UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

RECORDING AND REPLAYING USER INTENTIONS IN COORDINATED MULTIPLE VIEW VISUALIZATIONS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE

By

SUDARSHAN REDDY VANGALA
Norman, Oklahoma
2016

RECORDING AND REPLAYING USER INTENTIONS IN COORDINATED MULTIPLE VIEW VISUALIZATIONS

A THESIS APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY

Acknowledgements

I owe my deepest gratitude to my thesis advisor Dr. Chris Weaver. I am fortunate to have an advisor who allowed me to explore the jungle of research on my own and at the same time guided me in the right direction. Chris inculcated in me a sense of research by teaching how to question ideas and thoughts from different perspectives. Thanks to the committee members, Dr. Krishnaiyan Thulasiraman and Dr. Christan Grant for their time, effort, and valuable suggestions.

I would like to express my gratitude to DLL project members, especially Dr. Samuel Huskey and Dr. June Abbas for their valuable suggestions and feedback throughout this research. I would also like to thank my fellow group members Vamshi Sunchu, Bharathi Ashokarajan, Jaychandran Ratnem, Emily Grimes, Silvia Shejuti, and Akshay for being helpful and supportive. Special thanks to Dr. Mahendran Veeramaneni for helping with Latex.

I also wish to express my gratitude to staff members Virginie Perez Woods, Jamie Hoots, Emily Pierce, Lindsay Rice, Ashley Hill, and James Cassidy for their support in various stages of my masters study.

It is a pleasure to thank my dear friends Omkar Gurram, Rajasekhar Kota, Alekhya Kammadanam, Keerthi Seetha, Sree Harsha Mandem, Rajiv Janupala, Vijaykanth Sreelam, Nayan Ravuluri, Richa Saraswat, Ankitha Sinha, Isha Bhatia, Rohan G, Akash Krishnamurthy, Navesh Vijaykumar, Raghuram Vutukuru, Jayaram Patibandla, and Anish Badri for the wonderful times we have shared. You guys gave me the necessary distractions from my research and made my grad school experience a memorable and joyous one.

Special thanks to my friends at Infy – Sartre Chagantipati, Jayaditya Bandi, Sandeep Vemulapati, Nitish Neelagiri, Lakshmi Goli, Shravani Chintha, and Nazia Shareef for your encouragement, support and most of all your humor.

To my dear friend Rashmi Nayak, thank you for being there to support me through hard times even though you had your own share of problems. Our conversations have always lightened painful events in my life and helped me focus on studies.

Most of all, I wish to express my gratefulness to my parents and brother for their unconditional love, support, patience, and many sacrifices. I would not be standing here today if it wasn't for your support and encouragement. Also my sincere thanks to extended family members, especially my beloved cousins Madhusudhan Vangala and Venkat Kandukuri for their support to my family while I am away.

Contents

List of Figures

Abstract

Visualization tools help people gain insights into data. Analysts often want to revisit and review previously visited visualization states to make sense of their previous observations. Browsing a history of visualization interactions is often useful. Traditionally, history mechanisms are based on either undo-redo or replay of low-level keyboard and mouse interactions. In this thesis, we examine the feasibility of translating low-level interactions into high level user intentions for the purpose of recording user actions at a semantic level. Yi, et al. [22] taxonomize low-level interactions into seven higher level user intents: **Select, Connect, Encode, Filter, Explore, Reconfigure, and Abstract/Elaborate**. Our hypothesis is that a rule-based system can translate low-level mechanical interactions into user intentions under the Yi taxonomy.

Many visualizations are designed around the data state model [3], in which visualizations are composed of parameters, operators, datasets, and views. Dependencies between these objects define a *coordination query graph.* When a user interacts with a visualization, these objects get modified in particular sequences to process, render, and display the information. Our core idea is to define rules that map the activity in sets of connected objects in a visualization coordination graph into corresponding user intentions. By dissecting existing visualization designs, we identified and characterized distinct mapping functions for each type of intention. We then collected these functions in a set of rules for deducing user intentions.

Based on the identified mapping functions, we implemented a rule system as a new capability in the Improvise visualization environment [20], for discerning user intentions behind user interactions. User intentions detected by the rule system are recorded in an automatically generated data set to allow a user to revisit earlier visualization states. We designed a user interface to let the user query the intent data set and restore and replay past visualization states. Finally, we assessed the utility of the system for performing queries and replaying visualization history at the level of intentions.

Chapter 1

Introduction

Visualization tools help analysts discern patterns present in data by providing visual representations of data. During the visual exploration process, an analyst interacts with a visualization to pose new queries. During this exploration process, the analyst may often want to revisit and examine earlier visited visualization states. History tools can be helpful to focus on the exploration process without having to worry about remembering past results.

History tools are used in many applications. MosaicG [12] visualizes a history of documents visited in a browser in a two dimensional graphical history view. A graphical history view presents history in the form of a tree diagram. A node in the tree diagram represents page details such as title, URL address, and thumbnail image to aid the user in recognizing the documents. Users can revisit a document by double-clicking on the tree nodes. Heer, et al. [9] have investigated the design space of history mechanisms for the visualization systems and developed a branching undo-redo based history mechanism in Tableau. Vistrails [2] proposed a provenance mechanism for capturing the parameter and data flow changes in a scientific visualization to allow an analyst to navigate through different versions of data flows of an exploration task and compare them. The Revise toolkit [15] captures low-level interactions to allow users to replay, review, and explore their past interactions. Roth, et al. [6] have implemented a time travel history mechanism in the Visage visualization system to address the limitations of the ubiquitous linear undo-redo mechanism. Users can also perform selective undo operations.

The history mechanisms we looked at are based on low-level mechanical interactions, parameters and data flows, or on undo-redo stacks. Though these history mechanisms are easy to implement, we propose that users can benefit from semantically meaningful history models that aggregate low-level user actions into higher level operations which are analytically meaningful and are easier to associate with analytical tasks. Several taxonomies have been proposed to categorize low-level interactions into higher level operations, at different granularity. Shneiderman [17] proposed a taxonomy of tasks - overview, zoom, filter, details-on-demand, relate, history, and extract based on the data type of the information. Chuah, et al. [4] categorized user actions into graphical operations, set operations, and data operations. These and a few other taxonomies ([1], [7], [11]) are organized around low-level interactions and are only distantly related to a user's analytical tasks. A captured history is most useful if it is relevant to a user's own understanding of the tasks they perform.

According to Norman's action cycle [13], user goals are translated into a set of intentions. Intentions are carried out by performing a set of action commands. In visualization tools, users perform action commands by interacting with input devices such as mouse and keyboard. Though users are aware of their goals and intentions, it is not obvious how to identify and record intentions and the user's goals motivating their interactions. Previous history mechanisms are based on low-level keyboard, mouse, and other mechanical aspects of user interactions. We argue that it is desirable to record history in terms of higher level user intentions and goals, in opposition to the traditional history models.

Yi, et al. [22] categorize low-level user interactions into seven higher level user intentions: **Select, Connect, Encode, Filter, Explore, Reconfigure, and Abstract/Elaborate**. This taxonomy is interesting to us because interactions are categorized around user intentions, which are closely related to analytical tasks. Hence we chose this taxonomy of interaction as our base model for developing a new history mechanism. Our aim is to identify the user intention(s) behind user interactions under the Yi, et al. taxonomy, and record those intentions for later replay. To realize this aim, we developed a set of rules to translate low-level interactions into each of the seven user intentions. We present a user interface to allow users to query the intention log and replay the interactions involved in selected intentions. The rest of the thesis document is organized as follows:

- Chapter 2 provides background information on interactive visualization, interaction parameters, visualization operators, coordinated multiple views, and coordination graphs.

- Chapter 3 discusses the intentions in the Yi taxonomy, the approach we use to detect user intentions and the rules we identified to detect user intentions.

- Chapter 4 describes implementation of the coordinated graph structures as sets of rules to deduce user intentions, logging of the user intentions, and the user interface for browsing and revisiting histories.

- Chapter 5 provides an analysis of our rule system and user interface. We analyze the user interface in terms of the queries it can and cannot express.

- Finally, Chapter 6 outlines future directions for this work and concludes.

Chapter 2

Background

2.1 Information Visualization

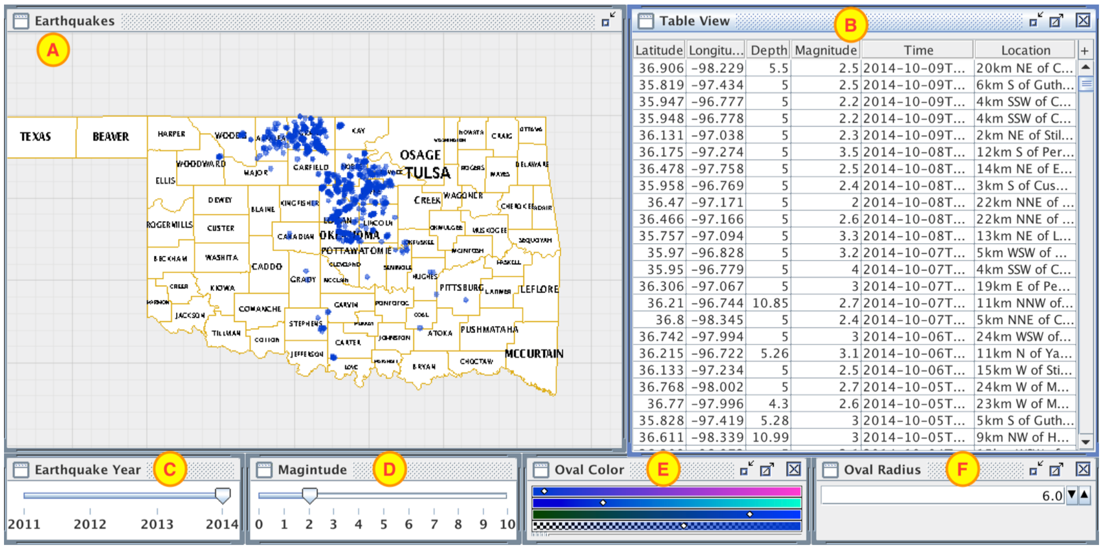
Information visualization (InfoVis) is the study and practice of using visual representations of data to help people gain insights into the data. Histograms, bar charts, pie charts, line plots, scatter plots, box plots, and tree maps are the most commonly used visual representations. A visualization can be either static or interactive. As the name suggests, a static visualization provides an unchanging image, while an interactive visualization allows the user to control aspects of the visual representation and the data it visualizes. Though a static visualization helps user understand the data, often it does not provide enough ground to answer all the questions user may have about the data especially when the data set is large and highly dimensional. Interactive visualization overcomes the limitations of static visualization by allowing the user interact with the visualization. By the virtue of interaction users can explore different dimensions and aspects of data by dynamically altering their queries.

Each visualization is a collection of one or more views. A *view* represents data by encoding it in visual channels such as shape, position, color, size, and orientation. Different attributes of the data set can be mapped to these visual encoding channels. Apart from the views, a visualization often contains additional components called *Controls*. Controls allow a user to modify input parameters that affect one or more views. Range sliders, alpha sliders, checkboxes, combo boxes, text boxes, and radio buttons are examples of controls. The difference between controls and views is that views display data where as a control shows a value of an input parameter and allows the user to modify that parameter value interactively.

Interactions define the behavior of views. User perform interactions to manipulate where they are looking and what they are looking at. The process of interaction happens in an iterative manner, in which the current state of the visualization serves as both motivation and context for the next user interaction. In other words, user interactions are motivated by the information a user is already viewing and information he would like to view.

If we consider an example of online shopping, a user interacts with a website by first picking the main category of a product they are looking for. As a result of user interaction, the website updates the list of visible items to show only relevant products. An interaction to change any sub category or price range further alters the products and updates the display. The user considers the list of products being displayed and internalizes these findings before performing another interaction. This process of information exploration repeats until user is satisfied.

Interactions transform a visualization from one state to another. Each interaction can be thought of as a function that maps a visualization from an input state to an output state. The resulting visualization state depends upon the in-



Latitude	Longitu...	Depth	Magnitude	Time	Location
36.906	-98.229	5.5	2.5	2014-10-09T...	20km NE of C...
35.819	-97.434	5	2.5	2014-10-09T...	6km S of Guth...
35.947	-96.777	5	2.2	2014-10-09T...	4km SSW of C...
35.948	-96.778	5	2.2	2014-10-09T...	4km SSW of C...
36.131	-97.038	5	2.3	2014-10-09T...	2km NE of Stil...
36.175	-97.274	5	3.5	2014-10-08T...	12km S of Per...
36.478	-97.758	5	2.5	2014-10-08T...	14km NE of E...
35.958	-96.769	5	2.4	2014-10-08T...	3km S of Cus...
36.47	-97.171	5	2	2014-10-08T...	22km NNE of ...
36.466	-97.166	5	2.6	2014-10-08T...	22km NNE of ...
35.757	-97.094	5	3.3	2014-10-08T...	13km NE of L...
35.97	-96.828	5	3.2	2014-10-07T...	5km WSW of ...
35.95	-96.779	5	4	2014-10-07T...	4km SSW of C...
36.306	-97.067	5	3	2014-10-07T...	19km E of Pe...
36.21	-96.744	10.85	2.7	2014-10-07T...	11km NNW of...
36.8	-98.345	5	2.4	2014-10-07T...	5km NNE of C...
36.742	-97.994	5	3	2014-10-06T...	24km WSW of...
36.215	-96.722	5.26	3.1	2014-10-06T...	11km N of Ya...
36.133	-97.234	5	2.5	2014-10-06T...	15km W of Sti...
36.768	-98.002	5	2.7	2014-10-05T...	24km W of M...
36.77	-97.996	4.3	2.6	2014-10-05T...	23km W of M...
35.828	-97.419	5.28	3	2014-10-05T...	5km S of Guth...
36.611	-98.339	10.99	3	2014-10-05T...	9km NW of H...

Figure 2.1: Example of an interactive visualization designed for exploring Earthquakes that occurred in Oklahoma between 2011 and 2014. The earthquake view (A) is a scatterplot in which earthquakes are represented as circles at locations identified by longitude and latitude attributes in the data set. A table view (B) lists the earthquakes in the data set used in the visualization. Views A and B display only the records which satisfies user-controlled filter critieria **(Year = 2014 and Magnitude >= 2)**. C and D are slider controls which allow the user to modify the filter criteria. Color control E, and Value control F allow the user to pick the color and the radius of earthquake glyphs respectively, to suit their preferences.

put state and the user interaction. An input visualization state, given different user interactions, usually leads to distinct visualization states. Common interactions that users perform in a visualization include panning and zooming in a scatter plot, changing a slider value, checking a check box, picking an item from a drop-down list, rotating the coordinate system of a view, marking data items, and many more.

2.2 Visualization Parameters

State of a visualization depends upon values of its parameters. Parameters are user inputs to the visualization, which can be modified by interacting. Every parameter has a data type and a value associated with it. Parameters are connected to the views. Often a parameter is connected to more than one view. Parameter values can also be referred to by data transformation operators in the data flow pipeline. A view redraws when any value of one or more parameters connected to it changes. Jankun Kelly, et al. model the visualization process, including data processing and interaction in the data flow pipeline, in terms of sets of visualization parameters (P-Set) [10]. User interaction with a view can modify one or more parameters connected to the view. When a parameter is modified, it broadcasts a change notification to all views and operators that are connected to it. Views redraw themselves upon receiving a parameter change notification.

Parameter changing allows user to dynamically choose visual queries. For example, in figure 2.1, controls C, D, E, and F allow the user to modify input parameters Earthquake Year, Magnitude, color, and radius, respectively. The Earthquake Year and Magnitude parameters currently have values 2014 and 2, respectively. The views only display records that satisfy the filter condition **(Year =2014 and Magnitude >= 2)**. If the user chooses to view only the earthquakes that have magnitude 3 and above, they can do so by moving the position of slider D from 2 to 3.

The number and types of parameters that a view accepts depends on the type of the view itself. For example a scatterplot has two (min, max) range parameters to determine the cartesian coordinate region. A timeline plot has only one navigation parameter, time range.

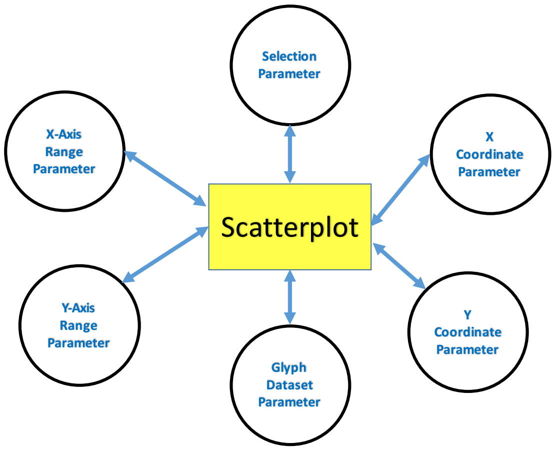


Figure 2.2: Necessary parameters of scatterplot view: X-axis and Y-axis range parameters are (min, max) range parameters to determine the cartesian region. X and Y coordinate parameters are real valued parameters for determining coordinate point inside a view. Selection parameter to capture the data items that the user has selected in scatterplot view. Glyph dataset parameter contains glyphs of all records drawable in the view.

Certain parameters are common to all views. All views have at least one glyph data set parameter which will be used to draw the information. A glyph is a graphical mark such as circle, rectangle, or text. Most views have a selection parameter to capture the data items that the user has selected in the view. Figure 2.2 shows the necessary parameters of a scatterplot view.

2.3 Operators

In visualization tools, source data often undergoes a sequence of transformations leading up to display. Common operations include filtering a data set, concatenating datasets row-wise, merging datasets column-wise, sorting a data set, removing unwanted columns, and transforming records into drawable glyphs. Entities that perform these operations are called *Operators*. Operators abstract transformations by taking inputs and generate output that can be expected by performing the operation. Commonly used operators in visualization tools include filter, sort, projection, union, join, and visual mapping(record ->glyph).

Filtering is performed to create a subset of the original data set, such that every record in the subset satisfies the filter condition. The inputs to a filter operator are a data set and a filter condition. As explained in the previous section, a filter condition may depend on multiple input parameters. A sorting operator takes a data set and a binary comparison criteria as inputs and orders the data set accordingly. A projection operator maps each record in one data set into a record in a new data set and is often employed to generate derived attributes and delete unnecessary columns prior to other transformations including glyph generation. A union operator takes two or more column-compatible data sets as input and generates a data set comprising non-redundant records from either dataset. The union operator concatenates data sets row wise. The merge operator merges two input datasets column wise and generates a data set that contains columns from both data sets.

Visual mapping operator converts each record in the input data set into a glyph to be drawn in a view. This operator has a template that defines the visual encoding channels such as shape, location, size, color, and orientation of each glyphs. Attributes of the record and parameters can both be used to define these visual encoding channels.

Our research focuses on filter, sort, projection, and visual mapping data set operators, because these are essential to define intention rules.

2.4 Coordinated Multiple Views (CMV)

Analyzing datasets is challenging. Difficulty in analyzing these datasets arises from both a large number of attributes and the complex relationships between them. Another problem is that attributes can be individually complex and may each require special attention. Some attributes, such as geographical information and time series are better suited to special types of views. To address these problems, researchers have proposed techniques to create coordinated multiple views (CMV). Using CMV, users can explore relationships between attributes by interacting with records displayed in *multiple views*. Each view may display attributes different from those displayed in other views. View can be *coordinated* so that interactions performed in a view affects other views.

The visualization in figure 2.1 is an example of CMV. When the user manipulates Earthquake Year value in slider control C, the earthquakes map view and the table view both filter the data to show only the records corresponding to that year. In other words, the slider control and the two views coordinated through the Earthquake Year parameter. Dependencies between parameters, operators, and views in a visualization form the coordination graph of the visualization. For multiple views to coordinate, views should either directly share the parameters or indirectly, the data transformations of one view depend on the parameters of another view. To create CMV, vis designers need to compose the coordination graph with the parameters, operators, data sets and views in a meaningful manner(without creating cyclic dependencies). There can be multiple data sources, interaction sources, and display sinks in a coordination graph.

Figure 2.3 shows coordination graph for the example visualization. Arrow direction represents the direction of parameter change propagation. The filter criteria of the filter operator is a function of input parameters Magnitude and Year. Similarly, the visual mapping operator of the Earthquake View depends on the oval color and oval radius parameters. The coordination graph specifies which parameters, operators, data sets, and views should update whenever user interactively modifies a parameter. For example, if the user alters the Magnitude slider's position, the Magnitude parameter changes as a result. The filter operator then updates due to the dependency on the Magnitude parameter in it's filter condition, by generating a new data set. The visual mapping operator takes the new data set as input and calculates a new glyph data set which will be delivered to the views for display. The region outlined by the dotted red line in figure 2.3 is the subgraph of the coordination graph that recalculates when user interacts with the Magnitude slider.

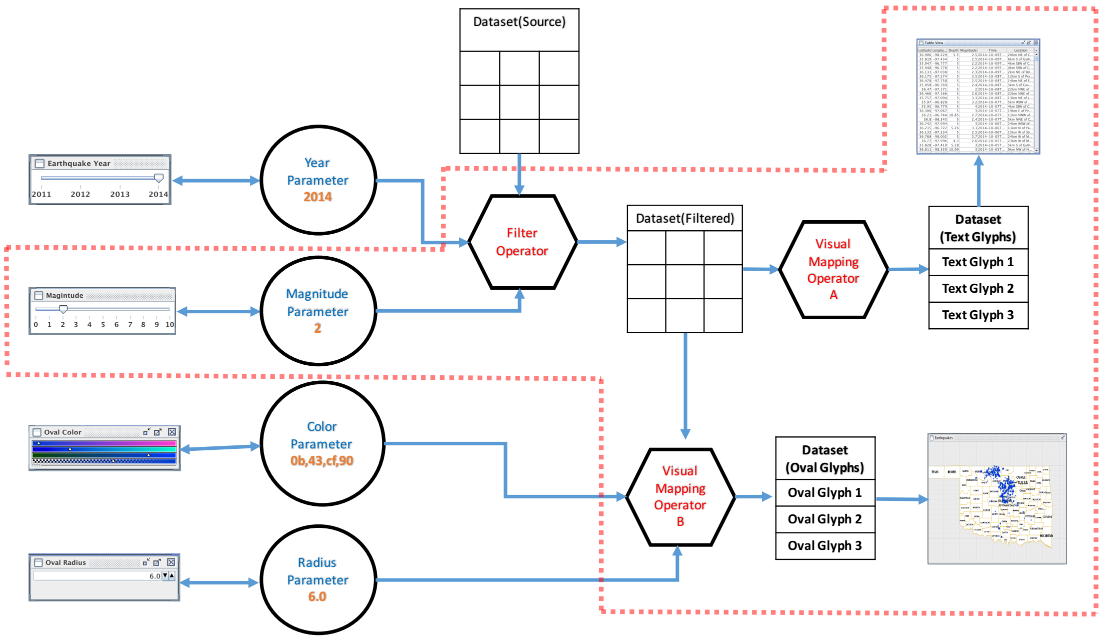


Figure 2.3: Coordination Graph for the visualization in figure 2.1. The dotted red line region contains the interaction subgraph that is dependent upon Magnitude parameter

$$\begin{tikzcd}
\text{Magnitude Slider} \arrow[r] & \text{Magnitude Parameter} \arrow[r] & \text{Filter Operator} \arrow[r] & \text{Dataset(filtered)} \arrow[r] & \text{Visual Mapping Operator A} \arrow[r] & \text{Dataset(Text Glyphs)} \arrow[r] & \text{Table View} \\
\text{Magnitude Slider} \arrow[r] & \text{Magnitude Parameter} \arrow[r] & \text{Filter Operator} \arrow[r] & \text{Dataset(filtered)} \arrow[r] & \text{Visual Mapping Operator B} \arrow[r] & \text{Dataset(Oval Glyphs)} \arrow[r] & \text{Earthquake View}
\end{tikzcd}$$

Figure 2.4: Coordinated query paths resulting from user interaction in magnitude slider

Figure 2.4 further dissects the subgraph into two individual paths that define the individual effects of interaction on the earthquake and table views. The view in which the user interacted, and the view in which interaction effects are seen, are referred as the source view and the target view, respectively. In the example, the Magnitude slider is the source view. The table and earthquake views are the target views.

Chapter 3

User Intentions

Our goal is to log user actions and allow users to revisit the history by searching the action log. To achieve our goal, we can track mouse and keyboard changes that happen in the system whenever the user interacts. However, these interactions are too mechanical and low-level to track. Interactions can also be tracked at the parameter level by logging parameter changes. Though this approach is feasible, it requires a lot of effort and memory to use the log. Hence we aim to log user interactions at a level higher than raw interaction events. Yi, et al. categorized low-level user interactions into seven higher level interactions based on user intentions. The seven higher level intentions are:

- **Select**: Mark something as important.
- **Connect**: Show me related items.
- **Encode**: Show me a different representation.
- **Filter**: Show me something conditionally.
- **Abstract/Elaborate**: Show me more or less details.

- **Explore**: Show me something else.
- **Reconfigure**: Show me a different arrangement.

To log user interactions at the user intention level, we need to deduce the analytic connotation of user interactions. When a user interacts with the system, data available for probing user intention come from keyboard, mouse, and parameter change events. It is hard to deduce user intention from keyboard and mouse events directly, as these events are mechanical. However, keyboard and mouse interactions modify parameters. Parameter changes reflect user intention more meaningfully. However, looking at modified parameter value changes alone is still not sufficient to identify user intention.

Many coordinated multiple view visualizations are built around the data state model [3], in which data undergoes analytical, visualization, and visual mapping transformations before being displayed in views. According to this model, when a parameter value changes, it affects other entities such as operators, datasets, and finally views themselves. Changes in parameters, operators, datasets, and views taken together, and the sequence in which these entities have changed, define the effects that user interaction has on the visualization overall. We use the coordination subgraph involved in any given parameter change to determine user intentions. (We use the terms *interaction subgraph, coordination query subgraph*, and *subgraph* interchangeably to refer to the interconnected components of a visualization that change as a result of user interaction.)

Our hypothesis is that the coordination query sub graph involved in a given user interaction provides a sufficient fingerprint to identify and characterize user intention. To develop this hypothesis, we looked at the coordination graphs of existing visualizations to dissect the various ways that visualization designers compose coordination graphs to allow user to express the seven intentions. We identified patterns of subgraph structure for all seven higher level intentions. Each of these seven higher level intentions have one or more coordination subgraphs, all distinguishable from each other. Having distinguishable coordination structures allow us to map interactions to intentions with sufficiently low ambiguity. In the following sections we briefly explain each of the seven user intentions and the coordination subgraphs that we identified for each one.

3.1 Select

Users perform selection to highlight data records of interest. Selection helps user keep track of particular items during ongoing navigation and other kinds of visual querying, such as rotating or filtering views. Typically, visualization designers chose a distinct fill or edge color to differentiate the glyphs of selected records from unselected ones. Designers can use other visual encoding channels, such as shape, size, or orientation, to distinguish selected from non-selected data items. For instance, selected records can be encoded as a rectangular shape, different from the circular shape of a non-selected record, or the size of the selected records could be much larger than the non-selected records.

Users can perform selection intention in variety of ways, such as clicking, making a lasso or a rubber band selection, or by hovering over data items. In visualization systems, selections made using such methods are represented internally as a special parameter value that parallels records in the visualized dataset. The selection parameter could be represented either as an array of booleans or an array of bits (in Improvise, selection parameters are based on the Java BitSet class). An on at index position i in the selection array means that the record

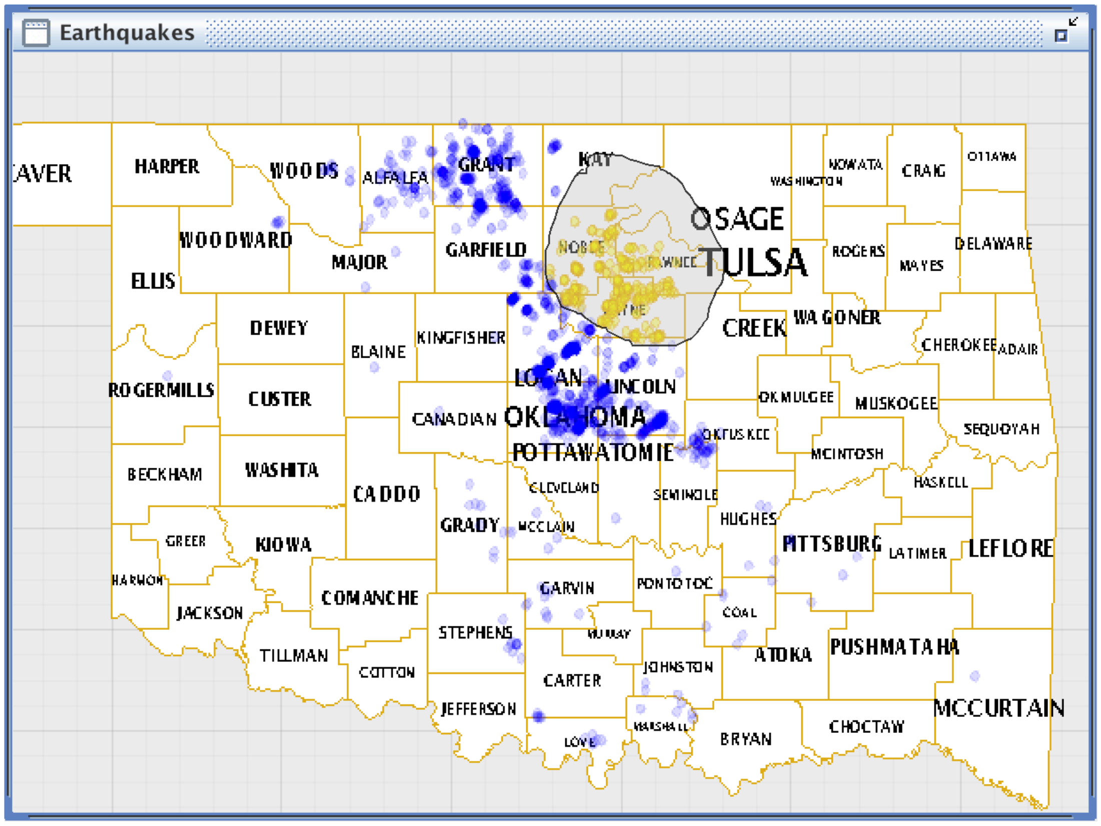


Figure 3.1: Example of the Select interaction. All the earthquake glyphs lying within the lasso region are selected. Selected earthquakes are highlighted in yellow. Non-selected ones are represented in blue.

with record ID i in the data set is currently selected.

When a user performs selection in a view, the selection parameter bound to the view (if any) gets updated. Figure 3.1 shows an example of a select interaction. After selection is performed, to differentiate the selected from non-selected records, the visual mapping operator checks if the record is set to on in the selection parameter. It uses this information to determine the shape, color, and size of the record's glyph. Because the visual mapping operator is dependent on the selection parameter, whenever the selection parameter changes, the visual mapping operator is notified and recalculates it's glyph data set. The updated glyph data set will be given as input to the view for rendering. Figure 3.2

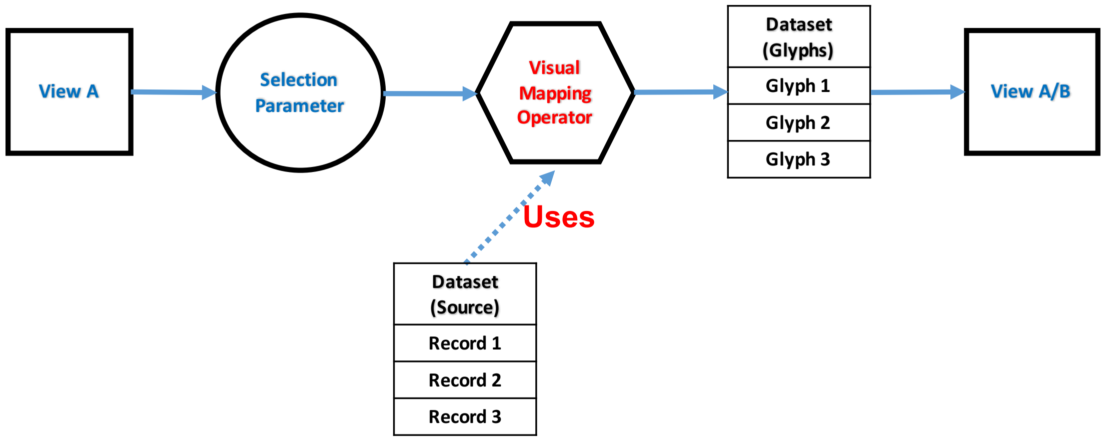


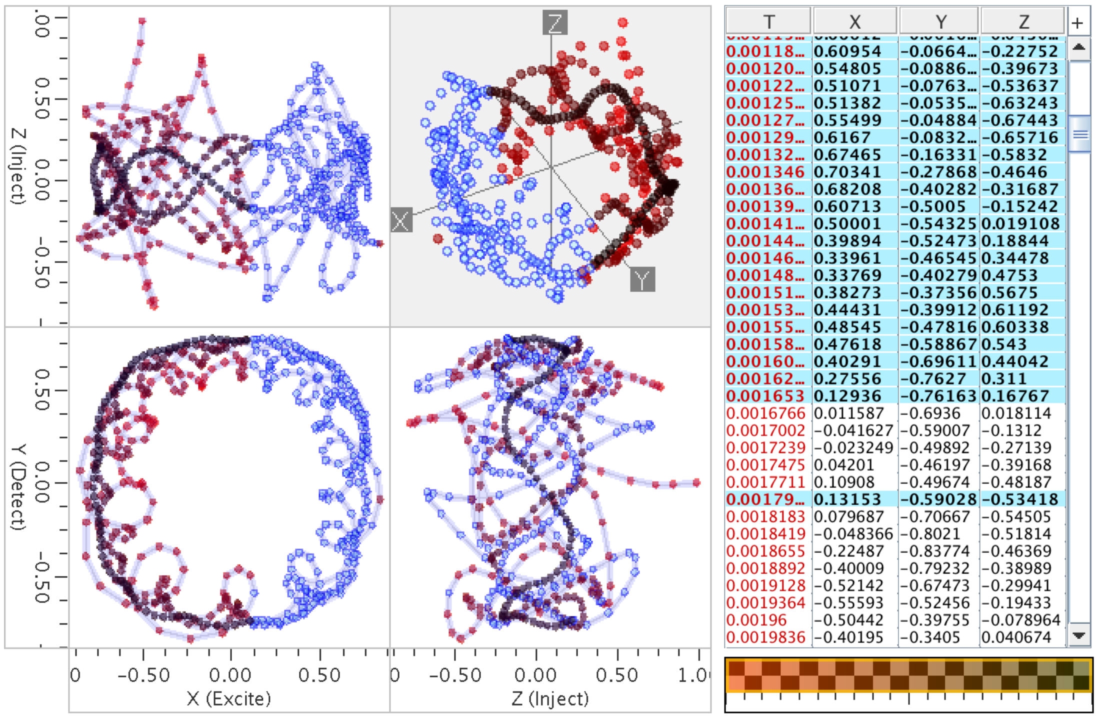
Figure 3.2: Subgraph for common Select intentions

shows the generic structure of coordination query subgraph that is impacted when selection is performed in a typical Improvise visualization.

Brushing is another form of selection interaction. In a CMV visualization, when two or more views visually represent the same records, brushing is often used to visually highlight the selected records in some or all of those views at the same time. Figure 3.3 shows an example of visually linking multiple views through selection. Selection performed in one view highlights the records in all other views that share same data set and selection parameter. As shown in figure 3.2, the target view can be the locus of interaction (view A) or a different view (view B). If the target view is A, highlighting is observed directly as selection is performed. If the target view is B, records are highlighted indirectly as selection is performed elsewhere.

Yi, et al. categorize brushing as a connect intention. A connect intention explicitly reveals relationships between different records. However, in the case of brushing, the same records are implicitly connected by virtue of being highlighted in different views. Hence we treat brushing as a selection intention.

The selection methods discussed up to now directly affect the selection pa-



T	X	Y	Z
0.00118...	0.60954	-0.0664...	-0.22752
0.00120...	0.54805	-0.0886...	-0.39673
0.00122...	0.51071	-0.0763...	-0.53637
0.00125...	0.51382	-0.0535...	-0.63243
0.00127...	0.55499	-0.04884	-0.67443
0.00129...	0.6167	-0.0832...	-0.65716
0.00132...	0.67465	-0.16331	-0.5832
0.001346	0.70341	-0.27868	-0.4646
0.00136...	0.68208	-0.40282	-0.31687
0.00139...	0.60713	-0.5005	-0.15242
0.00141...	0.50001	-0.54325	0.019108
0.00144...	0.39894	-0.52473	0.18844
0.00146...	0.33961	-0.46545	0.34478
0.00148...	0.33769	-0.40279	0.4753
0.00151...	0.38273	-0.37356	0.5675
0.00153...	0.44431	-0.39912	0.61192
0.00155...	0.48545	-0.47816	0.60338
0.00158...	0.47618	-0.58867	0.543
0.00160...	0.40291	-0.69611	0.44042
0.00162...	0.27556	-0.7627	0.311
0.001653	0.12936	-0.76163	0.16767
0.0016766	0.011587	-0.6936	0.018114
0.0017002	-0.041627	-0.59007	-0.1312
0.0017239	-0.023249	-0.49892	-0.27139
0.0017475	0.04201	-0.46197	-0.39168
0.0017711	0.10908	-0.49674	-0.48187
0.00179...	0.13153	-0.59028	-0.53418
0.0018183	0.079687	-0.70667	-0.54505
0.0018419	-0.048366	-0.8021	-0.51814
0.0018655	-0.22487	-0.83774	-0.46369
0.0018892	-0.40009	-0.79232	-0.38989
0.0019128	-0.52142	-0.67473	-0.29941
0.0019364	-0.55593	-0.52456	-0.19433
0.00196	-0.50442	-0.39755	-0.078964
0.0019836	-0.40195	-0.3405	0.040674

Figure 3.3: Brushing Interaction: Selection is performed in top left scatterplot of the visualization to highlight the records. Selected records are highlighted in blue in remaining scatterplots and table view.

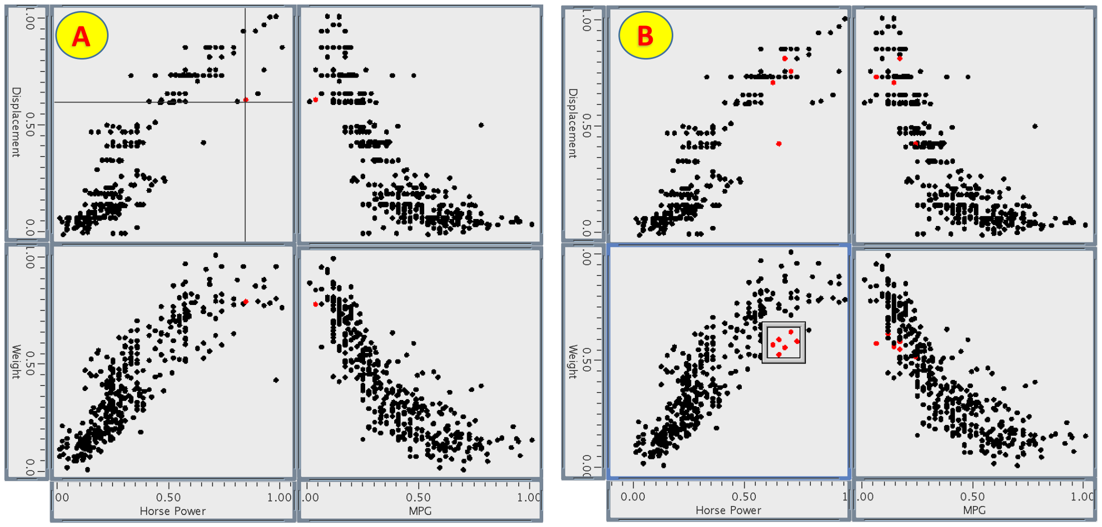


Figure 3.4: (A) Selection defined by mouse hover coordinates (x, y). (B) Selection defined by a movable rectangular region. Selected records are highlighted in red color.

rameter in the visualization system. Visualization designers provide alternative ways to define selection without depending on the selection parameter. For instance visualization designers may allow users to drag and stretch a persistent rectangular region (a Lens) in a view, to highlight the records present within the region (see figure 3.4). Similarly, users can hover over a point in the view to highlight the data item(s) beneath it. In these situations users are not manipulating a selection parameter, but rather non-selection parameters that determine record highlighting. Figure 3.5 shows the interaction subgraphs involved in select interaction performed using a rectangular region and a point.

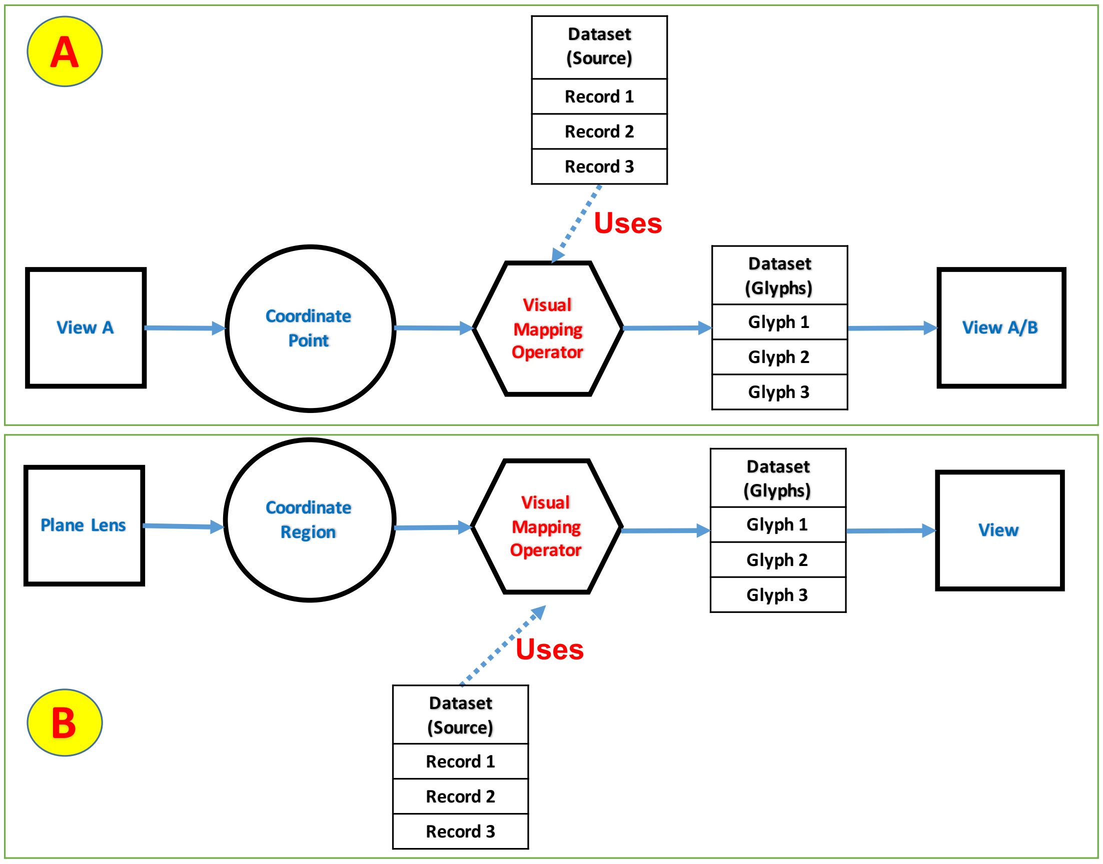
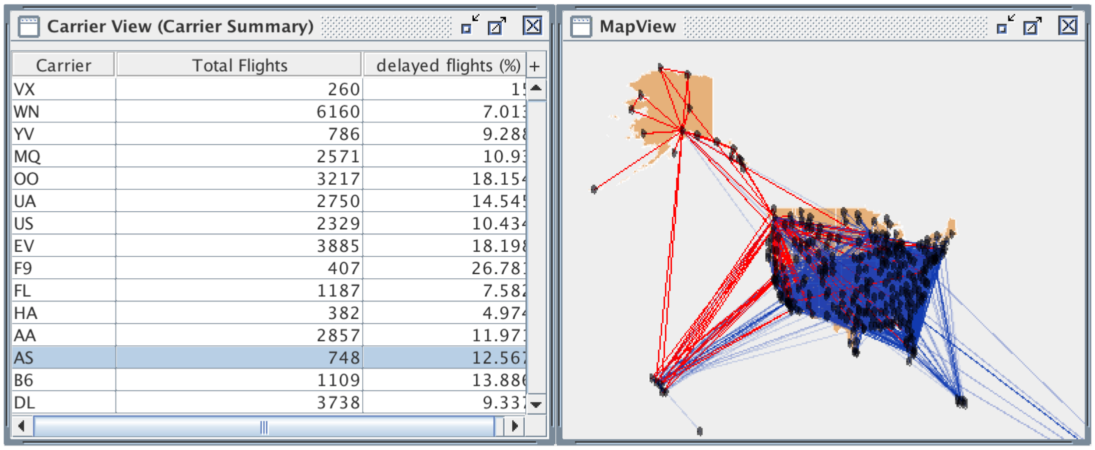


Figure 3.5: (A) Coordination subgraph of selection performed by dragging and stretching a persistent rectangular ”lens”. (B) Coordination subgraph of selection performed by hovering over coordinate point (x, y).

3.2 Connect

A connect intention reveals relationships between data entities through interactive selection. When user performs selection, the entities related to the selected ones get highlighted through some common characteristic. This is in contrast to selection intentions, in which identity determines highlighting. Figure 3.6 shows an example of connect activity involving two distinct datasets. A map view visualizes commercial flights data by drawing a line for each flight from origin to destination. A table view shows aggregate flights by carrier. When a user selects a carrier in the table view, the flights associated with the selected carrier are highlighted in red color on the map. Flight records and carrier summary records are connected visually through carrier name attribute using color. By selecting a carrier, the user is able to see airports between which the carrier operates flights. In this example, the user can perform connect interactions to explore carrier-flight relationships.

Figure 3.7 shows the subgraph that is involved in the connect interaction. To



Carrier	Total Flights	delayed flights (%)
VX	260	1
WN	6160	7.01
YV	786	9.28
MQ	2571	10.9
OO	3217	18.15
UA	2750	14.54
US	2329	10.43
EV	3885	18.19
F9	407	26.78
FL	1187	7.58
HA	382	4.97
AA	2857	11.97
AS	748	12.56
B6	1109	13.88
DL	3738	9.33

Figure 3.6: Example of a connect interaction between different Datasets/Entities. Map view highlights the flights that belong to the selected carrier(s) in red color to reveal carrier-flight relationships.

connect the selected entities of data set A with the non-selected related entities of data set B, a new data set that contains only the selected items of data set A needs to be generated by applying a filter operation. The filter condition tests if the record is selected. Once the data set of selected records is generated, a lookup operator uses this data set to generate an internal index table on the characteristic on which the connect needs to be performed. The visual mapping operator uses the index to look up whether the record in data set B has a relationship to the selected records. The visual mapping operator highlights the corresponding glyph if it's record in data set B is related to any of the selected records.

Connect reveals relationships between different types of objects present in two datasets. The datasets are typically distinct but may well be the same. For instance, a view shows employee records of an organization and the user interaction is to highlight all employees who belong to the same department as a selected employee. In this case, the user is trying to explore a relationship between objects (employees in the same department) within a single data set using a connect interaction.

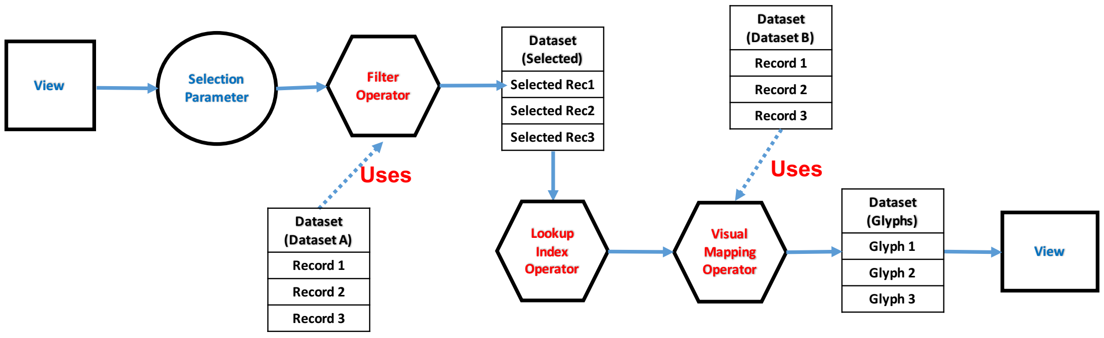


Figure 3.7: Coordination query subgraph for the Connect interaction

3.3 Encode

According to Yi, et al., any interaction that modifies the visual representation of the data is categorized as encode intention. An encode intention allows users to choose the visual representation that works best for them. As explained in the previous sections, visual mapping operator determines shape, color, size, position, and orientation of the glyphs using data attributes, parameters, and constant values. Altering any parameter which in result changes visual encoding of the data items in one or more views is treated as encode intention. Users can change visual encoding of data in two ways: Firstly, updating the parameters that are used to determine the encoding channels, and secondly, by updating the attributes that are used to determine encoding channels.

Figure 3.8 shows the coordination query subgraph that involves in the encode intention of first type. Example interactions that have this type of sub graph are: changing the color scheme used for representing the glyphs, modifying the width and hight of the glyphs as shown in figure 2.1. By interacting, users modify parameters which are used by visual mapping operator. Visual mapping operator creates new glyph data set because the input parameters have changed. However, selection parameter is an exception to the encode intention. As explained in the previous section, when user performs selection, visual mapping parameter changes as a result of change in selection. Selection can be thought of as a special attribute of data set that contains value as either true/false or 0/1. So any changes to visual mapping parameter as a result of change in selection, can be considered as a change in data rather than change in the encoding of the data.

Second type of Encode interaction involves changing the attributes assigned to encoding channels color, width, and hight. Visualization in figure 3.9 provides

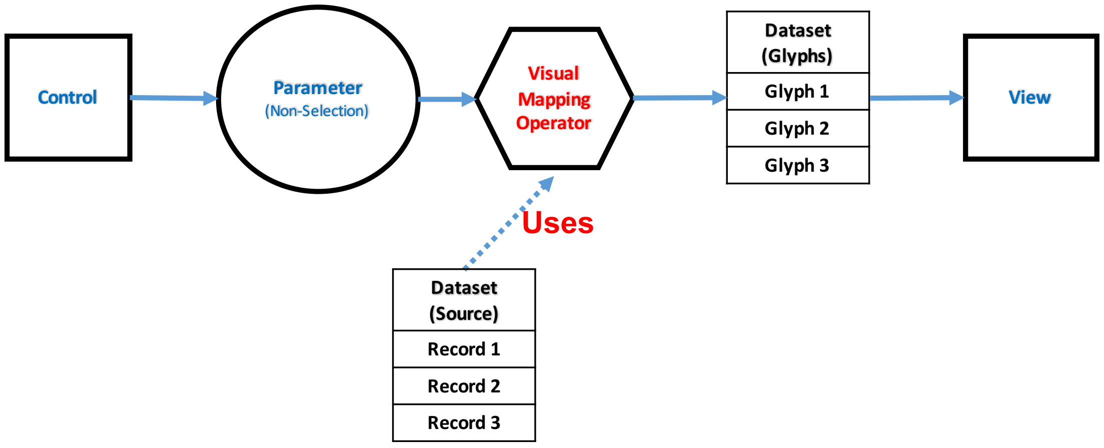


Figure 3.8: Coordination query subgraph for the Encode interaction

a way for user to choose the attributes used for determining color and size (width and hight) of the oval glyphs. According to the current encoding scheme, depth and magnitude attributes of earthquake data set are determining size and color of the oval glyphs respectively. Earthquakes having more depth are represented with bigger ovals compared to the ones having lesser depth. Similarly, color of the earthquakes having more magnitude are drawn in darker red color compared to earthquakes having smaller magnitude. Using this encoding, users can easily see that the earthquakes having higher magnitude have less amount of depth to them and vice versa. With the current encoding scheme, users can explore Magnitude, Depth attributes, and relationships between them. With the help of encode interaction, users can explore different attributes and relationships between them by interactively changing the encoding scheme.

Figure 3.10 shows the coordination query subgraph that involves in dynamically changing encoding scheme. User updates encoding scheme by changing attribute selection. As attribute selection has changed, selection index operator identifies the index number (attribute ID) of the selected attribute and inputs the same to visual mapping operator. The visual mapping operator encodes at-

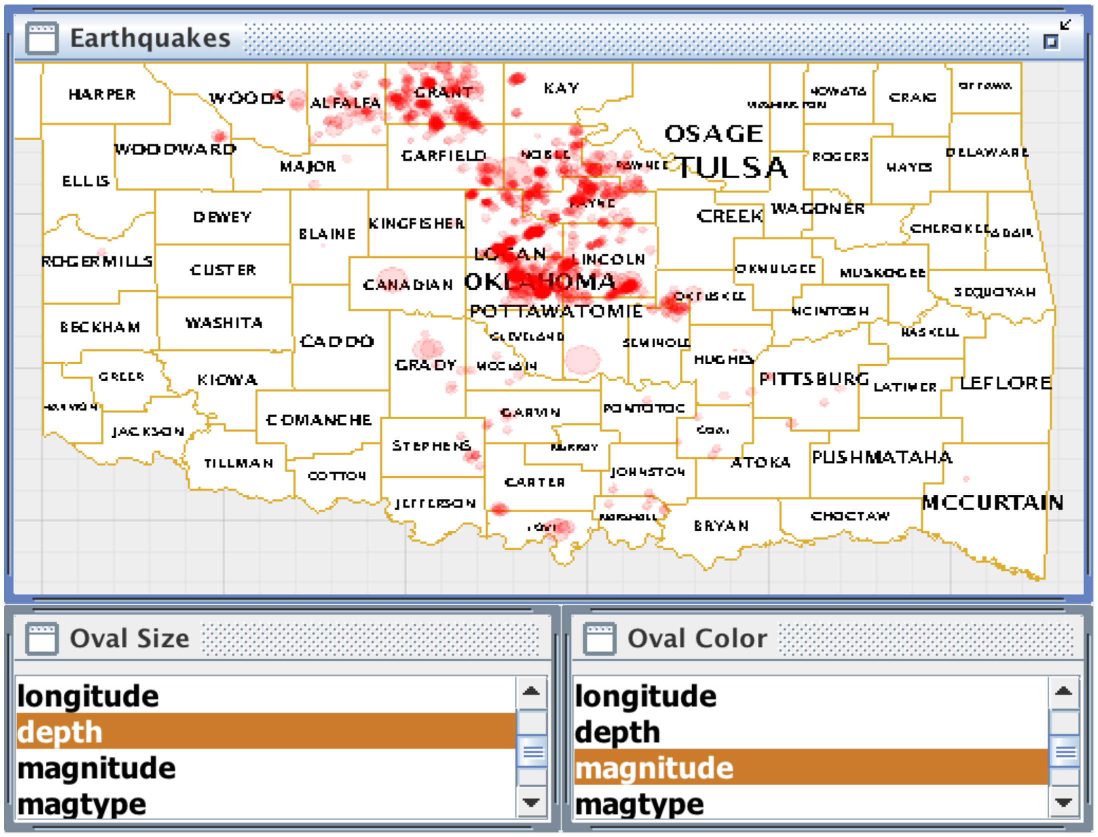


Figure 3.9: Encode intention: Changing data attributes used to calculate size and color of the earthquake glyphs.

tribute value present at the position given by index number to either size, color, or shape of the glyphs.

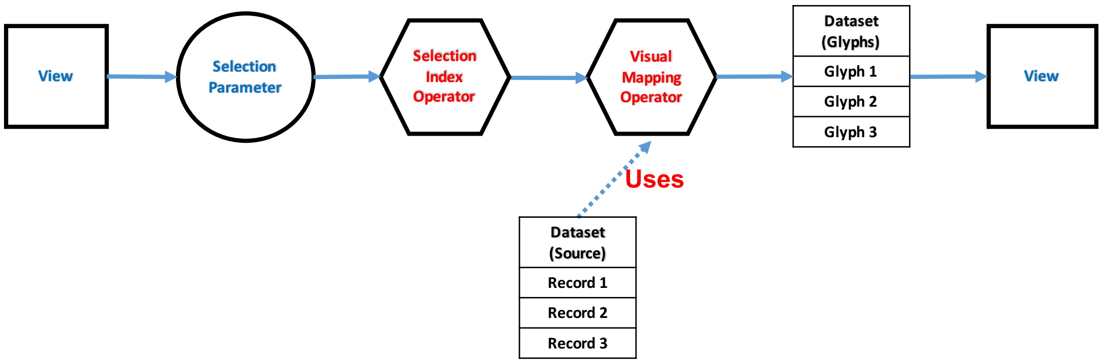


Figure 3.10: Coordination query subgraph for encode intention.

3.4 Filter

Filter interactions allow users to visualize subset of data items which are satisfying certain criteria specified by filter condition. If a record satisfies condition specified by the filter operator, then the record will qualify to be presented in the view, otherwise it will be ignored. Users can change the filter conditions dynamically by manipulating the parameters on which filter operator is depending on. Filter is applied on the input data set to create a new data set that contains all the records satisfying the filter criteria. Once the new data set is generated, visual mapping operator transforms the data set into glyphs to be visualized in views.

Figure 3.11 shows the rule for detecting the filtering interaction. Users interact with controls such as range sliders, check boxes, and dropdown menus to filter the records. Direct manipulation of these controls modify parameters connected to them. Parameter change propagates to filter operator as the filter condition depends on the value of the parameter. Change of the filter condition forces the filter operator to generate new data set. '*' on the edge suggests that the new data set may undergo optional operations such as sorting and projection before being visualized.

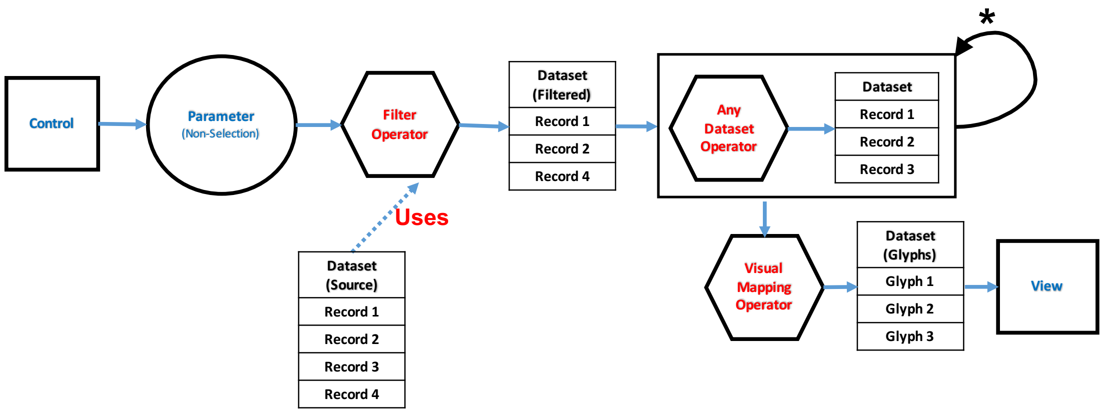


Figure 3.11: Coordination query subgraph for the Filter interaction

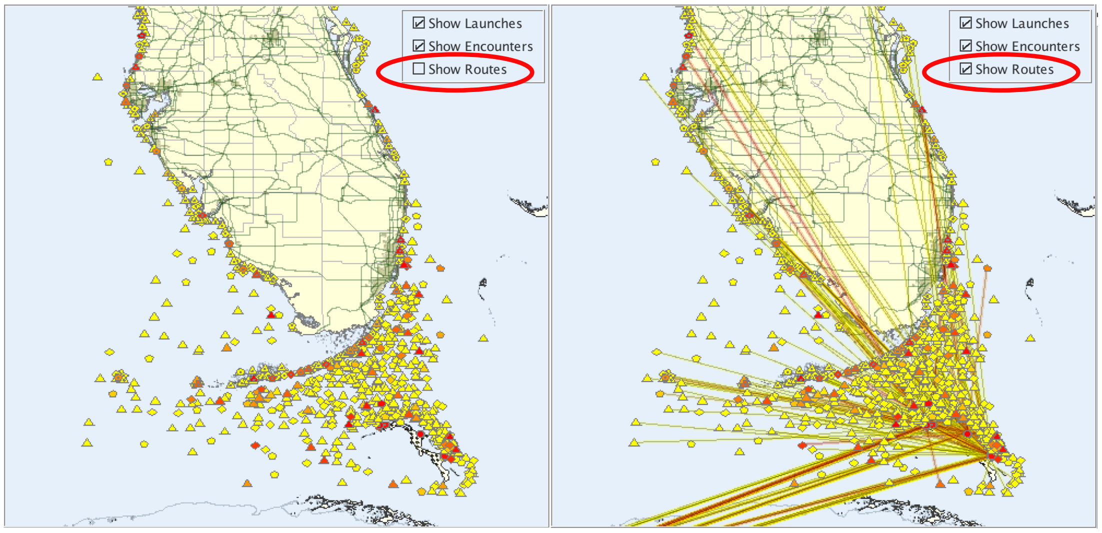


Figure 3.12: Example for filter interaction using masking approach

Masking is another filtering mechanism vis designers employ when layered approach is used to construct a view. As shown in figure 3.12, multiple layers of a view can be used to show different information. In the example, 'Show Routes' check box allows user to make layer showing information about boat routes either visible or invisible. This filtering allows user to make unwanted information on a layer invisible to reduce cluttering in a view. Figure 3.13 shows the interaction subgraph that involves in masking filter interaction. When user interacts with control, the boolean parameter connected to the view gets updated. Modified boolean parameter then notifies view that it has changed. View then makes the corresponding layer either visible or invisible based on the value of the boolean parameter.

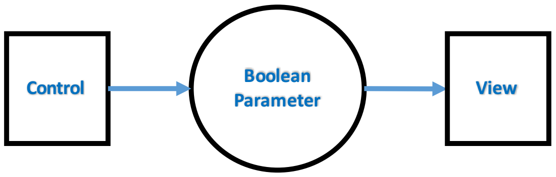


Figure 3.13: Coordination query graph for filter interaction using layer masking

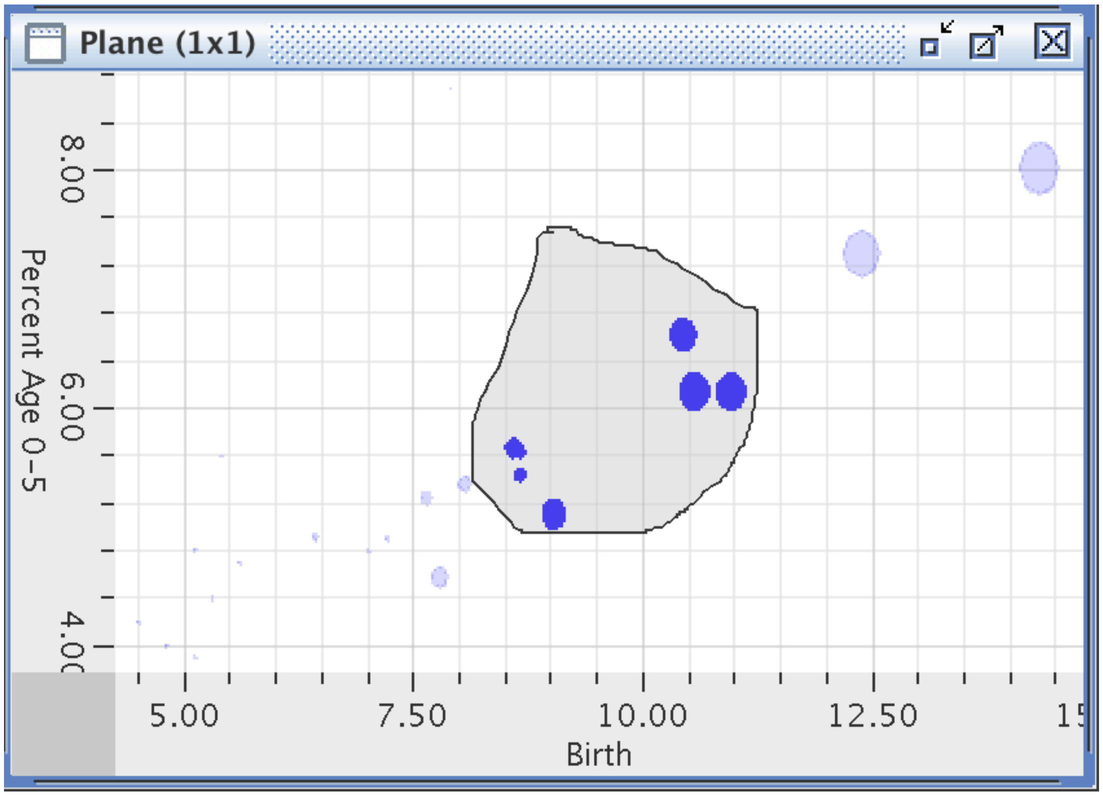


Figure 3.14: Perceptual filter: Selected records are shown in bright colors. Non-selected records are shown in translucent colors.

Perceptual Filter is another filtering mechanism which makes non-selected entities almost transparent and selected entities brighter as shown in figure 3.14. Since visual mapping operator calculates translucency based on selection parameter of the view, interaction subgraph of perceptual filter matches with that of Select interaction. In scenarios like these, it is not possible to distinguish select from perceptual filter without analyzing the color values visual mapping operator uses to represent both selected and non selected. However, analyzing the individual parameter and operator values is beyond the scope of this research. To address collision between the two graph structures of these two interactions, we chose to log only select interaction because of two reason, firstly user has performed selection to pick objects he is interested in and secondly, the non-selected items are not completely removed from the view, instead these are made more transparent than the selected ones.

3.5 Abstract/Elaborate

Abstract/Elaborate interaction enables user to view data at different levels of abstraction. Elaborate is to view information at detail level while Abstract is to visualize data at a overview or summary level. Because of the limited screen size and perceptual and cognitive limitations of human, all the information cannot be visualized at once. It makes sense to first get a overview of the data before delving into the details. overview - select - filter - details, Semantic Zooming, and Tooltip interactions are the examples of Abstract/Elaborate.

In case of overview-select-filter-details, overview often presents aggregated information in the forms of categories that best represents the information. Detail View presents elaborate information of the selected category in the overview. Upon selecting one or more categories in overview, detail view visualizes information specific only to the selected ones. Figure 3.15 shows the interaction subgraph of overview - select - filter - details interaction. Filter operator A takes selection parameter and data set of Overview and generates a new data set which contains list of selected overview categories. Filter operator B takes detail data set and data set of selected overview records as input and generates a new data set which contains records of detail data set which belongs to selected overview categories only. Filter criteria is to check whether or not record from detail view belongs to any one of the selected categories.

When user performs selection in overview, the selection parameter of the view gets updated. As the selection parameter of overview is changed, filter operator

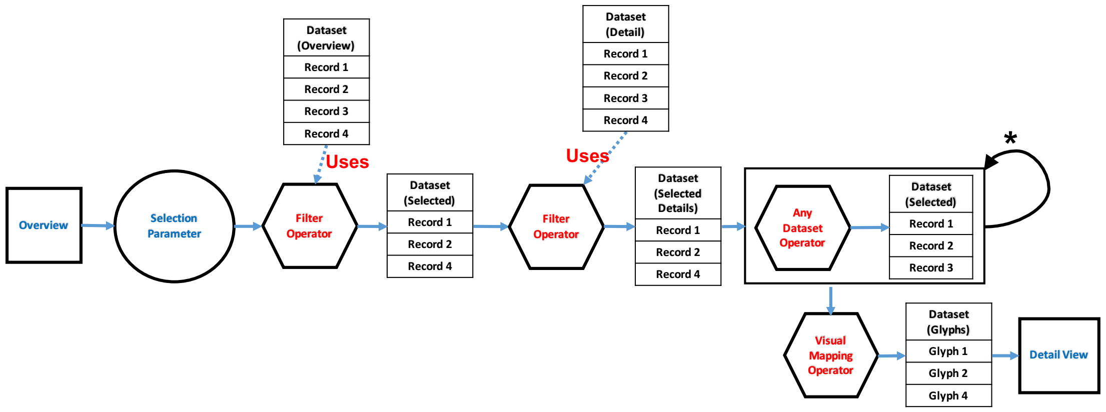


Figure 3.15: Interaction subgraph for the Overview - Select - Filter - Details interaction

A generates a new data set of selected categories. Since the selected categories data set has changed, filter operator generates a new detail data set to reflect the newly selected overview categories. As explained in previous section, the subsequent optional operations such as sorting and projection can be applied on new detail data set before visually transforming the data set into glyphs.

Tooltip details is another widely used Abstract/Elaborate interaction. Using Tooltips, a user can mouse hover on a data item to view additional details of the item as shown in figure. Figure 3.16 shows interaction subgraph for the tool tip interaction. Hovering can be captured in different ways. Two most commonly observed mechanisms to capture hovering are through coordinate point (x, y) and selection parameter. Coordinate point (x, y) parameter which is location of mouse in a scatterplot view, can be used to determine whether or not a data item is close to point (x, y). A record can be treated as hovered item if it is beneath mouse position. Capturing mouse hover through selection parameter is explained in previous section. Filter operator generates a new data set that contains only hovered item. The hovered record then visually transformed into a glyph to be visualized in tooltip view.

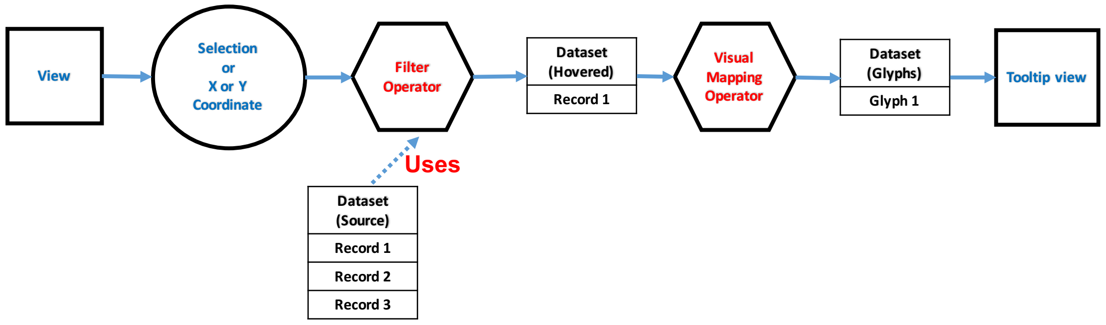


Figure 3.16: Coordinate query subgraph for the Tooltip(Elaborate) interaction

Semantic Zooming is another commonly used technique which can be classified under Abstract/Elaborate interaction. Semantic zooming technique is applied to views where range is one of the fundamental parameter. Scatterplots visualize information present with in coordinate region defined by X and Y-axis extent. Similarly timeline plots presents information within a time range. In these kind of views, Semantic zoom-in on horizontal and/or vertical axis will provide more details, while zooming out would provide less details. The idea is to decide on what level of details need to be presented based on the extent of the axis or zoom level. So the extent of the axis decides the records to be visualized in the view. Google maps zoom-in and zoom-out is an example for semantic zoom interaction. Depending on the amount of zoom, level of information varies from countries, states, cities, and streets.

When user zooms in or zooms out, the extent of the axis is modified. Change in the extent of the axis results in recalculation of the data set that needs to be visualized. Hence parameter change graph looks as shown in figure 3.17. The filter operator calculates extent/zoom level using input range parameters and calculates new data set to be visualized at that zoom level. The new data set may undergo optional data transformations such as sorting and projection before getting visualized.

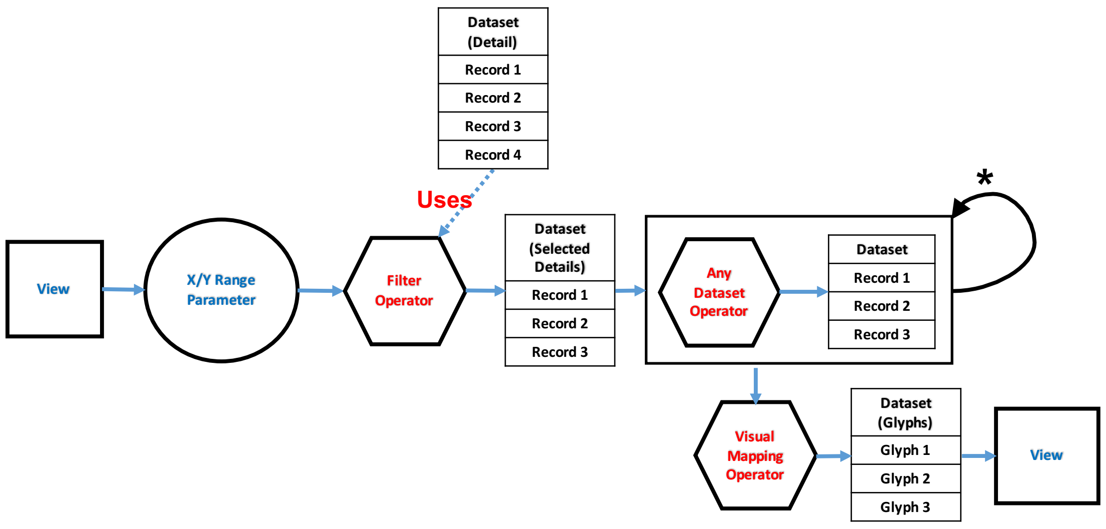


Figure 3.17: Parameter change graph for the Semantic Zoom(Abstract/Elaborate) interaction

3.6 Reconfigure

According to Yi, et al., reconfigure intention allows users to alter data arrangement in a view. This interaction is useful because at times single arrangement of the data might be not be sufficient for the data exploration and require different arrangements of the records. Ordering a data set is an example of reconfigure interaction. Records can be ordered based on particular characteristics of the data set either in increasing or decreasing order. Figure 3.18 provides an example of reconfigure interaction. Records are currently ordered on horsepower attribute of the vehicles. Users can change the attribute to be used for sorting by altering selection in the ‘Order By’ control. Users can also choose the order by checking or unchecking the ‘Order’ checkbox. When user checks the checkbox, records are sorted in ascending order, otherwise ordered in descending order.

Figure 3.19 shows the rule for detecting sort reconfigure intention. When user changes attribute selection, selection index operator identifies the index of

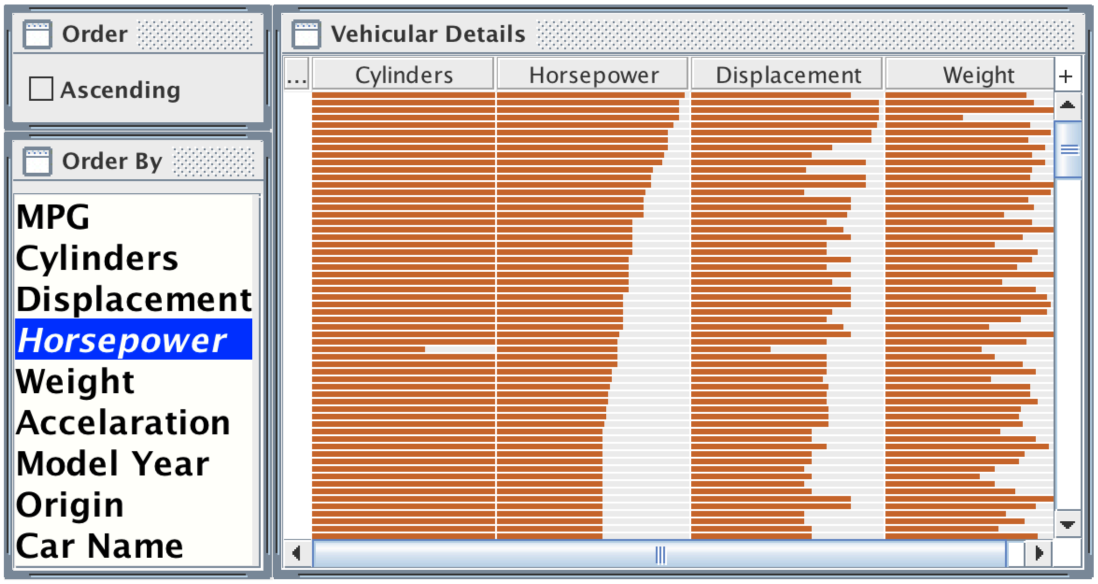


Figure 3.18: Example for reconfigure intention.

selected column. Since index of the selected column changes, sort operator orders the data set with respect to the attribute present at the index given by selection index parameter.

User reconfigures data set by changing attribute selection. As attribute selection has changed, selection index operator identifies the index number (attribute ID) of the selected attribute and inputs the same to sort operator. Sort operator generates a new data set by ordering the input data set based on the attribute present at location given by index number. The new data set may undergo optional data set operations such as projection and filtering before getting visualized. Similarly, user can alter a parameter to decide the order in which data set should be sorted.

Changing the data attributes assigned to position (x, y) of glyph is another interaction Yi, et al. classified as reconfigure. However we already categorized this interaction under encode intention because, assigning attributes to position of glyph is encoding of data attributes into position.

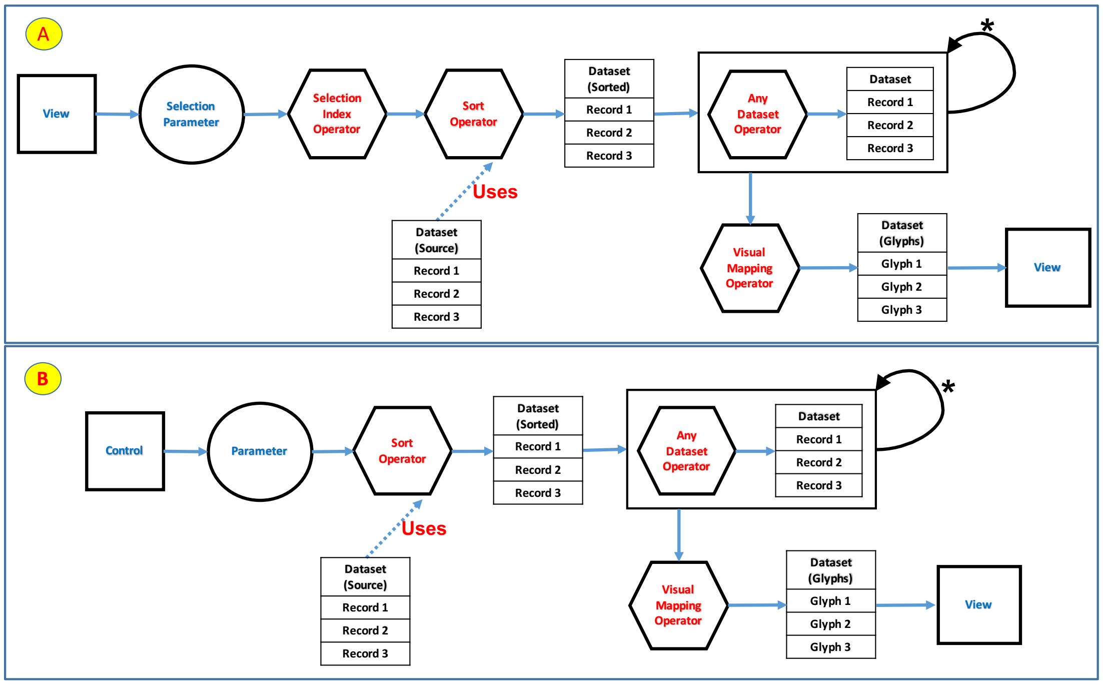


Figure 3.19: Interaction subgraphs for reconfigure intention. A: Changing attributes used for ordering. B: Changing the order(Increasing or Decreasing).

3.7 Explore

Due to the limitations of screen size, large datasets and human perceptual and cognitive limitations, often views display only part of the data set. Explore interaction lets users analyze a section of data set before move onto the another section of data set. Most common explore intention technique is panning and zooming interaction. Panning is performed by grabbing the scene and moving it while camera stays still. Panning alters the X-Range and/or Y-Range of the view, i.e. changes region of the view to be shown. Hence panning is most suited to the views which display information within a range such as scatter plots and timeline plots. Users perform panning in google maps to view the geographic region of their interest. In timeline plots users alter time range by panning to visualize records within the modified time range. Figure 3.20 shows the subgraph

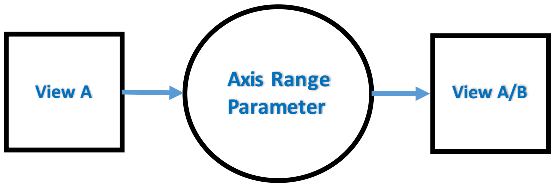


Figure 3.20: Interaction subgraph for explore intention.

involved in panning explore interaction. When user is panning, view assigns new value to the axis range parameter. Since the range parameter is modified, it sends parameter change notification to all the views it is connected to including the view in which panning is performed.

Yi, et al. have categorized syntactic zooming as Abstract/Elaborate intention. However, we consider zooming as explore interaction because when user is zooming, he is navigating in the information space.

Chapter 4

System Implementation

This chapter describes design and implementation details of our system. We implemented our history mechanism as part of Improvise [20] visualization environment. Implementation task of system is divided into subtasks as follows:

1. Identification of coordination query path for user interaction
2. Implementation of rule system
3. Detecting user intentions by matching coordination of query paths with the rules
4. Recording user intention details
5. Querying and Revisiting earlier visualization states

In the following sections, we will explain design and implementation choices we made to achieve the above tasks.

4.1 Coordination Query Path

As explained in section 2.4, when user interacts with the visualization, only subgraph of the coordination query graph involves in user interaction. However it is difficult to construct the entire interaction subgraph by using the parameter notification mechanism since parameter notifications moves forward and never retreats like functional calls does. It is possible to construct the coordination query path from source view to target view using the notification mechanism. In the following section, we will explain Improvise coordination architecture used to extract the coordination query paths involved in user interaction.

Improvise

Improvise is a coordinated multiple view visualization system. Improvise architecture facilitates seamless coordination between multiple views through shared parameters. Figure 4.1 shows the coordination architecture of improvise. Terminology of improvise architecture is different from data set model. Parameters in Improvise are referred as *variables*. Similarly filter, sort, projection operators are referred as *Lexical Filter, Lexical Sort, and Lexical Projection* respectively. Datasets are referred as *Lexical Info*.

Improvise system brings parameters, operators and datasets under same umbrella by assigning lexical as value to the variable. By assigning lexicals to variables, Improvise treats operators and datasets as special parameters. As shown in figure, parameter change notification goes through all the variables of parameters, operators and datasets before it reaches view. With the help of notification protocol, we can easily identify the objects of the coordination graph that involve in user interaction.

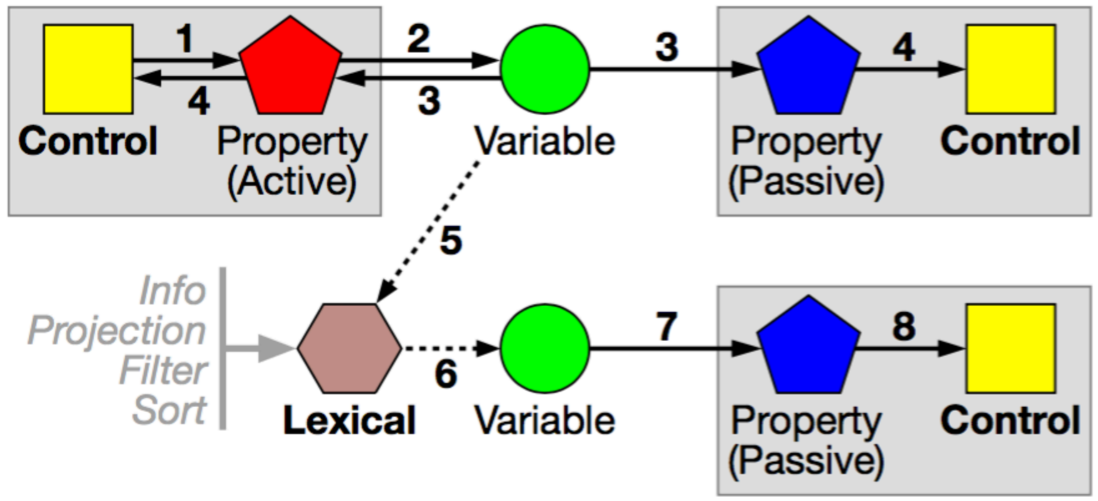
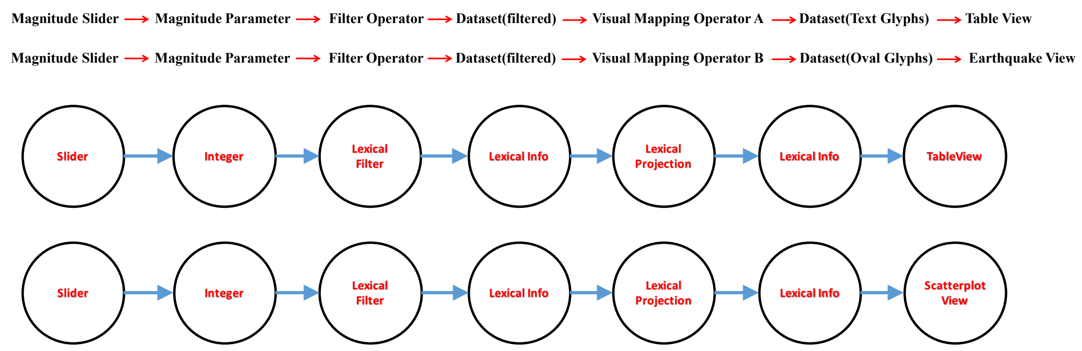


Figure 4.1: Used with permission from Chris Weaver [21]. Improvise notification mechanism. 1: User interaction within a control modifies active properties of control. 2: Changed properties assign new value to variable. 3: Variable sends a value change notification to all properties connected to it. 4: Properties assigns variable value to itself. 5: Variable also sends value change notification to all lexicals that refer to the variable in their expressions. 6: Lexical sends change notification to variable it is assigned as value. 7, 8: View updates itself to reflect the change in value of live property.

Coordination Query Paths

Improvise change notification protocol facilitates identification of coordination paths involved in user interaction. When a variable changes, it sends a change event notification to all the interested listeners. We enhanced notification mechanism to capture list of types of all the variables which change along the path, from view in which user interacted to the view in which interaction has effects. When a variable receives change notification, it adds its type to the list and pass the list to the next variable. This process continues until the notification reaches property of the view. When property receives the notification, list has types of all the parameters and operators in the occurrence order along the coordination query path.



Magnitude Slider → Magnitude Parameter → Filter Operator → Dataset(filtered) → Visual Mapping Operator A → Dataset(Text Glyphs) → Table View
Magnitude Slider → Magnitude Parameter → Filter Operator → Dataset(filtered) → Visual Mapping Operator B → Dataset(Oval Glyphs) → Earthquake View
Slider
Integer
Lexical Filter
Lexical Info
Lexical Projection
Lexical Info
TableView
Slider
Integer
Lexical Filter
Lexical Info
Lexical Projection
Lexical Info
Scatterplot View

Figure 4.2: Coordination query paths: Two coordination query paths are observed one for each target view when user interacts with the magnitude slider of the visualization shown in figure 2.3.

As shown in figure 4.2, each node in the path contains a type of the variable and reference to the succeeding node.

4.2 Rule System

Rule system holds the repository of the rules and matches those rules with coordination query paths of the interaction. This section first outlines implementation of the rules and then explains rule matching algorithm.

4.2.1 Rule Implementation

Our rule system implements the mapping functions we identified in chapter 3 as set of rules. Idea is that when user interacts with the visualization, the coordination query paths corresponding to the interaction can be matched with these rules to identify the user intention. The natural choice for implementing the rules is graph data structures as the functions are themselves graphs.

A rule system often enforces conditions such as an object can only be of certain types and/or not allowed to be of certain types to match a rule. In our

Attribute	**Type**	**Description**
Allowed Types	List	List of allowed types (parameters, operators, datasets or views)
Non-Allowed Types	List	List of non-allowed types
Optional	Boolean	Indicates whether a node is optional or not
Forward Edge	Node	Reference to the succeeding node
Back Edge	Node	Reference to the back edge

Table 4.1: Attributes of the nodes in a rule graph

rules we have cases where an operator can be any one out of filter, sort, union, and join operators. Similarly we also have cases where a parameter can not be of certain types. *Allowed Types* contains list of all acceptable types for that node, where as *Non-Allowed Types* contains list of types that node cannot be. A *type* is considered to be a match for a *node* in rule graph only if the type is compatible with one of the Allowed Types and not compatible with Not Allowed types of the list. Figure 4.3 shows conversion of conceptual filter interaction graph into rule. As shown in figure, *Backward Edge* allows the rule to match multiple instances of the nodes present in between the nodes connected by it. Subsequence *Lexical Projection –>Lexical Info –>Lexical Sort –>Lexical Info* can be matched twice with back edge ***'D'*** shown in figure.

If there is a mismatch between the type and the node, *Optional* attribute identifies whether or not rule matching procedure should continue. If the attribute value is *true*, rule matching procedure continues to match the type with *Forward* and *Backward Edges* of the node. Rule matching fails if a node is not optional.

Value of *Any* in *Allowed Types* indicates that any type is compatible with that node. However, if the *Non-Allowed Types* list is not empty for such a node, node matches with all the types except the non-allowed types.

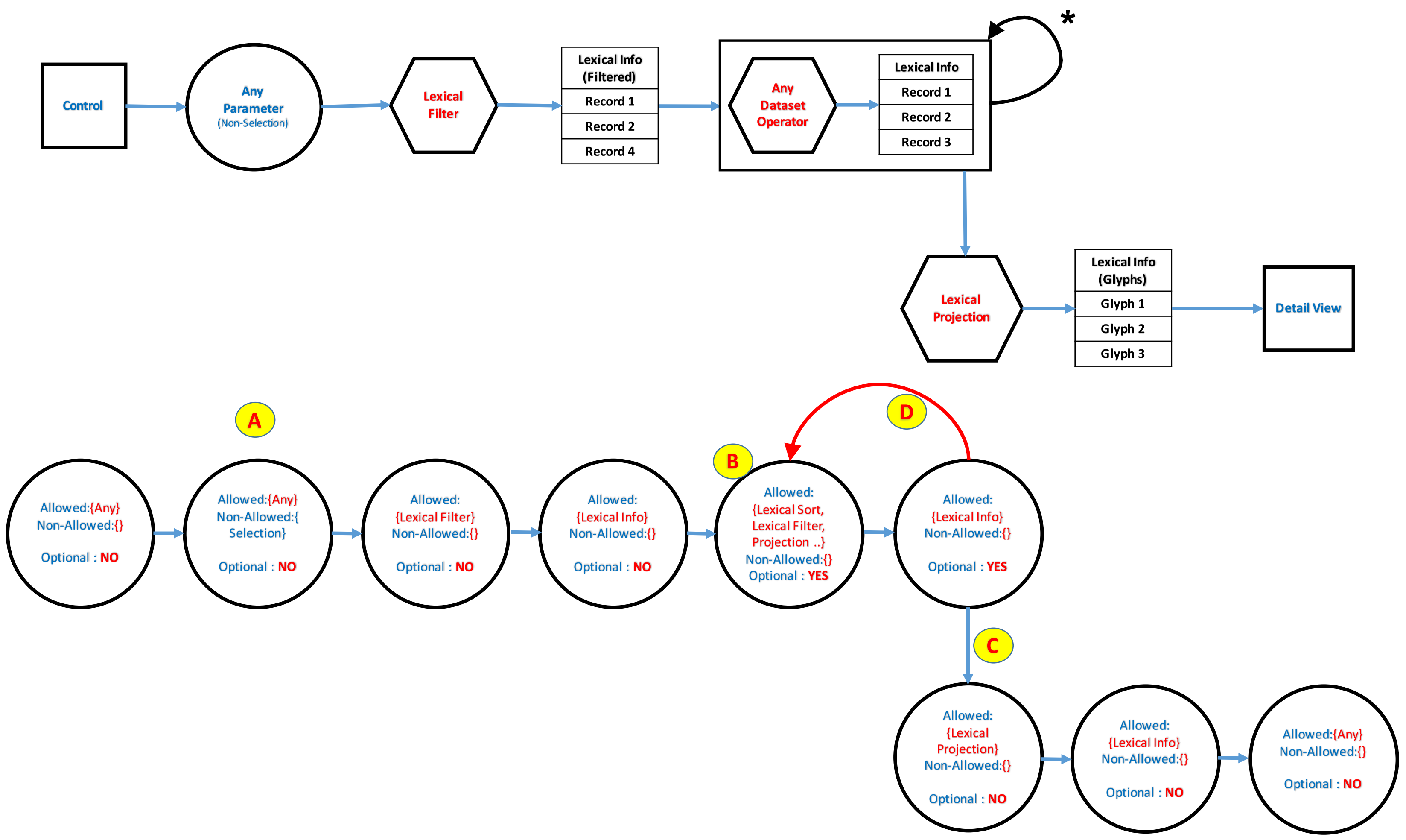

Control
Any Parameter (Non-Selection)
Lexical Filter
Lexical Info (Filtered)
Record 1
Record 2
Record 4
Any Dataset Operator
Lexical Info
Record 1
Record 2
Record 3
*
Lexical Projection
Lexical Info (Glyphs)
Glyph 1
Glyph 2
Glyph 3
Detail View
A
B
C
D
Allowed:{Any} Non-Allowed:{} Optional : NO
Allowed:{Any} Non-Allowed:{ Selection} Optional : NO
Allowed: {Lexical Filter} Non-Allowed:{} Optional : NO
Allowed: {Lexical Info} Non-Allowed:{} Optional : NO
Allowed: {Lexical Sort, Lexical Filter, Projection ..} Non-Allowed:{} Optional : YES
Allowed: {Lexical Info} Non-Allowed:{} Optional : YES
Allowed: {Lexical Projection} Non-Allowed:{} Optional : NO
Allowed: {Lexical Info} Non-Allowed:{} Optional : NO
Allowed:{Any} Non-Allowed:{} Optional : NO

Figure 4.3: Rule construction for filter interaction. A: Parameter of any type is allowed except selection parameter. B: Any node with data set operator types such as Lexical Filter, Lexical Sort, Lexical Projection etc. can be matched with this node. Node B and node succeeding it are also optional. C: Forward edges enforces the succession in types D: Back edge addresses the zero or more occurrences of the types in between the nodes connected by it.

Rule Repository

System maintains rule repository as a list of *Intent Name and Rule reference* pairs. When user interacts with visualization, system can match coordination query paths of user interaction with the rules present in the repository. When ever system identifies a match, intent corresponding to the rule can be recorded. In the following subsection, we will present the graph matching algorithm.

4.2.2 Rule Matching

For coordination query path to match rule's graph, there should be a valid path from source node to target node in the rule's graph resembling the coordination query path. Resemblance of the two paths is decided based on the types of the nodes. While validating a Type Node with a Rule Node, it is considered to be a match only if the type is present in the allowed list and not present in the non-allowed list.

Approach of the rule match algorithm is that all possible paths from source node to target node in the rules graph are traversed and matched against coordination query path. If at least one path of the rule graph matches with coordination query path, intent corresponding to the rule graph is detected. Following is the approach of our rule matching algorithm:

- *If RuleNode and TypeNode are Empty*
    - *return True*
- *If RuleNode is Empty and TypeNode is Not Empty or vice versa*
    - *return False*
- *If the RuleNode is optional*
    - *If RuleNode and TypeNode are mismatched*
        - *Match RuleNode.next and TypeNode*
        - *Match RuleNode.back and TypeNode*
    - *If RuleNode and TypeNode are matched*
        - *Match RuleNode.next and TypeNode.next*
        - *Match RuleNode.back and TypeNode.next*
        - *Match RuleNode.next and TypeNode*
        - *Match RuleNode.back and TypeNode*
- *else (Rule node is mandatory)*
    - *If RuleNode and TypeNode are mismatched*
        - *Return False*
    - *If RuleNode and TypeNode are matched*
        - *Match RuleNode.next and TypeNode.next*
        - *Match RuleNode.back and TypeNode.next*

Rule matching procedure terminates in 2 scenarios:

1. After all possible paths in the rule graph are explored

2. If a mandatory node(Optional attribute value is False) is mismatched

Once the system identifies a match, an intent is detected and the details of the that intent need to stored. Next sections describes design and implementation details of logging mechanism.

4.3 Intent Logging

Our rule system generates intents by matching the subgraph affected by the user interaction with the graphs of the rules. The identified user intention needs to be recorded in a log to access it at a later point in time. Capturing these details efficiently and cleanly decides the ability to query and revisit history. Intent name, parameter user interacted with, value of the parameter, view in which user interacted, view in which intent is observed, and time at which intent is performed are details that can be captured when an intent is identified. Parameter user interacted with, view in which user interacted, and view in which user intention is observed are referred as *Source View, Source Parameter, and Target View* respectively.

Logging intent details alone will not be sufficient to replay the intents. State of visualizations need to be saved in parallel to the intents recorded. According to the P-Set model [10], state of a visualization can be captured by storing parameter values of that visualization. As user interaction modifies state of the visualization, we capture the changing states by recording parameter changes. Parameters can be stored along with the identified user intentions in a single data set, but to separate the concerns related to state of the visualization and intent recording, we log the parameter value changes and intent details as separate datasets.

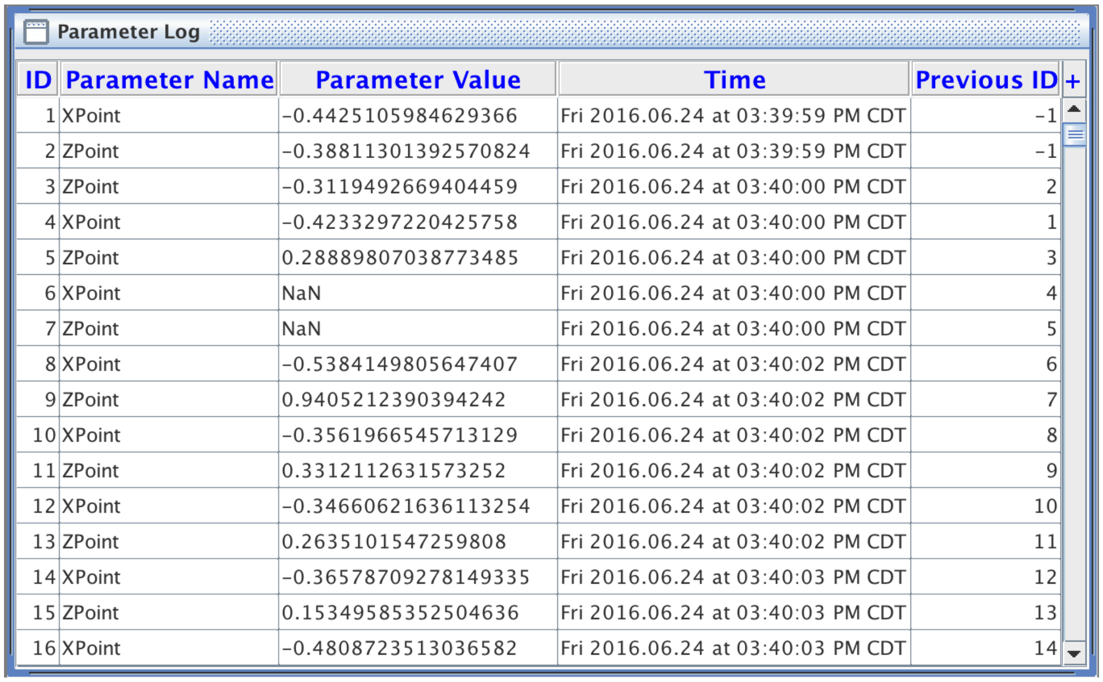
4.3.1 Logging State Changes

Objective of our system is to allow users revisit earlier visualization states by playing user intentions. To revisit a visualization state, system needs to restore the values of the parameters corresponding to that state. Our system facilitates this by capturing all the visualization states in a parameter log data set. Parameter log stores all the parameter changes in a visualization as shown in figire 4.4. When user interaction modifies a parameter, the value of that parameter is captured in parameter log. Along with value of the parameter, parameter log captures other details described in table 4.3.1.

Attribute	**Description**
ID	Unique identifier of parameter change
Parameter Name	Name of the parameter modified
Parameter Value	Value of the parameter after user interaction
Time	Time at which parameter is modified
Previous ID	ID of the record holding previous value of that parameter

Table 4.2: Data set schema of Parameter Log

Figure 4.4 shows parameter log of a visualization. ID uniquely identifies an entry in parameter change log. As *ID* is sequential and non-decreasing, it provides random access to entries in parameter log. *Time* attributes provides reference to the time at which the parameter is modified. *Previous ID* attribute identifies the log record that has previous value for that parameter.


Parameter Log

ID	Parameter Name	Parameter Value	Time	Previous ID
1	XPoint	-0.4425105984629366	Fri 2016.06.24 at 03:39:59 PM CDT	-1
2	ZPoint	-0.38811301392570824	Fri 2016.06.24 at 03:39:59 PM CDT	-1
3	ZPoint	-0.311949266940445 9	Fri 2016.06.24 at 03:40:00 PM CDT	2
4	XPoint	-0.4233297220425758	Fri 2016.06.24 at 03:40:00 PM CDT	1
5	ZPoint	0.2888980703877348 5	Fri 2016.06.24 at 03:40:00 PM CDT	3
6	XPoint	NaN	Fri 2016.06.24 at 03:40:00 PM CDT	4
7	ZPoint	NaN	Fri 2016.06.24 at 03:40:00 PM CDT	5
8	XPoint	-0.5384149805647407	Fri 2016.06.24 at 03:40:02 PM CDT	6
9	ZPoint	0.9405212390394242	Fri 2016.06.24 at 03:40:02 PM CDT	7
10	XPoint	-0.3561966545713129	Fri 2016.06.24 at 03:40:02 PM CDT	8
11	ZPoint	0.3312112631573252	Fri 2016.06.24 at 03:40:02 PM CDT	9
12	XPoint	-0.34660621636113254	Fri 2016.06.24 at 03:40:02 PM CDT	10
13	ZPoint	0.2635101547259808	Fri 2016.06.24 at 03:40:02 PM CDT	11
14	XPoint	-0.36578709278149335	Fri 2016.06.24 at 03:40:03 PM CDT	12
15	ZPoint	0.15349585352504636	Fri 2016.06.24 at 03:40:03 PM CDT	13
16	XPoint	-0.4808723513036582	Fri 2016.06.24 at 03:40:03 PM CDT	14

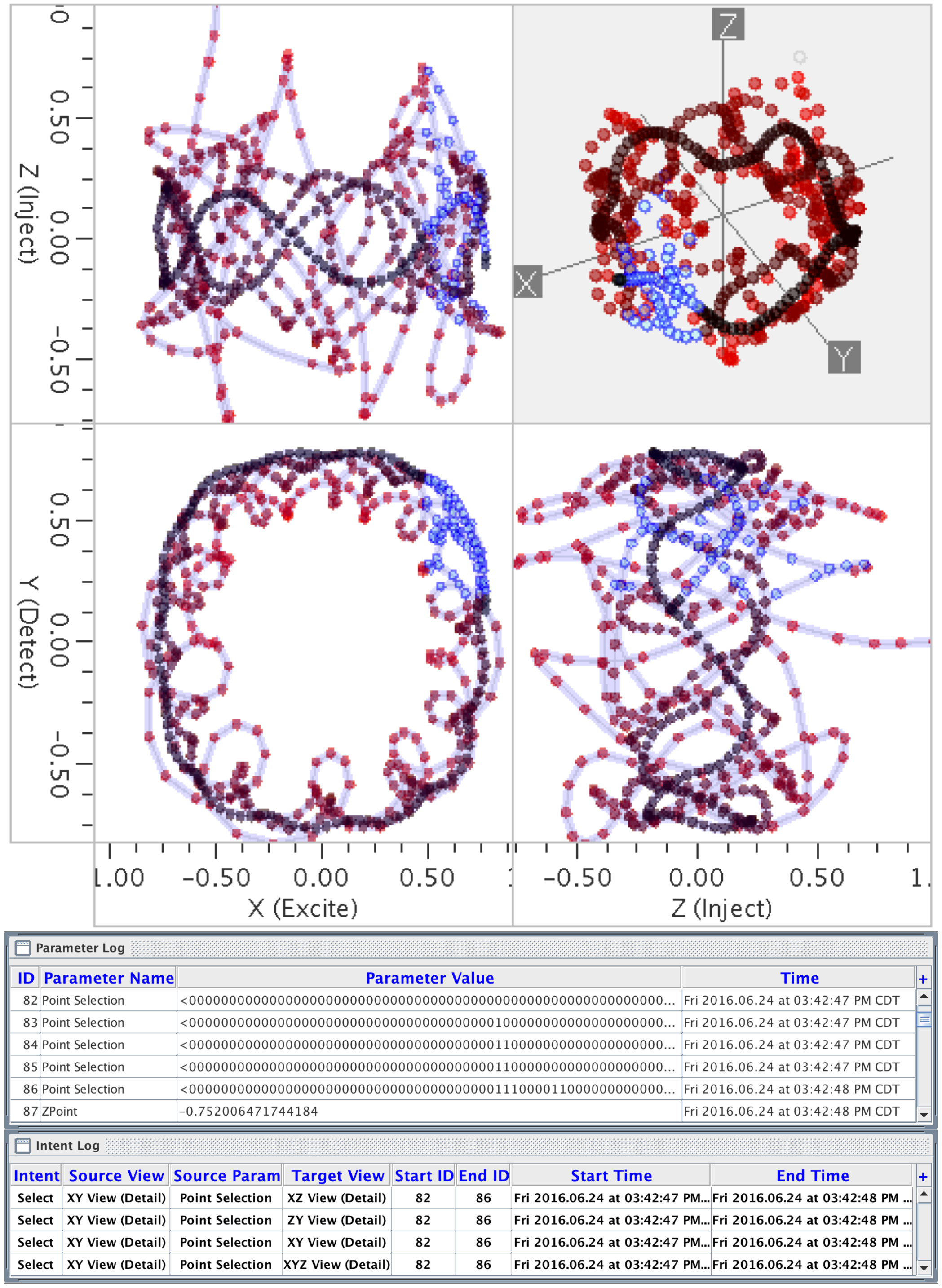
Figure 4.4: Parameter change log.

4.3.2 Logging User Intentions

Every time intent is detected, details of it can be added to user intention data set but, this approach creates an intent record for every parameter value change. User interaction often modifies same parameter in a sequence and adding an intent record for every parameter value change can be avoided by ***coalescing*** the user intentions resulted from continuous interaction. For example, as shown in figure 4.5, when user is selecting in a view, the selection parameter of the view changes multiple times in the course of interaction. In this scenario, instead of logging Select intent for every change in selection parameter, identified intents can be compressed to a single Select intent to significantly reduce the log size while capturing the information completely. The term 'continuous' doesn't represent time or any other dimension instead, it refers to the continuous behavior of user interaction. Selection parameter is modified 5 times in succession with ID's from

82 to 86 in parameter change log. Since our system identifies user intention every time parameter get modified, 5 select intents will be detected. As shown in figure, 5 intents can be coalesced to a single intent with the help of Start ID, and End ID attributes. These attributes are foreign keys to the parameter log and identify the parameter change log interval of the interaction.

In the above example, parameter is the criteria used for coalescing intents. Intentions identified in a sequence get coalesced if those intentions are performed by modifying same parameter. User selection events from 82 to 86 correspond to same parameter *Point Selection*. However, it is possible that certain parameters in a visualization may get modified in a interleaved fashion. For example, while panning in map view, user interaction may modify *X-Axis* and *Y-Axis range* parameters in a interleaved fashion. Similarly, while moving mouse in a scatterplot, *X* and *Y coordinate* parameters change alternatively. In these scenarios, continuity of intentions cannot be determined based on the parameter criteria. This can be addressed by coalescing intentions detected for interleaved parameters at different levels as shown in figure 4.6. At intermediate level, intentions interleaved for both X-axis and Y-axis range parameters can be coalesced into intentions of coordinate region and interleaved intentions of X and Y coordinates can be coalesced into intentions of coordinate point. At the view level, interleaved intentions of all parameters can be coalesced. Though coalescing of intentions above parameter level solves the interleaved parameter problem, it reduces the specificity of intention querying. Users cannot query the intention log to view only intentions corresponding to specific parameters as intentions of multiple parameters are coalesced at higher levels of coalescing. Hence we chose to coalesce and log the detected user intentions at two different levels, parameter and view level.



Parameter Log

ID	Parameter Name	Parameter Value	Time
82	Point Selection	<00000000000000000000000000000000000000000000000000000000000000...	Fri 2016.06.24 at 03:42:47 PM CDT
83	Point Selection	<00000000000000000000000000000000000000001000000000000000000000...	Fri 2016.06.24 at 03:42:47 PM CDT
84	Point Selection	<00000000000000000000000000000000000000001100000000000000000000...	Fri 2016.06.24 at 03:42:47 PM CDT
85	Point Selection	<00000000000000000000000000000000000000001100000000000000000000...	Fri 2016.06.24 at 03:42:47 PM CDT
86	Point Selection	<00000000000000000000000000000000000000001110000110000000000000...	Fri 2016.06.24 at 03:42:48 PM CDT
87	ZPoint	-0.752006471744184	Fri 2016.06.24 at 03:42:48 PM CDT

Intent Log

Intent	Source View	Source Param	Target View	Start ID	End ID	Start Time	End Time
Select	XY View (Detail)	Point Selection	XZ View (Detail)	82	86	Fri 2016.06.24 at 03:42:47 PM...	Fri 2016.06.24 at 03:42:48 PM ...
Select	XY View (Detail)	Point Selection	ZY View (Detail)	82	86	Fri 2016.06.24 at 03:42:47 PM...	Fri 2016.06.24 at 03:42:48 PM ...
Select	XY View (Detail)	Point Selection	XY View (Detail)	82	86	Fri 2016.06.24 at 03:42:47 PM...	Fri 2016.06.24 at 03:42:48 PM ...
Select	XY View (Detail)	Point Selection	XYZ View (Detail)	82	86	Fri 2016.06.24 at 03:42:47 PM...	Fri 2016.06.24 at 03:42:48 PM ...

Figure 4.5: Coalescing of Select interaction: User interaction updated Point Selection parameter five times. 5 select intents are coalesced to a single Select intent

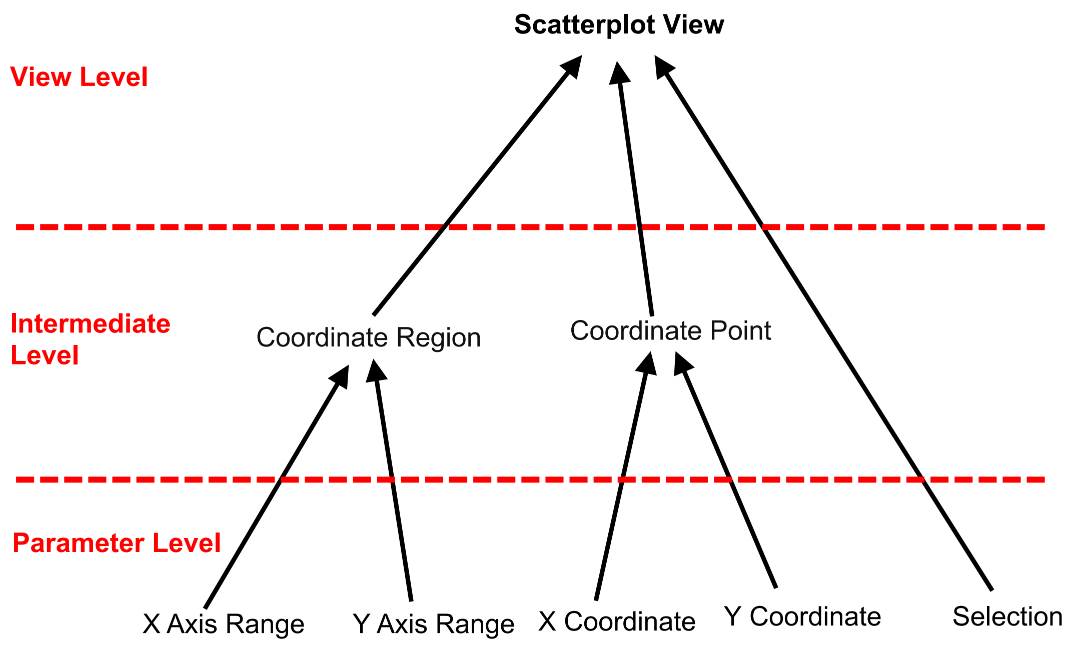


Figure 4.6:

Parameter Level Coalescing

At parameter level, criteria for coalescing two sequential intents are source parameter, and source view. Intents of same type get coalesced when two successive intents in a target view have same source parameter and same source view. Table 4.3.2 describes the intent information captured in parameter intent log record at parameter level.

Attribute	**Description**
Intent	Name of the deduced user intention
Source View	View in which user interacted
Source Param	Parameter modified in user interaction
Target View	View in which user intention is detected
Start Time	Time at which user intention has stated
End Time	Time at which user intention has completed
Start ID	Starting ID of the user intention
End ID	Ending ID of the user intention

Table 4.3: Data set schema of Intent Log captured at parameter level

View Level Coalescing

Similarly at view level, intents of same type in a target view get coalesced if they are successive. Advantage of view level coalescing is that, Intents of same type but are performed by interacting with different source parameters get coalesced. Table 4.3.2 describes the information captured at view level intent logging.

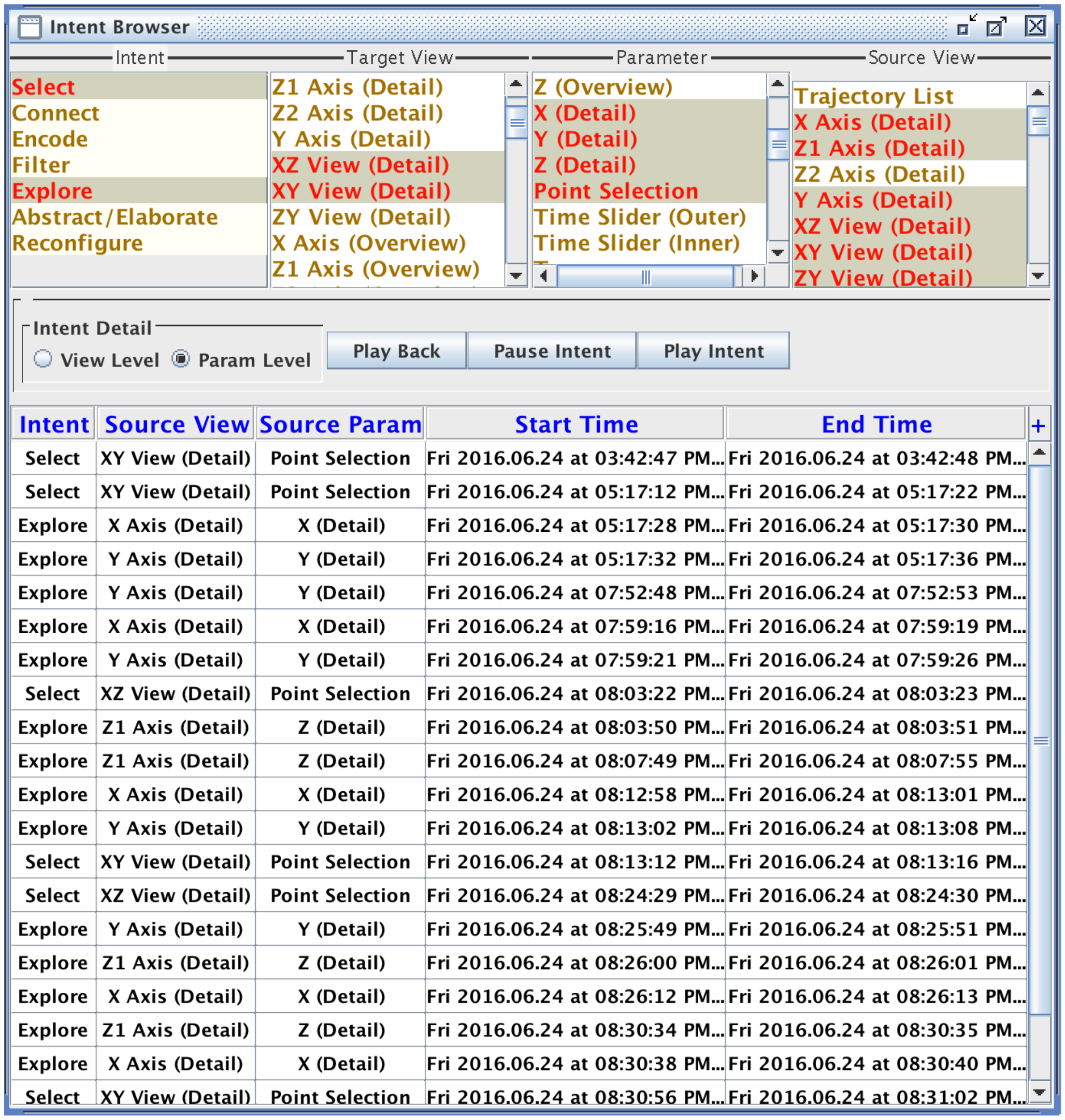
Attribute	**Description**
Intent	Name of the deduced user intention
Target View	View in which user intention is detected
Start Time	Time at which user intention has stated
End Time	Time at which user intention has completed
Start ID	Starting ID of the user intention
End ID	Ending ID of the user intention

Table 4.4: Data set schema of Intent Log captured at view level

4.4 Intent Browser

Our main objective is to allow user query captured history and replay it. To facilitate this, we developed an user interface called *Intent Browser* to query the intention logs and revisit earlier visualization states. User interface design is strongly motivated by the information being captured. Figure 4.7 shows the user interface. Since the intentions are captured at two different levels, our interface allows user decide which log to query. Choice selected in *Intent Detail* radio button decides intention log to be queried.

Figure 4.8 shows the data flow pipeline of our user interface. Either parameter or view level intention data set is chosen based on the choice selected in *Intent Detail* control in user interface. Filter operation is applied on the chosen data set. *Intent, Target View, Parameter, and Source View* controls in user interface



Intent: Select, Connect, Encode, Filter, Explore, Abstract/Elaborate, Reconfigure

Target View: Z1 Axis (Detail), Z2 Axis (Detail), Y Axis (Detail), XZ View (Detail), XY View (Detail), ZY View (Detail), X Axis (Overview), Z1 Axis (Overview)

Parameter: Z (Overview), X (Detail), Y (Detail), Z (Detail), Point Selection, Time Slider (Outer), Time Slider (Inner)

Source View: Trajectory List, X Axis (Detail), Z1 Axis (Detail), Z2 Axis (Detail), Y Axis (Detail), XZ View (Detail), XY View (Detail), ZY View (Detail)

Intent Detail: View Level [✓ Param Level]

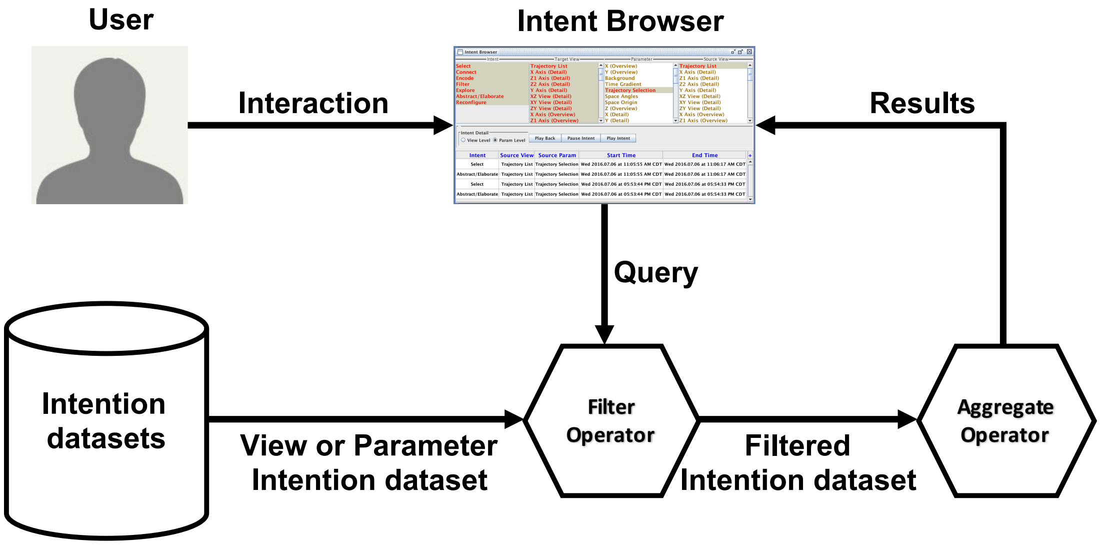
Intent	Source View	Source Param	Start Time	End Time
Select	XY View (Detail)	Point Selection	Fri 2016.06.24 at 03:42:47 PM...	Fri 2016.06.24 at 03:42:48 PM...
Select	XY View (Detail)	Point Selection	Fri 2016.06.24 at 05:17:12 PM...	Fri 2016.06.24 at 05:17:22 PM...
Explore	X Axis (Detail)	X (Detail)	Fri 2016.06.24 at 05:17:28 PM...	Fri 2016.06.24 at 05:17:30 PM...
Explore	Y Axis (Detail)	Y (Detail)	Fri 2016.06.24 at 05:17:32 PM...	Fri 2016.06.24 at 05:17:36 PM...
Explore	Y Axis (Detail)	Y (Detail)	Fri 2016.06.24 at 07:52:48 PM...	Fri 2016.06.24 at 07:52:53 PM...
Explore	X Axis (Detail)	X (Detail)	Fri 2016.06.24 at 07:59:16 PM...	Fri 2016.06.24 at 07:59:19 PM...
Explore	Y Axis (Detail)	Y (Detail)	Fri 2016.06.24 at 07:59:21 PM...	Fri 2016.06.24 at 07:59:26 PM...
Select	XZ View (Detail)	Point Selection	Fri 2016.06.24 at 08:03:22 PM...	Fri 2016.06.24 at 08:03:23 PM...
Explore	Z1 Axis (Detail)	Z (Detail)	Fri 2016.06.24 at 08:03:50 PM...	Fri 2016.06.24 at 08:03:51 PM...
Explore	Z1 Axis (Detail)	Z (Detail)	Fri 2016.06.24 at 08:07:49 PM...	Fri 2016.06.24 at 08:07:55 PM...
Explore	X Axis (Detail)	X (Detail)	Fri 2016.06.24 at 08:12:58 PM...	Fri 2016.06.24 at 08:13:01 PM...
Explore	Y Axis (Detail)	Y (Detail)	Fri 2016.06.24 at 08:13:02 PM...	Fri 2016.06.24 at 08:13:08 PM...
Select	XY View (Detail)	Point Selection	Fri 2016.06.24 at 08:13:12 PM...	Fri 2016.06.24 at 08:13:16 PM...
Select	XZ View (Detail)	Point Selection	Fri 2016.06.24 at 08:24:29 PM...	Fri 2016.06.24 at 08:24:30 PM...
Explore	Y Axis (Detail)	Y (Detail)	Fri 2016.06.24 at 08:25:49 PM...	Fri 2016.06.24 at 08:25:51 PM...
Explore	Z1 Axis (Detail)	Z (Detail)	Fri 2016.06.24 at 08:26:00 PM...	Fri 2016.06.24 at 08:26:01 PM...
Explore	X Axis (Detail)	X (Detail)	Fri 2016.06.24 at 08:26:12 PM...	Fri 2016.06.24 at 08:26:13 PM...
Explore	Z1 Axis (Detail)	Z (Detail)	Fri 2016.06.24 at 08:30:34 PM...	Fri 2016.06.24 at 08:30:35 PM...
Explore	X Axis (Detail)	X (Detail)	Fri 2016.06.24 at 08:30:38 PM...	Fri 2016.06.24 at 08:30:40 PM...
Select	XY View (Detail)	Point Selection	Fri 2016.06.24 at 08:30:56 PM...	Fri 2016.06.24 at 08:31:02 PM...

Figure 4.7: User interface for browsing and replaying the user intentions. 'Intent' control lets user view only Intents of selected types. Selection in 'Target View' control shows intents correspond to those views. Selection in 'Parameter' control shows only intents detected when the selected parameters are modified. 'Source View' allows the user to view intents performed by interacting with those views only. 'Intent Detail' radio button allows user to query either parameter intent log or view intent log. Buttons 'Play Intent', 'Pause Intent', and 'Play Back' allow user to play, pause, and play back the intents respectively.

allow user to modify filter criteria. Filtering criteria is different at parameter and view level querying. At view level querying, an intention record is shown only if it's *Type* and *Target View* attribute values are present in set of intent types and target views selected in Intent Browser respectively. At parameter level querying, an intention record is shown only if it's *Type, Target View, Source View, and Source Param* attribute values are present in set of intent types, target views, source views, and parameters selected in Intent Browser respectively. *Parameter* and *Source View* controls in the user interface are not functional when user is querying at view level as source view and parameter details are captured only at parameter level intent log.

As shown in figure 4.5, user selection in one view might brush items in multiple views. Since our approach is based on coordination query paths, our rule system detects select intention for every view in which brushing is observed. All these intention records captured for different target views are part of single selection intention performed by user. Similarly, in CMVs, often interaction has effects in multiple views. In this scenario, information in intention records is duplicate except for the *Target View* attribute. It will confuse the user if multiple intention records of same type are shown in the search results. Hence we chose to aggregate similar intention records belonging to one single user interaction. Records are aggregated based on Intent *Name, Source Parameter, Source Target, Start ID, and End ID.* This means, if two or more intention records are captured of same user interaction, they will be aggregated.

Name of the Intent, Source view from which intent is performed, Parameter, and timing information are the details shown in the displayed log at parameter level querying. As shown in figure 4.9, view level querying displays only Intent, and timing information of the log. Difference in the displayed logs is due to



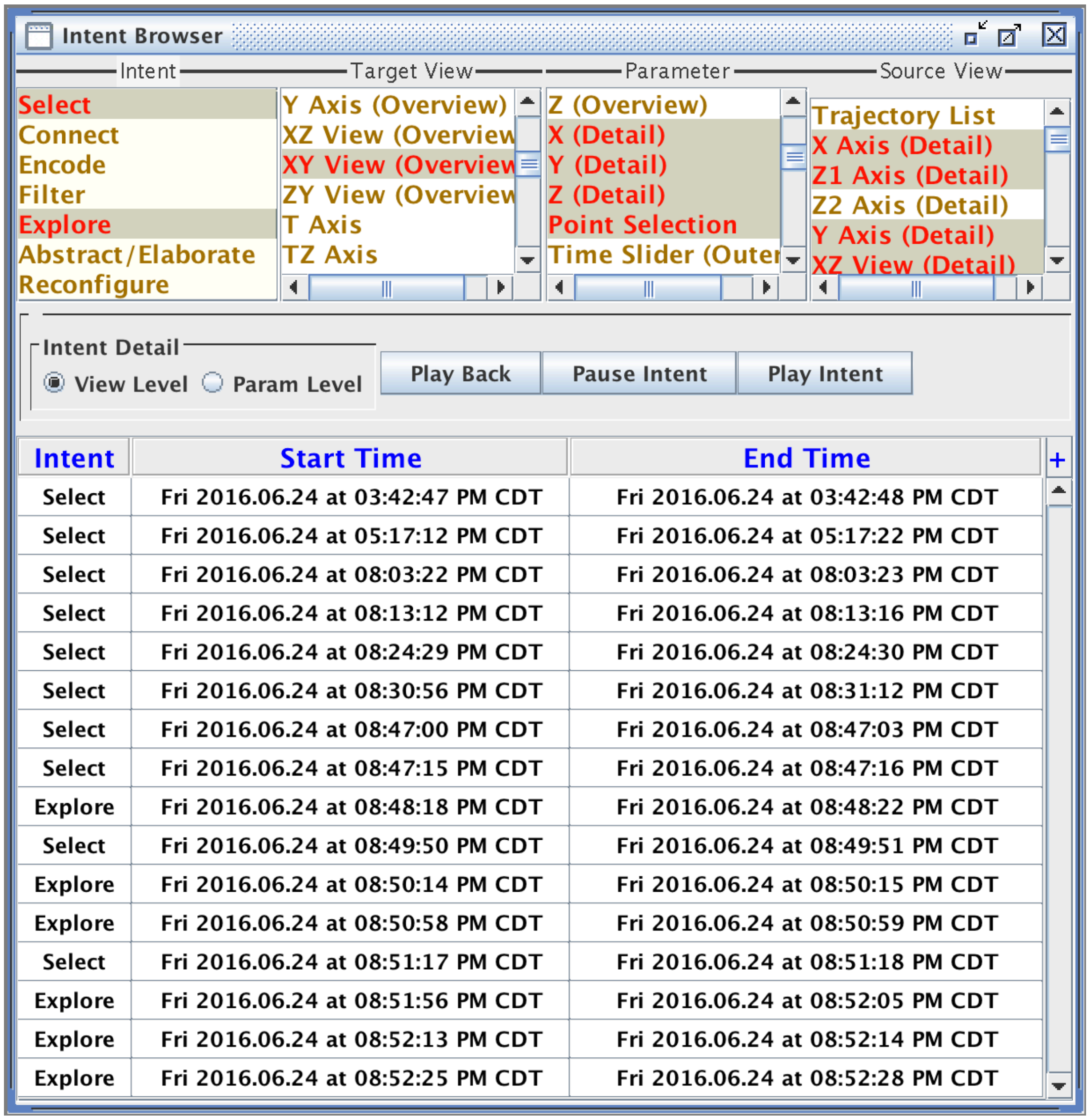
Intent	Source View	Source Param	Start Time	End Time
Select	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 11:05:55 AM CDT	Wed 2016.07.06 at 11:06:17 AM CDT
Abstract/Elaborate	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 11:05:55 AM CDT	Wed 2016.07.06 at 11:06:17 AM CDT
Select	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 05:53:44 PM CDT	Wed 2016.07.06 at 05:54:33 PM CDT
Abstract/Elaborate	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 05:53:44 PM CDT	Wed 2016.07.06 at 05:54:33 PM CDT

Figure 4.8: Intent Browser query data flow pipeline

the different coalescing criteria. Also interesting readers might have noticed that *Target View* attribute is not shown in displayed results at both parameter and view level querying even though this attribute is recorded in both the intention logs. This is because of the aggregation operation performed in the query pipeline.

User can select one of the displayed intent records and chose to play it by clicking either ***Play Intent*** or ***Play Back*** button. *Play intent* button plays the user intention in forward direction where as *Play Back* button plays user intention from end to the starting of the user intent. ***Pause Intent*** button allows user to pause while revisiting history.

When user choses to play selected intent, Start ID, and End ID can be obtained. History browsing should show all the visualization states between Start ID and End ID. To show the states within the interval, system first needs to be brought to a state same as the one just before interval. To achieve this, our system uses parameter log and key frame log. Keyframe stores state of all the parameters at the time of capturing it. As shown in figure 4.10, a keyframe



Intent	Target View	Parameter	Source View
Select	Y Axis (Overview)	Z (Overview)	Trajectory List
Connect	XZ View (Overview	X (Detail)	X Axis (Detail)
Encode	XY View (Overview	Y (Detail)	Z1 Axis (Detail)
Filter	ZY View (Overview	Z (Detail)	Z2 Axis (Detail)
Explore	T Axis	Point Selection	Y Axis (Detail)
Abstract/Elaborate	TZ Axis	Time Slider (Outer	XZ View (Detail)
Reconfigure			

Intent Detail: [✓ View Level] Param Level

Intent	Start Time	End Time
Select	Fri 2016.06.24 at 03:42:47 PM CDT	Fri 2016.06.24 at 03:42:48 PM CDT
Select	Fri 2016.06.24 at 05:17:12 PM CDT	Fri 2016.06.24 at 05:17:22 PM CDT
Select	Fri 2016.06.24 at 08:03:22 PM CDT	Fri 2016.06.24 at 08:03:23 PM CDT
Select	Fri 2016.06.24 at 08:13:12 PM CDT	Fri 2016.06.24 at 08:13:16 PM CDT
Select	Fri 2016.06.24 at 08:24:29 PM CDT	Fri 2016.06.24 at 08:24:30 PM CDT
Select	Fri 2016.06.24 at 08:30:56 PM CDT	Fri 2016.06.24 at 08:31:12 PM CDT
Select	Fri 2016.06.24 at 08:47:00 PM CDT	Fri 2016.06.24 at 08:47:03 PM CDT
Select	Fri 2016.06.24 at 08:47:15 PM CDT	Fri 2016.06.24 at 08:47:16 PM CDT
Explore	Fri 2016.06.24 at 08:48:18 PM CDT	Fri 2016.06.24 at 08:48:22 PM CDT
Select	Fri 2016.06.24 at 08:49:50 PM CDT	Fri 2016.06.24 at 08:49:51 PM CDT
Explore	Fri 2016.06.24 at 08:50:14 PM CDT	Fri 2016.06.24 at 08:50:15 PM CDT
Explore	Fri 2016.06.24 at 08:50:58 PM CDT	Fri 2016.06.24 at 08:50:59 PM CDT
Select	Fri 2016.06.24 at 08:51:17 PM CDT	Fri 2016.06.24 at 08:51:18 PM CDT
Explore	Fri 2016.06.24 at 08:51:56 PM CDT	Fri 2016.06.24 at 08:52:05 PM CDT
Explore	Fri 2016.06.24 at 08:52:13 PM CDT	Fri 2016.06.24 at 08:52:14 PM CDT
Explore	Fri 2016.06.24 at 08:52:25 PM CDT	Fri 2016.06.24 at 08:52:28 PM CDT

Figure 4.9: User interface for browsing and replaying the view level user intentions.

is captured after every 100 parameter value changes. Also an initial keyframe is captured the moment visualization is loaded to preserve the initial state of visualization before user starts to interact.

To play an intent with parameter log interval between 144 and 226, visualization needs to be brought to a state same as the one after parameter change id 143. To do so, system takes copy of nearest keyframe i.e. kf1 and modifies it to

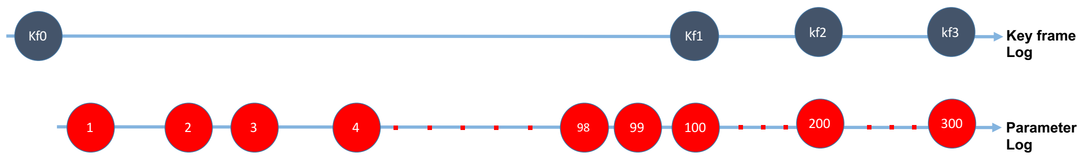


Figure 4.10: Keyframe capturing mechanism

reflect parameter value changes from 101 to 143. Once the keyframe represents state corresponding to event id 143, parameters of visualization will be assigned with values present in the keyframe. After bringing visualization to state after parameter change id 143, parameter changes from 144 to 226 will be replayed one by one.

Similarly, in case of replaying back, visualization first needs to be brought to a state from which user wish to replay back. After the visualization state is established, Previous ID attribute of parameter change log can be used to replay intent backwards.

Chapter 5

System Analysis

In this chapter, we present an analysis of our history system. First, we present the effectiveness of the rule system. Then, we describe the effectiveness of coalescing intentions at two different levels. Finally, we demonstrate the expressiveness of user interface in terms of questions it can and cannot express.

5.1 Rules Analysis

We loaded existing visualizations and performed various interactions to see if the rule system works as expected. We observed that the rule system correctly translates low-level interactions into user intentions, except for a few instances of false positives. These false positives result from the rudimentary way in which we match transformation functions. Our rule system translates interactions into intentions only based on the types of coordination graph components that change as a result of user interaction, rather than the internal characteristics of those components.

As shown in figure 5.1, a user can select data items in a view by hovering the

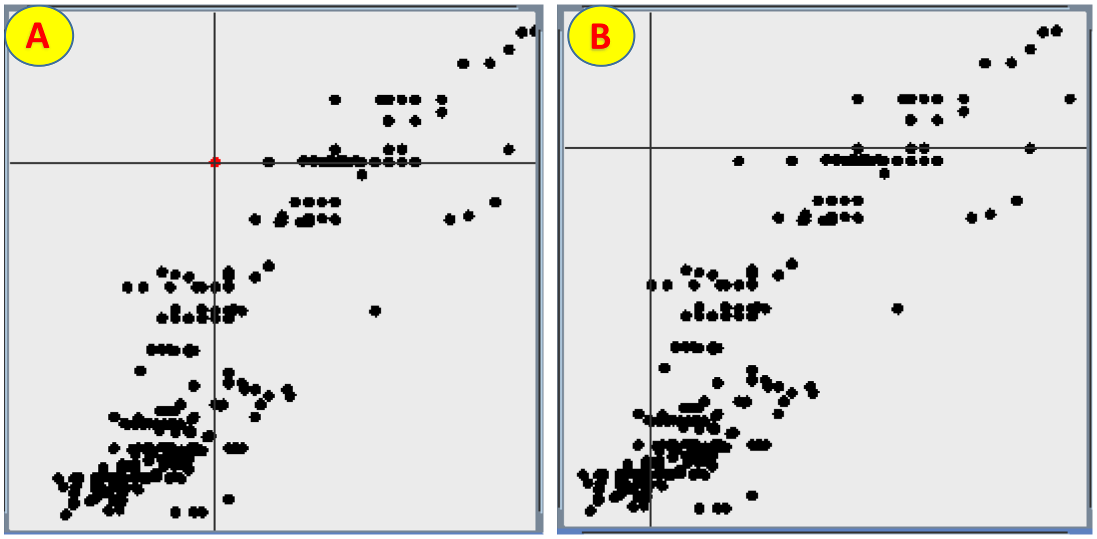
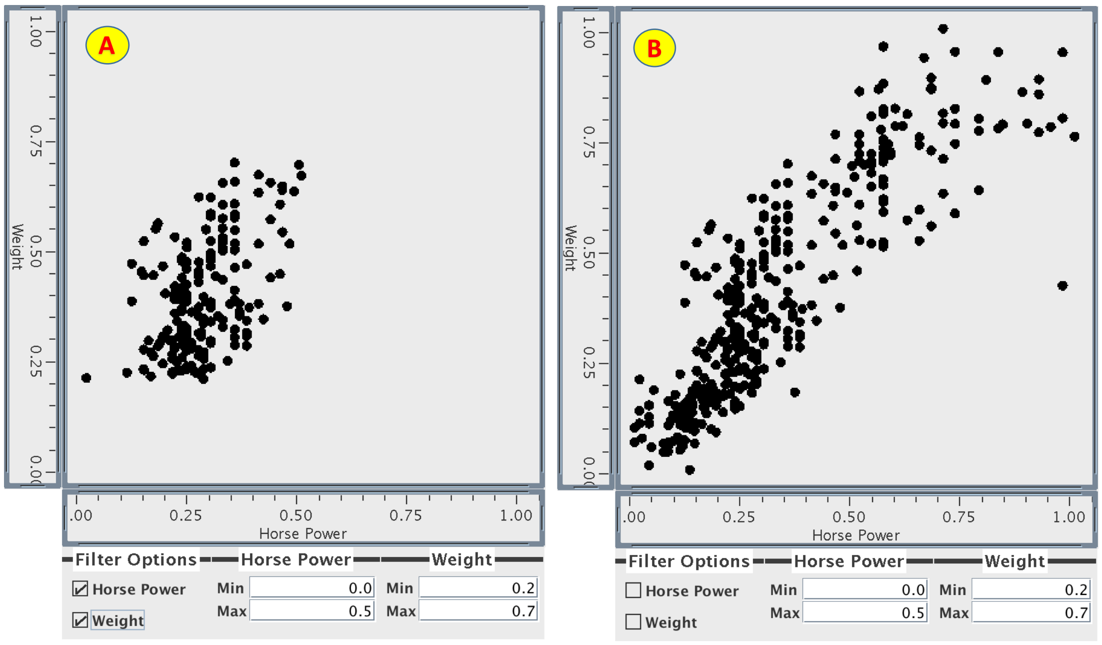


Figure 5.1: Selecting Items by mouse hovering. (A) A select intention is correctly detected. (B) A select intention is incorrectly detected, since none of the items are selected.

mouse over them. Our rule system matches the selection rule with the coordination query path of user interaction when the user moves mouse inside the view. Mouse hovering highlights items if the position of the mouse is inside the corresponding glyphs. Items are not highlighted when the mouse position is not inside any of the data items in the view. Since our approach of deducing user intention is based on types of components present in coordination query path, but not on the internal characteristics of those components, the rule system detects a select intention irrespective of whether any items are highlighted or not.

Similarly, we observed a false positive user intention when selective filtering is employed, as shown in figure 5.2. Users can optionally filter the information present in the view by checking the *Horse Power* and *Weight* checkboxes. The filter intentions identified by the rule system for any changes in horse power and weight ranges are true positives when these options are selected. When these options are disabled, any user interaction to modify horse power and weight



A:

Filter Options — Horse Power — Weight

☑ Horse Power   Min 0.0   Min 0.2
☑ Weight   Max 0.5   Max 0.7

B:

Filter Options — Horse Power — Weight

☐ Horse Power   Min 0.0   Min 0.2
☐ Weight   Max 0.5   Max 0.7

Figure 5.2: A: User Interaction to modify the range of Horse Power and Weight parameters, filters the data shown in view(**True Positive**). B: Modifying parameters does not filter data, but rule system identifies Filter intention(**False Positive**).

ranges, does not filter the data in the view, but the rule system detects a filter interaction nevertheless. These false positives might be avoided by incorporating some additional rules that more deeply examine individual parameters, operators, and datasets to make sure that a detected intention is not only syntactically valid but also semantically correct.

5.2 Coalescing Analysis

Capturing an intention log at multiple levels provides flexibility in querying. At the view level, performed intentions are highly summarized in the log. In parameter level coalescing, captured information is at a finer grain of detail. Figure 5.3 shows the difference between parameter level and view level coalescing.

Event identifiers from 41 to 70, and from 123 to 166, represent user interaction with the X-axis range parameter. Similarly, event identifiers from 71 to 122, and from 167 to 184, correspond to user interaction with the Y-axis range parameter.

As a result of panning along the axis, the system detects explore intentions. At the view level, all explore intentions are coalesced into a single explore intent. At the parameter level, four distinct explore interactions are recorded as parameters and source view from which user interacted has changed. At the parameter level, users can query the intent log to see only the intents performed in the selected source view and involving selected parameters. At view level, users can get a high level summary of all the intents performed in a view over time.

5.3 User Interface Analysis

Queries Intent Browser Can Express

By varying the search criteria, users can effectively search the intention logs from several perspectives. Some of the queries users can express are:

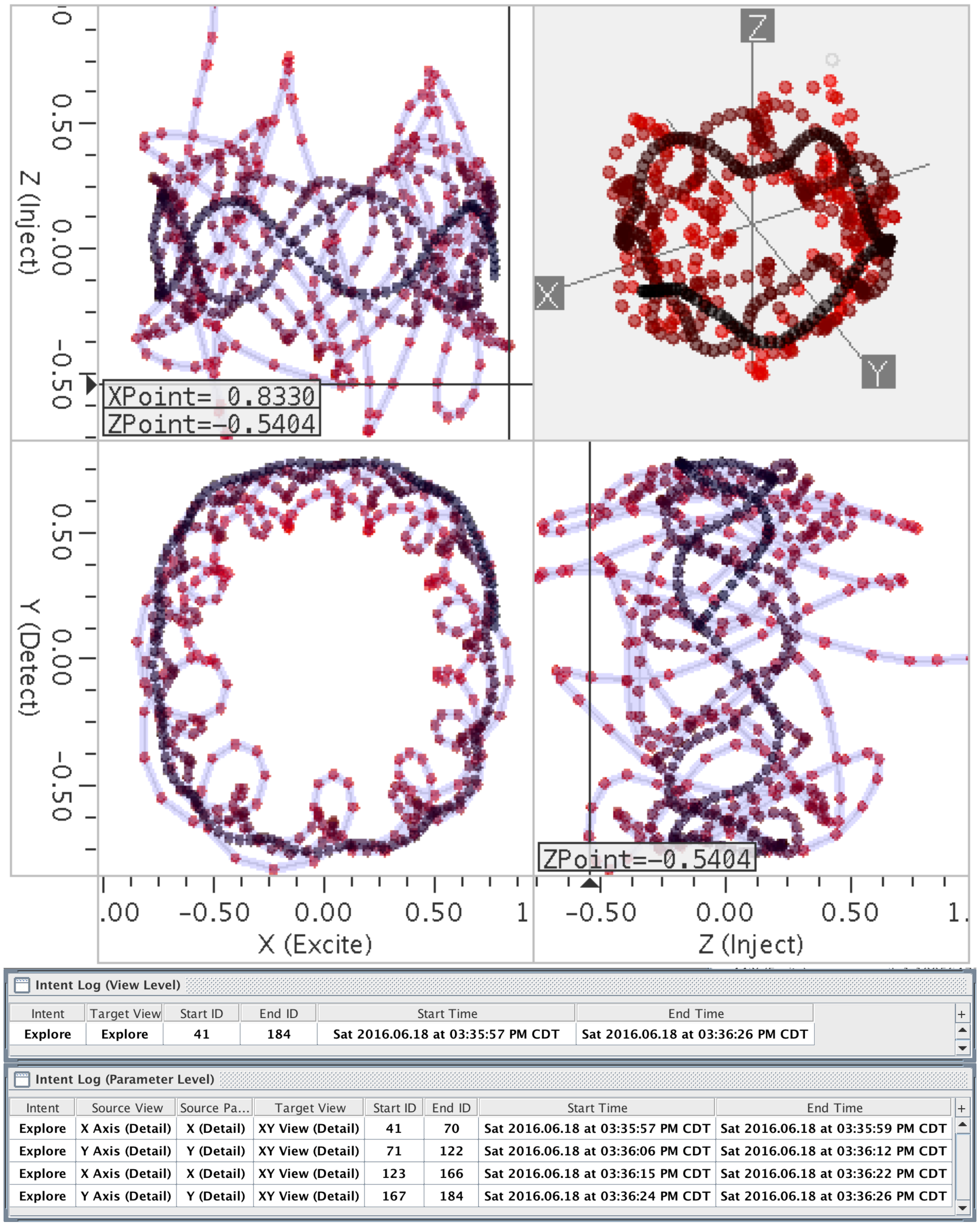
1. *Display all intentions performed in the visualization (Figure 5.4, 5.5).*

2. *Display all selections performed in particular views (Figure 5.6, 5.7).*

3. *Display selected intentions that were performed in particular target views by modifying a parameter from any source view (Figure 5.8).*

4. *Display selected intentions performed in particular target views by modifying a parameter from particular source views (Figure 5.9).*

5. *Display all intentions performed by modifying particular parameter from selected source views (Figure 5.10).*



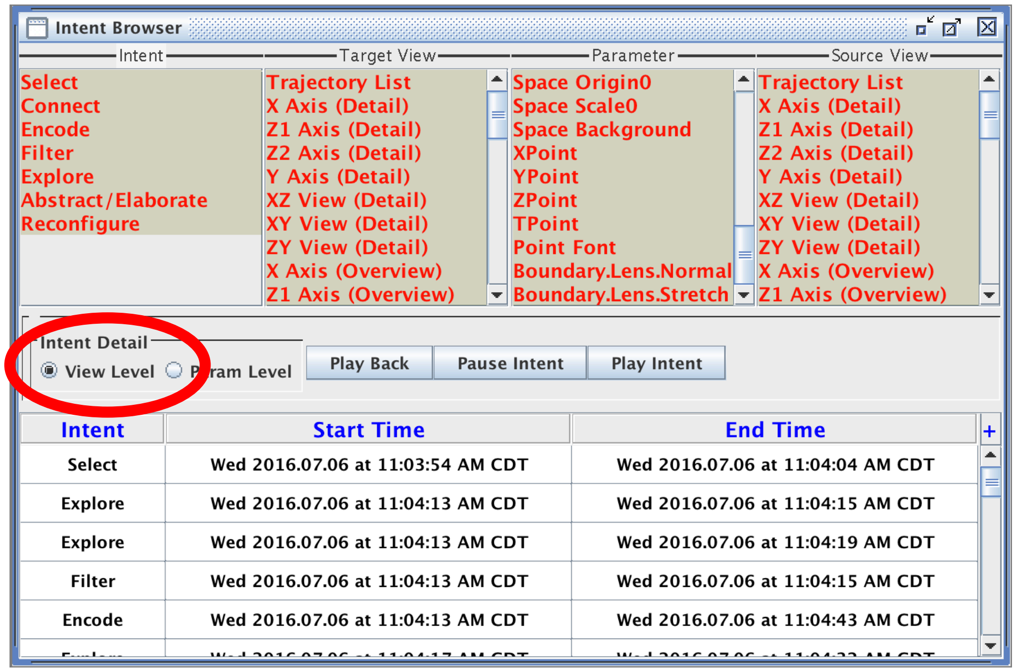
Intent Log (View Level)

Intent	Target View	Start ID	End ID	Start Time	End Time
Explore	Explore	41	184	Sat 2016.06.18 at 03:35:57 PM CDT	Sat 2016.06.18 at 03:36:26 PM CDT

Intent Log (Parameter Level)

Intent	Source View	Source Pa...	Target View	Start ID	End ID	Start Time	End Time
Explore	X Axis (Detail)	X (Detail)	XY View (Detail)	41	70	Sat 2016.06.18 at 03:35:57 PM CDT	Sat 2016.06.18 at 03:35:59 PM CDT
Explore	Y Axis (Detail)	Y (Detail)	XY View (Detail)	71	122	Sat 2016.06.18 at 03:36:06 PM CDT	Sat 2016.06.18 at 03:36:12 PM CDT
Explore	X Axis (Detail)	X (Detail)	XY View (Detail)	123	166	Sat 2016.06.18 at 03:36:15 PM CDT	Sat 2016.06.18 at 03:36:22 PM CDT
Explore	Y Axis (Detail)	Y (Detail)	XY View (Detail)	167	184	Sat 2016.06.18 at 03:36:24 PM CDT	Sat 2016.06.18 at 03:36:26 PM CDT

Figure 5.3: Difference between view level and parameter level coalescing. At the view level, the explore intentions of interleaved X(Detail) and Y(Detail) parameters are coalesced. At the parameter level, intentions are coalesced only if they are successive and involve the same parameter.

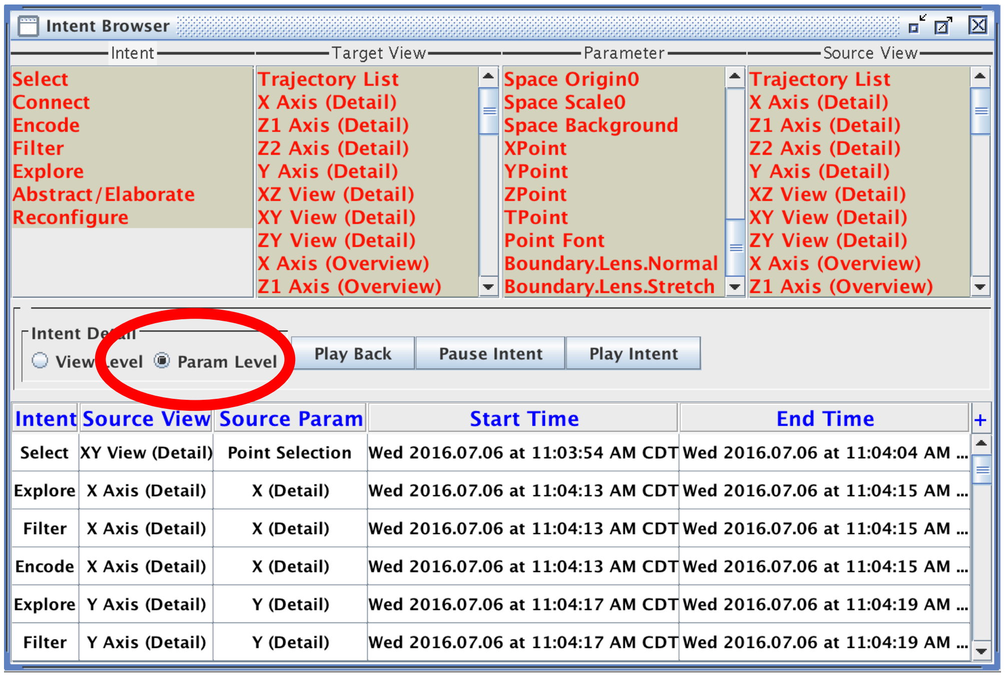


Intent	Target View	Parameter	Source View
Select	Trajectory List	Space Origin0	Trajectory List
Connect	X Axis (Detail)	Space Scale0	X Axis (Detail)
Encode	Z1 Axis (Detail)	Space Background	Z1 Axis (Detail)
Filter	Z2 Axis (Detail)	XPoint	Z2 Axis (Detail)
Explore	Y Axis (Detail)	YPoint	Y Axis (Detail)
Abstract/Elaborate	XZ View (Detail)	ZPoint	XZ View (Detail)
Reconfigure	XY View (Detail)	TPoint	XY View (Detail)
	ZY View (Detail)	Point Font	ZY View (Detail)
	X Axis (Overview)	Boundary.Lens.Normal	X Axis (Overview)
	Z1 Axis (Overview)	Boundary.Lens.Stretch	Z1 Axis (Overview)

Intent Detail: [✓ View Level] Param Level

Intent	Start Time	End Time
Select	Wed 2016.07.06 at 11:03:54 AM CDT	Wed 2016.07.06 at 11:04:04 AM CDT
Explore	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM CDT
Explore	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:19 AM CDT
Filter	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM CDT
Encode	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:43 AM CDT

Figure 5.4: At the view level, selecting all values present in both the intent and the target view lists displays a summary of all intentions performed in all views. Parameter and source view list selections are ignored at the view level.

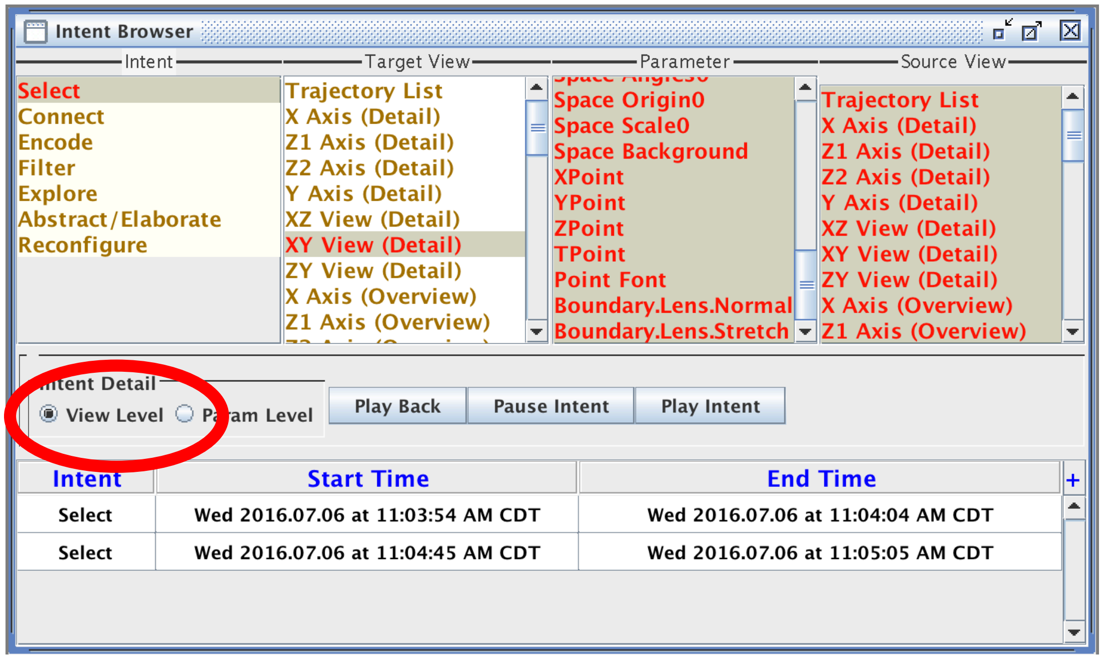


Intent	Target View	Parameter	Source View
Select	Trajectory List	Space Origin0	Trajectory List
Connect	X Axis (Detail)	Space Scale0	X Axis (Detail)
Encode	Z1 Axis (Detail)	Space Background	Z1 Axis (Detail)
Filter	Z2 Axis (Detail)	XPoint	Z2 Axis (Detail)
Explore	Y Axis (Detail)	YPoint	Y Axis (Detail)
Abstract/Elaborate	XZ View (Detail)	ZPoint	XZ View (Detail)
Reconfigure	XY View (Detail)	TPoint	XY View (Detail)
	ZY View (Detail)	Point Font	ZY View (Detail)
	X Axis (Overview)	Boundary.Lens.Normal	X Axis (Overview)
	Z1 Axis (Overview)	Boundary.Lens.Stretch	Z1 Axis (Overview)

Intent Detail: View Level [✓ Param Level]

Intent	Source View	Source Param	Start Time	End Time
Select	XY View (Detail)	Point Selection	Wed 2016.07.06 at 11:03:54 AM CDT	Wed 2016.07.06 at 11:04:04 AM ...
Explore	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM ...
Filter	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM ...
Encode	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM ...
Explore	Y Axis (Detail)	Y (Detail)	Wed 2016.07.06 at 11:04:17 AM CDT	Wed 2016.07.06 at 11:04:19 AM ...
Filter	Y Axis (Detail)	Y (Detail)	Wed 2016.07.06 at 11:04:17 AM CDT	Wed 2016.07.06 at 11:04:19 AM ...

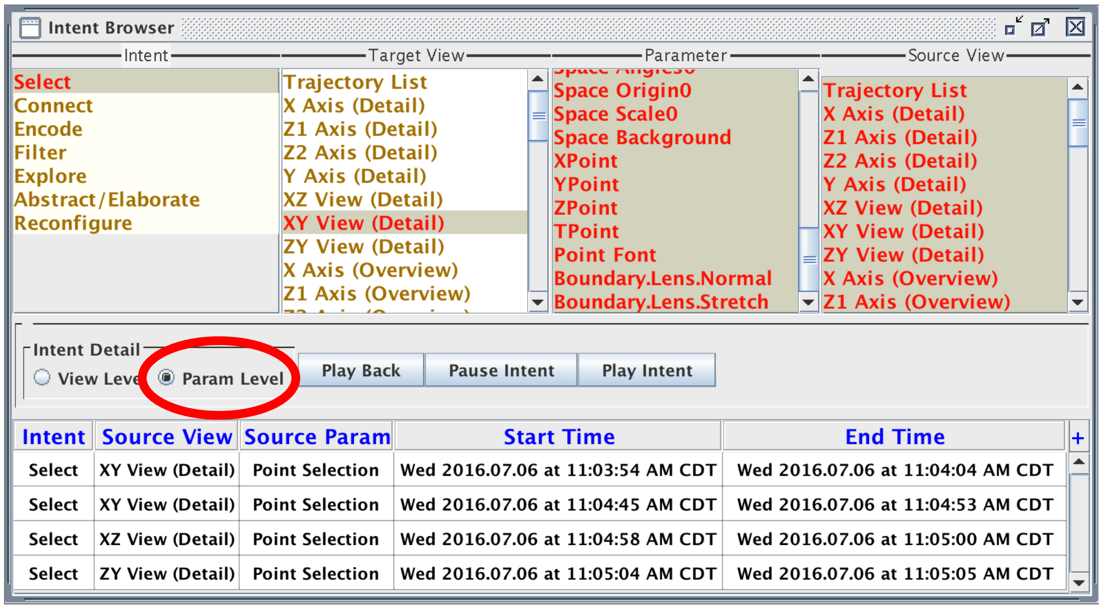
Figure 5.5: At the parameter level, selecting all values in all lists displays fine-grain details of all user intentions.



Intent Detail: [✓ View Level] Param Level

Intent	Start Time	End Time
Select	Wed 2016.07.06 at 11:03:54 AM CDT	Wed 2016.07.06 at 11:04:04 AM CDT
Select	Wed 2016.07.06 at 11:04:45 AM CDT	Wed 2016.07.06 at 11:05:05 AM CDT

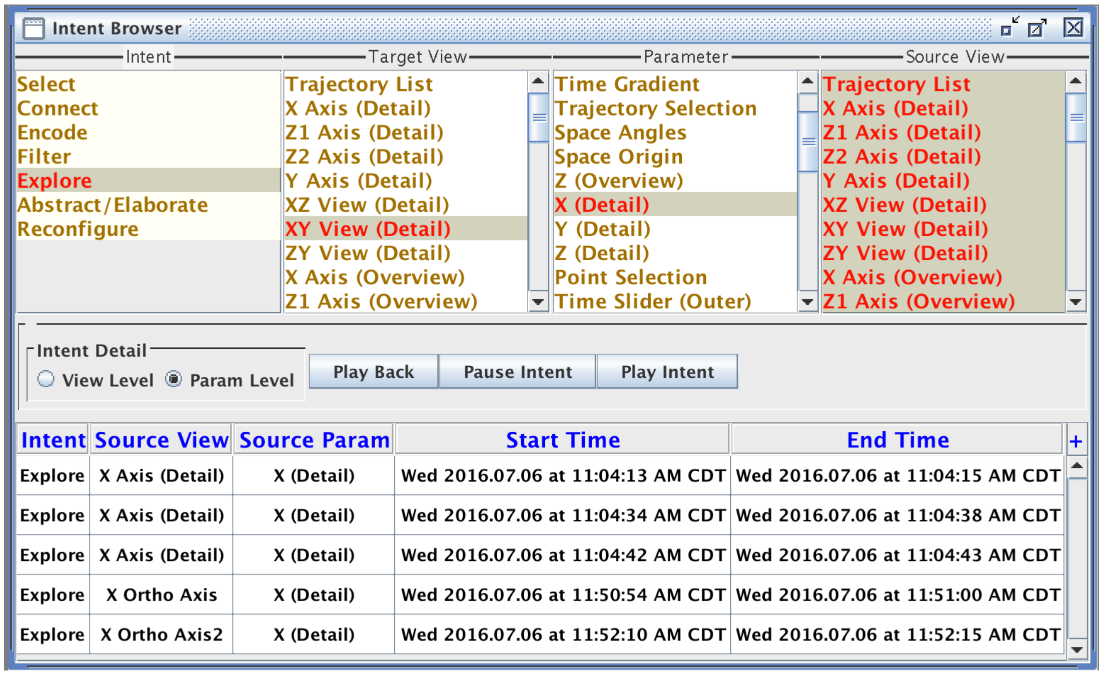
Figure 5.6: By selecting 'Select' intention in the intent list and the 'XY View (Detail)' view in the target view list, users can see all the selections performed in the selected target view.



Intent Detail: View Level [✓ Param Level]

Intent	Source View	Source Param	Start Time	End Time
Select	XY View (Detail)	Point Selection	Wed 2016.07.06 at 11:03:54 AM CDT	Wed 2016.07.06 at 11:04:04 AM CDT
Select	XY View (Detail)	Point Selection	Wed 2016.07.06 at 11:04:45 AM CDT	Wed 2016.07.06 at 11:04:53 AM CDT
Select	XZ View (Detail)	Point Selection	Wed 2016.07.06 at 11:04:58 AM CDT	Wed 2016.07.06 at 11:05:00 AM CDT
Select	ZY View (Detail)	Point Selection	Wed 2016.07.06 at 11:05:04 AM CDT	Wed 2016.07.06 at 11:05:05 AM CDT

Figure 5.7: At the parameter level, users can see fine-grain details of all selections performed only in 'XY View (Detail)'.

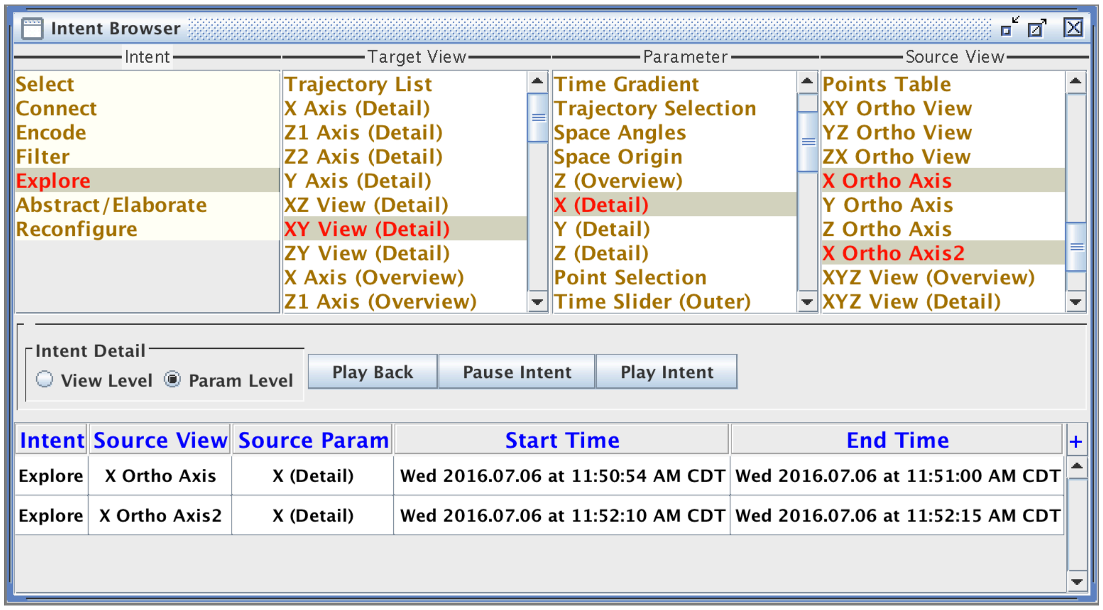


Intent	Target View	Parameter	Source View
Select	Trajectory List	Time Gradient	Trajectory List
Connect	X Axis (Detail)	Trajectory Selection	X Axis (Detail)
Encode	Z1 Axis (Detail)	Space Angles	Z1 Axis (Detail)
Filter	Z2 Axis (Detail)	Space Origin	Z2 Axis (Detail)
Explore	Y Axis (Detail)	Z (Overview)	Y Axis (Detail)
Abstract/Elaborate	XZ View (Detail)	X (Detail)	XZ View (Detail)
Reconfigure	XY View (Detail)	Y (Detail)	XY View (Detail)
	ZY View (Detail)	Z (Detail)	ZY View (Detail)
	X Axis (Overview)	Point Selection	X Axis (Overview)
	Z1 Axis (Overview)	Time Slider (Outer)	Z1 Axis (Overview)

Intent Detail: View Level [✓ Param Level]

Intent	Source View	Source Param	Start Time	End Time
Explore	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:13 AM CDT	Wed 2016.07.06 at 11:04:15 AM CDT
Explore	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:34 AM CDT	Wed 2016.07.06 at 11:04:38 AM CDT
Explore	X Axis (Detail)	X (Detail)	Wed 2016.07.06 at 11:04:42 AM CDT	Wed 2016.07.06 at 11:04:43 AM CDT
Explore	X Ortho Axis	X (Detail)	Wed 2016.07.06 at 11:50:54 AM CDT	Wed 2016.07.06 at 11:51:00 AM CDT
Explore	X Ortho Axis2	X (Detail)	Wed 2016.07.06 at 11:52:10 AM CDT	Wed 2016.07.06 at 11:52:15 AM CDT

Figure 5.8: Querying all 'Explore' intentions that were performed in 'XY View (Detail)' by modifying the parameter 'X (Detail)' in any source view.

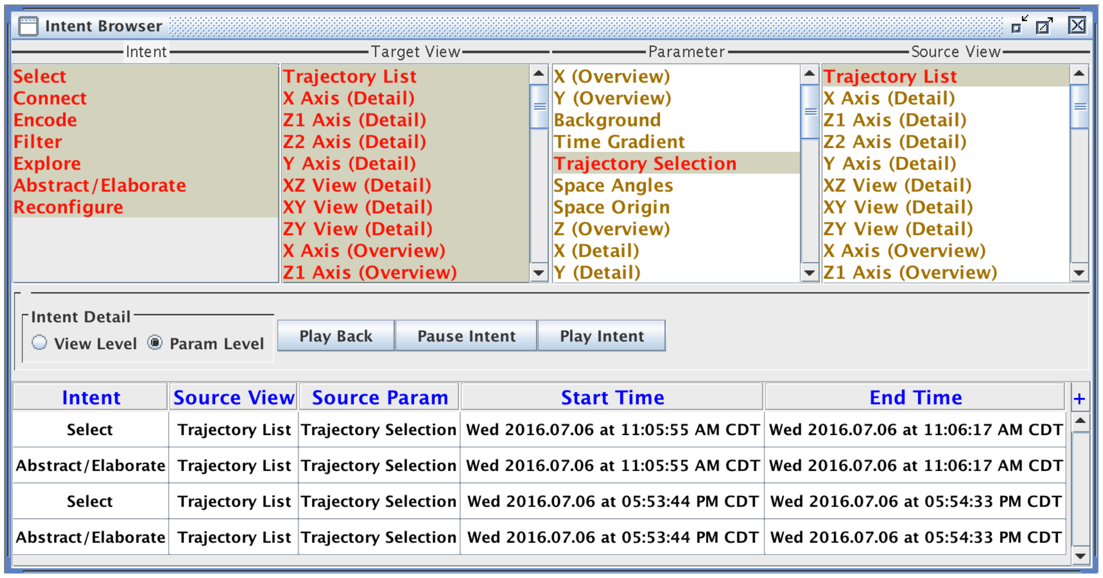


Intent	Target View	Parameter	Source View
Select	Trajectory List	Time Gradient	Points Table
Connect	X Axis (Detail)	Trajectory Selection	XY Ortho View
Encode	Z1 Axis (Detail)	Space Angles	YZ Ortho View
Filter	Z2 Axis (Detail)	Space Origin	ZX Ortho View
Explore	Y Axis (Detail)	Z (Overview)	X Ortho Axis
Abstract/Elaborate	XZ View (Detail)	X (Detail)	Y Ortho Axis
Reconfigure	XY View (Detail)	Y (Detail)	Z Ortho Axis
	ZY View (Detail)	Z (Detail)	X Ortho Axis2
	X Axis (Overview)	Point Selection	XYZ View (Overview)
	Z1 Axis (Overview)	Time Slider (Outer)	XYZ View (Detail)

Intent Detail: View Level [✓ Param Level]

Intent	Source View	Source Param	Start Time	End Time
Explore	X Ortho Axis	X (Detail)	Wed 2016.07.06 at 11:50:54 AM CDT	Wed 2016.07.06 at 11:51:00 AM CDT
Explore	X Ortho Axis2	X (Detail)	Wed 2016.07.06 at 11:52:10 AM CDT	Wed 2016.07.06 at 11:52:15 AM CDT

Figure 5.9: Querying all 'Explore' intentions that were performed in 'XY View (Detail)' by modifying the parameter 'X (Detail)' in selected source views only.



Intent	Source View	Source Param	Start Time	End Time
Select	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 11:05:55 AM CDT	Wed 2016.07.06 at 11:06:17 AM CDT
Abstract/Elaborate	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 11:05:55 AM CDT	Wed 2016.07.06 at 11:06:17 AM CDT
Select	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 05:53:44 PM CDT	Wed 2016.07.06 at 05:54:33 PM CDT
Abstract/Elaborate	Trajectory List	Trajectory Selection	Wed 2016.07.06 at 05:53:44 PM CDT	Wed 2016.07.06 at 05:54:33 PM CDT

Figure 5.10: Displaying all intentions that were performed in all views by modifying the parameter 'Trajectory Selection' in 'Trajectory List' view only.

The above queries are a subset of queries user can pose using the intent browser user interface. Queries 1 and 2 allow the user to get a sense of the intention log overall. These two queries can be expressed on both the view level and the parameter level intention data sets. A summary of intentions is provided at the view level. At the parameter level, fine-grain details of intentions are revealed. Queries 3 through 5 reduce search hits a great deal by allowing the user to filter the intentions data set in terms of details such as the parameter and/or source view invloved. These queries can only be expressed on the parameter level intention data set. Queries 1 and 2 can be used to get an overview of all intentions when the user does not know or remember the big picture of intentions performed. Queries 3 through 5 can be used to drill down into intentions having particular characteristics.

Queries that the Intent Browser Cannot Express

Like with every user interface, there are certain queries our user interface cannot express. For instance, our rule system cannot express **compound logic** queries. At times, the rule system may identify more than one user intention for a given interaction. In an *overview+detail* situation [17], when the user selects records in the overview, details are updated in the detail view. The rule system identifies a *Select* intention with the overview as source view and an *Abstract/Elaborate* intention with the detail view as target view. The user interface does not currently allow the user to search these types of intentions using conjunctive queries like *'Display all the states of visualization in which selection is detected in Overview* ***AND*** *Abstract/Elaborate is detected in Detail View.'* Similarly, users cannot express disjunctive queries such as *'Display all the states of visualization in which selection is detected in the ABC view* ***OR*** *Connect is detected in the XYZ view.'*

Chapter 6

Conclusion

There are several clear future directions for this work.

- Our current implementation of the system does not record low-level interactions. We can extend the system to record the low-level interactions, along with state changes and user intentions, to provide direct visual cues to the user when replaying interactions as described in the revise technique [15]. Visual cues will help the user in effectively reviewing search hits of the query.

- As explained in the previous chapter, the intent browser does not allow the user to perform conjunctive queries. We can extend the querying mechanism to allow the user express such queries.

- Another future direction would be to visualize and analyze the intention logs to study patterns in the way intentions are performed. Intention logs of different users can also be visualized and compared to discern similarities and dissimilarities in their interaction sequences and styles of visual exploration.

In this thesis, we have presented a novel history mechanism for supporting users in the sense-making activities of visual data analysis. Our history mechanism combines the Yi, et al. interaction taxonomy with the data state model to capture interaction histories at a semantic level. We believe that recording visualization activities in this way can help users recall and incorporate knowledge of the tasks performed into their broader data analysis processes.

The major contributions of this thesis are:

- a set of rules to convert low-level interactions into user intentions under the Yi, et al. taxonomy;
- an implementation of a rule system for detecting user intentions;
- the design and implementation of a coalescing mechanism for aggregating continuous user interactions;
- a system to record user intentions and corresponding state changes, integrated into the Improvise visualization environment; and,
- a user interface for querying and replaying recorded user intentions.

Bibliography

[1] D. F. S. Andreas Buja, Dianne Cook. Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, 5(1):78–99, 1996.

[2] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. Managing the evolution of dataflows with vistrails. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages 71–71, April 2006.

[3] E. H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Vizualization 2000*, INFOVIS '00, pages 69–, Washington, DC, USA, 2000. IEEE Computer Society.

[4] M. C. Chuah and S. F. Roth. On the semantics of interactive visualizations. In *Information Visualization '96, Proceedings IEEE Symposium on*, pages 29–36, Oct 1996.

[5] P. Cowley, L. Nowell, and J. Scholtz. Glass box: An instrumented infrastructure for supporting human interaction with information. In *System Sciences,*

2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, pages 296c–296c. IEEE, 2005.

[6] M. Derthick and S. F. Roth. Enhancing data exploration with a branching history of user operations, 2001.

[7] A. Dix and G. Ellis. Starting simple: Adding value to static visualisation through simple interaction. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '98, pages 124–134, New York, NY, USA, 1998. ACM.

[8] W. Dou, D. H. Jeong, F. Stukes, W. Ribarsky, H. R. Lipford, and R. Chang. Recovering reasoning processes from user interactions. *IEEE Computer Graphics and Applications*, (3):52–61, 2009.

[9] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1189–1196, Nov 2008.

[10] T. J. Jankun-Kelly, M. Kwan-Liu, and M. Gertz. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):357–369, March-April 2007.

[11] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, Jan. 2002.

[12] A. O. Mendelzon. Visualizing the world wide web. In *Proceedings of the Workshop on Advanced Visual Interfaces*, AVI '96, pages 13–19, New York, NY, USA, 1996. ACM.

[13] D. A. Norman. *The Design of Everyday Things.* Basic Books, Inc., New York, NY, USA, 2002.

[14] M. Pohl, S. Wiltner, and S. Miksch. Exploring information visualization–describing different interaction patterns. 2010.

[15] A. C. Robinson and C. Weaver. Re-visualization: Interactive visualization of the process of visual analysis. In *Proceedings of GIScience Workshop on Visual Analytics & Spatial Decision Support*, Münster, DE, September 2006.

[16] R. Shadoan and C. Weaver. Visual analysis of higher-order conjunctive relationships in multidimensional data using a hypergraph query system. *Visualization and Computer Graphics, IEEE Transactions on*, 19(12):2070–2079, Dec 2013.

[17] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, Sep 1996.

[18] Y. B. Shrinivasan and J. J. Van Wijk. Supporting exploration awareness in information visualization. *Computer Graphics and Applications, IEEE*, 29(5):34–43, 2009.

[19] J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, Jan. 1976.

[20] C. Weaver. Building highly-coordinated visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis)*, pages 159–166. IEEE Computer Society, October 2004.

[21] C. E. Weaver. *Improvise: A User Interface for Interactive Construction of Highly-Coordinated Visualizations*. PhD thesis, University of Wisconsin–Madison, Madison, WI, June 2006.

[22] J. S. Yi, Y.-a. Kang, J. T. Stasko, and J. A. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.