An Operator Interaction Framework for Visualization Systems

Ed Huai-hsin Chi, John T. Riedl
University of Minnesota
Department of Computer Science and Engineering
{echi,riedl}@cs.umn.edu

Abstract

Information visualization encounters a wide variety of different data domains. The visualization community has developed representation methods and interactive techniques. As a community, we have realized that the requirements in each domain are often dramatically different. In order to easily apply existing methods, researchers have developed a semiology of graphic representations. We have extended this research into a framework that includes operators and interactions in visualization systems, such as a visualization spreadsheet. We discuss properties of this framework and use it to characterize operations spanning a variety of different visualization techniques. The framework developed in this paper enables a new way of exploring and evaluating the design space of visualization operators, and helps end–users in their analysis tasks.

Keywords: information visualization, operators, user interactions, view/value, framework, spreadsheet, design, extensibility, visualization systems.

1 Introduction

Why the need for an operator framework? Imagine a visualization application with two views of the same source dataset, say a HomeFinder application. In one view, the dataset is visualized using a scatter plot display with dynamic query sliders [2], while the other view shows the values using a sorted numeric table. Now let's use the sliders to filter out some data points. The scatter plot view changes accordingly. However, a question of semantics arises for the table view. One possible interpretation of this action is that the table view is a totally independent view of the original dataset, and therefore should not change its view. The other possible interpretation is that the original data source is being modified by this interaction, which means the table view should change accordingly! Which of these two possibilities is the correct interpretation? Let's try to solve this problem of contradictory semantics by examining the application domain. Assume the user is interested in selecting homes that fall in her price range. If the user is merely interested in how the plot view changes while manipulating the price sliders, we would then argue that the table view should not change at all, because the task semantics do not require the original data source to be modified. If the user is actually interested in creating a new dataset that only contains homes in her price range, then we would argue that the original dataset is indeed being modified, and therefore the table view should change accordingly. Both interpretations of the interaction are valid under this task scenario! The user needs a "Do What I Mean" key that requests the behavior she intends.

Problem for End–Users The above example shows that end–users often have difficulty interacting with visualization systems because there is a wide 'gulf of execution'—"a difference between the intentions and the allowable possible actions" [23]. Sometimes the semantics of operations are imprecise, or worse, impossible to achieve. The user is often left with no way of predicting the result of her actions, or may even be incapable of selecting the operation she desires from among several alternatives. The operation and interaction model often hampers the analysis process because it does not fulfill the needs of the analysis. We can construct similar scenarios by examining other data domains, such as visualizing hierarchical structures such as file systems or organizational charts, slices of a 3D human brain, or world-wide web linkage structures. So this problem exists even after careful consideration of the application task domain.

Problem for Designers Information visualization has made great strides in the development of a semiology of graphical representation methods [5, 20, 6], but lacks a framework for studying visualization operations. Visualization system designers have three problems that can be improved by such a framework: operator reuse, view/value separation, and operand focus.

Operator reuse is an important consideration in developing visualization systems that can be applied to many different application domains. Consider our Spreadsheet for Information Visualization system (SIV), for example [8]. Spreadsheet environments are powerful because of a rich set of operators. One of the challenges of applying a spreadsheet to information visualization is the wide variety of data domains that are dealt with in information visualization. Therefore, the flexibility and the generalizability of the spreadsheet hinge on the application programmers' ability to extend the spreadsheet with additional operators as needed for their application domain. However, the disparities between different types of operators and their applicability in different situations make operators difficult to implement and reuse.

The HomeFinder example emphasizes the difference between view and value. The *value* within each cell is the dataset being visualized. The *view* controls the way the data is represented on the screen. In information visualization, since the data is represented abstractly on the screen, there is a distinct separation between the value of the data and the view of the data, and it is especially useful to represent the same data in many different ways. This is different from numerical data or other areas of visualization, such as volume visualization, where there is a tighter coupling between the value of the data and its visual representation. In HomeFinder, the user needs the ability to decouple the view and the value, so that they can be specified and changed independently. We call this *view/value separation*.

In scientific visualization, past work in data–flow networks and the visualization pipeline helped users to focus on the visualization process. These models have enabled us to better understand how operators interact with each other, because they were designed with the goal of solving the difficult problem of how to provide a user model for the analysis process.

In different problem–solving situations, users prefer to focus variously on the operations to achieve a single desired result, or on the operands at various stages in the computation. Instead of focusing on the process, sometimes the user is more concerned with what is the state of her data. For example, in numeric spreadsheets, instead of showing the explicit relationships between variables (the numeric equations), the system hides those relationships in favor of showing the operands of the formulas. This enables the user to focus on the intermediate computational results. Numeric spreadsheets emphasize the operands. This mode of interaction is especially useful in situations where the next analysis step is not always immediately apparent. By showing the state of the data, the user gets visual feedback that helps bridge the gulf of evaluation. The user can evaluate the result of her last action and choose the next step based on the results of the computation. Therefore, in addition to describing the analysis process, we seek a model that allows us to capture the data states so we can accurately provide feedback and support these exploratory interactions. We call this *operand focus*. We need a model that not only describes the data transformation

process such as the visualization pipeline, but that also models the data states.

So from the viewpoint of end–users and designers, we see that there is a need to construct a general operator framework—a conceptual model that enables us to clearly classify and organize different operators. While we were motivated by our research in the Visualization Spreadsheet [8], this is a general question about the utility of visualization systems, because it is often unclear how domain–specific operators are to be integrated into visualization systems. Without the ability to incorporate domain specific operators, a visualization toolkit or system would be useless. Furthermore, there are many operators that are not domain–specific but are not effectively reused in different applications. We need an effective operator framework in order to better understand these issues. The framework must enable us to better understand the interaction between data, view, and the operators.

What is an operator framework and how does it help us? An operator framework is a conceptual model for all possible visualization operations. By *operation*, we mean all user interactions, whether based on direct manipulation or other interaction. Our motivation is two–fold: (1) to develop a framework that is sufficiently clean and simple that it enables end–users to choose which operator to apply for a desired result, and to predict the results of their interactions with the visualization system; and (2) to develop a general interaction model for information visualization that helps visualization designers classify and understand the relationships between operators and the composition of interactions.

The biggest benefit of achieving our goal is establishing a user conceptual model that allows us to bridge this conceptual gulf of execution for the end–user. It helps us eliminate errors caused by imprecise or incorrect conceptual models. It also helps us bridge the 'gulf of evaluation'—the feedback from the system is "directly interpretable in terms of the intentions and expectations of the person" [23]. The model helps users in performing the actions appropriate to the task.

This model will also enable us to organize operators by classifying and taxonomizing the space of possible operations. Herb Simon once said that the inherent value of classification is that, in understanding any phenomenon, the first step is to "develop a taxonomy" [29]. This understanding is what enables us to isolate the important artifacts for design. In information visualization, an operator framework allows us to build interaction models for new data domains.

In summary, the operator framework should enable both end–users and designers to better understand the situations in which operators can be applied, how they can be applied, and what operators do.

In this paper, we contribute to a new way of thinking about the operator model that applies over a range of data domains, with some specific discussion as applied to visualization spreadsheets:

- Establishing a new operator–centric framework for designers to explore the following properties of operators in visualization systems: view vs. value, domain dependence/independence, breadth of applicability, amount of direct manipulation possible, and implementation choices. In particular, we show three different operator implementation choices (inside a data repository such as a DBMS, inside the visualization system, or outside of both the systems as a separate module.)
- Developing a new end–user interaction model that establishes user expectation, thus enabling users to apply and predict the result of operators and the relationships they establish between views and values. We form an analysis process model for users to apply to their task scenario in their particular data domain.
- Focusing on end–users' need for viewing intermediate results in determining subsequent analysis steps. We use a visualization state model with multiple data values and views to bridge the "gulf of execution".
- Applying this framework to past visualization systems and techniques, including visualization spreadsheets. We demonstrate the framework by enumerating the interaction techniques in many past visualization projects.

The rest of the paper is organized as follows. In section 2, we discuss some related work in this area. In section 3, 4, and 5, we present our operator model and some analysis and discussion of its properties. In section 6, we illustrate our model by analyzing it using multiple existing interaction techniques. Finally, we have some concluding remarks.

2 Related Work

Many have observed the intricate relationship between the view and the value associated with that view. In particular, a good observation was made in [3] that when using a brushing technique with a group of scatterplots, the effect of an operation on a data point appears simultaneously on all scatterplots in the other views. They termed this "coordinated". This is a simple, yet–powerful, notion of view and value, where the view is always tied to the underlying value. This binding is never broken. While this model does not take multiple data sources into account, the advantage is that the user has a very concise and clear model of how the system works. The disadvantage is that the other opposing semantic is impossible, which is that the user may just want to temporarily change one view, but not all of them. For example, the user may simply want to select a group of data points in one particular view to highlight it to discuss it outside the context of other scatter plots.

In scientific visualization, many have examined the visualization data–flow network for constructing visualizations [31, 14, 35]. Schroeder et. al. [28] described a fairly complete conceptual dataflow model in the context of scientific visualization for applying operations to generate a visualization. The model consists of a visualization network that can contain multiple sources and sinks. Every step in the middle of the network consists of filters that have inputs and outputs.

Similar concepts have led to other work on building operator models for these kinds of dependency issues. Lee and Grinstein [18] presented a conceptual model for database visual exploration, which describes the analysis process as a series of value-to-value, value-to-view, view-to-value, and view-to-view transformations. They also describe the concept of generating metadata using database queries to aid in this process.

Chuah and Roth [11] extends Foley et. al.'s user interaction framework [13] by incorporating BVI (Basic Visualization Interactions), which is a more detailed characterization of data filters in the context of information visualization. They also presented a basic classification taxonomy for BVIs (shown in Figure 1). We were motivated by this work, and explored to what extent this taxonomy suits our needs. However, we were unclear about the relationship between 'set operations' and 'data operations', and the semantics of view/value filtering ('shift' as defined in Figure 1) appears confusing in this model. the 'encode data' subtree is dramatically more complex than 'set–graphical–value' and 'object–manipulation' operations. The class of visual mapping operators needs more examination.

Tweedie [30] presented a data transformation model similar to [18], a simple interactivity model that basically classifies the interactions based on the amount of control the user has over the process, and a simple state model similar to [11].

Past spreadsheet work has focused mostly on data that can readily be visualized with a straight mapping, e.g. numeric, or images.

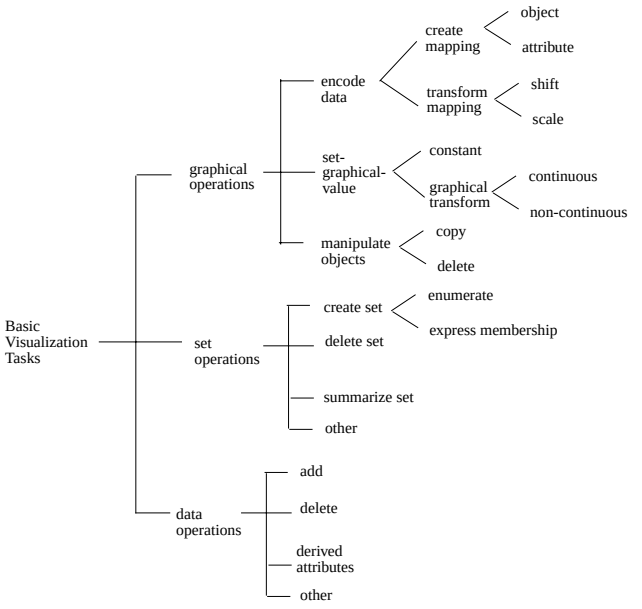

Figure 1: Chuah and Roth's Basic Visualization Interaction taxonomy

Levoy [19] describes a tool called "Spreadsheet for Images" that mentions the importance of data flow in spreadsheet. Levoy also briefly mentions volume visualization in the context of his tool. Most of the interactions in his tool are implemented as Tcl [24] commands, with certain geometric operations implemented using direct manipulation. Images, however, are rather straight forward mapping from value to view. In a sense, the view is the value, and there is very little discern-able differences between the two. Therefore, the operator model for an image spreadsheet is relatively simple in comparison to a full–blown visualization system.

As we developed the Visualization Spreadsheet [8], we noticed some deficiencies of past interaction models. They were not sufficiently detailed for describing operators and interactions in the spreadsheet. For instance, while the BVI model tied the interaction model with a state model, the state model lacks detail to help capture domain–specific designs. Moreover, it does not appear to handle operations with multiple semantics. For example, the filter example at the beginning of the introduction section suggests that we could interpret dynamic query filters as either a value operation or a view operation. Under the BVI architecture, dynamic querying is classified as a graphical operation, which does not affect the underlying data. On the other hand, Tweedie's model was so general that it offered little evaluation potential in the context of information visualization systems that have multiple data sources and views.

Most importantly, past models also failed to unify the interaction model with the visualization process. The visualization pipeline as described by Stuart Card's [6] is a rich design space that has yet to be unified with an interaction model. Chuah and Roth's work, which unified the low-level keyboard/mouse interactions, did not incorporate this visualization pipeline. We need a model that describes how the graphical, data, and control states are affected by the operators. Here we are trying to extend past work by unifying a taxonomy of operators with the visualization pipeline that uniquely solves the above problems.

3 Operator Model

3.1 Properties of operators

In order to develop an operator framework, we first start by observing some fundamental properties of operators from the visualization spreadsheet point of view. One property is whether an operator is a view or value operator—whether it modifies the underlying data set or not. The other property is degree of functional similarity with other operators. These two properties are important because functional similarity deals with an operator's degree of applicability, and view/value have deep implications regarding the semantics of the operator.

3.1.1 Functional versus Operational Similarity

In developing our model, we made the first observation that some operators are **operationally similar** across applications—operations whose underlying implementations are exactly the same from application to application such as rotations, scaling, translation, camera position manipulations, geometric object manipulations, and lighting. The entire class of geometric and scene operators are operationally similar across applications, because we can make a fundamental assumption that once we obtain a view, we are dealing with graphic primitives such as lines and polygons that we can operate on in exactly the same ways. There are other operators that belong in this class, such as duplicating or deleting a view or value, and renaming a data source.

We also observed that there are operators that are only **functionally similar**—operations that are semantically similar across applications, but the underlying implementation are actually different for different types of data sets. For example, filtering a data set is a common and extremely useful operation, but different application domain have different ways of filtering the data set. Another example is the class of algebraic operators such as adding or subtracting data sets, which is again domain specific. The way we add two for-sale real estate property list together is not the same as combining the web linkage structures from two different crawls of the web.

Finally, there are operators that are only application task dependent—operations that are specially designed for a specific task in a particular application domain. An example of this class of operations is an electrical probe inserted during a heart electrical pulse visualization. We could specify the positioning of a probe using the mouse and trigger a new heart pulse simulation. The simulation is a domain task specific operation. Another example is a specific implementation of the HTML document parsing operation for multi–dimensional scaling to compute similarity of documents.

3.1.2 View versus Value

Another dimension of operators is whether it is view or value–oriented. By **value**, we mean the raw data, whereas **view** is the visualization end–product. A *value operator* changes the data source by such processes as adding or deleting subsets of the data, filtering or modifying the raw data, and performing a Fourier Transform on an image. A value operator fundamentally generates a new data set.

A *view operator*, on the other hand, changes the visualization content only. Examples of such operators include 3D rotation, translation, and zooming, a horizontal or vertical flip of an image, and changing transparency values of a surface in order to see the underlying structures better. A view operator fundamentally does not change the underlying data set.

The distinction between a view and value operator is not always clear. For instance, the modification of the colormap represents a raw pixel value modification for an image, and therefore, should be classified as a value operator. However, in a 3D surface heat map, the modification of the heat scale appears to be a view change that does not fundamentally change the underlying surface values. Another example is the motivational example from the Introduction Section. Sometimes we would like to apply filtering to generate a data set. Other times we just like to temporarily make certain data points invisible without affecting the underlying data source. The same filtering operation appears to change its property depending on the user's intentions! How do we unify such seemingly contra-

dictory classification of operators according to this important property? View/value does not appear to be a black and white property for operators.

3.2 A Visualization State Model for Operators

Visualization Pipeline The concepts of functional and operational similarity are related to the concepts of view and value operators. On the one hand, view operations tend to be more operationally similar across application domains. On the other hand, value operators tend to appear functionally similar but implemented differently for each data domain. Even though there are classes of value operators, such as combining data sets, a value operator must perform on the specific data structures from the application domain. But view operators, such as scene, geometric, and pixel operations, performs on the displayed end–product, which we can assume to be graphic primitives such as points, lines, polygons, or voxels. We need a model that fits with this observation. Our discovery is that the solution to this dilemma comes from a non–intuitive source—the visualization pipeline.

On one end of the pipeline, we have the data (value), while on the other end, we have the visualization (view). We propose that view/value property for operators is a fundamental classification for what stage the operator is in the visualization pipeline. On the one end of the spectrum, we have full view operators that can only be interpreted as view operators, such as rotation. On the other extreme, we have full value operators that can only be interpreted as value operators, such as expanding an existing data set by adding a new data set. Operators that are not full view or full value operators lie in between the two extremes.

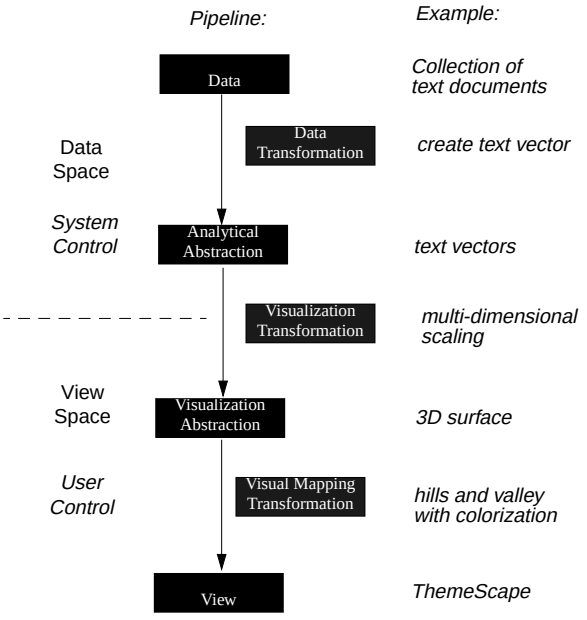

Figure 2: The information visualization pipeline (slightly modified from Stuart Card's model)

In information visualization, data domains usually contain more complicated pipelines (a model is shown in Figure 2, which is slightly expanded from Stuart Card's information visualization pipeline). Raw data are usually first processed into some form of analytical abstraction (a.k.a. metadata, data about data, or information) through a data transformation process. This analytical abstraction is often further reduced using a visualization transformation into some form of visual abstraction, which is information content that is visualizable. Usually this process contains a dimension reduction step, because the data sets in information visualization are usually complex and multi-dimensional. An example of visualization transformation is multi-dimensional scaling and clustering. From the visualization abstraction, there is a further step of visual mapping transformation that brings a view that is presentable to the user on the computer display.

State Model In order to accurately emphasize the end–user's analysis process as well as the intermediate results, we constructed a new model based on the visualization pipeline (see Figure 3). The modifications are two–fold.

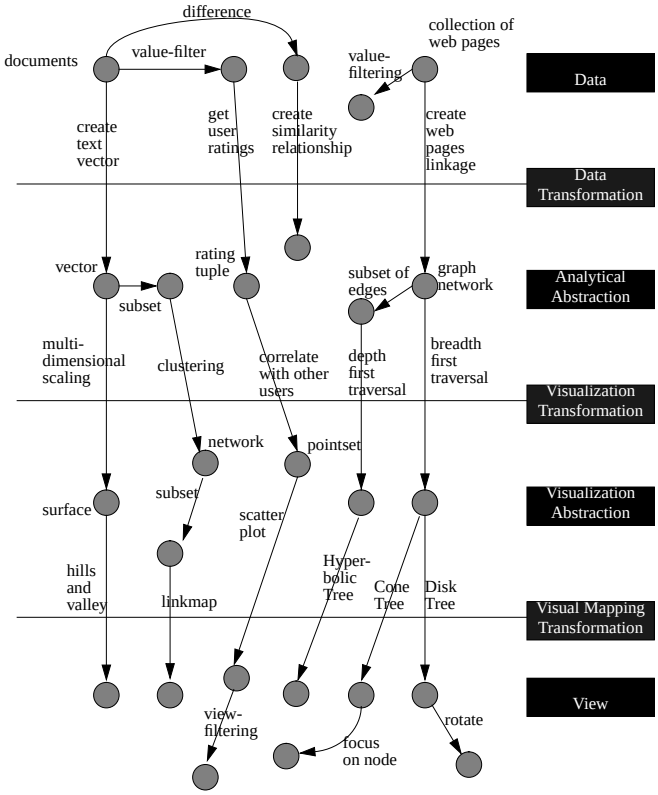

Figure 3: Our visualization operator model

First, while the visualization pipeline handles a large variety of operators, the model does not take multiple values and multiple views into account. If two separate data sets go through two different pipelines to contribute to a single visualization, the model breaks down. In order to ameliorate this problem, we expand the above pipeline into a network that allows as many values and as many views as needed. To this extent, our model is similar to data–flow networks as presented in [28].

Second, the visualization pipeline uses nodes to represent operators, and edges to represent flow of data. Instead, we use a state model, where each node represents a certain data state, and each edge is an operator transforming the data from one state to the next. Instead of stages, each node in the network is a state describing the status of the data. Each directed edge from a state to another state describes the operator that is applied to modify the data. The source data states are the raw values, whereas the sinks are the views of the data sets.

In some sense, our model is similar to a dual of the traditional scientific visualization data–flow network. Our model is not an exact dual, because we use a state model. In data–flow networks, the data state is not explicitly represented. For example, a different data set can flow down the same exact pipes. Also, a view rotation does not take the data to a new state. If we construct an exact dual

of the data–flow network model, we would get nodes that represent pipeline *stages*, instead of intermediate result *states*. Our state model encompasses data–flow concepts, while being more detailed at the same time. The state model has advantages for some visualization tasks, because it makes the intermediate results explicit to the user, which enables the user to view intermediate results in planning later operations.

Let's examine an example of applying this framework to a specific application domain—visualization of web site structures (see right side of Figure 3). The raw data set is a collection of web pages generated by crawling a web site. We can first perform a value–filtering operator where we search for documents that contain the word "Hewlett–Packard" or "HP". This would be an example of within data stage operator. We can then use this collection of web pages and generate a graph network (the analytical abstraction) from the linkages between pages. This is a data transformation operation.

Using the network, we can again select only subsets of the edges, such as choosing only the first three levels of documents from the root node. This subset operation is an example of a within analytical abstraction stage operator. We can then create a tree by doing a breadth first traversal (a visualization transformation operation). The breadth first traversal generates a visualization abstraction, a hierarchical tree of the web pages, that can be easily visualized. There are many visual mapping techniques that can be applied to this visualization abstraction, such as Cone Tree [27], Disk Tree [9], TreeMap [16], Hyperbolic Tree [17]. Within View Stage Operators such as focusing and brushing nodes, or rotating the cone tree can then be applied to this visualized content.

4 Analysis of Operators with the Framework

4.1 Classification of Operators

Using the above model, we can classify operators according to what stage the operator is involved in. Note that some operators work within a single stage, while other operators work across different stages:

Data Stage Operators (DSO) .
General: value-filtering, subsetting
Domain Algebraic: difference or addition of two data sets
Image: flip, rotate, crop, fourier transform, etc.
Point set: value–filter
Web: collection of web pages generated by crawling a web site

Data Transformation Operators (DTO) .
Textual: computing textual vectors
Grid: iso-surface extraction
Point set: triangulation
Web: hypertext document network

Analytical Abstraction Stage Operators (AASO) .
Vector: select a subset of the vectors
Surface: divide region
Web: select subset of the the nodes in the network

Visualization Transformation Operators (VTO) .
Dimension Reduction: multi-dimensional scaling or principal component analysis
Clustering: association rule, multi-modal clustering, spreading activation
Network: breadth first traversal, depth first traversal

Visualization Abstraction Stage Operators (VASO) .
Grid: simplify by reducing number of regions
Network: simplify by consolidating nodes
Hierarchy: cut–off depth of tree

Visual Mapping Transformation Operators (VMTO) .
Point set: scatter plot
Multi–dimensional Surfaces: World–within–World,
Hierarchy: Cone trees [27], Hyperbolic Trees [25], TreeMaps [16] and Disk Trees [9]
Network: GV3D [1], NVB [22], SeeNet [4]

View Stage Operators (VSO) .
Object Manipulation: rotation, translation, scale, zoom
Camera: position and orientation
General: view–filter

4.2 Why is this framework powerful?

This operator model provides a classification that is powerful because it makes these following properties of operators explicitly clear:

View vs. Value The closer an operator is to the view end of the pipeline, the more it takes on view operator properties. Similarly, the closer an operator is to the value end of the pipeline, the more it takes on value operator properties.

Operator–centric approach Note that we are explicitly taking an operator–centric approach. If an operator appears to be able to operate on multiple types of data, we separate the single operator idea into several different operator implementations. This approach is the opposite of the data–centric approach, which favors overloading operators so that they can function with multiple data types. As an example, if a filter can be viewed as both a value operator as well as a view operator, we separate these two meanings into a value–filter and a view–filter operator.

Applicability of operators The breadth (the amount of operational similarity, or wide–applicability) of an operator is dependent on how late it comes in the visualization pipeline. All Within-Stage Operators take their respective data type from the stage, and output the same data type. Moving down the pipeline gets us closer and closer to a generalized data type that is applicable over a wider range of data domains. In order of decreasing breadth of applicability, we list some examples of the following different levels of applicability:

- As mentioned in Section 3.2, *view operators*, such as rotation, are applicable across a large set of data domains.
- *Visualization Transformation Operators* can be applied to data domains with similar goals. For example, multi-dimensional clustering can be used to reduce the dimensionality of any problem that can be formulated using a feature space. Breadth first traversal can be used to produce a hierarchy out of a graph network.
- *Visual Mapping Transformation Operators* are usually applicable to a wide variety of data types. For example, glyphs, icons, streamlines can be employed to show multi–dimensional data at a particular spatial point. Worlds–within–Worlds [12] can be used to visualize high dimensional surfaces. Cone Trees [27], Hyperbolic Trees [17], TreeMaps [16] and Disk Trees [9] can be used to visualize a wide variety of hierarchical data.
- *Data Transformation operators* are specific to the particular data structure from an application domain, because they take the data structure as input, and output an analytical abstraction or metadata. Examples include creating text vectors from a list of documents, or creating a graph network from a web site.
- *Value Operators* are specific to its associated data type, as discusssed in Section 3.2.

Direct manipulation The amount of direct manipulation that is possible in an operator also lies on this visualization pipeline scale. The closer the operator is to view, the higher the amount of interactivity. For example, the geometric position and orientation operators are easily directly manipulated. Variable-to-axis mapping,

a visualization transformation operation is also easily specified using a point-and-click approach. As we move up the pipeline toward value operators, the amount of domain–dependency increases, making the specification of these operations more and more difficult. For example, the parsing of a file for data extraction is a data transformation operator, and it is extremely hard to design an interface that allows the user to specify the file format. Interestingly, MS Excel has certain amount of automatic parsing capabilities using an 'Import Wizard'. This is because the need to import data is especially important to its users. Such capabilities are hard to design for information visualization, because the wide variety of data domains has different data and information structures. Indeed, for many visualization systems, the hardest part of the visualization process is importing the data!

Implementation Choices With the knowledge of the above model, the choices for the implementation of operators are clearer. There are three basic choices for implementation:

- inside the visualization system. Examples of operators appropriate for this choice are scene operators, camera operators, color scale operators.
- using queries inside a data management engine, such as a DBMS database. For example, we can use the full power of relational algebra to organize and use OnLine Analytical Processing (OLAP) to analyze and generate the metadata.
- using an analytical engine outside of both the data depository and the visualization system. Examples are differential equation solvers for fluid–flow simulations, web crawlers, numerical analysis in Mathematica, Maple, Matlab, business computation using numeric spreadsheet in MS Excel, or image, sound, video processing in Khoros.

The operator model also helps us in choosing between these choices. If the operator is closer to the view stage in the pipeline, then it is most efficient and most easily implemented in the visualization system. For example, systems such as AVS [33] and Data Explorer [34] data–flow systems and the Visualization Spreadsheet [8] implement most, if not all, of its visual mapping transformation operators and view operators in the visualization.

5 Discussion

5.1 Three classes of users of this framework

Three kinds of people can benefit from this framework. First, visualization system developers can use this framework to make the system extensible to the application programmer. For example, we have applied this framework to our visualization spreadsheet system.

Second, once a visualization system has been built, application programmers can use this model to extend the system to new data domains. This framework enables programmers to identify operators that are not domain specific and, hence, that they can easily reuse.

Last, but not least, this framework provides a clean and concise model for end–users to understand how to operate a system such as the visualization spreadsheet, and to predict the results of applying operators.

5.2 Framework, User, and the Visualization Spreadsheet

How does this framework help the end–user when using the visualization spreadsheet? The development of this framework enhances the usability of the spreadsheet by solving the following three problems.

First, the framework provides a user interaction model so that users can understand what they have to do to get a visualization. This is accomplished by incorporating the visualization pipeline process model. By following the steps in the pipeline, the user can perform the actions required to create a desired visualization in the correct order.

Second, the framework establishes users' expectation of the flow of changes to the data. This enables users to understand how the system works, and how the data flow can be manipulated to perform the correct analysis action. For example, let us create a set of textual feature vectors from a set of documents (see the left side of Figure 3). In one analysis, we choose to do multi–dimensional scaling, and in another analysis we choose to first create a subset of these textual vectors before applying a clustering algorithm. Because both states are dependent on the same data source, if the set of documents change, both states would change as well. This is made explicit by the state model.

Third, the framework cleanly solves the operator semantics problem, because it models the separation between view and value. The view versus value filtering example mentioned in the introduction is an excellent example of how the framework forces interaction designers to realize potential ambiguity in the semantic of operators. By forcing designers to think about where operators exist in the pipeline, the operator semantics are made explicit. By having a cleaner model, the end–user can now choose among several operational semantics that correspond to the correct action that she desires. The user can interact more accurately because she understands how operators in different stages of the pipeline fit together.

6 Framework Illustrated with Coverage Examples

In this section, we illustrate the power of the above framework by applying it to the design of the following visualization techniques. Using example data domains for each technique, we describe the operations that are possible using the framework. We also comment on any technique changes we made while making the classifications. For many of these techniques, we added the two different view/value filtering semantics. For a list of the acronyms we are using for the operators, see Section 4.1.

Dynamic Querying [2]
. Example data: Home, Movies sales data
. Analytical abstraction: parsed feature records
. Visualization abstraction: multi–dimensional point sets
. VASO: dynamic value-filtering
. VMTO: scatter plot, choosing variables-to-axes mappings
. VSO: dynamic view-filtering
Comment: could also apply unmapped variable filtering (see [10] for a discussion).

AlignmentViewer [10]
. Data: similarity reports from comparing a single sequence against a database of many other sequences
. DTO: parsing textual reports
. Analytical abstraction: alignment records (data structure representing parsed information)
. DTSO: addition, subtraction between different reports, unmapped variable value–filtering
. VTO: information extraction from records
. Visualization abstraction: feature point set with vector
. VMTO: comb–glyphs
. VSO: rotation, zoom, focus on a single alignment, detail-on-demand, animation (by using an iterator over the view–filtering operation)

Parallel Coordinates [15]
. Example data sets: production run of VLSI chip yield and its defect parameters

. Analytical abstraction: corresponding yield and parameter feature set
. AASO: choosing a subset of records using dynamic value–filtering
. Visualization abstraction: point set
. VMTO: parallel coordinate plot
. VSO: dynamic view–filtering

SeeNet [4]
. Example data sets: phone calls made, Internet packet flows, Email communication patterns
. Analytical abstraction: parsed records of source and destination and associated feature sets
. AASO: unmapped variable value–filtering, choice of displayed statistics, record aggregation
. Visualization abstraction: graph network
. VMTO: matrix display, geographical link maps, node maps
. VSO for all three views: sound feedback, unmapped variable view–filtering (they called it 'conditioning')
. VSO for nodemaps and linkmaps: size, color, zoom parameter focusing, identification by brushing, animation speed, line thickness, line length, dynamic query threshold view–slider
. VSO for nodemaps: symbol size, color sensitivity view–slider
. VSO for matrix display: time and threshold view–slider, permute rows and columns
Comment: added view/value filtering semantics, aggregation is mentioned as implemented using data management software (see discussion about implementation in Section 5), several different views of the data sets.

ThemeScape and Galaxies [36]
. Data: CNN news stories
. Analytical abstraction: text vectors
. AASO: choose an item and then perform weighted query
. Visualization abstraction: multi-dimensional scaling, principal component analysis
. VMTO: hills and valleys
. VSO: zoom, rotate, focus on spot
. VSO for ThemeScape: slices
. VSO for Galaxies: animation of scatter plot

Hierarchical Techniques : Cone tree [27], Hyperbolic Browser [17], TreeMap [16], Disk Tree [9]
. Data: file system, organization charts, web structure
. Analytical abstraction: graph
. AASO: dynamic value–filtering
. VTO: breadth first traversal
. Visualization abstraction: Tree hierarchy
. VMTO: 3D cone layout and hyperbolic tree layout, Disk layout of tree
. VSO: focus node, hide subtree, orientation and position of tree, dynamic level–filtering
Comment: multiple techniques for views

Perspective Wall [21]
. Data: schedule, file system
. Analytical abstraction: parsed record set
. AASO: dynamic value–filtering
. Visualization abstraction: linear list with item features
. VMTO: wall panels in 3D with glyphs, focus+context distortion–based
. VSO: focus on a particular wall, focus an item, dynamic view–filter, different levels of detail

WebBook and WebForager [7]
. Data: URLs for web pages
. Analytical abstraction: Images of HTML pages generated by getting the web pages
. Visualization abstraction: linear page lists, collection of page lists.
. VASO: aggregation into a book or a pile, place on book shelf (creating list of lists), crawl from a URL and create a book from the collection
. VMTO: books with multiple pages, Document Lens, bookshelf, table, piles
. VSO: focus on a book, focus on a page, flip through pages in a book, view as a Document Lens, history pile

Table Lens [25, 26]
. Data: baseball player statistics
. Analytical abstraction: numeric records
. AASO: sort
. Visualization abstraction: constructed numeric table
. VMTO: number represented using bars, with focus+context distortion–based table:
. VSO: change distortion focus

Time Tube [9]
. Data: web structure evolving over time
. Analytical abstraction: evolving graph represented as ordered collection of graph
. VTO: breadth first traversal with global node position over time
. Visualization abstraction: evolving tree as ordered list of trees
. VMTO: Time Tube represented using an aggregation of Disk Trees (invisible tube–like shelf)
. VSO: gestures for focus on a slice, bring slices back into the Time Tube, right–click for zooming focus on the connectivity of a node, rotate slices, brushing on pages by highlight URL on all slices, animation through the slices

Spreadsheet for Images [19]
. Data, analytical and visualization abstraction: pixels, voxels
. DSO: rotate image, filter, change color scale
. View: images from pixels, volumes from voxels (direct mapping from data to view)
. VSO: rotate image, filter, change color scale, rocking the volume visualization

FINESSE [32]
. Data: financial data
. DTO: compute call and put option prices
. Analytical abstraction: matrix records, mathematical functions
. AASO: change parameter of functions, change arithmetic relationships, load, copy, paste, cut, move, clear cell, input math function
. Visualization abstraction: matrix, computed curves
. VMTO: heat map, surfaces in 3D, 3D bar charts, 2D line plots, text for filenames, value sliders
. VSO: orientation of geometric objects, common colormap or font, same geometric orientation, show cell dependency relationships, picking a data item

Spreadsheet for Information Visualization [8]
. Example data sets: point sets, matrix, sequence similarity reports
. DTO: normalize, parse textual report
. Analytical abstraction: normalized matrix and point sets, value tuples
. AASO: dynamic value–filter, algebraic set operators
. VTO: Delaunay Triangulation, data feature extraction
. Visualization abstraction: point set, matrix, triangulated surface, point set with feature vector
. VMTO: heat map, cube visualization, bar visualization, cone tree, disk tree, glyphs, scatter plot, choosing variable-to-axes mapping
. VSO: dynamic view–filter, object position and orientation, pixel image addition between cells, geometric object addition between cells, animation
Comment: allows coordinated direct manipulation, value and view dependencies between cells, change cells to have same visual mapping transformation

7 Conclusion

In the past several years, researchers have made great advances in information visualization. Semiologies of graphic representation methods have been developed by various researchers [5, 20, 6] to gain understanding of the visualization design space. A major difference between current information visualization work and past work on graphic design is the development of interactivity. The dialog between human and computer enriches the communication of information.

In this paper, we examined recent work on visualization interaction frameworks and then developed a novel operator and user interaction model. Our state model unifies the data analysis process and the complex relationship between view and value to characterize the interactive and non–interactive operations in a visualization system. Using the visualization pipeline as a basis, we developed a way to classify operators. We examined not just view and value, but how metadata is generated in the analysis process. We also suggested three possible ways of implementing operators based on where they are involved in the visualization pipeline. Finally, we illustrated the usage of this framework by applying it across a large number of different visualization techniques. We showed that this visualization operator model allows us to characterize the operations that are possible in each technique.

Using the state model, this method forms the basis of an evaluation technique for operators in visualizations. By applying this operator analysis to various visualizations, we can point researchers toward areas where particular operators are missing from a given system or technique. We can also use this model to compare different interaction models in visualizations.

The framework in this paper facilitates a new way of exploring the space of visualization operators. We hope this will enable other researchers to characterize various interaction techniques, and capture design requirements for new application domains, and develop new and novel operators.

Acknowledgments This work has been supported in part by the National Science Foundation under grant BIR 9402380. We like to thank the reviewers, Stuart Card, Joseph Konstan, and Phillip Barry for comments, inspiration, and discussion.

References

[1] Graph Visualizer 3D. http://www.omg.unb.ca/hci/projects/gv3d/, March 1998.

[2] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 313–317, 1994. Color plates on pages 479-480.

[3] R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[4] R. A. Becker, S. G. Eick, and A. R. Wilks. Visualizing network data. *IEEE Transaction on Visualization and Computer Graphics*, 1(1):16–28, 1995.

[5] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. University of Wisconsin Press, Madison, WI, 1967/1983.

[6] S. K. Card and J. Mackinlay. The structure of the information visualization design space. In *Processings of Information Visualization Symposium (InfoVis '97)*, pages 92–99. IEEE, IEEE CS Press, 1997.

[7] S. K. Card, G. G. Robertson, and W. York. The webbook and the web forager: An information workspace for the world-wide web. In *Proceedings of ACM CHI'96 Conference on Human Factors in Computing Systems*, pages 111–117. ACM, ACM Press, 1996.

[8] E. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *Proc. Information Visualization Symposium '97*, pages 17–24,116. IEEE CS Press, 1997.

[9] E. H. Chi, J. Pitkow, J. Mackinlay, P. Pirolli, R. Gossweiler, and S. K. Card. Visualizing the evolution of web ecologies. In *Conference on Human Factors in Computing Systems (CHI 98)*. ACM, ACM Press, April 1998.

[10] E. H. Chi, J. Riedl, E. Shoop, J. V. Carlis, E. Retzel, and P. Barry. Flexible information visualization of multivariate data from biological sequence similarity searches. In *IEEE Visualization '96*, pages 133–140, 477. IEEE CS Press, 1996.

[11] M. C. Chuah and S. F. Roth. On the semantics of interactive visualization. In *Proceedings of IEEE Visualization (Vis '96)*, pages 29–36. IEEE, IEEE Press, 1996.

[12] S. Feiner and C. Beshers. Visualizing n-dimensional virtual worlds with n-vision. *Computer Graphics*, 24(2):37–38, 1990.

[13] J. D. Foley, A. vanDam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[14] P. E. Haeberli. ConMan: A visual programming language for interactive graphics. In *Computer Graphics*, volume 22, pages 103–111. ACM SIGGRAPH, August 1988.

[15] A. Inselberg. Multidimensional detective. In *Proc. Information Visualization Symposium (InfoVis '97)*, pages 100–107. IEEE, IEEE CS Press, 1997.

[16] B. Johnson and B. Shneiderman. Tree-maps: A space-filling approach to the visualization of hierarchical information structures. In *Proc. IEEE Visualization '91*, pages 284–291, Piscataway, NJ, 1991. IEEE.

[17] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Information Visualization*, pages 401–408, 1995.

[18] J. P. Lee and G. G. Grinstein. An architecture for retaining and analyzing visual explorations of databases. In *Proceedings of IEEE Visualization (Vis '95)*, pages 101–108. IEEE, IEEE Press, 1995.

[19] M. Levoy. Spreadsheet for images. In *Computer Graphics (SIGGRAPH '94 Proceedings)*, volume 28, pages 139–146. SIGGRAPH, ACM Press, 1994.

[20] J. Mackinlay. Automating the design of graphical presentation of relational information. *ACM Transaction on Graphics*, 5(2):110–141, Apr. 1986.

[21] J. D. Mackinlay, G. G. Robertson, and S. K. Card. The perspective wall: Detail and context smoothly integrated. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 173–179, 1991.

[22] S. Mukherjea, J. D. Foley, and S. Hudson. Visualizing complex hypermedia networks through multiple hierarchical views. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1 of *Papers: Creating Visualizations*, pages 331–337, 1995.

[23] D. A. Norman. *The Design of Everyday Things*. Doubleday, 1988.

[24] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[25] R. Rao and S. K. Card. The Table Lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems*, volume 1 of *Information Visualization*, pages 318–322, 1994. Color plates on pages 481-482.

[26] R. Rao and S. K. Card. Exploring large tables with the table lens. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 2 of *Videos*, pages 403–404, 1995.

[27] G. G. Robertson, J. D. Mackinlay, and S. K. Card. Cone trees: Animated 3d visualizations of hierarchical information. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, Information Visualization, pages 189–194, 1991.

[28] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. Prentice Hall, 1996.

[29] H. A. Simon. *The Sciences of the Artificial*. MIT Press, 1969.

[30] L. Tweedie. Characterizing interactive externalizations. In *conference proceedings on Human factors in computing systems (CHI '97)*, pages 375–382. ACM, ACM Press, 1997.

[31] C. Upson, T. Faulhaber, Jr., D. Kamins, D. Laidlaw, D. Schlegel, J. Vroom, R. Gurwitz, and A. van Dam. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, pages 30–42, July 1989.

[32] A. Varshney and A. Kaufman. FINESSE: A financial information spreadsheet. In *IEEE Information Visualization Symposium*, pages 70–71, 125, 1996.

[33] Advanced Visualization System home page. http://www.avs.com, Feb. 1997.

[34] IBM Visualization Data Explorer (DX). http://www.almaden.ibm.com/dx/, Feb. 1997. (current as of date).

[35] IRIS Explorer home page. http://www.nag.co.uk:80/Welcome_IEC.html, Feb. 1997.

[36] J. A. Wise, J. J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. In *Proc. Information Visualization Symposium (InfoVis '95)*, pages 51–58. IEEE, IEEE CS, 1995.