An Intelligent Assistant for Creating Data Flow Visualization Networks*

Peter Kochevar
Digital Equipment Corporation
La Jolla, California

Zahid Ahmed
Len Wanger
San Diego Supercomputer Center
La Jolla, California

Abstract

Non-visualization experts, including most scientists, find visualization systems like AVS too difficult to use. One approach to assisting these end-users in doing interactive visualization is to embed the knowledge of visualization experts into an intelligent system. A prototype, called *Tecate*, of such a system has been developed as part of the Sequoia 2000 Project. In this system, a Planner makes use of expert knowledge stored in a Knowledge Base to create data-flow visualization programs. The Planner takes as input a description of the data to be visualized and an indication of the data analysis goals of an end-user. From this information, an AVS network script is produced that when executed, builds an appropriate visualization of the indicated data set. The networks so produced make use of both a restricted set of standard AVS modules and a collection of custom ones which operate on data structured as fiber bundles.

Introduction

The eternal problem in developing visualization systems is to provide those that end-users will actually use. For applications that are specific to a particular discipline and a particular data set format, useful systems abound. With these systems, end-users simply tweak parameters through a graphical interface and "canned" visualizations appear on the screen.

But, such systems are constraining to end-users who frequently would like to explore, compare, and contrast a variety of data sets using many different visualization techniques. Current "general" visualization systems, of which AVS is a primary example, provide this capability through a visual programming interface. Unfortunately, the visualization primitives that form the basis of the visual language are couched in the terminology of the visualization expert not in that of the end-user. Thus many end-users have trouble constructing salient visualizations. In addition, it is tedious and time-consuming to build and alter programs each time a new data set format is encountered or an old one is to be visualized in a new way. Finally, the user-interfaces to most visualization systems are overly complex in that they provide too many options to a user even when most of those options do not apply in a given circumstance.

For these reasons, many scientists and policy-makers employ visualization experts to use a system like AVS to create a limited set of pre-determined visualizations for them. This situation is frustrating to most end-users because acting through intermediaries to make changes beyond the mere alteration of parameters is far too time-consuming thus curtailing exploration. One solution to this problem is to embed the knowledge of the visualization expert into an intelligent system, a strategy that has been tried by others [5, 8, 11, 13]. Such a system can be thought of as an "assistant" that would be responsible for actually creating visualizations based on high-level cues from an end-user. These cues can be delivered via an interface that has domain-specific elements drawn from the lexicon of an end-user.

*This paper was presented at the 1994 AVS Users Conference held in Boston, Massachusetts May 2-4, 1994.

The balance of this paper describes a simple prototype for an intelligent visualization system called *Tecate* that has been developed as part of the Sequoia 2000 Project [6]. The heart of the system is a *Planner* which structures visualizations of data selected by an end-user. The inputs to the Planner consist of a description of the data that is to be visualized and a task specification outlining the data analysis goals of an end-user. The task specification must be supplied by an end-user while the data descriptions are provided by a data source such as a database management system.

From a data description and task specification, the Planner produces an AVS network script that when executed, creates a visualization of a selected data set. In fashioning the script, the Planner makes use of a collection of facts and rules that reside in a *Knowledge Base* that is managed by a *Knowledge Representation System*. These rules dictate how data can be represented graphically using a subset of the standard AVS module set augmented with a collection of custom-designed modules.

The Data Model, Calculus, and Exchange Format

All data that is to be visualized by Tecate's intelligent visualization system is assumed to be structured according to a data model based on the mathematical notion of *fiber bundles* [3, 4, 7]. A fiber bundle is determined by the specification of two spaces, a *base space* and a *fiber space*, in which elements from the fiber space are assigned to each element in the base space. As such, fiber bundles generalize the notion of the graph of a function, the base space being analogous to the independent function variables while the fiber space corresponds to the function's dependent variables. The assignment of a fiber space element to each base space element is represented by the function itself whereas the entire fiber bundle is the space where the function is graphed.

Although convenient to associate independent and dependent variables of functions with base and fiber spaces, the definition of fiber bundles makes no assumptions about the structure of either type of space. In fact, base and fiber spaces can be arbitrary topological spaces which may or may not have coordinate systems defined on them. A fiber bundle whose base space consists of an unordered set of elements and whose fiber space is the Cartesian product of the field domains of a record, is exactly a *relation*, the theoretical underpinning of relational database management systems. By using more complex spaces, topological information can be encoded thus allowing the definition of arbitrary polyhedra, say, which can be used to represent such objects as the topography of the Earth or streamlines corresponding to the winds in the atmosphere. Finally, fiber spaces can themselves be collections of fiber bundles thus allowing the representation of hierarchical structures.

An Implementation of Fiber Bundles

Tecate currently supports four classes of fiber bundles: *sets*, *index-sets*, *point-sets*, and *index-point-sets*. These classes reflect the character of the base spaces underlying each type of fiber bundle. A *set* is simply a fiber bundle whose base space is an unordered collection of elements, in other words, a set is a relation in the sense of relational database management systems. A *point-set* is a set in which each base space element resides in some coordinate system but there is not necessarily any topological relationship among the elements. In general, point-sets are used to represent scattered data, and the coordinate space where the data points reside is called the *embedding space*.

An *index-set* is a set in which each base space element is identified with a *multi-index* consisting of a vector of integer values that resides in an *index space*. Index-sets are used primarily to represent multi-dimensional arrays. Finally, an *index-point-set* is a set that combines features of both index-sets and point-sets. Base space elements of an index-point-set reside in some coordinate system like point-sets but each element is also identified with a multi-index as with index-sets. Conceptually, index-point-sets are rectangular arrays of points that have been warped in an embedding space. The index space of an index-point-set is a discrete parameter space in which multi-indices are mapped to points in an associated embedding space.

For ease of implementation, the Tecate prototype assumes that the fiber space for any fiber bundle class is the Cartesian product of the real number line. In other words, each base space element is associated with a tuple of floating point values. Furthermore, the embedding spaces of point-sets and index-point-sets are also assumed to be Cartesian products of the reals. In future versions of Tecate, these restrictions will be lifted.

Using AVS

The Tecate prototype deals with two representations of fiber bundles: an in-memory format and an “on-the-wire” encoding. The in-memory format is that of the AVS field type with a few non-standard interpretations of some of the components of the field structure, for instance, the name of a fiber bundle class is stored in the “units” slot. Sets and point-sets are mapped into 1-dimensional uniform and irregular fields respectively by imposing an arbitrary ordering on the base space elements. Index-sets and index-point-sets are mapped into n-dimensional uniform and irregular fields respectively where n is the dimension of the index space.

The “on-the-wire” encoding is a self-describing file format suitable for transporting fiber bundles across a network. This format consists of an ASCII header followed by a binary encoding of the fiber space elements, and when appropriate, a binary encoding of embedding space coordinates. The header is a specification written in a *Data Description Language* that is defined in [10]. Within this language, the particular class of fiber bundle is indicated as well as the structure, extent, and number of base and fiber space domains. The data specifications also indicate the number of elements present in the base space of a fiber bundle and whether or not the base elements are indexable using multi-indices defined in a specified index space.

A number of user-defined AVS modules have been created that operate on fiber bundles represented in either of the two representations. **Reader** and **Writer** modules are provided to transform fiber bundles represented in the “on-the-wire” encoding into the in-memory format, and vice versa. In addition, a small collection of AVS modules transform fiber bundles in one in-memory format into those of another. For instance, the **Promoter** module takes specified fiber space domains and makes them domains in a fiber bundle’s embedding space. In doing so, **Promoter** converts set and index-set fiber bundles into point-sets and index-point-sets respectively while point-set and index-point-set bundles remain as such albeit with more general embedding spaces. These transformational modules collectively represent an implementation of a “fiber bundle calculus” akin to the relational calculus that underlies the query languages of relational database management systems.

The Visualization Planner

Tecate’s Visualization Planner is a CLOS [9] program which utilizes the inferencing capabilities of a knowledge representation system. The Planner produces an AVS data flow network that when executed, creates a visualization of a given data set. The Planner takes as input a *data descriptor* and a *task specification*.

The data descriptor is a description of a fiber bundle data set written in the Data Definition Language mentioned in the last section. This descriptor is the header from the “on-the-wire” encoding for a given data set. The Planner uses the descriptor to gain an understanding of a data set’s structure and semantics. Semantic information is primarily conveyed through pre-defined domain names for which visualization rules exist in the knowledge base. For example, the Planner might choose to use a vegetative index color map for elevation data, whereas a color map representing black-body radiation might be used to visualize temperature data.

The task specification, provided by the end-user, is a textual description of the end-user’s data analysis goal for a selected data set. Since any data set can be visualized in a myriad of ways, the art of creating a visualization is to determine which of these possibilities will be most effective for satisfying the stated task. For example, a visualization should not map a data variable to color if precise measurements are intended since people generally are less accurate in determining quantitative values from color than spatial differences, say [1].

Task Specification Language

Tecate's task specification language is designed to express tasks in a way that a scientist might describe a data analysis goal to a colleague. With this in mind, the task language is intended to be natural-like and to use the lexicon of a scientist. Each task specification is composed of a *task operator*, and a *domains-of-interest (DOI)* list which indicates the fiber bundle domains upon which to operate. The Tecate task language currently supports the following task operators which are only a small set of those that are possible:

Select: The end-user wishes to utilize an individual instance of a domain from a data set. This operator indicates that a visualization should be formulated such that individual data elements remain autonomous and easily distinguishable. A scatter plot is an example of such a visual paradigm.

Search: The end-user wishes to see overall trends in a data set as a whole as opposed to individual values within the set. The resulting visualization favors visual paradigms which show high-level features of the set, such as minima and maxima, over the autonomy of individual elements. Surface fitting methods, such as iso-surfacing, are appropriate for many search tasks.

The DOI list indicates which subset of domain variables are of interest within a data set. The ordering in the list is used as an indication of the importance an end-user assigns to the domains. For example, the task, "**search (temperature, water-content)**," specifies that temperature is more important than water-content. Since most people have a greater visual acuity for determining differences in position than color differences, the Planner would be more apt to map temperature to the position of some glyph while using a water-content value to determine its color.

Planning Algorithm

The process of planning a visualization makes use of a context-sensitive grammar that is depicted graphically in Figure 1. The Planner constructs a "sentence" within the language defined by the grammar that represents a data-flow network for a particular visualization. These sentences are created using the following algorithm:

1. **Instantiate a blank network.** An internal representation for a network structure is created. This internal data structure, which will eventually hold a complete specification of the data-flow network, has four phases: input, global transformation, glyph generation, and output (see Figure 1A).

2. **Determine a visualization technique.** Based on rules in the knowledge base, a *generalized visualization technique* is selected using information primarily about the type, structure, and extent of a fiber bundle's base space. A generalized visualization technique specifies the dimensionality and structure of the visualization scene space without specifying the geometric primitives used in its construction.

3. **Fill in the input and output phases.** The input phase consists of a single module, which fetches a data set and outputs it as a fiber bundle encoded in the in-memory format. The input module that is used depends on whether a data set is stored in a file system or a database management system. The output phase consists of a single module which inputs and displays AVS geometry primitives. Currently the AVS Geometry Viewer is the only available output module.

4. **Add Global transformation modules.** The *Global Transformation List* contains a serial list of modules which transform the fiber bundle data into a form suitable for the generalized visualization technique to be performed. Modules are added as necessary to convert the form of the fiber bundle coming from the input phase to match the form required by the technique. For example, a scattered data set may need to be discretized into a uniform grid before a volume visualization technique can be performed. Other global transformations include the promotion of fiber domains into embedding space domains, imposing order on unordered sets, and various projection operations.

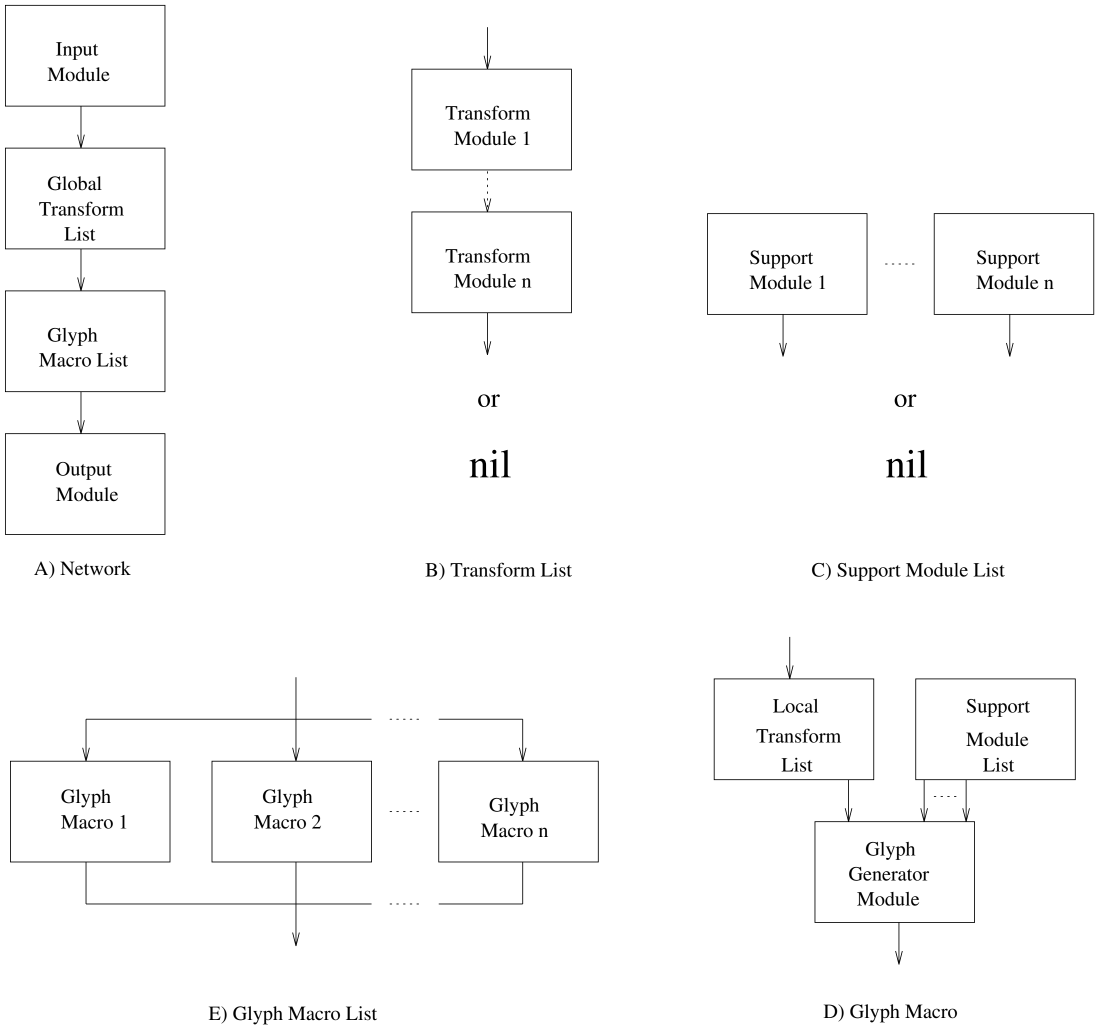

Figure 1: A graphical representation of the visualization network grammar used in the Tecate system. Each data-flow network contains four phases (A): input, global transformation, glyph generation, and output. The input and output phases contain a single reader and writer module respectively. (B) shows the expansion of a Global Transformation List which may be nil, and (D) shows the expansion of a Glyph Macro List. (E) shows the composition of each Glyph Macro in this list which contains a Glyph Generator Module, a Local Transform List (B), and a Support Module List (C). Both the Local Transform and Support Module Lists may be nil. Although not shown in the figure, flow connections between the modules are also specified in the grammar.

5. **Add glyph macros.** A *Glyph Macro* structure is created for each group of geometric primitives used in a visualization. Glyph Macros are expanded to include modules that generate geometric primitives representing both data points, and “informational” components which clarify visualizations, such as axes, labels, and legends.

   Each Glyph Macro contains three parts:

   - The *Glyph Generator Module* creates AVS geometry primitives from fiber bundle data. The parameters of this module determine the mapping of the domains-of-interest to the geometric, appearance, and behavioral attributes of the primitive. The particular primitives used are determined by the knowledge representation system based on factors such as the generalized technique, and the task and data semantics.
   - A serial chain of local transformation modules is added, as necessary, based on the input requirements of the individual Glyph Generation Module. For example, a data domain for a particular module may need to be normalized or mapped into an integer range for color mapping, or a filter may need to be added to remove data elements not to be represented by the glyph.
   - A list of *Support Modules* is added to each glyph generation module as needed. Support modules provide supplementary data, such as color maps, that are needed to produce geometric primitives that do not directly represent items within a given data set.

6. **Add flow connections:** Connections are added to string together components of a network in the following manner:

   - Add a connection from the output of the Input Module to the input of the first transform of the Global Transform List.
   - Add connections to daisy-chain all of the transforms in the Global Transformation List.
   - For each Glyph Macro:
     – Add a connection from the output of the last module in the Global Transformation List to the input of the first module in the Local Transformation List.
     – Add connections to daisy chain all of the modules in the Local Transformation List.
     – Add a connection from the output of the last module of the Local Transformation List to the fiber bundle input of a Glyph Generator Module.
     – Add connections from the output of each module in the Support Module List to the appropriate input of a Glyph Generator Module.
     – Add a connection from the last module of each Glyph Macro to the input of the Output Module.

7. **Create an AVS CLI script:** The final stage of the planning algorithm converts the internal data structure for a network into an AVS CLI script. Each module is represented by a CLI **module** command, each connection by a **port_connect** command, and each module parameter by a **parm_set** command. A **manipulator** command is added for each parameter to be modified by the user. Finally a pre-determined set of commands is added to initialize such display module parameters as lights and the viewing matrix.

The Knowledge Base

The Tecate knowledge base consists of a collection of facts, classifications, and rules regarding the construction of data-flow networks to visualize Earth science data sets. The knowledge base is managed by a *knowledge representation system (KRS)* which is a kind of database management system specialized to efficiently store and query for knowledge data. In addition, a KRS provides the ability to infer new information based on existing knowledge and it is this inferencing capability that is used to answer questions posed to the KRS by the Planner.

Knowledge Basics

The Tecate knowledge base is implemented using the CLASSIC KRS [12, 2]. In CLASSIC, a knowledge base is composed of the following classes of knowledge:

Concepts. Concepts represent classes of objects in the knowledge base. Each concept possesses a series of descriptive clauses describing the qualities of the object. A hierarchy representing a taxonomy of related concepts can be formed to express the relationships between various objects. For example, the Tecate knowledge base contains the concept **MODULE-THING** to represent data flow modules. **MODULE-THING**s are specified to have a set of input ports, output ports, and parameters. Reader modules are represented by the concept **INPUT-MODULE-THING** which is a sub-concept of **MODULE-THING** with zero input ports and a single output port of type **FIBER-BUNDLE-PORT**.

Roles. Roles represent properties of concepts that have associated values. Clauses can be added to restrict the values that can fill a role. For example, the **GLOBAL-TRANSFORM-LIST** role in a **NETWORK-THING** is restricted to be a list of zero or more **TRANSFORM-MODULE-THING**s.

Rules. Rules are essentially if-then statements that are used to invoke inferences. Each rule has a set of antecedent clauses which, when satisfied, cause the rule to be fired–a process called *forward chaining*. The action to be performed by the rule is specified in a set of consequential clauses. Rules can be used for a number of functions including the filling or restricting of roles, and the classification of individuals. For example, a rule to help determine the generalized visualization technique to use might say:

> if data-set type is **SET-THING**, then set the **GENERALIZED-VISUALIZATION-TECHNIQUE** to **SCATTER-PLOT-THING**.

Individuals. Individuals are instances of concepts. For example, the Tecate knowledge base defines **FILE-READER-MODULE** and **DBMS-READER-MODULE** as individuals of type **INPUT-MODULE-THING**.

Knowledge Base Contents

The planning process can then be thought of as defining an individual for a data-flow network, a **NETWORK-THING**, and allowing the KRS to fill the network's roles using forward chaining inferences. These inferences are based on role restrictions and the set of rules which together are classified into the following categories:

Data Model and Characteristics. Knowledge regarding the fiber bundle data model and its calculus. This knowledge includes concepts representing the various fiber bundle classes, rules regarding the visualization techniques applicable to those classes, rules for performing fiber bundle transformations, and individuals representing modules to perform those transformations. For example, the rule that determines when a scatter plot should be used fires when a fiber bundle individual classified as a **SET-THING** is detected. The consequent of this rule sets the network's generalized visualization technique role to **SCATTER-PLOT-THING**, and sets the **NUM-PROMOTIONS** role to the number of fiber domains to be promoted to embedding space domains. A non-zero **NUM-PROMOTIONS** role causes the **ADD-PROMOTER** rule to fire thus adding a **PROMOTER-MODULE** individual to the network's **GLOBAL-TRANSFORMATION-LIST** role.

Task Specification. Knowledge regarding the syntax and semantics of the task specification language. This knowledge includes rules mediating the visualization techniques applicable to various task operators. Currently, only one rule is defined for each operator in the task language. These rules dictate that **select** tasks use glyphs of type **SPHERE-GLYPH-THING**, and **search** tasks use glyphs of type **ISO-SURFACE-GLYPH-THING**.

Data Flow Networks. Knowledge regarding the construction of data-flow networks, and the functional specification of the components from which they are built. This knowledge includes concepts and individuals describing networks, modules, flow connections, glyphs and parameters, and rules for creating sentences in the language defined by the network grammar shown in Figure 1.

The bulk of the rules in the knowledge base is comprised of rules to manipulate the network grammar. In particular, each generalized visualization technique specifies a set of constraints for the filling of roles on its associated **NETWORK-THING**. These rules mediate such factors as the fiber bundle class of the data flowing into the input ports of the glyph macros. Transformation roles, such as the network's **GLOBAL-TRANSFORM-LIST**, are filled by re-writing rules which fire when there is a mismatch between the types of an output port and a subsequent input port. For instance, when an output port type is a **SET-THING** and an input port is an **INDEX-SET-THING**, the rule named **ADD-LINEARIZER-MODULE** is triggered which causes a **Linearizer** module to be interposed. The act of "linearization" transforms a fiber bundle of class set into one of class index-set by creating a one-dimensional index space and then arbitrarily assigning an index value to each element of the set.

Problem Domain. Knowledge specific to the data, units, conventions, and symbology commonly found in a particular problem domain. The Tecate knowledge base contains a small amount of knowledge directly pertaining to Earth science data sets. For example, the Planner is able to recognize data domains named **latitude**, **longitude**, and **elevation** in a DOI list. These domains are given preference for mapping into a visualization's x, y, and z scene coordinates.

Results

Figure 2 shows a visualization created by the prototype Tecate system for a relational data set representing California-based hydrological station measurements. Figure 3 depicts the AVS network generated by the system and whose execution produced the image. The task specification that was input to the Planner was, "**select (longitude, latitude, elevation, stationType)**." Although relational data does not have an intrinsic coordinate space associated with it, the Planner chose to represent the data set using a scatter plot by restructuring the data set as scattered data embedded in a 3D coordinate space. This visualization technique required manipulating the data set by "promoting" to embedding space domains some of the fiber domains listed in the task specification. Based on the rules in the knowledge base, the longitude, latitude, and elevation domains were promoted to be the x, y, and z axes of the scene space respectively. In addition, the Planner decided to use spheres to represent the location of individual stations (retaining autonomy for selection), and to map the station type to the color of the spheres.

Figure 4 shows another visualization created by the Tecate system for a data set output from a regional climate model program. The data set is a 3D array indexed by latitude, longitude, and elevation in which each array element is a record containing wind vector, cloud density, water content, and temperature values. Given the task specification, "**search (water-content)**," the Planner took advantage of the 3D structure of the base space to create a coordinate space plot. In the case of base spaces which inherently contain a coordinate space, it is assumed that the base space is to be used for the spatial component of the visualization. With the given task, the Planner chose to represent the water content data as an iso-surface. The AVS network that was generated by the Planner in this case is depicted in Figure 5.

Conclusion

The present Tecate prototype was intended to demonstrate the feasibility of using a knowledge-based system to help end-users develop data visualizations. Although fully capable of structuring visualizations for many of the types of data encountered in the sciences, the quality of the visualizations can be improved. One way to do this is to simply place more varied and robust knowledge into the knowledge base. Beyond this obvious improvement, the design of the Planner and the way knowledge is encoded in the knowledge base can be improved as well. Some lessons that have been learned in developing and using the prototype include the following:

Modularity. Knowledge engineering is a complex undertaking primarily because there is little methodology to guide developers. To help build intelligent systems, it is important to design modularity into a knowledge base and planner as one would in developing a large piece of software. The current Tecate Planner creates a data-flow network in a single call to the inferencing engine. This procedure is like writing a large C program by placing all of the code in the main routine and not using any function calls. As would be expected, the resulting knowledge base is unnecessarily complicated and difficult to debug. One outcome of this lack of modularity is that "meta-rules" had to be added to insure that regular visualization rules were applied in proper order. A better alternative is to partition the Planner so that a sequence of smaller inferences would be made instead.

Multiple Entry Points. Because planning takes place in one large inference, there is no facility for an end-user to interrupt the planning process in mid-stream to make individual preferences known. For example, if the Planner

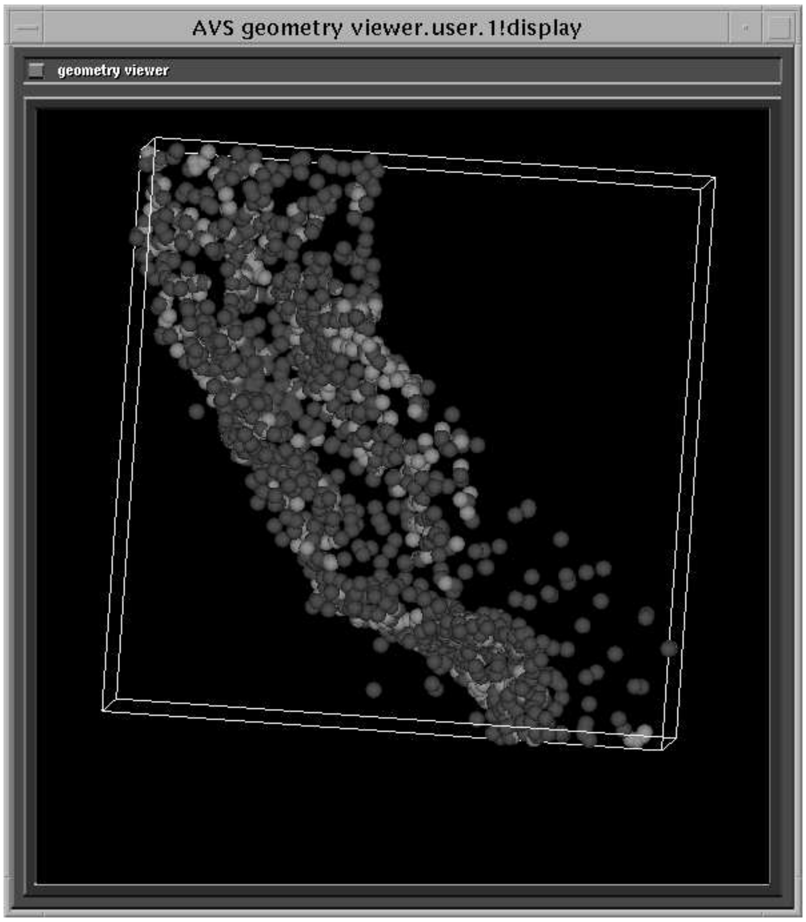

Figure 2: Visualization of hydrological survey stations in California color coded by station type. This image was produced by the Tecate system given the task, "**select (latitude, longitude, elevation, stationType)**."

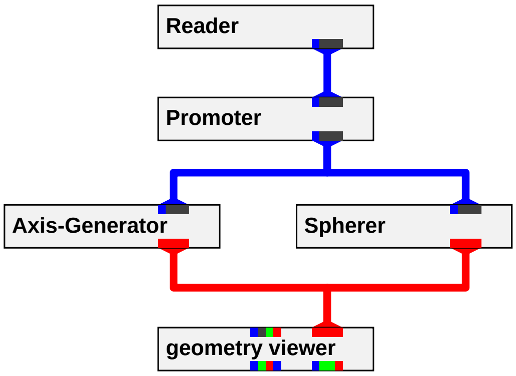

Figure 3: The AVS network that produced the hydrological station visualization.

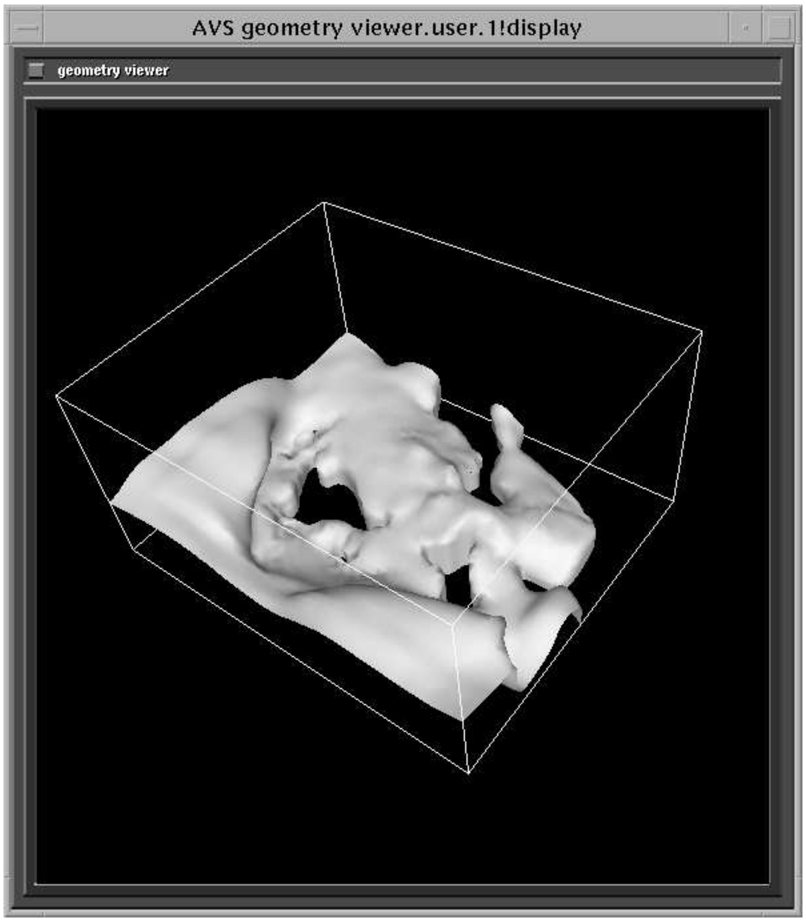

Figure 4: An iso-surface visualization of regional climate model output. This image was produced by the Tecate system given the task, "**search (water-content)**."

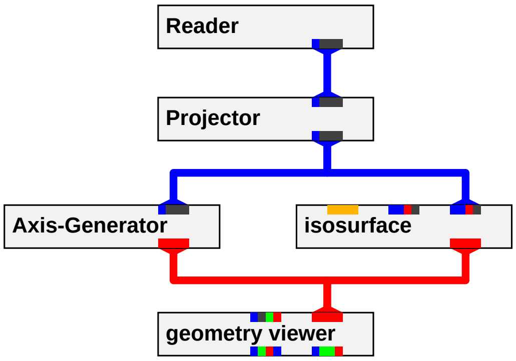

Figure 5: The AVS network that produced the iso-surface visualization.

would have chosen to use an iso-surface in a particular circumstance but an end-user would have preferred to use another technique, there is no mechanism with which to indicate this preference. Having this capability would also simplify adding learning to Tecate's Planner since the Planner then would know which of its inferences were overridden by an end-user.

Multiple Tool Support. The current Tecate Planner is structured so that for any given data set class, there is only a single visualization technique that can be applied. In the future, the Planner needs to be extended so that it can choose from a collection of techniques that are applicable to a given data set class and task specification. To do so requires that an *effectiveness metric* [5, 11] be developed to determine which of the possible techniques would be the most appropriate to accomplish a stated task. The definition of an effectiveness metric for three-dimensional visualization is an open research issue.

User-Computer Dialog. The current task specification language is textually-oriented and limited in scope. To more fully aid end-users, the task language should be enriched to allow the articulation of a broader range of data analysis goals. In addition, a graphically-oriented task editor should be provided to simplify the creation of task specifications.

As data sets become more complex and as the body of knowledge to visualize them grows, intelligent systems will become increasingly important. For this reason, it is hoped that the Tecate prototype will serve as a springboard from which the design of useful production systems will abound.

Acknowledgements

This work was supported by the Digital Equipment Corporation, the University of California, and the San Diego Supercomputer Center. Special thanks to the rest of the Sequoia Visualization team, Mike Bailey, Mike Kelley, Jonathan Shade, and Colin Sharp, for their continued support.

References

[1] Jacques Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[2] A. Borgida et al. CLASSIC: A structural data model for objects. In *Proceedings of the 1989 ACM SIGMOD International Confernece on Management of Data*, 1989.

[3] D. M. Butler and M. H. Pendley. A visualization model based on the mathematics of fiber bundles. *Computers in Physics*, pages 45–51, Sep/Oct 1989.

[4] David M. Butler and Steve Bryson. Vector-bundle classes form powerful tool for scientific visualization. *Computers in Physics*, 6(6):576–584, Nov/Dec 1992.

[5] Stephen M. Casner. A task-analytic approach to the automated design of graphic presentations. *ACM Transactions on Graphics*, 10(2):111–151, April 1991.

[6] Jeff Dozier. How Sequoia 2000 addresses issues in data and information systems for global change. Sequoia 2000 Technical Report 92/14, University of California, Berkeley, August 1992.

[7] R. B. Haber, B. Lucas, and N. Collins. A data model for scientific visualization with provisions for regular and irregular grids. In *Proceedings Visualization '91 Conference*, 1991.

[8] Eve Ignatius and Hikmet Senay. Visualization assistant. In *Proceedings IEEE Visualization Workshop on Intelligent Visualization Systems*, October 1993.

[9] Guy L. Steele jr. *Common LISP: The Language, 2nd Edition*. Digital Press, 1990.

[10] Peter Kochevar et al. A visualization architecture for the Sequoia 2000 project. Sequoia 2000 Technical Report 93/35, University of California, Berkeley, September 1993.

[11] J. D. Mackinlay. Automating the design of graphical presentations of relational information. *ACM Transactions on Graphics*, 5(2):110–141, 1986.

[12] Lori Resnick et al. *CLASSIC Description and Reference Manual for the Common LISP Implementation*. AT&T Bell Laboratories, Murray Hill, New Jersey, 1993.

[13] Hikmet Senay and Eve Ignatius. VISTA: A knowledge based system for scientific data visualization. Technical Report GWU-IIST-92-10, George Washington University, March 1992.