Chapter VI: Conclusion & Future Work

In this thesis we extended automatic visualization design to include all four phases of the visualization creation process: data, mapping, graphical, and rendering. Together with this expansion, we also enable input devices to be added during design, thus enabling interactive visualization interfaces to be created automatically. Previous automatic systems focussed only on the use of mapping operations. We show that by including the full set of visualization functions we expand the range of designs that can be generated and this allows us to address data analysis tasks more effectively. Specifically we can offload cognitive tasks onto the computer system with data transform techniques as well as address readability issues such as occlusion, display density, dwarfing, and spatial separation. We show in appendix E (the evaluation section) that this added functionality can significantly decrease total task performance time. Such improvements in the quality and breadth of designs will enable automatic visualization systems to better communicate information to users as well as provide better assistance to designers for creating visualizations. The focus of our work is on the domain of exploratory data analysis however many of the theoretical concepts developed is applicable to visualization design in general.

VI-1 Summary & Relevancy of Work

We expand automatic design systems to include the four phases of the visualization creation process by developing three core technologies:

1. A framework of the visualization creation process,
2. Metrics and heuristics for measuring the goodness of visualization designs,
3. An automatic visualization design system (AVID) that utilizes our theoretical framework and heuristics to generate rendered visualization interfaces.

Our framework and heuristics are necessary for enhancing automatic visualization design research, however, they are also applicable for aiding human designers in creating and prototyping visualizations. Specifically they provide a structure and methodology for creating visualization techniques and systematically exploring the design space.

VI-1.1 Methods: Framework of the Visualization Creation Process

In chapters II and III we developed a framework for characterizing commonly used functions in each of the four visualization phases. We also show how these functions can be combined with each other, with input devices and with visualization elements through a set of composition rules. This framework is **flexible** so that as new techniques get developed, additional functions can be included with minimal effort. In addition by composing the new functions with existing functions, we can leverage off of previous operations to generate a wide range of new visualization techniques. The framework is also **general** in that it encompasses a wide range of visualization techniques and can be integrated with a wide range of visual representations. Throughout this document we have presented *chart*, *map*, *table* and *grid* visualizations. These visualizations may contain *marks*, *bars*, *lines* or *text*. We have also presented many different visualization techniques including *dynamic query sliders*, *painting*, *aggregation*, *drag & drop*, *SDM techniques*, etc. Finally the framework is also **practical** for three primary reasons:

1. *Tailoring visualization techniques*: The framework provides designers with a visualization technique toolkit. This allows designers to easily modify and tailor visualization techniques to suit different task requirements. In addition it also simplifies the sharing and transfer of functionality across techniques.
2. *Design methodology*: Our framework divides the design of visualization techniques into two levels: functional and instantiation. At the functional level of design we populate the technique with the necessary functions to perform our desired tasks. At the instantiation level of design we expand the functional design with input devices and visualization elements and properties, thereby establishing the "look and feel" of the technique. This two tier design process allows functional decisions to be made free from hardware and aesthetic constraints so that we do not falsely constrain function based on form. It also allows us to identify functionally similar techniques that may have very different "look and feel" so that we may more accurately compare and borrow design strategies across techniques.
3. *Systematic exploration of visualization techniques design space*: Our framework also allows us to define and lay out the current explored areas in visualization technique design and identify areas that are less populated. By using this "map" we can systematically expand the visualization techniques design space by combining existing methods or by crafting new techniques in the less explored areas.

VI-1.2 Principles: Metrics & Heuristics for Measuring the Goodness of Visualization Designs

Our visualization technique framework provides a language for describing and creating visualization techniques. However, it does not tell us which techniques are the most effective or appropriate for solving our data analysis tasks. For any particular task, there are commonly many alternative techniques that can be used. Thus it is crucial that we have some way of measuring the goodness of these various alternatives and some guidelines for directing us towards the more promising design paths. Earlier work on automatic visualization design considered metrics and design rules for using mapping transforms based on data and task requirements. In our work we expand on previous work and develop metrics and guidelines for evaluating all phases of the visualization process including data, mapping, graphical, and rendering transforms.

Our metrics framework determines the effectiveness of a visualization design based on the four distances: articulatory, functional, expressive, and observational. This metrics framework is used by our automatic system to evaluation the effectiveness of possible design alternatives. In addition, these metrics can also be used by designers as general design yardsticks to help them create more effective visualizations and avoid design mistakes. Note that it is also possible to evaluate the effectiveness of visualization designs by using more procedural methods such as GOMS [*Card, 1983*]. This was attempted by Lohse [*Lohse, 1993*] who automatically decomposed each task and visualization design pair into a set of GOMS operators (similar to the GOMS sequences shown in appendix E of this thesis). The problem with using GOMS in an automatic design system, however, is that it is time consuming and difficult to apply to partial designs. It can also be difficult to isolate from a GOMS evaluation which particular design decision resulted in the ultimate high or low cost of a visualization.

Based on our metrics framework we also develop a set of design rules or heuristics that help direct our automatic design system towards more promising design paths. It would be very time expensive and unfeasible to explore and rate **all** possible design alternatives before presenting them to users. As a result it is important that we have some guidelines to help focus our design efforts on the more promising design possibilities while culling out design spaces that are ineffective or inexpressive of our tasks. In chapter IV we presented design rules that help us determine when it might be more useful to perform a task or subtask perceptually by mapping it to graphics, and when it might be more advantageous to let the system compute the task through data transforms and only visualize the pre-computed results. In appendix F, we present additional design rules that consider readability issues such as occlusion, display density, data dwarfing, and information presence, and how graphical and rendering transforms can improve the readability of a visualization design.

VI-1.3 Systems: AVID – Automatic Visualization Interface Designer

To showcase the applicability of our visualization framework and heuristics in "real" systems, we implemented an automatic design system (AVID). AVID accepts one or more tasks as input and produces a set of visualization designs as output, ranked according to effectiveness of the designs with respect to the input task(s). Our AVID system was used to generate most of the visualization designs shown in this document. The wide range of examples generated by our system shows the flexibility and generality of our theoretical framework and heuristics. Our ability to translate the theory into active systems indicate that our theory is relatively complete. And appendix E (GOMS evaluation) shows that AVID produces practical design results that do indeed conform to cognitive, perceptual and articulatory complexity. Our evaluation results also show that our work significantly expands the design space of automatic visualization systems and enables more effective designs to be generated than was previously possible.

This design system can ultimately be integrated with a system like AutoBrief to enable higher level analysis and planning systems to automatically communicate complex information and relationships to users in the form of both text and graphics. Our work can also be integrated with editing and browsing interfaces similar to SageBrush and SageBook [*Roth, 1994*] to help provide design assistance to users so that the creation and prototyping of visualization interfaces can be performed more quickly, easily, and with more effective results.

VI-2 Scope of Work

We describe the scope of our work based on the three components described above. For each component we present its limitations and point to possible directions for future work.

VI-2.1 Limitations of the Framework

The visualization technique framework presented in this thesis covers many current visualization techniques. Specifically, it focuses on those operators that can be applied to data concepts, graphical objects, and the relationships between graphics and data. There are however several distinct areas that are not addressed, namely:

1. Complex Mapping Transforms

The mapping transforms considered were limited to mapping data concepts to graphical objects and mapping data attributes to graphical properties. We need not, however, limit ourselves to only graphical objects. In the *Worlds within Worlds* system [*Feiner, 1990*], for example, three-dimensional charts can be mapped within other three-dimensional charts. In this case data is not only mapped to the graphical objects within each chart region but also to attributes of the chart region itself. Through multiple embeddings the authors of the *World within Worlds* system were able to analyze a large multidimensional space.

2. Specialized transformation functions

The framework also does not explore specialized transformation functions in detail. An example class of specialized functions are those used for animations such as fade in/outs. Other specialized transformations not dealt with are space distortion techniques such as those used in the Fisheye lens [*Furnas, 1991*], and the Hyperbolic space [*Lamping, 1995*]. These distortion techniques have been analyzed to some degree by Leung et.al. [*Leung, 1994*]. Even though the framework does not provide a characterization of these animation and distortion techniques, they can be integrated into the system as additional transformation operators or as black-box operators, if need be. However, more work still needs be performed within each of these specialized areas (e.g. animation, space distortion) to define the types of functions that are common and useful.

3. Windowing operators

The framework also does not deal with windowing operators such as popping up windows, raising or lowering windows, or changing the size of windows. Primarily, this is because such low-level operations should or are already captured within the specification of the virtual input devices. While designing visualization techniques, we should not have to concern ourselves with these low-level interface operations.

4. Workspace metaphors

The framework does not deal with the issue of designing a set of consistent visualization techniques that fit within a common workspace metaphor (e.g. the *Mac* user interface metaphor or the *Windows* metaphor). This as an important and significant area of study, but is beyond the scope of this thesis.

5. Conflict resolution

At the end of appendix B-6.2 we began to deal with conflicts that may occur between visualization techniques. However we only considered conflicts along five different dimensions and between pairs of primitive techniques. More conflicts will be revealed with a deeper study of this issue. In addition, our framework does not deal with conflict resolution.

6. Usage information

We do not consider collecting or applying usage information in our visualization system. Some example usage information includes which objects were selected most, which objects were last selected, which objects were commonly grouped together, etc. Such information may be very useful for providing good defaults to users, and may also be useful for informing the designer of work practices within a domain.

7. Scientific Visualization vs Information Visualization

The visualization technique framework presented in this thesis only deals with information visualization. Another large area of study is scientific visualization. Scientific data usually has a strong physical correspondence and contains very spatially oriented information. Information visualizations, however, represent abstract data that do not have a physical correspondence and are not inherently spatial. It is therefore not surprising that the requirements for these two areas can be quite different. There are several commercial frameworks available for describing scientific visualization techniques with limited interactions [*Brodlie, 1991*]. In appendix B-1 we compare our framework to these other existing frameworks.

8. Functions within functions

All of the object definition and transformation functions considered are applied to either data, graphical or annotation objects. We do not consider functions that can be applied to other functions or that generate new functions as output. Most common visualization techniques do not require such complex functional interactions. This class of operations, however, are interesting to consider and may produce very powerful visualization techniques.

VI-2.2 Limitations of the Metrics & Heuristics

In our work we identified two areas where the functional expansion enabled by our work can have the most impact over previous systems:

1. By offloading task cognitive operations onto the computer system with data transform techniques in addition to offloading them onto the user's perceptual system using mapping transform techniques.
2. By considering readability issues and examining when and how the four visualization transformation classes can be used to solve these issues.

We therefore only consider design rules and heuristics for these two areas. Other interesting design areas that we do not consider include:

1. Design heuristics that use graphical and rendering transforms to solve tasks rather than just readability problems

In addition to solving readability issues, graphical and rendering transforms can also be used manipulate the elements within a visualization to change the general goals addressed by that visualization (i.e. to change what is expressed by that visualzation). Such techniques however tend to be difficult to use, and specialized to the task domain. Therefore, to integrate such design techniques into an automatic design system requires that we have a powerful model for understanding user knowledge and expertise as well as capturing specific domain knowledge. The system would also need to provide better instructions to coach users on how the techniques should be used. Because of these complexities we save consideration of these issues for future work. In our work we focus on considering how graphical and rendering transforms can be used to only solve readability problems.

2. Design heuristics for three and four dimensional visualizations

We mainly focus our work on generating two-dimensional visualizations. Our heuristics and framework easily carries over to higher dimensional visualizations. However, if we are to effectively design such visualizations we need additional sets of design guidelines that specifically deal with the issue of when it is more expeditious to map data to the third positional dimension or the fourth time dimension, rather than using aligned spaces or a *retinal* attribute.

3. Design heuristics that minimize learning time rather than task performance time

In our work we focus on developing heuristics that reduce total task performance time. As is shown in appendix E, the design rankings generated based on our design guidelines conform in most part to the GOMS estimated total task performance time, excluding learning time. To limit our problem space to a reasonable size, we assume that the users of our system are expert users who are familiar with all the visual and interactive design classes generated by our system. Thus learning time for each design is consistent and negligible. In the future it would be interesting and useful to expand our set of heuristics to include rules that take learning time into account as well as task performance time. For example we may want to give a better ranking to designs that have a consistent look or interactive metaphor to a previous design because then the learning time for that design would be much smaller than a totally new and different design. We may also want to give preference to more conventional, well understood designs, because users would already know how to interpret and use them.

4. *Design heuristics for advanced tasks involving data patterns and trends*

Appendix C-3 describes two task classes: simple tasks that involve processing pairs of values, or more complex tasks that involve sensing gestalt patterns from a set of values, e.g. looking for data clusters, patterns or trends. The heuristics developed in this work is more focussed on simple value pair processing tasks, e.g. *lookup, find, AND, compute* and *compare*. This level of tasks is what was dealt with in previous automatic visualization research and we decided to focus on the same task classes. The focus of our work is on expanding the design space to include all the functions in the visualization creation process. Since we are taking the first steps in exploring the use of several new transform classes in automatic design we decided that it would better serve us to start with just the simpler tasks. Developing heuristics for the more complex trend tasks however would be a very challenging and interesting problem for the future. Note that heuristics for the complex tasks have no impact on our current heuristics for our simpler value pair processing tasks.

VI-2.3 Limitations of the System Implementation

With AVID we have expanded the design space of automatic visualization design systems. Previously, only mapping transformations were considered in the design process, but AVID is able to reason about many data manipulation and summarization operators as well as composite hybrid designs that use both mapping and data transformations to solve tasks. While we have expanded the functional design space from previous work, we can augment AVID further in the following areas:

1. *Integrating graphical and rendering techniques*

An important area that AVID currently does not address is in integrating *graphical* and *rendering* transformation decisions into the design process. Currently we only consider the use of data and mapping functions to solve tasks. We show in appendix F that *graphical* and *rendering* transformations can be very useful for solving *readability* issues that may arise in visualization designs and how our current system can be augmented with these graphical and rendering transforms. *Readability* refers to problems arising from constraints of the output medium and its interactions with our perceptual system that impede the optimal use of a visual design. Examples of *readability* impediments include occlusion among objects, display space that is too limited to show all the necessary design objects, or overly high ink density, producing visual interference.

2. *Translating high level tasks to lower level task operators*

We also do not deal with the issue of how higher level domain systems can translate their tasks into the task language required by AVID. Some discussion of this issue can be found in related research by Kerpedjiev et. al.[*Kerpedjiev, 1997*].

3. Error checking

We do very little error checking in our systems. Specifically our system does not have mechanisms for checking the syntax of the input task specifications. Error checking and reporting commonly get very involved and bring their very own set of research challenges that is beyond the scope of our work.

4. Limited range of visualization representations

Our system can generate common visualization representations such as *charts, maps, tables,* and *grids* as well as *marks, bars, lines* and *text*. Other common representation types that would be interesting to add in future work include *networks, pie charts* and richer glyphs that can encode many different data attributes simultaneously. Addition of specialized glyphs may require new heuristics and constraints to be added that are specific to these new representation types.

5. Limited range of navigation techniques

Currently, each of our visualization designs support some point of view navigation techniques including *zoom, pan* and *rotate.* In the future it would be interesting to integrate our automatic design system with a richer front-end data navigation environment such as the Visage system. Related to this issue it would be interesting to explore which visualization functions to include as default to all generated visualization designs and which functions to include on a case by case bases.

6. Interface for specifying visualization techniques

In the implementation chapter of this thesis (chapter V) we presented an interface for manually building visualization techniques by constructing data flow type diagrams similar to the specifications shown in chapters II and III. To bring visualization technique construction to more mainstream use however, a simpler, more intuitive interface will have to be designed that enables non-expert users to easily access the functionality provided by our visualization technique framework.