

The Add Techniques system implements a software component for defining additional analysis techniques that are used to evaluate samples of cover type data. The system provides a detailed, window driven, user interface which organizes the entry of the technique definitions. Dynamic error checking, file management, object creation, and definition management facilities are provided.

The technique definition has multiple components that include description, error message, function body, rule for determining when the technique can be used, and technique priority. The user follows instructions on various windows to input technique elements. Error checking is done interactively by the system. The function component of the definition is compiled for efficiency.

The new definition is managed so that it is logically isolated from the basic VEG system. In a separate step, the new technique may be loaded for use.

Testing of the Add Techniques system focused on the expected range of typical user behavior. It proved to be reasonably robust and user-friendly.

APPENDIX A

LISTING OF METHODS FILES FOR THE ADD TECHNIQUES SYSTEM

```
;;; veg-methods5.lisp
;;;
;;; Code to allow the user to add techniques to VEG
;;;
;;; Written by Ann Harrison
;;; Created April 1, 1993
;;; Last modified April 20, 1993

(in-package 'kee)

(defun open-add-techniques-menu ()
"Open the screen for adding techniques."
  (unitmsg 'viewport-add.techniques.1 'open-panel!)
  (remove.all.values 'add.techniques 'options))

;;;-----------------------------------------------------------------------
;;; Functions required to add previously defined techniques from files to VEG
;;;-----------------------------------------------------------------------

(defun add-previously-defined-techniques ()
"Loads previously defined additional techniques from a file."
  (cond ((and (probe-file "new-tech.sbin")
              (probe-file "new-tech-data"))
         (my-documentation-print "Loading ........")
         (load "new-tech")     ; Load the file containing the functions
                               ; required by the techniques
         (with-open-file (str "new-tech-data" :direction :input)
           (load-tech-data-from-file str))
         (clear-prompt))
        (t (my-documentation-print "No techniques available"))))

;;; Note that the function> read-file is in the methods file veg-methods1.lisp

(defun load-tech-data-from-file (str)
"Sets up the appropriate arguments and calls the function to create the units
to store the data for the technique and rule units in VEG."
  (do ((goal (read-file str)(read-file str)))
      ((null goal) nil) ;End of file
    (case goal
      (total.hemispherical.reflectance
            (my-documentation-print "This option is not yet implemented"))
      (spectral.hemispherical.reflectance
                (load-tech str
                           'spectral.hemispherical.reflectance.techniques
                           'hemispherical.reflectance.technique.rules
                           "HRTR."
                           (unit
              'windowpane-selected.techniques-of-6.generate.techniques.3)))
    (proportion.ground.cover.single.wavelength
             (load-tech str
               'proportion.ground.cover.single.wavelength.techniques
               'proportion.ground.cover.single.wavelength.rules
               "PGCSWR."
               (unit
               'windowpane-selected.techniques-of-portion.ground.cover.5)))
    (proportion.ground.cover.multiple.wavelength
             (load-tech str
               'proportion.ground.cover.multiple.wavelength.techniques
               'proportion.ground.cover.multiple.wavelength.rules
               "PGCMWR."
               (unit
         'windowpane-selected.mw.techniques-of-portion.ground.cover.5)))
    (view.angle.extension
             (load-tech str
                       'view.angle.extension.techniques
                       'view.angle.extension.rules
                       "VAER."
                       (unit
           'windowpane-selected.techniques-of-view.angle.extension.6))))))

(defun load-tech (str tech-class rule-class prefix window)
"Creates the units to store the data for the technique and rule in VEG. Loads
the data from the file."
    (let ((new-tech (read-file str)))
        (if (unit.exists.p new-tech)        ; Technique already read in
            (dotimes (n 9) (read-file str)) ; Read past this technique
            (let ((new-tech-unit            ; Read in technique
                   (create.unit new-tech 'veg nil tech-class))
                  (new-rule-unit
                   (create.unit (gentemp prefix) 'veg nil rule-class)))
              (put.value new-tech-unit 'description (read-file str))
              (put.value new-tech-unit 'error.message (read-file str))
              (put.value new-tech-unit 'technique.method (read-file str))
              (put.value new-tech-unit 'interpolate.extrapolate?
                    (if (eq (read-file str) 'YES)
                       t
                       nil))
              (cond ((eq (read-file str) 'YES)
                     (put.value new-tech-unit 'coeffs.p t)
                     (put.value new-tech-unit 'coeff.method (read-file str)))
                    (t (put.value new-tech-unit 'coeffs.p nil)
                       (read-file str))) ;Ignore coeffs method from file
              (put.value new-tech-unit 'ok.to.use (read-file str))
              (put.value new-tech-unit 'weight (read-file str))
              (put.value new-rule-unit 'external.form (read-file str))
              (slot-image-toggle-enable window) ; Update the user pick
                                    ; technique interface
              (slot-image-toggle-enable window)))))
```

```
;;;----------------------------------------------------------------------
;;; Functions required to define a new technique
;;;----------------------------------------------------------------------

(defun define-new-techniques ()
"Opens and initializes the interface to guide the user through entering the
required data for a new technique."
  (unitmsg 'viewport-add.techniques.1 'close-panel!)
  (unitmsg 'viewport-add.techniques.2 'open-panel!)
  (remove.all.values 'add.techniques 'tech.name)
  (initialize-add-techniques)
  (put.value 'add.techniques 'new.tech.options 'enter.technique)
  (store-previously-defined-tech-names)
  (my-documentation-print "Enter the technique name"))

(defun close-new-tech-windows ()
"Close the subwindows of the define new technique interface."
  (unitmsg 'windowpane-coeffs.p-of-add.techniques.4 'close!)
  (unitmsg 'windowpane-interpolate.extrapolate?-of-add.techniques.10 'close!)
  (unitmsg 'windowpane-weight-of-add.techniques.4 'close!)
  (unitmsg 'windowpane-goals-of-add.techniques.2 'close!)
  (unitmsg 'windowpane-enter.description-of-add.techniques.4 'close!)
  (unitmsg 'windowpane-enter.error.message-of-add.techniques.3 'close!)
  (unitmsg 'windowpane-enter.functions-of-add.techniques.6 'close!)
  (unitmsg 'windowpane-compile.functions-of-add.techniques.1 'close!)
  (unitmsg 'windowpane-enter.rule-of-add.techniques.7 'close!)
  (unitmsg 'windowpane-parse.rule-of-add.techniques.2 'close!))

(defun store-previously-defined-tech-names ()
"If any techniques have been defined, calls the function to collect the
technique names."
   (when (probe-file "new-tech-data")
    (with-open-file (str "new-tech-data" :direction :input)
        (remove.all.values 'add.techniques 'previous.techs)
        (read-tech-names-from-file str))))

(defun read-tech-names-from-file (str)
"Saves the names of the techniques that have already been defined in the slot
PREVIOUS.TECHS of the unit ADD.TECHNIQUES."
  (do ((data (read-file str)(read-file str)))
     ((null data) nil)                ; End of file
    (add.value 'add.techniques 'previous.techs (read-file str))
    (dotimes (n 9)(read-file str))))

(defun already-defined (tech-name)
"Returns t if a technique of the same name has already been defined and nil
otherwise."
  (or (unit.exists.p tech-name)
     (member tech-name (get.values 'add.techniques 'previous.techs)
            :test #'equal)))
```

```
(defun open-new-tech-windows ()
"Open the subwindows of the define new technique interace.  This function is
called after the new technique has been named."
   (unitmsg 'windowpane-coeffs.p-of-add.techniques.4 'open!)
   (unitmsg 'windowpane-interpolate.extrapolate?-of-add.techniques.10 'open!)
   (unitmsg 'windowpane-weight-of-add.techniques.4 'open!)
   (unitmsg 'windowpane-goals-of-add.techniques.2 'open!)
   (unitmsg 'windowpane-enter.description-of-add.techniques.4 'open!)
   (unitmsg 'windowpane-enter.error.message-of-add.techniques.3 'open!)
   (unitmsg 'windowpane-enter.functions-of-add.techniques.6 'open!)
   (unitmsg 'windowpane-compile.functions-of-add.techniques.1 'open!)
   (unitmsg 'windowpane-enter.rule-of-add.techniques.7 'open!)
   (unitmsg 'windowpane-parse.rule-of-add.techniques.2 'open!))

(defun initialize-add-techniques()
"Initializes the slots in the unit ADD.TECHNIQUES, ready for entering the new
technique.  If they exist, deletes the files that have temporarily held the
functions and the selection rule for a previously entered new technique."
   (remove.all.values 'add.techniques 'technique.method)
   (remove.all.values 'add.techniques 'coeffs.method)
   (remove.all.values 'add.techniques 'ok.to.use)
   (remove.all.values 'add.techniques 'goals)
   (put.value 'add.techniques 'description "")
   (put.value 'add.techniques 'error.message "")
   (put.value 'add.techniques 'interpolate.extrapolate? 'no)
   (put.value 'add.techniques 'coeffs.p 'yes)
   (put.value 'add.techniques 'weight 1)
   (put.value 'add.techniques 'initialized.function t)
   (put.value 'add.techniques 'initialized.rule t)
   (put.value 'add.techniques 'rule.parsed nil)
   (when (probe-file "temp.lisp")
    (lcl::shell "rm temp.lisp"))  ; Remove temporary function file
   (when (probe-file "temp.lsbin")
    (lcl::shell "rm temp.sbin"))  ; Remove temporary compiled function file
   (when (probe-file "temp-rule")
    (lcl::shell "rm temp-rule"))) ; Remove temporary rule file

(defun enter-description (self)
"Prompts the user to enter the description of a new technique into a file.
Then reads it from the file into the description slot of ADD.TECHNIQUES."
  (declare (ignore self))
  (my-documentation-print "Complete the description of the technique.
Save the file and exit the editor to save the description.")
  (sleep 1)
  (when (equal (get.value 'add.techniques 'description) "")
   (with-open-file (str "temp-desc" :direction :output :if-exists :supersede)
     (princ (format () "Technique ~A " (get.value 'add.techniques 'tech.name))
            str)))
  (lcl::shell "textedit temp-desc")
```

```
 (put.value 'add.techniques 'description
    (with-open-file (str "temp-desc" :direction :input)
         (let ((desc ""))
           (do ((dat (read-file str)(read-file str)))
              ((null dat) desc)
            (setf desc (format () "~A ~A" desc dat))))))
 (clear-prompt))

(defun enter-error-message (self)
"Prompts the user to enter the error message of a new technique into a file.
Then reads it from the file into the error.merssage slot of ADD.TECHNIQUES."
 (declare (ignore self))
 (my-documentation-print "Complete the description of the error message.
Save the file and exit the editor to save the error message.")
 (sleep 1)
 (when (equal (get.value 'add.techniques 'error.message) "")
   (with-open-file (str "temp-error" :direction :output :if-exists :supersede)
     (princ (format () "Technique ~A " (get.value 'add.techniques 'tech.name))
            str)))
 (lcl::shell "textedit temp-error")
 (put.value 'add.techniques 'error.message
    (with-open-file (str "temp-error" :direction :input)
         (let ((desc ""))
           (do ((dat (read-file str)(read-file str)))
              ((null dat) desc)
            (setf desc (format () "~A ~A" desc dat))))))
 (clear-prompt))

(defun enter-functions(self)
"Enter the functions required by the new technique."
 (declare (ignore self))
 (let ((tech-name (get.value 'add.techniques 'tech.name)))
   (cond (tech-name
           (when (get.value 'add.techniques 'initialized.function)
             (with-open-file (str "temp.lisp" :direction :output
                                        :if-exists :supersede)
               (princ (format ()
";;; Templates for adding technique ~A

(in-package 'kee)

;;; Replace the body of this example function with the correct function for
;;; the new technique.  The function checks whether the technique is suitable
;;; for the sample.  Here the sample is the wavelength level unit.  The
;;; function should return t if the technique is suitable for the sample and
;;; nil otherwise.
(defun ~A (sample)
\"Checks the suitability of the function ~A for the sample.\"
 (= (get.value sample 'number.view.angles) 1))
;;; Replace the body of this example function with the technique function for
;;; the new technique.  In this function the arguments are the sample unit at
;;; the wavelength level and the vector of coefficients, if any.  The function
;;; should return a number which is the result of applying the technique to the
;;; sample.
(defun ~A (thisunit coeffs)
\"Applies the function ~A to the sample.\"
  (declare (ignore coeffs))  ;Remove this line if technique uses coefficients
  (third (first (get.value thisunit 'reflectance.data))))

;;; If the technique uses coefficients, replace the body of this example
;;; function with the coefficient function.  Otherwise delete the template.
;;; In this function the argument is the list of restricted historical data
;;; units to be used for calculating the coefficients.  The function should
;;; return the vector of coefficients of the correct length for the technique.
(defun ~A (data)
\"Calculates the coefficients for the technique ~A.\"
  (declare (ignore data))    ;Replace these lines with the new function body
  nil)~%"

                        tech-name
                         (get.value 'add.techniques 'ok.to.use) tech-name
                        (get.value 'add.techniques 'technique.method) tech-name
                        (get.value 'add.techniques 'coeffs.method) tech-name)
                        str))
                   (put.value 'add.techniques 'initialized.function nil))
             (my-documentation-print
"Edit the file.  Then save the file and exit the editor.")
             (sleep 1)
             (lcl::shell "textedit temp.lisp")
             (clear-prompt))
           (t (my-documentation-print
                  "Enter technique name before entering the functions")))))

(defun compile-functions (self)
"Compiles the functions for the new technique."
  (declare (ignore self))
  (when (not (probe-file "temp.lisp"))
    (my-documentation-print
     "Functions not found - enter them before compiling")
    (return-from compile-functions nil))
  (my-documentation-print "Compiling the new functions")
  (when (probe-file "temp.sbin")
    (lcl::shell "rm temp.sbin"))
  (compile-file "temp.lisp" :messages nil :file-messages nil)
  (my-documentation-print "Finished compilation"))

(defun compiled-ok ()
"Returns t if the function complied correctly and nil otherwise."
   (when (probe-file "temp.sbin")
      (with-open-file (str "temp.sbin" :direction :input)
         (let ((len (file-length str)))
           (and (numberp len)
                (> len 0))))))
```

```
(defun enter-rule (self)
"Sets up a file to temporarily store the new rule.  Prompts the user to enter
the rule.  Attempts to parse it.  If parsing fails, prompts the user to correct
the rule until it parses correctly."
 (declare (ignore self))
 (put.value 'add.techniques 'rule.parsed nil)
 (let ((tech-name (get.value 'add.techniques 'tech.name)))
   (cond (tech-name
          (when (get.value 'add.techniques 'initialized.rule)
            (with-open-file (str1 "temp-rule" :direction :output
                                  :if-exists :supersede)
              (princ (format ()
";;; Template for rule for selecting technique ~A

;;; Edit the lefthand side of this example rule to create the required rule
(IF (THE CURRENT.SAMPLE.WAVELENGTHS OF
                    ESTIMATE.HEMISPHERICAL.REFLECTANCE IS ?X)
       (THE NUMBER.VIEW.ANGLES OF ?X IS 1)
     THEN (LISP (ADD.VALUE ?X (QUOTE TECHNIQUES)
                    (QUOTE ~A))))"
             tech-name tech-name)
            str1))
         (my-documentation-print
"Edit the file.  Then save the file and exit the editor.")
         (put.value 'add.techniques 'initialized.rule nil))
       (sleep 1)
       (lcl::shell "textedit temp-rule"))
      (t (my-documentation-print
          "Enter technique name before entering the rule")))))

(defun parse-rule (self)
"Returns t if the rule parses correctly and nil otherwise.  Note that parsing
is not a complete test of correctness for a rule."
 (declare (ignore self))
 (my-documentation-print "Parsing rule")
 (when (not (probe-file "temp-rule"))
  (my-documentation-print
   "Rule not found - enter it before parsing")
  (put.value 'add.techniques 'rule.parsed nil)
  (return-from parse-rule nil))
 (with-open-file (str1 "temp-rule" :direction :input)
  (cond  ((not (parens-ok str1))
          (my-documentation-print
           "Rule has unequal number of left and right parens - edit again")
          (put.value 'add.techniques 'rule.parsed nil))
         ((test-rule-parses str1)
          (my-documentation-print "Rule parsed OK")
          (put.value 'add.techniques 'rule.parsed t))
         (t
          (my-documentation-print
           "Rule does not parse correctly - edit again.")
          (put.value 'add.techniques 'rule.parsed nil)))))
```

```
(defun test-rule-parses (str1)
"Sets up a temporary unit to hold the new rule.  Attempts to parse it.  Deletes
the temporary rule unit.  Returns t if the rule parsed OK and nil otherwise."
  (let ((new-rule-unit
          (create.unit 'TEMP 'veg nil 'vegrules)))
    (put.value new-rule-unit 'external.form (read-file str1))
    (prog2 (unitmsg new-rule-unit 'parse)
         (not (get.value new-rule-unit 'parse.errors))
      (delete.unit new-rule-unit))))

(defun store-data ()
"Stores the data about the new technique in the file."
  (let ((goal (get.value 'add.techniques 'goals))
        (tech-name (get.value 'add.techniques 'tech.name))
        (description (get.value 'add.techniques 'description))
        (error-message (get.value 'add.techniques 'error.message)))
    (cond ((not tech-name)
            (my-documentation-print
             "Technique name not found - data not stored"))
          ((not goal)
           (my-documentation-print
            "Goal not found - data not stored"))
          ((eq goal 'total.hemispherical.reflectance)
           (my-documentation-print
            "Techniques for this goal are not yet implemented - not stored"))
          ((equal description "")
           (my-documentation-print
            "Description not found - data not stored"))
          ((equal error-message "")
           (my-documentation-print
            "Error message not found - data not stored"))
          ((not (compiled-ok))
           (my-documentation-print
            "Functions not correctly compiled - data not stored"))
          ((not (probe-file "temp-rule"))
           (my-documentation-print "Rule not found - data not stored"))
          ((not (get.value 'add.techniques 'rule.parsed))
           (my-documentation-print
            "Rule not successfully parsed - data not stored"))
          (t (store-data-on-file goal tech-name description error-message)))))

(defun store-data-on-file (goal tech-name description error-message)
"Stores the technique data in the file new-tech-data.  Calls the function to
store the technique functions."
  (my-documentation-print "Saving the new technique")
  (with-open-file (str "new-tech-data" :direction :output :if-exists :append
                        :if-does-not-exist :create)
    (princ (format ()
                   "~A~%~A~%\"~A\"~%\"~A\"~%~A~%~A~%~A~%~A~%~A~%~A~%"
                   goal
                   tech-name
                   description
                   error-message
                   (get.value 'add.techniques 'technique.method)
```

```
                    (get.value 'add.techniques 'interpolate.extrapolate?)
                    (get.value 'add.techniques 'coeffs.p)
                    (get.value 'add.techniques 'coeffs.method)
                    (get.value 'add.techniques 'ok.to.use)
                    (get.value 'add.techniques 'weight))
          str)
   (add.value 'add.techniques 'previous.techs tech-name)
   (store-functions str)))

(defun store-functions (str)
"Adds the function for the new technique to the file new-tech.lisp.  Compiles
the file.  Adds the new rule to the file new-tech-data."
 (lcl::shell "cat new-tech.lisp temp.lisp > temp1")
 (lcl::shell "mv temp1 new-tech.lisp")
 (compile-file "new-tech.lisp" :messages nil :file-messages nil :warnings nil)
 (with-open-file (str1 "temp-rule" :direction :input)
   (princ (read-file str1) str) ; Read the rule from the temporary file and
   (terpri str))                ; store it in the file new-tech-data
 (clear-prompt)
 (remove.all.values 'add.techniques 'tech.name)
 (initialize-add-techniques)
 (close-new-tech-windows)
 (my-documentation-print "Enter the name of the new technique")
 (put.value 'add.techniques 'new.tech.options 'enter.technique))

(defun abandon-data ()
"Initializes the values in the add.techniques unit.  Deletes any recently
entered but not yet stored functions or rules.""
stored."
 (remove.all.values 'add.techniques 'tech.name)
 (initialize-add-techniques)
 (close-new-tech-windows)
 (my-documentation-print "Enter the name of the new technique")
 (put.value 'add.techniques 'new.tech.options 'enter.technique))

(defun read-char-file (str)
"Reads a charcter from a file.  Returns the character read, or nil if the end
of the file has been reached."
  (flet ((eof-p (obj)
            (eq obj '*eof*)))
    (let ((obj (read-char str () '*eof* ())))
      (if (eof-p obj)
          nil
          obj))))

(defun parens-ok (str)
"Returns t if the file contains the same number of left and right parens and
nil otherwise."
 (let ((left 0)
       (right 0))
   (do ((char (read-char-file str)(read-char-file str)))
       ((null char) (if (zerop (- left right))
                         t
                         nil))
      (case char
           (#\) (incf right))
           (#\( (incf left))))))

(defun purge-previously-defined-techniques ()
"Opens the required subwindows ready to remove all previously defined
new techniques from the files."
   (remove.all.values 'add.techniques 'yes.no)
   (put.value 'add.techniques 'message
"Are you sure you want to permanently delete all previously defined new techniques?")
   (unitmsg 'windowpane-message-of-add.techniques.2 'open-panel!)
   (unitmsg 'windowpane-yes.no-of-add.techniques.3 'open-panel!))

(defun purge-techniques()
"Removes all previously defined new techniques from the files so they are no
longer available to be added to VEG."
  (when (probe-file "new-tech-data")
    (lcl::shell "rm new-tech-data"))
  (with-open-file (str "new-tech.lisp" :direction :output
                                       :if-exists :supersede)
    (princ (format ()
";;; new-tech.lisp
;;;
;;; Holds Functions Required by Newly Defined Techniques
;;; Functions are entered through the Define New Technique Interface

") str)))
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Expert System Shell for Inferring Vegetation Characteristics - Interface for the Addition of Techniques (Task H)		5. Report Date April 1993
		6. Performing Organization Code
7. Author(s) P. Ann Harrison		8. Performing Organization Report No. C931021-U-2R07
		10. Work Unit No. 462-61-14
9. Performing Organization Name and Address JJM Systems, Inc. One Ivybrook Blvd., Suite 190 Ivyland, PA 18974		11. Contract or Grant No. NAS5-30127
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 NASA/Goddard Space Flight Center Greenbelt, MD 20771		13. Type of Report and Period Covered Task Report for Task H March - April 1993
		14. Sponsoring Agency Code

15. Supplementary Notes

The Lisp and KEE code for this work is available on a Sun Cartridge Tape.

16. Abstract

VEG is an expert system that infers vegetation characteristics from reflectance data. VEG provides the scientist with several different analysis techniques which are stored in the knowledge base. When VEG is run, rules assist the scientist in selecting the best of the available techniques to apply to the sample of cover type data being studied. In the previous version of VEG, the addition of a new technique was a complex process. A new interface that enables the scientist to add techniques to VEG without assistance from the developer has been designed and implemented. It guides the scientist through entering the data, Common Lisp functions and the rule required by the new technique. Once the technique has been defined, adding it to VEG requires only the selection of the appropriate menu option. The Add Techniques System was tested using both valid and invalid data. The tests were designed to test the typical range of user behavior. They confirmed that the interface was operating correctly.

17. Key Words (Suggested by Author(s)) EXPERT SYSTEM, ARTIFICIAL INTELLIGENCE, REMOTE SENSING		18. Distribution Statement UNCLASSIFIED - UNLIMITED	
19. Security Classif. (of this report) UNCLASSIFIED	20. Security Classif. (of this page) UNCLASSIFIED	21. No. of pages 33	22. Price

NASA FORM 1626 OCT 86

For sale by the National Technical Information Service, Springfield, VA 22161-2171

APPENDIX D

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - PROTOTYPE HELP SYSTEM (TASK I)

JJM SYSTEMS INC

N94-11236

C931032-U-2R08

AN EXPERT SYSTEM SHELL FOR INFERRING VEGETATION CHARACTERISTICS - PROTOTYPE HELP SYSTEM (TASK I)

July 1993

Prepared for:

National Aeronautics and Space Administration
Goddard Space Flight Center
Greenbelt, MD 20771

Prepared by:

JJM Systems, Inc.
1225 Jefferson Davis Hwy., Suite 412
Arlington, VA 22202

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

KEE	Knowledge Engineering Environment
VEG	VEGetation Workbench

SECTION 1.0

INTRODUCTION

The NASA VEGetation Workbench (VEG) infers vegetation characteristics from reflectance data. VEG was developed using the Intellicorp, Knowledge Engineering Environment (KEE). KEE is a mature development platform that supports a number of well-engineered components including inference engines, windows, graphics tools, objects and inheritance, procedural attachments and other support needed for prototyping expert systems using object-oriented programming.

An extensive, window-oriented interface system was constructed for VEG using the KEE graphics package called "ActiveImages." This interface provides a variety of screens to enhance dialogue between the scientist and the system. The interface is a key feature of this system. It was designed to focus the scientist on the appropriate level of organization to carry out scientific work without attention to "housekeeping" functions. The interface allows the scientist to run VEG and select options at all stages of the run by clicking the mouse over the appropriate menu option. The interface further allows the scientist to focus on the data and the functions performed by VEG as it abstracts away most of the underlying, detailed complexity of the VEG system.

A prototype Help System has been designed and implemented. The Help System allows the scientist to get more information about each screen in the VEG interface. It was designed to help the new user of VEG to learn how to operate the system. The Help System is stored in separate files from the VEG knowledge base and it is loaded only when needed. An interface that allows the scientist to add and modify help messages has also been integrated into the "Administration" part of the VEG system. This enables the scientist to evolve the Help System over time.

Task I of this project required the design and implementation of a prototype Help System. This task has been completed. The code for the Lisp methods used in this task is included in Appendix A. A Sun cartridge tape containing the Lisp methods and the current version of VEG, including the Help System has been delivered to the NASA GSFC technical representative.

SECTION 2.0

DESCRIPTION OF THE HELP SYSTEM

The storage of help messages in the VEG system and the method of operation of the Help System are described in this section.

2.1 STORAGE OF HELP MESSAGES IN VEG

The screens in the VEG interface were built using the KEE ActiveImages package. The attributes of each screen are stored in KEE units known as "Viewports" and a number of other units that hold the attributes of the screen subwindows. The Help System provides help for each VEG screen. When the Help System is loaded, an extra slot called "HELP" is created in each viewport unit. This slot holds the help message for the screen to which the viewport unit refers.

The unit HELP.SYSTEM has been created in VEG. This unit holds the slots required by the Help System. The slots in this unit are shown in Figure 2-1. The HELP.LOADED slot is initialized with the value NIL to indicate that the Help system is not loaded. The scientist must explicitly load the Help System. When the Help System is loaded, the value of HELP.LOADED is changed to T for true. The rationale for this approach is that the Help System will be used less as the scientist gains proficiency with the VEG system. Therefore, when possible, the VEG system avoids the overhead of having the Help System loaded.

```
HELP.LOADED
MESSAGE
OPTIONS
```

Figure 2-1
Slots in the Unit HELP.SYSTEM

The VEG system uses four separate knowledge bases: VEG, LEARN, AZIMUTH.PLOT and POLAR.PLOT. Each knowledge base contains at least one viewport. The help messages for the four knowledge bases in the VEG system are stored in the files "help-messages-veg," "help-messages-learn," "help-messages-azimuth," and "help-messages-polar," respectively. The files hold the viewport name and the appropriate help message for each viewport that has help available. When the Help System is loaded, the slot "HELP" is first added to the units that are parent units of the viewport units so that the HELP slot is inherited by each viewport unit. The help messages for the VEG knowledge base are then read from the "help-messages-veg" file and stored in the newly created HELP slots. Checks are then made to determine whether any of the other VEG knowledge bases have been loaded. If any additional knowledge bases have been loaded, then the help messages for these knowledge bases are also read from the files and stored in the knowledge bases. The value of the HELP.LOADED slot of the unit HELP.SYSTEM is then changed to T. If any of the knowledge bases LEARN, AZIMUTH.PLOT or POLAR.PLOT are subsequently loaded, the help messages for the additional knowledge base are read from the appropriate help message file and stored in the knowledge base immediately after it has been loaded.

2.2 THE OPERATION OF THE HELP SYSTEM

The user can activate the Help System at any time when VEG is loaded, by left clicking on the HELP.SYSTEM option in the Tool Box Menu as shown in Figure 2-2. If HELP.SYSTEM is moused in the Tool Box Menu, a Lisp method checks the value of the HELP.LOADED slot of the HELP unit and loads the Help System if it has not been loaded. The Help Screen is then opened as shown in Figure 2-3. The ActiveImages in this screen are attached to slots in the HELP.SYSTEM unit. The user is prompted to mouse on the window he/she needs help with, and the cursor changes to a cross shape. When the user left-clicks on a window, the Help System identifies the window that has been moused on. If the window is a KEE viewport, the help slot of the viewport unit corresponding to the window is examined. If a help message is found, the help message is put into the MESSAGE slot and consequently displayed on the Help System Screen, as shown in Figure 2-3. If no help for the window is available, the message "Sorry, no help is available for this window" is displayed. If the user has selected a window that is not part of the VEG system, such as the KEE typescript window or the Open Windows Workspace, the message "Not a VEG window" is displayed on the Help System Screen.

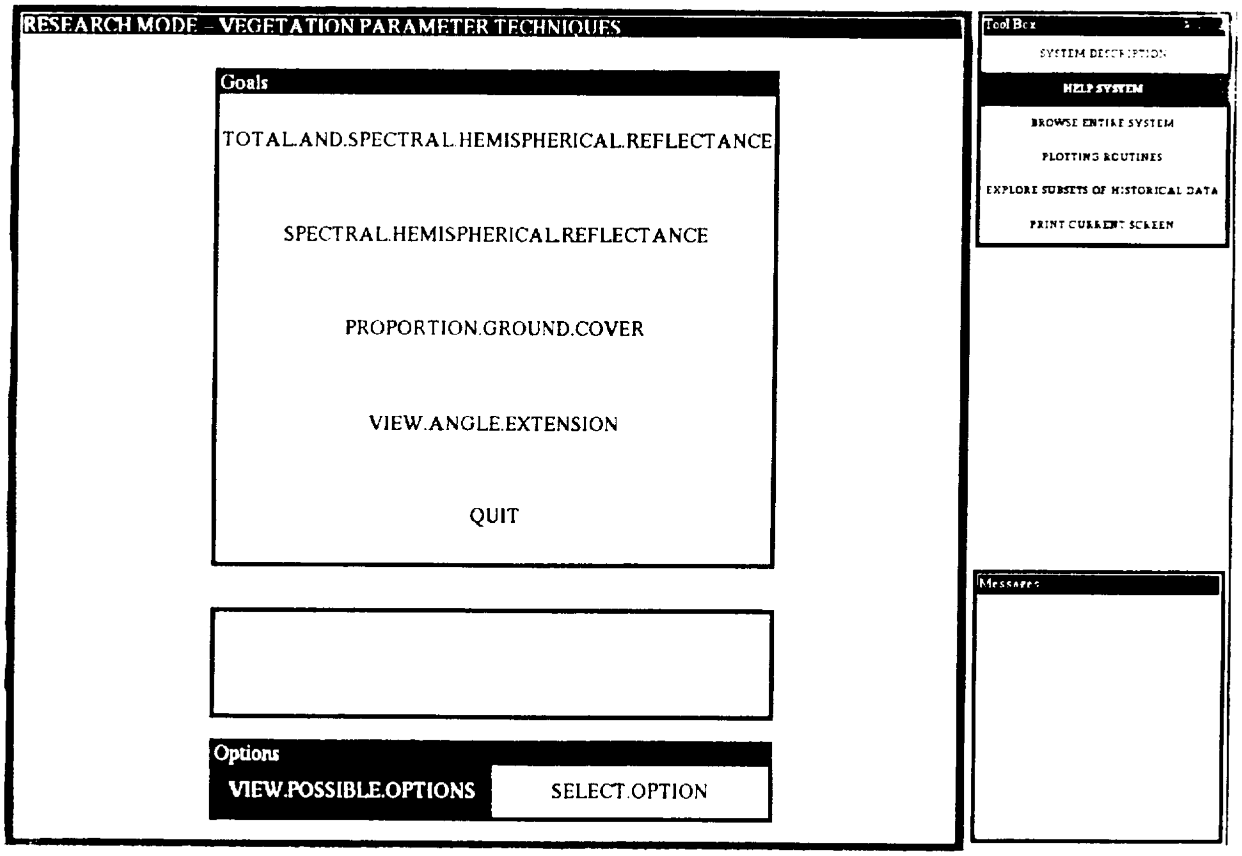
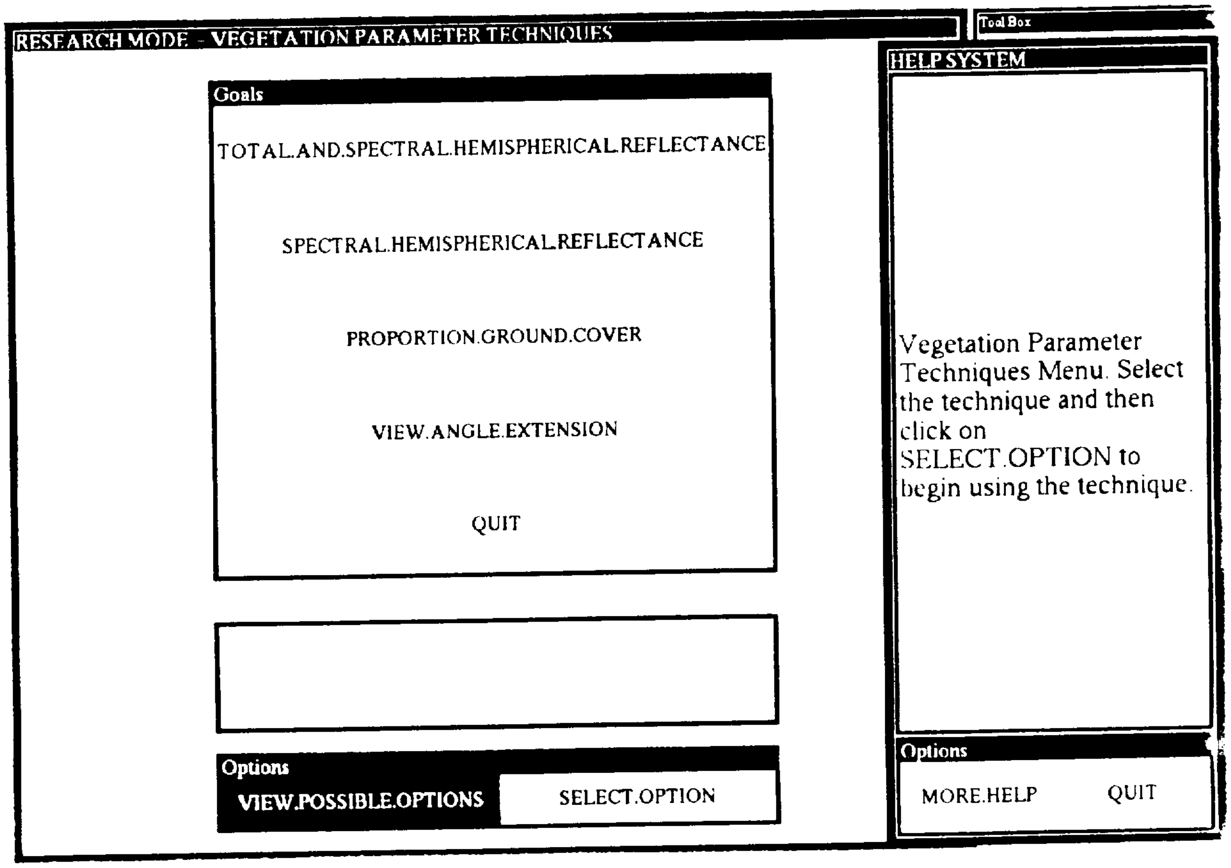


Figure 2-2
Using the Help System

The user can continue to use VEG while the Help System Screen is open, although one or more windows might be partially occluded by the Help System Screen. If the user left clicks on MORE.HELP, the cursor changes to a cross again and the user is prompted to select another screen for help. Once opened, the Help System Screen remains open until the user closes it by left clicking on the QUIT option at the bottom right of the screen.



Vegetation Parameter Techniques Menu. Select the technique and then click on SELECT.OPTION to begin using the technique.

Figure 2-3
The Help Screen

SECTION 3.0

ADDING AND MODIFYING HELP MESSAGES

An option that allows the scientist to add or modify help messages has been added to the Administration part of VEG. This allows the scientist to evolve the Help System interactively. It recognizes that help concepts will evolve and change as the scientist gains experience using various system functions. VEG is also extensible in certain ways which might require the addition and/or modification of help messages.

If the user left clicks on RUN.VEG and then selects ADMINISTRATION from the Processing Mode menu, the Administration Screen is opened as shown in Figure 3-1. The option CHANGE.HELP.MESSAGES has been added to the Administration Menu for handling message changes. When the user selects CHANGE.HELP.MESSAGES, the Administration Screen is closed and the Add/Change Help screen is opened, as shown in Figure 3-2. The user is prompted to navigate through the VEG system until the screen is reached that is to be associated with the new or modified help message. When the screen level is located, the user left clicks on MODIFY.HELP in the option section of the Add/Change Help window. When this is done, the cursor changes to a cross and the user is prompted to left click on the screen for which help is to be added or modified. When the cursor is brought over the appropriate window and left clicked, the New Help Message window is opened. This is shown in Figure 3-3. The Add/Change Help Window allows the user to iterate the process of modifying and saving help messages. The New Help Message window displays the value "Unknown" if this is a new help message as shown in Figure 3-3. If the user is modifying an existing message, the current help message is displayed in the window as shown in Figure 3-4. Once the new message is typed into the window, as shown in Figure 3-5, or the existing message has been modified, left clicking on DONE will close the window. This process can be continued by left clicking on MODIFY.HELP in the options section of the Add/Change Help window. The user can choose to click on SAVE.HELP at any time in order to save any changes made to the Help System. Finally, when the user is done adding and modifying the Help System, left clicking on QUIT in the options section of the Add/Change Help window will close the window and return control to the Administration window. If messages are not saved, they will be lost when VEG application is exited.

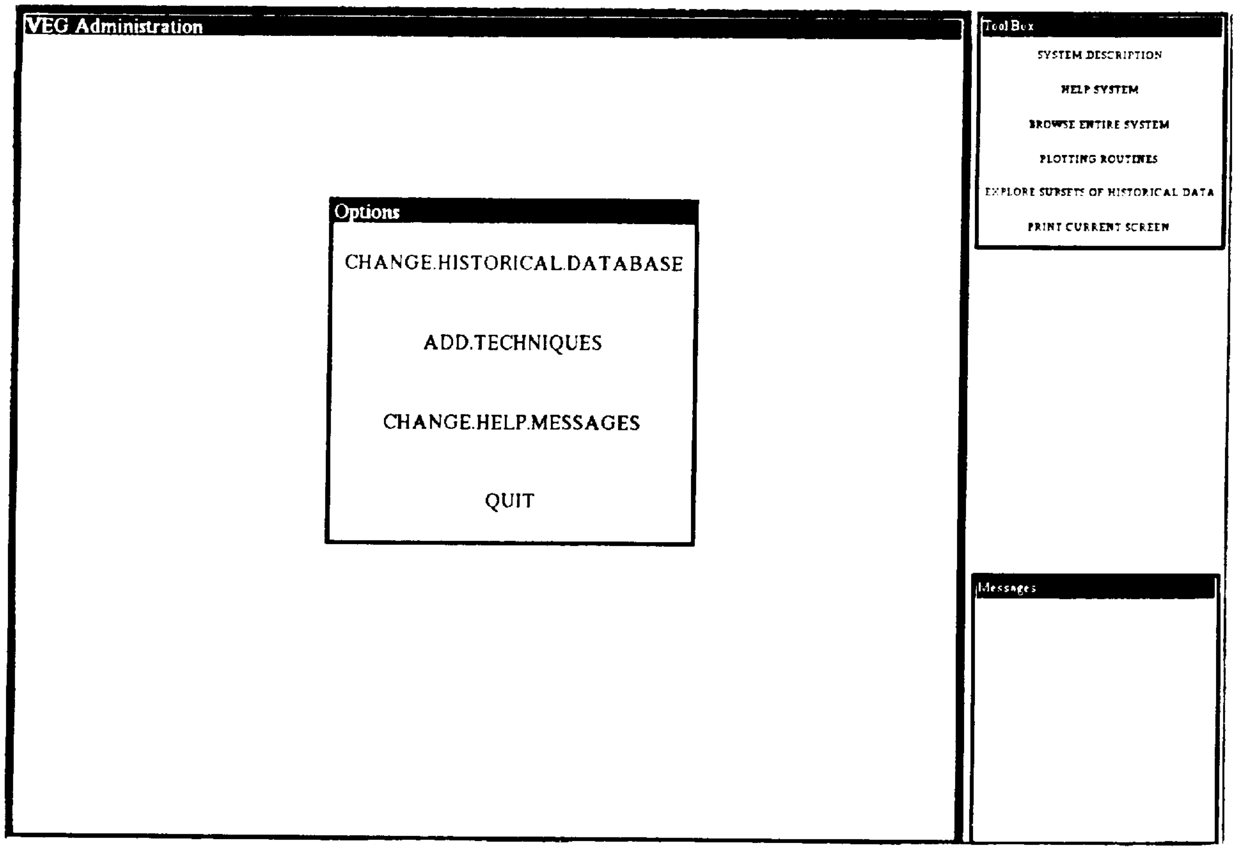
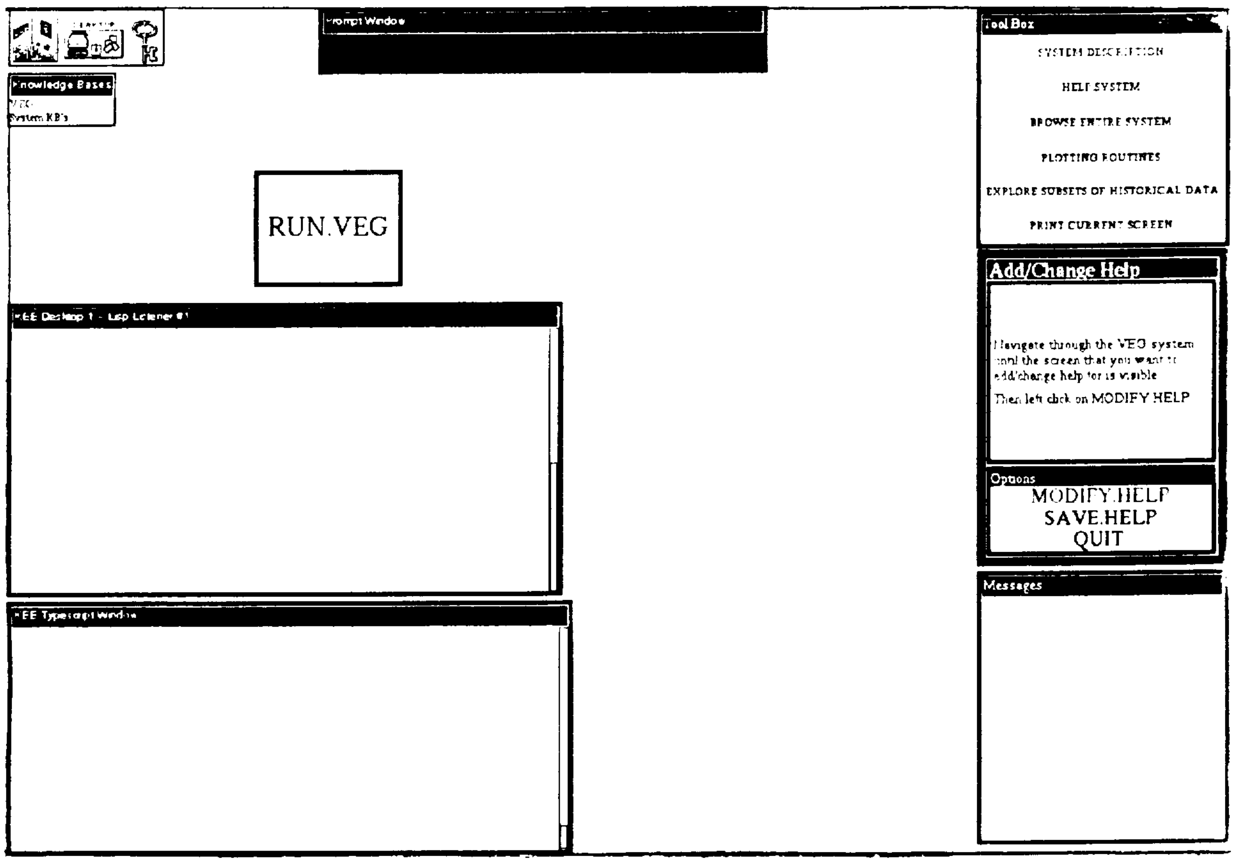


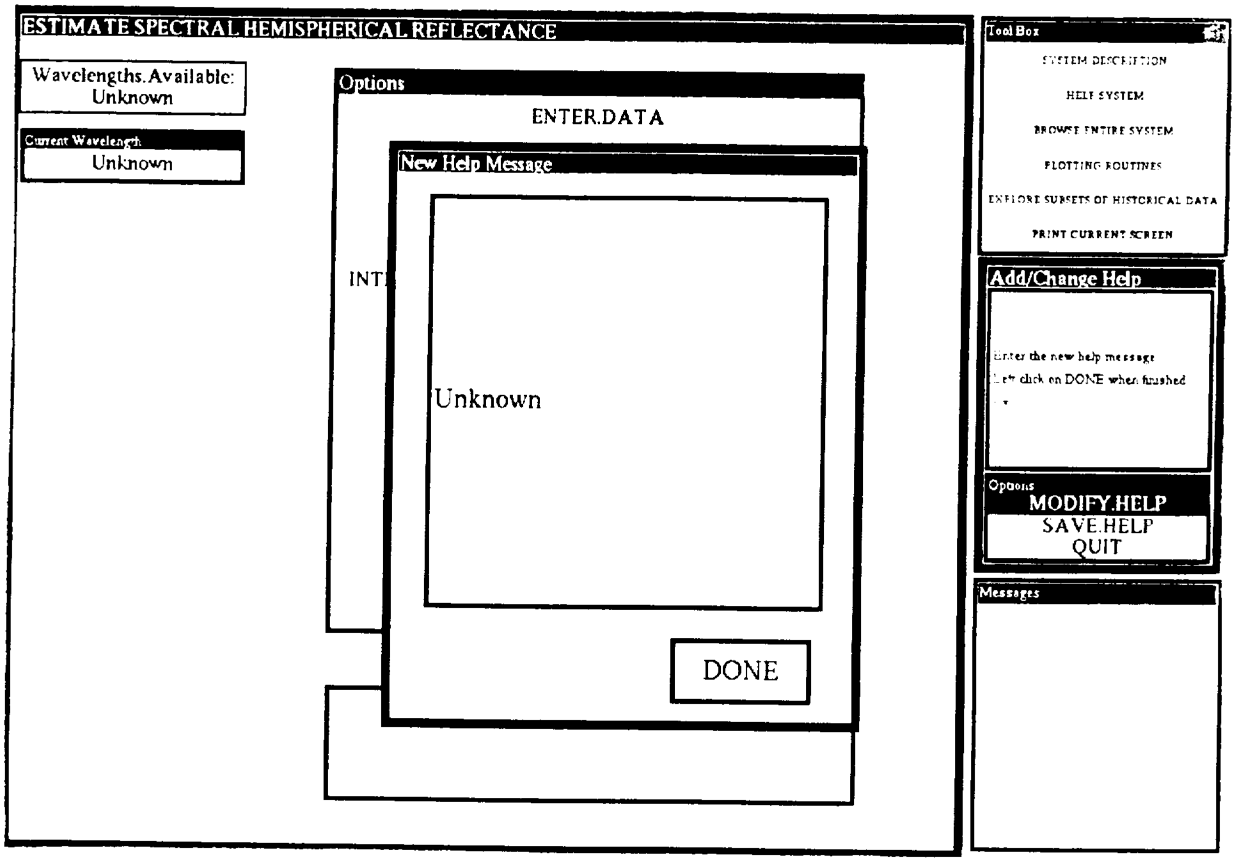
Figure 3-1
VEG Administration Screen with CHANGE.HELP.MESSAGES



Navigate through the VEG system until the screen that you want to add/change help for is visible

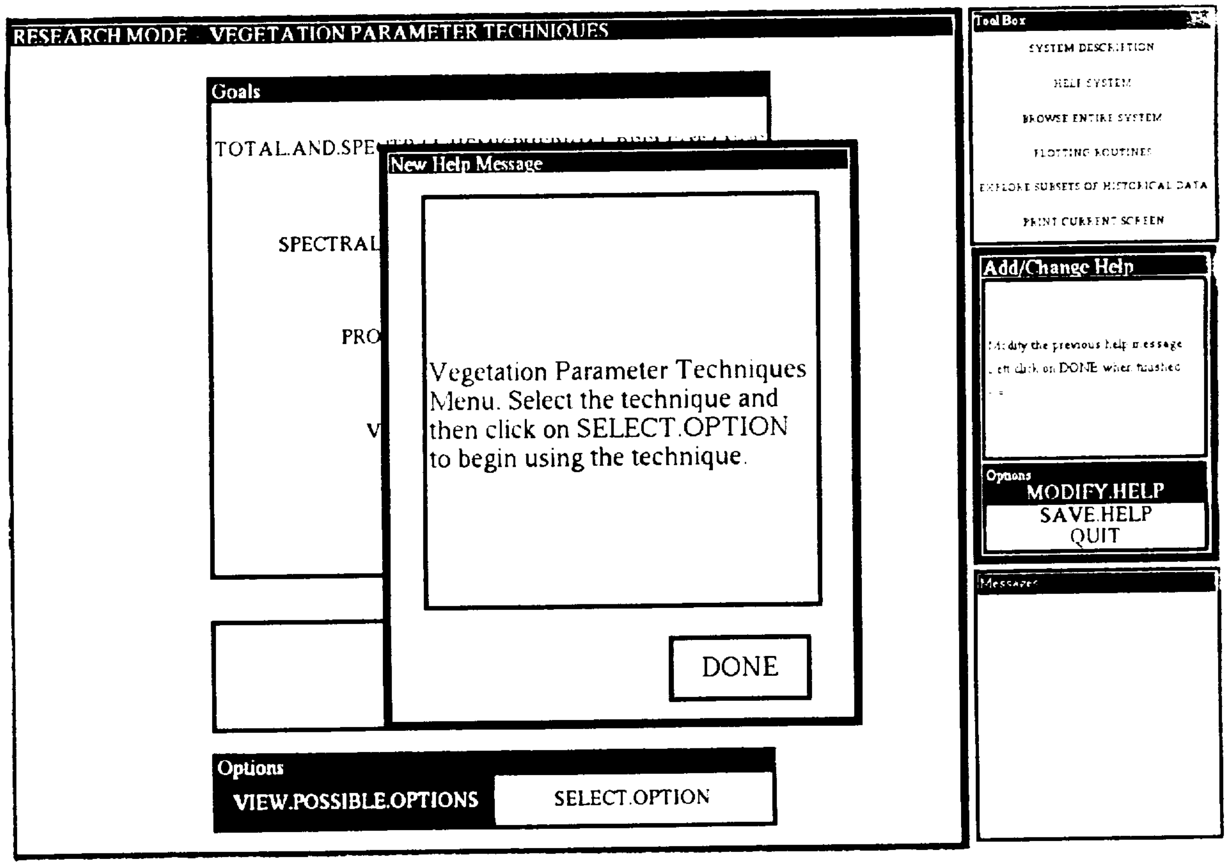
Then left click on MODIFY HELP

Figure 3-2
Add/Change Help Screen with Options



Enter the new help message
Left click on DONE when finished

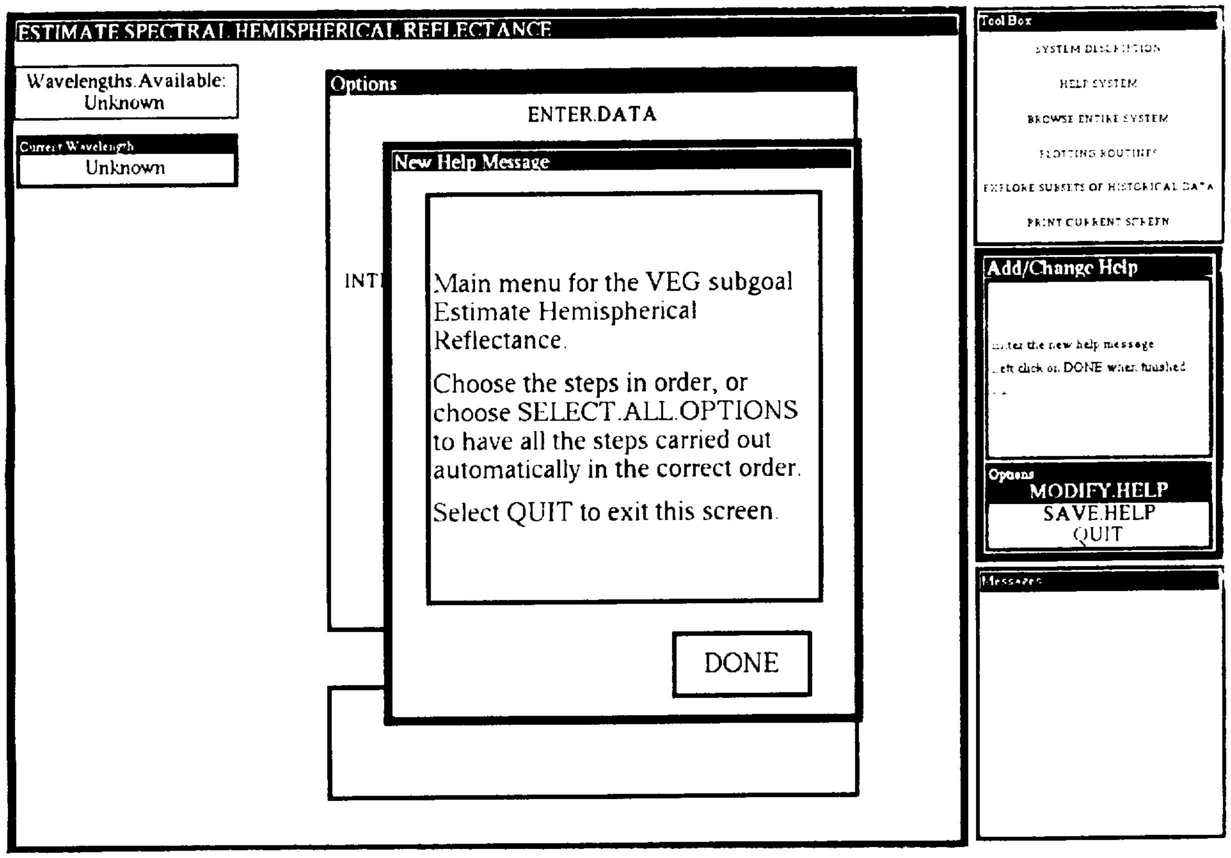
Figure 3-3
New Help Message Screen before a Message is Added



Vegetation Parameter Techniques Menu. Select the technique and then click on SELECT.OPTION to begin using the technique.

Modify the previous help message. Then click on DONE when finished.

Figure 3-4
New Help Message Screen Before an Existing Message is Modified



Main menu for the VEG subgoal Estimate Hemispherical Reflectance.

Choose the steps in order, or choose SELECT.ALL.OPTIONS to have all the steps carried out automatically in the correct order.

Select QUIT to exit this screen.

Enter the new help message
Left click on DONE when finished

Figure 3-5
New Help Message Screen with New Message

The Help System works equally well if multiple knowledge bases (application components) are loaded. The Help System automatically determines the knowledge base with which a window is associated. When a save is initiated, the help message is saved in a file using a name that includes the knowledge base name. For example, if a help message was added to a Learning System window, then an ASCII file named "help-messages-learn" would be created (if it did not already exist) and the message would be saved in the new file or appended to the existing file. Currently, help files are saved as text files. The help files can be inspected outside of the VEG system. It is possible to modify the help messages in the files using an editor. However, this is not recommended. Any editing should be done with great care. The help messages are stored in the files together with the object identifiers for the screens to which the messages apply. Changing the organization of the help message files would cause errors which would prevent the Help System being loaded. When help messages are added or modified and then saved using the VEG interface, they are automatically saved in the correct format.

It should be noted that with minimal effort, the new help message entry window could be replaced by an editor window tied to the editor favored by the user. This would allow more extensive editing capabilities than are presently available.

SECTION 4.0

TESTING AND RESULTS

The following capabilities of the VEG Help System and the Add/Modify Help option were tested:

- Test 1 - Navigate through an empty Help System.
- Test 2 - Add help messages to windows. Save help messages and inspect saved files.
- Test 3 - Navigate through the Help System and read previously saved help messages.
- Test 4 - Modify help messages in existing windows.
- Test 5 - Add help messages to multiple knowledge bases.

All the tests were successful, showing that the system was working correctly. The tests are described in detail in this section.

4.1 TEST 1

Test 1 simply navigated through the Help System before any help messages were added. This was done with one and two knowledge bases active at the same time. Application windows as well as KEE and OPENWIN application icons were tested. After left clicking on HELP.SYSTEM in the Tool Box menu, the message "Click on the window you need help with" appeared. The cursor became a cross shape. Placing the cross over the window of interest and then left clicking produced the message, "Sorry, no help currently available for this window" if the window was an application window, or "Not a VEG window" if the window was a KEE or OPENWIN window. No errors occurred. When an application window was clicked regardless of the knowledge base, the system behaved as expected. This test demonstrated the basic functionality of the HELP system for navigating through an application.

4.2 TEST 2

Test 2 activated the ADMINISTRATION window and then activated the CHANGE.HELP.MESSAGES window. The tester then navigated to the Automatic Mode Screen and clicked left on the MODIFY.HELP option in the Add/Change Help Screen. The cursor changed to a cross and the tester left clicked on the Automatic Mode Screen to indicate that help for this screen was to be added. A New Help Message window appeared and a new message was added. DONE was left clicked in the New Help Message window to indicate the message was complete. The procedure was repeated using several different windows at different levels in the VEG application. Each time a new message was added, MODIFY.HELP was left clicked to reiterate the process. Once messages had been added to different screens, the Help System was again invoked and the help messages successfully displayed. Finally, the SAVE.HELP option was activated to save the help messages that had been entered so far. A file called "*help-messages-veg*" was created and contained the help messages that had been entered.

4.3 TEST 3

The VEG system was exited and re-loaded. The HELP.SYSTEM option was selected from the Tool Box Menu. The Help System was loaded and the messages that had been added in Test 2 were successfully accessed through the Help System.

4.4 TEST 4

In Test 4, the ADMINISTRATION window was again activated, and the CHANGE.HELP.MESSAGES window activated. The tester then navigated to a window, for which a help message existed, and clicked left on the MODIFY.HELP option. The cursor changed to a cross and the tester left clicked on the window whose help message was to be modified. A New Help Message window appeared and the current message was displayed. The message was changed by writing a new message. DONE was left clicked in the New Help Message window to indicate the modification was complete. This was repeated using several different windows at different levels in the VEG application. Each time a message was modified, MODIFY.HELP was left clicked to reiterate the process. Once this process was completed, the Help System was again tested and the help messages successfully displayed. Finally, the SAVE.HELP option was activated to append the modified help messages to the existing file. The file was inspected and properly saved. The VEG system was exited and reloaded. The Help System was loaded and the messages were successfully accessed through the Help System.

4.5 TEST 5

Test 5 replicated the elements of tests 1 through 4 with multiple knowledge bases (modules) loaded in the VEG system. In addition to the VEG core, the AZIMUTH PLOT, POLAR PLOT and LEARN knowledge bases were loaded. Then tests 1 through 4 were repeated using windows from the four components. The system again performed as expected. The VEG help file was properly updated and new help files called "*help-messages-azimuth*," "*help-messages-polar*," and "*help-messages-learn*" were created.

4.6 RESULTS

The test suite demonstrated the ability of the Help System to provide the range of behavior expected of the Help System prototype.

SECTION 5.0

CONCLUSIONS

The prototype Help System provides an interactive tool for adding help support to the VEG system. It was designed to enable the scientist to control and shape the help facility without bothering with the details of implementation. The Help System provides both a help system and a tool for developing new help messages and modifying existing help messages. File management and object management issues are transparent to the user. Currently, the editing facilities for message modification are minimal. The Help System was designed so that it would be simple to replace the current editing window with whatever editor (emacs, textedit, vi) the user might favor for adding or modifying messages.

Since the Help System may not be needed by an experienced user, it was configured so that it is loaded only when the user initially clicks on the Help System option in the Tool Box Menu. This minimizes the overhead for the VEG environment.

APPENDIX A

LISP CODE FOR THE PROTOTYPE HELP SYSTEM

```
;;; veg-methods7.lisp
;;;
;;; Created April 27, 1993
;;; Last Modified July 22, 1993

(in-package 'kee)

(defun start-help-system ()
"Starts the help system."
  (remove.all.values 'help.system 'options)
  (cond ((get.value 'help.system 'help.loaded)
         (put.value 'help.system 'message
           "Click on the window that you need help with")
         (unitmsg 'viewport-help.system.1 'open-panel!))
        (t (put.value 'help.system 'message "Loading help ........")
           (unitmsg 'viewport-help.system.1 'open-panel!)
           (load-help)
           (put.value 'help.system 'message
             "Click on the window that you need help with")))
  (show-text))

(defun load-help()
"Call the function to load help messages from the appropriate files into the
help slots of viewports."
  (add-help-slots-to-viewports)
  (add-help-messages "help-messages-veg")
  (when (kb.exists.p 'learn)
    (add-help-messages "help-messages-learn"))
  (when (kb.exists.p 'azimuthplot)
    (add-help-messages "help-messages-azimuth"))
  (when (kb.exists.p 'polarplot)
    (add-help-messages "help-messages-polar"))
  (put.value 'help.system 'help.loaded t))

(defun add-help-messages (file)
"Load help messages from a file into the help slots of viewports."
  (with-open-file (str file :direction :input :if-does-not-exist :nil)
    (when str
      (do ((win (read-file str)(read-file str)))
          ((null win) (values)) ; End of file
        (if (unit.exists.p win) ; Window is found
            (put.value win 'help (read-file str)) ; Read & store message
            (read-file str))))))   ; Read past unused message

(defun add-help-slots-to-viewports()
"Modify the viewport parent units in the ACTIVEIMAGES knowledge base in
preparation for storing the help messages in the slots of each viewport unit."
  (create.slot 'ai3-kb-viewports 'help 'member "")
  (add.value 'ai3-kb-viewports 'local.compact.unit.slotnames 'help)
  (create.slot 'ai3-unit-viewports 'help 'member "")
  (add.value 'ai3-unit-viewports 'local.compact.unit.slotnames 'help)
  (create.slot 'ai3-slot-viewports 'help 'member "")
  (add.value 'ai3-slot-viewports 'local.compact.unit.slotnames 'help))
```

```
(defun get-more-help ()
"Prompts the user to select the screen for additional help."
  (put.value 'help.system 'message
     "Click on the window that you need help with")
  (show-text))

(defun mouse-top-window ()
"This function allows the user to mouse directly on the window that represents
the object he needs help with."
   (let ((pos (get-position)))
     (window-stream-under-position pos)))

(defun unit-from-stream (window)
"Returns the name of the viewport corresponding to the window."
   (getf (kwin-plist window) 'viewport))

(defun show-text ()
"Returns the help message from the moused window."
   (let ((unit (unit-from-stream (mouse-top-window))))
     (put.value 'help.system 'message
           (if (not unit)
              "Not a VEG window"
              (let ((help (get.value unit 'help)))
               (if help
                     help
                     "Sorry, no help currently available for this window"))))))

;;; -----------------------------------------------------------------------
;;; Methods for Changing or Adding Help Messages
;;; -----------------------------------------------------------------------

(defun open-change-help-menu ()
"Opens the top screen for changing or adding help messages."
  (remove.all.values 'add.help 'options)
  (cond ((get.value 'help.system 'help.loaded)
          (put.value 'add.help 'message
"Navigate through the VEG system until the screen that you want to add/change help for is visible.
Then left click on MODIFY.HELP.")
          (unitmsg 'viewport-add.help.2 'open-panel!))
         (t (put.value 'add.help 'message "Loading help ........")
            (unitmsg 'viewport-add.help.2 'open-panel!)
            (load-help)
            (put.value 'add.help 'message
"Navigate through the VEG system until the screen that you want to add/change help for is visible.
Then left click on MODIFY.HELP.")))
  (remove.all.values 'workbench 'run.veg)
  (unitmsg 'viewport-run.veg-of-workbench.1 'open-panel!))

(defun modify-help ()
  (put.value 'add.help 'message
     "Left click on the window that you want to change the help on")
 (add-help))
```

```
(defun save-help ()
"Saves the modified help messages to the help file."
  (with-open-file (strv "help-messages-veg" :direction :output
                                    :if-does-not-exist :create :if-exists :supersede)
    (with-open-file (strl "help-messages-learn" :direction :output
                                    :if-does-not-exist :create :if-exists :supersede)
      (with-open-file (stra "help-messages-azimuth" :direction :output
                                    :if-does-not-exist :create :if-exists :supersede)
          (with-open-file (strp "help-messages-polar" :direction :output
                                    :if-does-not-exist :create :if-exists :supersede)
            (dolist (uni (unit.children 'ai3-kb-viewports 'member))
              (let ((mes (get.value uni 'help)))
                (when mes
                     (let ((str (get-correct-stream uni strv strl stra strp)))
                       (princ uni str)
                       (princ " \"" str)
                       (princ mes str)
                       (princ "\" " str)))))
            (dolist (uni (unit.children 'ai3-unit-viewports 'member))
              (let ((mes (get.value uni 'help)))
                (when mes
                     (let ((str (get-correct-stream uni strv strl stra strp)))
                       (princ uni str)
                       (princ " \"" str)
                       (princ mes str)
                       (princ "\" " str)))))
            (dolist (uni (unit.children 'ai3-slot-viewports 'member))
              (let ((mes (get.value uni 'help)))
                (when mes
                     (let ((str (get-correct-stream uni strv strl stra strp)))
                       (princ uni str)
                       (princ " \"" str)
                       (princ mes str)
                       (princ "\" " str))))))))))

(defun get-correct-stream (uni strv strl stra strp)
"Returns the correct stream for the file holding the help messages for the
knowledge base containing the viewport."
  (case (unit.kbname uni)
    (VEG strv)
    (LEARN strl)
    (AZIMUTHPLOT stra)
    (POLARPLOT strp)))

(defun add-help ()
"Adds help for a viewport."
  (let ((unit (unit-from-stream (mouse-top-window))))
    (if unit
          (get-new-help unit)
          (my-documentation-print "Not a VEG window - help cannot be stored"))))
```

```
(defun get-new-help (unit)
"Prompts the user to enter the new help message and then accepts the new
message."
  (let ((old-mes (get.value unit 'help)))
    (put.value 'add.help 'unit unit)
   (cond ((or (null old-mes)
                 (equal old-mes "")(equal old-mes " ")(equal old-mes "  "))
           (remove.all.values 'add.help 'help.message)
           (put.value 'add.help 'message
"Enter the new help message.  Left click on DONE when finished.  <="))
          (t (put.value 'add.help 'help.message old-mes)
             (put.value 'add.help 'message
 "Modify the previous help message.  Left click on DONE when finished  <=")))
    (unitmsg 'viewport-add.help.4 'open-panel!)))

(defun make-one-long-string (list-of-strings)
"Concatenates a list of strings into one long string."
  (make-one-long-string-aux "" list-of-strings))

(defun make-one-long-string-aux (result remaining-strings)
 (if (null remaining-strings)
    result
    (make-one-long-string-aux (string-append result " "
                                        (first remaining-strings))
                              (rest remaining-strings))))

(defun wipe-out-help ()
"Removes all the help slots and help messages from all loaded knowledge bases."
  (delete.slot 'ai3-kb-viewports 'help)
  (remove.value 'ai3-kb-viewports 'local.compact.unit.slotnames 'help)
  (delete.slot 'ai3-unit-viewports 'help)
  (remove.value 'ai3-unit-viewports 'local.compact.unit.slotnames 'help)
  (delete.slot 'ai3-slot-viewports 'help)
  (remove.value 'ai3-slot-viewports 'local.compact.unit.slotnames 'help)
  (put.value 'help.system 'help.loaded nil))
```

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle An Expert System Shell for Inferring Vegetation Characteristics - Final Report 1993		5. Report Date November 1993 6. Performing Organization Code
7. Author(s) P. Ann Harrison Patrick R. Harrison		8. Performing Organization Report No. C931033-U-2R00 10. Work Unit No. 462-61-14
9. Performing Organization Name and Address JJM Systems, Inc. 1225 Jefferson Davis Hwy, Suite 412 Arlington, VA 22202		11. Contract or Grant No. NAS5-30127
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 NASA/Goddard Space Flight Center Greenbelt, MD 20771		13. Type of Report and Period Covered Final Report March - November 1993 14. Sponsoring Agency Code

15. Supplementary Notes

The Lisp and KEE code for this work is available on a Sun Cartridge Tape.

16. Abstract

The NASA VEGetation Workbench (VEG) is a knowledge based system that infers vegetation characteristics from reflectance data. The report describes the extensions that have been made to VEG in 1993. The historical cover type database has been removed from VEG and stored as a series of flat files that are external to VEG. An interface to the files has been provided. The framework and interface for two new VEG subgoals that estimate the atmospheric effect on reflectance data have been built. A new interface that allows the scientist to add techniques to VEG without assistance from the developer has been designed and implemented. A prototype Help System that allows the user to get more information about each screen in the VEG interface has been added to VEG.

17. Key Words (Suggested by Author(s)) EXPERT SYSTEM, ARTIFICIAL INTELLIGENCE, REMOTE SENSING, LEARNING, DISCRIMINATION		18. Distribution Statement UNCLASSIFIED - UNLIMITED	
19. Security Classif. (of this report) UNCLASSIFIED	20. Security Classif. (of this page) UNCLASSIFIED	21. No. of pages 142	22. Price

NASA FORM 1626 OCT 86