Multiple and Coordinated Views in Information Visualization

Maximilian Scherr

Abstract— Multiple views are not merely isolated separate views on data but mighty tools which often share a relationship. This relationship is brought to attention and utilized by coordination. Hence, in this paper I present an outline of the field of multiple coordinated views, its reasons, anticipations, jargon and examples. Because of its general nature, a large portion of this text deals with abstract formalizations, models and architecture descriptions. The latter is mainly revolving around user-generated coordination. Here, the works of North and Schneiderman (Snap-Together Visualization), Boukhelifa (A Coordination Model for Exploratory Multiview Visualization) and Weaver (Improvise) get major focus as they address coordination itself in different views and show development in their field. To conclude this paper, a small collections of interesting uses of multiple and coordinated views is presented as well as a brief discussion on recent issues.

Index Terms—multiple views, coordination, information visualization, abstract models, customized coordination

1 Introduction

Sucessfully aiding the seeking and discovery of information, two major purposes of information visualization, not only asks for ingenious ideas for a direct mapping of raw data to pixels. Diversity and the arrangement of such generated views on data also play crucial roles, for however natural or efficient a single specific view might seem, it does not evidently have to fulfill these asumptions for every imaginable situation. This inherent bias can be countered by multiple views.

While multiple views on data appear as an improvement over a single view, they also gives rise to various possible issues concerning screen space, computer performance and user perception[2]. Additionally coordinating multiple views does not necessarily solve these issues but it can compensate for them by facilitating the recognition of previously hidden relationships within the observed data[2].

The concept itself is not quite a novelty. Nowadays an average computer user frequently stumbles upon applications using coordinated multiple views on a regular basis, including file browsers, text editors, 3D modelers and the likes. Though we have to keep in mind that these applications usually do not fall into the category of scientific information visualization, they do however show the same features concerning multiple views and coordination. Even simple and common parts of an application such as a text field with scrollbar can in fact be identified as coordinated views.

In recent years the study of such *multiple and coordinated views* (MCV) has variedly progressed and even spawned a conference (*Coordinated and Multiple Views in Exploratory Visualization*) largely dedicated to this particular field of information visualization. The main subject of this research paper will be an analysis of different concepts and descriptions revolving around MCV, following a description of (historic) ideas to formalize and generalize coordination based on these classifications and an introduction of a small collection of specific MCV systems.

2 Terminology and Guidelines

In order to approach the following concepts a few general definitions are provided here in this section. Along the lines of Baldonado et al.[2] a *single view* shall be defined as the combination of a set of data together with specifications on how to display this data. Imagine a set of pairs, for example, representing time and temperature, size and weight, or coordinates. This kind of data could now be displayed in a single view by utilizing a list, a scatter plot, or some other hopefully appropriate technique. Such type of data representation used by a single view is refered to by the term *form*[14].

Now, a system design in which two or more forms are used to display (the same) data is called *multiform*[14]. If two or more views enable users "to learn about different aspects of the conceptual entity"[2], they may be called *distinct views*. This definition is in some way more general but also more specific than multiform at the same time. For example two scatter plots visualizing the same data in exactly the same way would definitely neither classify as multiform, nor as distinct views. However, if one of these scatter plots displays data in greater detail, users could observe data at a higher granularity and might thus learn about a different aspect. At first sight this may not seem to be in direct accordance with the definition of distinct views but it is actually wide enough to extend to such cases. In the case of multiform, I tend to be believe that such a setup does not fall under the term's definition since in both cases the same technique, that is a scatter plot, is utilized. Argueably this depends on whether such modifications alter a form enough to make it "different".

According to Roberts[14] the term *multiple views* generally refers to "any instance where data is represented in multiple windows"[14] whereas Baldonado et al.[2] strictly require distinct views for a *multiple view system*. I will stick to the former, more general definition here since it is less likely to interfere as easily with potentially stricter definitions used in the following concepts.

2.1 Mutliple Views

The class of multiple views can now be further divided. Usually this is done on grounds of the relationship between two so-called *side-by-side views*. Systems which are only using two such side-by-side views are called *dual-view systems*[14] but the following definitions are not necessarily limited to such:

Overview & detail views use one view to display the whole or at least a very large portion of the dataset and another view for showing a part of the datasets in greater detail. The example earlier of two scatter plots with different resolution falls into this category[14].

Focus & context views is actually not too different from the above. Apart from the semantical focus, stressing the detail before the overview, a context view does not have to display as much data as an overview, but only hint on the context. An example could be a text reader using three views with one being the central, big focus view, displaying several lines of text in an easily readable manner, and two others above and below, showing a certain amount of lines before and after respectively but in a size not quite suited for long reading. As Roberts[14] points out this technique is often used (in single view systems) together with

- *Maximilian Scherr is studying Informatics at the University of Munich, Germany, E-mail: maximilian.scherr@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Information Visualization, 2008/2009*

distortion, for example fish-eye magnification. There also exists an application of focus & context to virtual reality, called "world in miniature"[17], where the contexting world is shown as a miniature model within or possibly next to a full-sized reality imitating viewport.

Difference views focus on highlighting differences in the observed data, usually achieved by merging several views together[14]. An example for such usage is having two similar, yet different images, wherein differing pixel regions are marked by color. The Subclipse plugin for the popular software IDE Eclipse uses such an approach for sourcecode merging in case of version control conflicts.

Small-multiples are "shrunken, high-density graphics based on a large data matrix"[18]. Encouraging comparison, this choice of arrangement usually implies that the views use the same visualization form. They are "often narrative in content, showing shifts in relatonship between variables as the index variable changes"[18]. Several arranged miniature views representing temperature or rainfall data on top of a map at different times would be an example for this kind of multiple views.

This list is by no means exhaustive. Rather than that it merely represents common appearances of multiple views.

2.2 Coordination

Intuitively, whenever such relationships exist one would expect them to be reflected while interacting with multiple views. In fact, according to Roberts[14], when talking about multiple view systems, coordination is often implied.

The coordination of views requires specifications or mappings that make changes in one view affect the others. Such mappings are specified by so-called *coupling functions*[2]. When exactly or under which conditions these coupling functions are called has to be determined separately in a so-called *propagation model*[2].

Coordination is apparent foremost when user interaction comes into play. A very common coordination is called *brushing*, which means that upon selection of elements in one view the same (or related) elements are simultaneously highlighted in other linked views[2], which is especially useful for *multiform* views to find similarities and anomalies in the data[14]. An extension relying on a repeated brushing technique has been described by Wright et al.[9]. The *brush* metaphor usually refers to how and how much data is selected[14]. When the data to be visualized can be filtered or constrained by sliders, input fields, drop-down menus and such, the term *dynamic querying* is used[14]. This can influence several views and also become part of coordination.

Next to *linking* data across views, *navigational slaving* constitutes yet another common interaction technique[2]. This term describes a relationship between views in which navigational actions in one view are propagated to linked views. For example synchronized scrolling in side-by-side difference views would be a form of two-way navigational slaving. As would be zooming, panning and similar, for example, in a multiple view map application.

Navigational slaving is not restricted to a mapping of navigation in one view to navigation in another. North[11] describes possible occurrences of such case with a 2x3 taxonomy (*see figure 1*). This definition of coordination is restricted to three types (on navigation and data items), that are *selecting items* $\leftrightarrow$ *selecting items*, *navigating views* $\leftrightarrow$ *navigating views* and finally *selecting items* $\leftrightarrow$ *navigating views*. Furthermore he classifies by whether the data collections are different or the same. In the former case, relationships between the collections have to be explicitly specified[11]. Imagine a setup in which one view lists art museums and further information about them, while another view shows a city map. For a coordination that highlights the location of a selected museum in the map, it has to be specified that the adress record in the museum's information is to be used and mapped to the other view. Same data collections already have an implicit relationship[11].

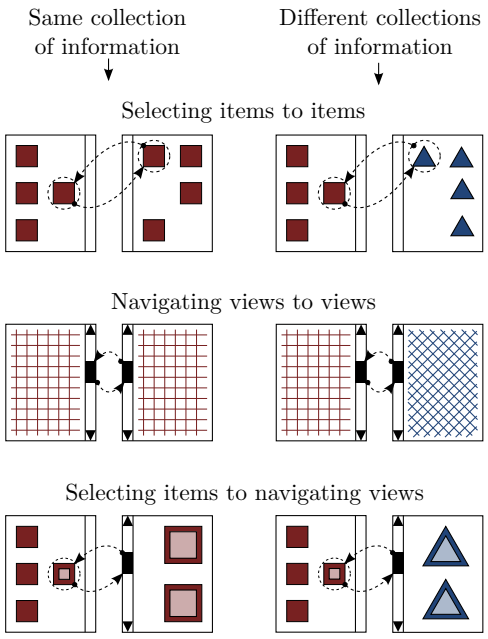

Fig. 1. "2x3 taxonomy of multiple window coordinations" (*modified after North[11], see diagram 1*)

We have to keep in mind, that although interaction is the first that comes into mind, coordination is more general. Whenever an event changes one view (for example an automated update every minute) and this change effects another view, coordination is partaking.

2.3 Issues and Guidelines

If used properly, multiple views can minimize cognitive overhead compared to a data visualization via only a single view. Used in the wrong way multiple views can have quite the opposite effect[2]. Baldonado et al.[2] identify the following aspects that influence utility of an MCV system and could become causes of issues:

- *Learning time and effort* required to use the system properly.
- *Load* on user's working memory when using the system.
- *Comparison effort* required when using the system.
- *Context switching effort* required when using the system.
- *Computational power* required by the system.
- *Display space* required by the system.
- *Design, implementation and maintenance* resources required by the system.

In order to avoid that, several guideline rules have been presented by Baldonado et al.[2]. The *rule of diversity* ("Use multiple views when there is a diversity of attributes, models, user profiles, levels of abstraction, or genres."), the *rule of complementary* ("Use multiple views when different views bring out correlations and or disparities."), the *rule of decomposition* ("Partition complex data into multiple views to create manageable chunks and to provide insight into the interaction among different dimensions."), and the *rule of parsimony* ("Use multiple views minimally.") address the fundamental conditions under which sensible usage of multiple views is recommended, or as Baldonado et a.l[2] put it, they cover the task of selecting views.

The second set of rules covers view presentation and interaction, which involves coordination: the *rule of space/time resource optimization* ("Balance the spatial and temporal costs of presenting multiple views with the spatial and temporal benefits of using the views."),

Table 1. Summary of guideline rules (*modified after Baldonado et al[2].*)

Rule of ...	Major positive impacts on utility	Major negative impacts on utility
... diversity	memory	learning, computational overhead, display space overhead
... complementary	memory, comparison, context switching	learning, computational overhead, display space overhead
... decomposition	memory, comparison	learning, computational overhead, display space overhead
... parsimony	learning, computational overhead, display space overhead	memory, comparison, context switching
... space/time resource optimization	learning, computational overhead, display space overhead	memory, comparison, context switching
... self-evidence	learning, comparison	computational overhead
... consistency	learning, comparison	computational overhead
... attention management	memory, context switching	computational overhead

the *rule of self-evidence* ("Use perceptual cues to make relationships among multiple views more apparent to the user."), the *rule of consistency* ("Make the interfaces for multiple views consistent, and make the states of multiple views consistent."), and the *rule of attention management* ("Use perceptual techniques to focus the user's attention on the right view at the right time.")

When applying these rules tradeoffs have to be considered. Applying the rule of parsimony, for instance, might put a strain on the user's ability to compare relevant data. For a summary of the rules and their possible effects on a MCV system see table 1.

3 Approaches to Abstract Models

Now that multiple views and coordination have been formally introduced I will go about introducing some of the major ideas researchers have come up with in recent years to generalize MCV. Not only can such formalization help implementing visualization frameworks themselves, it also provides a means to allow user defined coordination and thus customization of MCV systems.

3.1 North and Shneiderman.: Snap-Together Visualization

The first approach is the *Snap-Together Visualization* described by North and Shneiderman.[12] It tries to deal with the issue that users might be interested in coordinations not foreseeable by a developer for all possible tasks. It focuses mainly on information exploration and not manipulation/editing tasks.[12]

3.1.1 Ideas and Goals

Snap-Together Visualization both acknowledge user specific needs and "give users the coordinated multiple-view interfaces they want, yet, at the same time, save designers from endless development of coordinations."[12] Especially at the time of this system's devising most MCV systems had been static. Adding new, maybe not so common coordination required custom programming. The goals of this approach are not only simplified custom coordination, but also easy integration into projects, making it easy for developers to add "snap-ability" to their application. If actually performed this could give users a wide range of third party visualizations for utilization in their information exploration tasks[12].

3.1.2 Model and Terms

The model of North and Shneiderman's approach is based on the relational database model, which holds *information*, the basis for visualization, view generation and coordination. One information unit is called *object* and is represented as tuple inside of the database. Building on top of a relational database offers a main advantage in it having already formalized concepts such as unique identifiers (primary key) and relationships. We will see, that it also provides good methods to share queries when coordinating and updating views.

In this system the term *visualization* is used equally to the definition of view in section 2, a visual representation of a set of objects. This visualization can be filled with life by queries on the underlying database, which load the requested data.

North and Shneiderman present three major categories of *user actions*, namely *select* (for example clicking, hovering et cetera), *navigate* (for example scrolling, zooming et cetera) and *query* (as described above).

Finally, *coordination* is here defined on user actions on objects (again a limitation compared to the general definition in section 2) and their mapping across visualizations (views).

3.1.3 Usage

At the basis of this system's usage lies an application that serves as frontend to a relational database. Additionally creating and opening views as well as coordination are all integrated into this application and can be managed there by the user. This application may be regarded as a helper application for (third party) visualization applications.

The standard procedure for using the system is described by North and Shneiderman[12] as follows:

1. A user queries (or in the simplest case merely chooses to retrieve a table of) the database and thus creates a view. Existing views can also be updated with new data by queries. A drag and drop mechanism is employed to do this, as well as throughout most subsequent tasks. I will not discuss the exact usage details here.

2. Coordination is established by "snapping visualizations together" with help of the helper application. At this point the user has to choose what actions to coordinate. Again these coordinations can be modified later on.

3.1.4 Architecture

"[Snap-Together Visualization] is a centralized software system that acts as an intermediary among the visualization tools and the database."[12] The system is supposed to be informed by visualization tools about their snap-able actions upon initialization.

Actual snapping is performed between two visualizations vis_a and vis_b by a mapping of the form

$$(vis_a, action_a, objectid_a) \Leftrightarrow (vis_b, action_b, objectid_b)$$

where $action_a$ and $action_b$ are user defined actions to be coordinated and $objectid_a$ and $objectid_b$ are unique object identifiers, which in most cases are expected to be equal, "as in primary-key to foreign-key joins."[12]

Such information is stored in a coordination graph, the nodes of which are visualizations and the links of which are the described snap mappings for incident visualization nodes. Through inter-process communication the system is notified of user actions by the visualization (application), upon which this coordination graph is traversed and communicates the mapped actions to the snapped visualizations.

As mentioned earlier, applications have to be made snap-able before any of the described tasks can be performed. According to the authors, the role of their system can be compared to copy-and-paste features which are controlled by a centralized process in the background.

Such a system requires applications to implement simple hooks for interoperability. In case of Snap-Together Visualization these hooks

are *initialization* (notification of available user actions for coordination), *action notification* (propagating of events upon user action), *action invocation* (interface to methods resulting in an action on a given object) and finally *load* (reading and displaying given data).[12]

3.1.5 Evaluation

When designing such system directed at end-user utilisation it is of utter importance to evaluate acceptance and performance in user studies. As the authors mention in a study on their Snap-Together Visualization (or *Snap*), it is important to study its use for evaluation of usability, benefit, discovery "of potential user interface improvements", and gaining of "a deeper level of understanding about users' ability to understand, construct, and use coordinated-view strategies in general"[13].

This study investigated both, construction of MCV interfaces by users, and its subsequent operation. Participants were given different tasks to fullfil. While observing cognitive trouble spots and user interface problems the team measured the participant's background information, learning time, success, and time to completion[13].

In short, the results of this user-study are:

- Participants were excited to use the system.
- They were overall quick to learn it.
- Several felt satisfaction about being able to create exploration environments and with the ability to effectively use a coordinated visualization.
- They state that exploring with their custom-built system was "effortless compared to the standard tools they [were] used to."[13]
- A variety of solution processes for the given tasks indicated the ability to creatively handle the system, that is adjust them to their very own personal needs.

Despite some issues concerning the user interface (that I have chosen not to describe in detail here anyway) the overall and arguably most important result of this study was that the participants "did not have problems grasping the cognitive concept of coordinating views. They were able to generate designs by duplication and by abstract task description."[13]

3.2 Boukhelifa et al.: A Coordination Model for Exploratory Multiview Visualization

While Snap-Together Visualization provides easy to use methods for building own coordination designs, it is limited by its centering on a relational database backend. Exploratory visualization supports more types of interactions than Snap provides and thus requires a wider approach[3].

Boukhelifa et al.[3] introduce a model that, although similar to Snap-Together Visualization in some parts, "handles coordination from a more general viewpoint and takes in consideration exploratory visualization needs for rich and varied user interactions."[3]

3.2.1 Coordination in Detail

Acknowledging the need for freedom of coordination and an abstract definition thereof, Boukhelifa et al. define essential parts or design issues of a coordination for their system as follows[3]:

Coordination entities: This defines the exact subjects of coordination, for instance view, parameter, data, process, event, and of course aspects of the displaying window.

Type: Very close to the concept of a type in a programming language coordinations have a type as well, be it primitive or complex. Translation between types might also become necessary.

Chronology: This aspect covers a coordination's *lifetime*, dealing with how long a coordination persists and *scheduling*, dealing with matters such as synchronism (or asynchronism).

Scope: The scope of a coordination restricts in terms of link-ability, for example whether any arbitrary entity can be connected.

Granularity of links: This looks at how many entities and how many views there are in a coordination, as well as how many links an "entity contributes to coordination".

Initialization: How a coordination is created (also how links between entities are created) is contained in this aspect.

Updating: Several update models can be applied, for instance user *initiated updating*, *lazy updating*, or *greedy updating*. Updates can lead to inconsistencies that have to be resolved, or even better, avoided from the beginning.

Realization: This aspect decides on general realization matters such as if and how to convey which entities are linked to each other (for example by lines) as well as how users coordinate and interact.

3.2.2 Model

Ideally the model "should be flexible, adoptable, extensible and foster better visual exploration."[3] Additionally, a large degree of freedom to be able to describe all sorts of coordinations is desired, as well as the facilitation of their testing.

At the center of Boukhelifa et al.'s model lie the so-called *coordination objects*, which reside in a *coordination space* and manage (coordination) entities. For each type of coordination a single object is considered to be present, for instance, one object for selecting, responsible for selection-related coordinations (such as brushing), and one for rotation, responsible for all rotation-related coordinations[3].

Views are coordinated when they are linked to a common communication object. Figure 2 shows two views and two (different) coordination objects, which are linked to the maximal possible extent. This linking has two forms in the authors' model, namely translation functions (see $f_{i,j}$ with $i, j \in \{1,2\}$ in figure 2) and notifications, which are invoked in case an event makes changes to the linked coordination object[3].

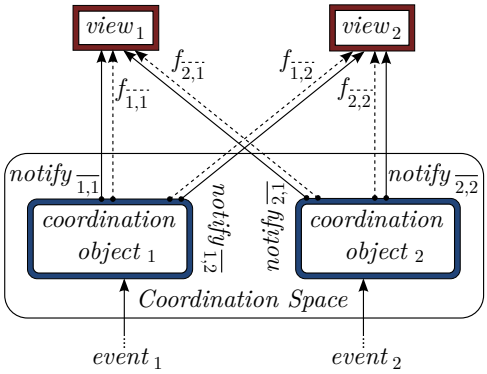

Fig. 2. Example of Boukhelifa et al.'s abstract model (*modified after Boukhelifa et al.[3], see figure 1*)

The authors stress the model's dynamic nature by stating that "views may be added and removed without other views that also access the same coordination object necessarily having knowledge of this activity. Importantly, views do not need to know about other views in the coordination."[3]

They go one step further by extending the abstract model applying it to the several stages of the so-called *dataflow* model. The dataflow paradigm for visualization described by the authors puts data at the first stage or layer. This data is *enhanced*, then *mapped* into an abstract visualization object. Finally this object can be *rendered* and subsequently *transformed* however often it is necessary[4][5]. Applying the earlier, abstract coordination model to the dataflow paradigm,

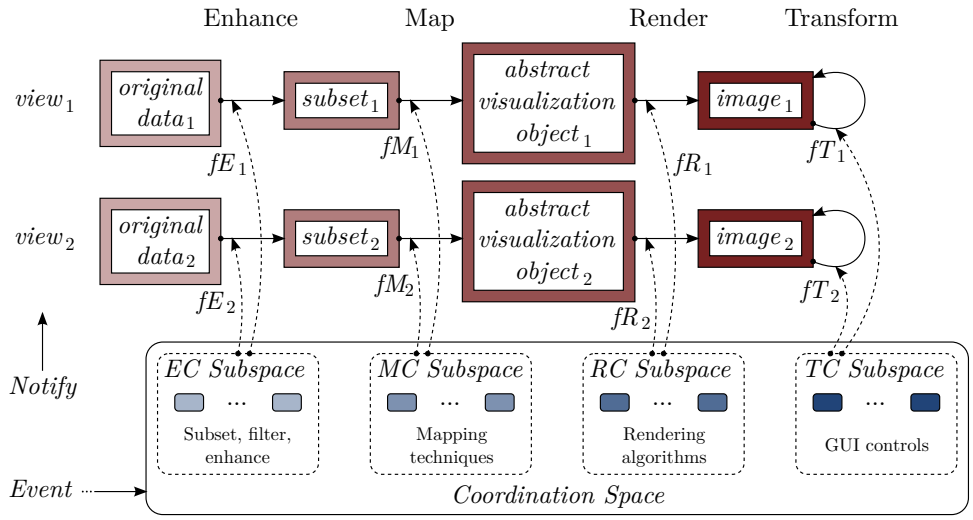

Fig. 3. Boukhelifa et al.'s layered model (*modified after Boukhelifa et al.[3], see figure 3*)

coordination can occur at any of these layers (*see figure 3*). However, it typically tends to happen at the mapping and rendering stages[3].

Since view parameters and coordination object parameters may not be identical, abstract parameters are introduced, which can be passed to translation functions (which in turn convert abstract parameters to view parameters). By these abstract parameters the coordination space can be divided into "four varieties of coordination sub-spaces"[3] (*see figure 3*).

What modifies those abstract parameters? Adequately typed events do, be they user action originated (in which case the views generate them) or occur automatically as result of some background process. As described earlier, notifications are responsible for updating linked views in response to events. It is not only mandatory to define translation functions to each view, it is also mandatory for views "to be registered to receive notify events."

After an event has occurred, the notify handler of a linked view is triggered and finally translation functions (according to the coordination object of origin) are called[3].

3.3 Weaver: Improvise

I have briefly introduced two approaches to MCV, one being simple and limited by its coordination actions and database, the other being a rather theoretical and abstract model. The approach presented by Weaver[19], called *Improvise*, adopts some traits of both and tries to balance between user coordination tradeoffs.

3.3.1 Goals

According to Weaver[19] visualization systems allowing for user controlled coordination are either limited in coordination options but on the other hand let users do this in a simple way (by offering a predefined set of coordinations) or flexible but on the other hand make it more difficult for users to coordinate (by requiring them to write coordination scripts). Thus, "the primary goal of Improvise is to enhance data exploration by offering users fine-grain control over the appearance of visualized data while preserving their ability to work quickly and easily."[19] To do this, the author proposes a visual abstraction language and a coordination mechanism based on shared-objects, which is combined with indirect coordination through a query mechanism[19].

3.3.2 Architecture

Improvise has two concepts that govern coordination, one being direct the other being indirect. The concept associated with direct coordination is called *live properties*.

Coordination in Improvise is performed on *controls*, for instance views, sliders et cetera. For each of these controls one or more live properties are defined. A single (live) property can bind to at most one shared object, called *variable* here (which can, by definition of a shared object, be bound by an arbitrary number of different live properties). A property that only accesses its bound variable is called *passive property*, one that also modifies is called *active property*[19] (*see figure 4*).

Controls are thus never directly modified (by variables) within the coordination mechanism. Instead variable changes are propagated to controls via changes of live properties (and vice versa for the relationship between controls and variables)[19]. Similar to *properties* in object-oriented programming, live properties are a way to conveniently access the control's data from the outside, and in this case data (at least that data, which is relevant for coordination) is actually stored in the property for the control (for instance a slider's position). Live properties are strongly typed and initialized with a default value in case of no variable binding[19]. If we regard control and properties as a single entity, view and variable as coordination object, similarity to Boukhelifa et al.'s abstract non-layered model becomes evident.

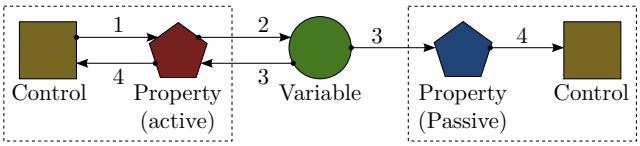

Fig. 4. Direct coordination (*modified after Weaver[19], see figure 1*)

Weaver describes the scenario depicted in figure 4 as follows[19]:

1. "A control modifies the value of one of its (active) live properties in response to interaction."
2. "The live property assigns the new value to its bound variable."
3. "The variable sends a change notification to all live properties bound to it."
4. "The live properties notify their respective parent controls of the change [and the] controls update themselves appropriately."

While live properties are a concept for *direct coordination*, *indirect coordination* can be achieved by so called *coordinated queries*, which "is a visual abstraction language based on the relational database model."[19] Here we can see a similarity to Snap-Together Visualization, yet the concept of coordinated queries appears to be more flexible. The main part of the abstraction language are query operations

that are made up by *expressions*. "An expression is a tree of operators that calculates the value of an output field using the fields of a [sic] input record."[19]

The Improvise implementation provides users with a dedicated expression editor, enabling them to construct complex queries from a variety of operators: *function operators*, *value operators*, *attribute operators*, *aggregate operators*, *constant operators*, *index operators*, and finally *variable operators*[1].

The key to indirect coordination lies in the variable operators, which during evaluation "take on the current value of their corresponding variable."[19]. A central database, called lexicon stores data, query operations et cetera. Its elements are called *lexicals*.

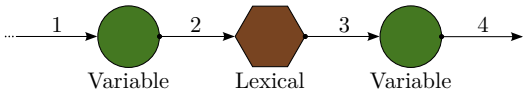

Fig. 5. Indirect coordination (*modified after Weaver[19], see figure 3*)

Weaver describes indirect coordination (situation in figure 5) as follows[19]:

1. "An upstream object propagates a value change to a variable."
2. "The variable notifies all lexical values that contain expressions which reference the variable."
3. "Each expression notifies variables to which it is assigned as a value."
4. "The variable sends a change notification to all downstream objects. Upstream and downstream objects can be live properties (as in [figure 4]), or other lexical values."

Figure 6 shows an example of direct coordination for a view that displays data in a scatterplot and two axis control views.

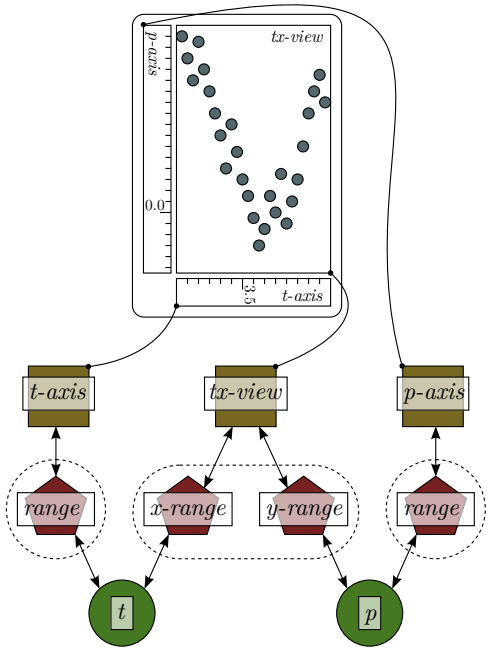

Fig. 6. Direct coordination example (*modified after Weaver[19], see figure 5*)

[1]Please refer to Weaver's paper[19] for detailed information.

3.3.3 Results

According to Weaver[19] "highly-coordinated visualizations appear to be much easier to build in Improvise than other visualization systems." He attributes this to indirect coordination, that links every aspect of a MCV system together in a flexible manner. The system was fully implemented, however, as of the time of writing his paper[19] no comparative user studies had been conducted yet.

4 Problem-specific Applications of MCV

Whether all the approaches in the previous section have actually been incorporated into MCV systems or not, one reason I chose to discuss them is, that they present what has to be considered when building a coordinated multiple view visualization and how coordination there can be improved.

Hence, while not directly related to the previous section, I tried to compile a small overview of problem-specific applications of MCV for this last section, to give an idea of what tasks it has been used for explicitly, and what research has resulted in.

4.1 Da Silva Kauer et al.: An Information Visualization Tool with Multiple Coordinated Views for Network Traffic Analysis

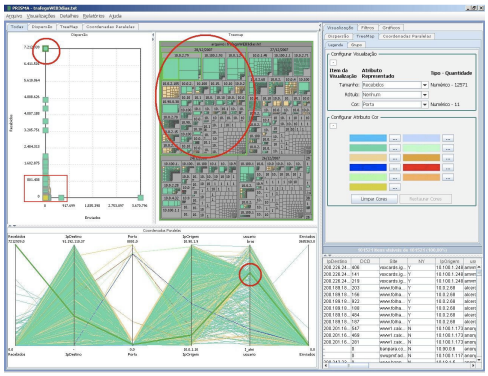

Fig. 7. Screenshot showing brushing (*see da Silva Kauer et al[6]., figure 2*)

The system by Kauer et al. was designed for network traffic analysis. It uses the PRISMA[8] visualization framework and represents a typical MCV design with static coordination between views. As you can see in figure 7, several views containing different types of visualization are used.

4.2 Shimabukuro et al.: Coordinated Views to Assist Exploration of Spatio-Temporal Data: A Case Study

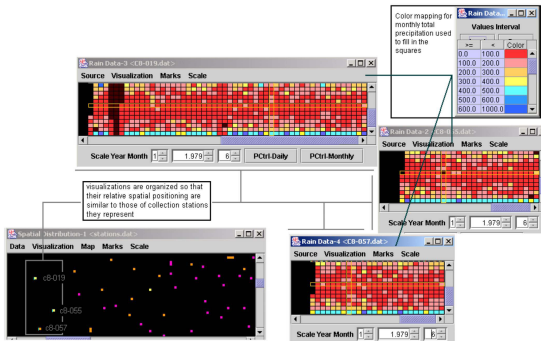

Fig. 8. Window placement reflects location on map (*see Shimabukuro et al.[15], figure 7*)

The system by Shimabukuro et al.[15], is interesting for applying a novel way of visualizing temporal data, and for using multiple views

enabling them to effectively combine this data with spatial data. For example, the user sees a map from which he chooses locations, to which certain temporal data has been collected (in this case climate data). Corresponding views are then arranged in a way that their position on screen reflects their locations' arrangement on the map (*see figure 8*).

4.3 Masui et al.: Multiple-View Approach for Smooth Information Retrieval

Fig. 9. Screenshot showing all four different views (*see Masui et al.[10], figure 1*)

Masui et al.[10] describe a tourist information system (for Nara, Japan) supporting tourists with a three-dimensional map view showing points of interest, category views and lists to find points of interests. One coordination is for instance, that details about data objects in the three-dimensional view are displayed in a separate window when coming into proximity of the view's center(*see figure 9*). It supports search for points of interest by employing real world search strategies, that is with the help of multiple and coordinated views "any vague knowledge about the data can be utilized to narrow the search space."[10]

4.4 Do Carmo et al.: Coordinated and Multiple Views in Augmented Reality Environment

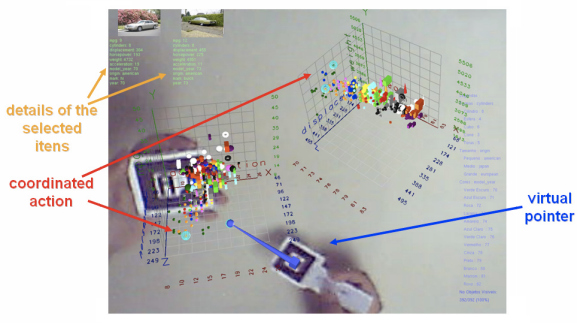

Fig. 10. Two coordinated scatter plots and in the top-left corner coordinated details (*see do Carmo et al[7]., figure 10*)

Combining multiple and coordinated views with augmented reality gives new display options concerning arrangement as well as new user interactions (such as navigation by perspective change). Results of a user study, conducted by do Carmo et al.[7] on their system, showed that most users did not encounter major difficulties adapting to the rather unusual setup. To manipulate views, the system relies on a set of markers and their handling(*see figure 10*). Participants of the user study reported to enjoy the mixture of virtual and real objects, the immersive environment in general, and profited from freedom of movement and manipulation.

5 Discussion and Final Words

In this paper I have tried to introduce the field of multiple and coordinated views. Though the abundance of systems using MCV may have prevented me from choosing the best ones, I hope I was able to give a cross-section through the field.

What struck me the most was that considering that some of the presented ideas are nearly ten years old, progress, especially for customized coordination in commercial systems seems to have been slow. In a recent paper Andrienko and Andrienko[1] attribute this to existing tools and approaches being "insufficiently suited to real-life problems."

To me this comes as a surprise, as approaches like the ones described in section 3 have become more and more general, in fact general enough to cover most cases or at least more and more cases as time passes. Apparently slow processing of large data volume, or in other words, non-scalability of MCV systems is the main cause for Andrienko and Andrienko's judgement. With more data to be visualized visualization itself and coordination come to a limit. In fact, while researching I also felt that most approaches fail to address this issue thoroughly enough.

What their paper also proposes is a deviation from Shneiderman's apparently (in the field of MCV) widely followed *information seeking mantra* "overview, zoom and filter, details-on-demand"[16]. They state that maybe applying "analyse first - show the Important - zoom, filter and analyse further - details on demand" (the *visual analytics mantra*) could be a suitable solution as this procedure "stresses the fact that fully visual and interactive methods do not properly work with big datasets."[1]

For me all the many different, yet not too different formalization approaches and visualization systems give off a general atmosphere of dissatisfaction, and in fact I have personally not come across any commercial application allowing for the described levels of customizability. In the end, one still has to decide what visualization method is best suited for a specific problem, which is not an easy task to begin with[1].

Despite these issues, multiple and coordinates views are still a hot topic and I anticipate further research and development in this area.

References

[1] G. Andrienko and N. Andrienko. Coordinated multiple views: a critical view. *Coordinated and Multiple Views in Exploratory Visualization, International Conference on*, 0:72–74, 2007.

[2] M. Q. W. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, 2000.

[3] N. Boukhelifa, J. Roberts, and P. Rodgers. A coordination model for exploratory multiview visualization. In *Coordinated and Multiple Views in Exploratory Visualization, 2003. Proceedings. International Conference on*, pages 76–85, 2003.

[4] S. Card, J. MacKinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think. Series in Interactive Technologies*, volume 340.

[5] E. Chi. A taxonomy of visualization techniques using the data state reference model. *Proceedings of the IEEE Symposium on Information Vizualization*, 2000.

[6] A. L. da Silva Kauer, B. S. Meiguins, R. M. C. do Carmo, M. de Brito Garcia, and A. S. G. Meiguins. An information visualization tool with multiple coordinated views for network traffic analysis. *Information Visualisation, International Conference on*, 0:151–156, 2008.

[7] C. do Carmo, R. Melo, B. Meiguins, A. Meiguins, S. Pinheiro, L. Almeida, and P. Godinho. Coordinated and multiple views in augmented reality environment. In *Information Visualization, 2007. IV'07. 11th International Conference*, pages 156–162, 2007.

[8] P. I. A. Godinho, B. S. Meiguins, A. S. G. Meiguins, R. M. C. do Carmo, M. de Brito Garcia, L. H. Almeida, and R. Lourenco. Prisma - a multidimensional information visualization tool using multiple coordinated views. *Information Visualisation, International Conference on*, 0:23–32, 2007.

[9] L. Lever and M. McDerby, editors. *Click and Brush: A Novel Way of Finding Correlations and Relationships in Visualizations*, University of Kent, UK, June 2005. Eurographics Association. (Electronic version www.diglib.eg.org).

[10] T. Masui, M. Minakuchi, I. George R. Borden, and K. Kashiwagi. Multiple-view approach for smooth information retrieval. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 199–206, New York, NY, USA, 1995. ACM.

[11] C. North. Generalized, robust, end-user programmable, multiple-window coordination. *Research Proposal, University of Maryland Computer Science Dept*, 1997.

[12] C. North and B. Shneiderman. Snap-together visualization: Coordinating multiple views to explore information. Technical report, 1999.

[13] C. North and B. Shneiderman. Snap-together visualization: Can users construct and operate coordinated visualizations? *International Journal of Human-Computers Studies*, 53(5):715–739, 2000.

[14] J. C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. *Coordinated and Multiple Views in Exploratory Visualization, International Conference on*, 0:61–71, 2007.

[15] M. H. Shimabukuro, E. F. Flores, M. C. F. de Oliveira, and H. Levkowitz. Coordinated views to assist exploration of spatio-temporal data: A case study. *Coordinated and Multiple Views in Exploratory Visualization, International Conference on*, 0:107–117, 2004.

[16] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, IEEE Symposium on*, 0:336, 1996.

[17] R. Stoakley, M. J. Conway, and R. Pausch. Virtual reality on a wim: interactive worlds in miniature. pages 265–272, 1995.

[18] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press LLC, 2nd edition, 2001.

[19] C. Weaver. Building highly-coordinated visualizations in improvise. *Information Visualization, IEEE Symposium on*, 0:159–166, 2004.