Chapter I: Introduction

Automatic Visualization Design

Automatic visualization systems have two primary goals: 1) to improve communication between the computer system and users both in terms of effectiveness and breadth and 2) to serve as a design assistant and help facilitate the creation of graphics for information presentation and analysis. To support the more complex and heavy demands that are made of information analysis systems today it is necessary to expand the effectiveness and flexibility with which computers can communicate with users. The range of tasks, data, media types and user preferences that these information systems must accommodate make it unfeasible to anticipate every possible output scenario. I.e. it is impossible to custom design graphics and interfaces here because there are too many possible output alternatives. Automatic visualization systems enable the flexible generation of information presentation graphics that are crafted on a case by case basis to suit the wide range of communication goals that may arise. Design rules and theories of cognition and perception are applied within these automatic systems to ensure the effectiveness and correctness of the graphics generated. Automatic visualization systems can also help users create and design graphic presentations. The SageBrush and SageBook interfaces [*Roth*, *1994*] show that automatic visualization research can be applied to help users complete partial designs or browse and adapt existing designs to show new data. This helps users with the more straightforward design operations, leaving them free to quickly explore many more design alternatives. It is important to stress that work in automatic visualization design is not meant to remove the "human" aspect from design and neither is its goal to design a "better" graphic than human designers. Instead, the power of automatic visualization systems is derived from the cooperative process between user and automatic system. The advantage of such systems as a design tool is that it can quickly generate a large range of possible design solutions and show them to the user who can then decide between similarly effective designs based on their preferences. In this way an automatic design system can aid a designer or less experienced user in creating graphics by performing the more mundane and simple visualization design tasks as well as give design suggestions that are based on compiled knowledge from graphic design and perceptual theory. Ultimately, the synergy between the user and the expert design system will be able to generate good designs more effectively and easily.

Previous research in automatic visualization design [*Mackinlay, 1986a, 1986b; Casner 1991; Roth 1994*] focussed on developing rules for mapping data to graphical elements effectively so that the generated designs support the desired user task(s) and can be clearly and correctly interpreted. However mapping data to graphics is only one step in the visualization creation process. Before we map data to graphical representations it is commonly effective to first process the data either by summarizing, computing, or culling out less relevant elements. By first massaging the data to a more appropriate form before presentation, we can construct more effective graphical designs for expressing user goals. This thesis expands the automatic design process to include these pre-processing data operations. This expansion allows a significant improvement over previous automatic design systems because it enables us to:

1. Generate more effective designs.
2. Address a larger range of tasks.
3. Produce a larger range of interesting design alternatives.
4. Usefully integrate input-devices into the design process.

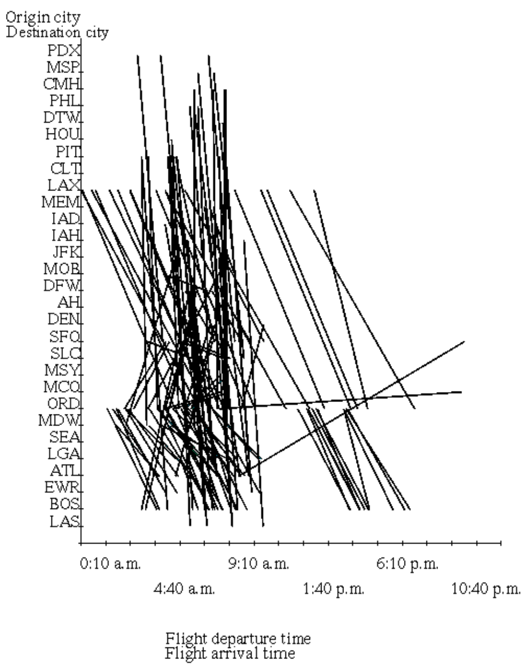
1. <u>Generate more effective designs</u>

The advantages of this work can be quickly seen in the following airline reservation task that was used in a previous automatic visualization system [*Casner*, *1991*]. In this task the user is interested in finding flights from *Los Angeles* to *Boston* with a layover in *Chicago* where they will be a meeting from *2 p.m.* to *4 p.m.*. The user would like to schedule the flights such that the total downtime in *Chicago* before and after the meeting is relatively small.

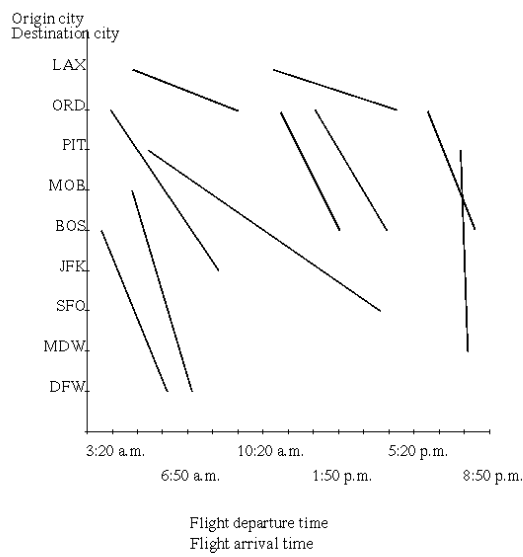
Flight number	Origin city	Destination city	Flight arrival time	Flight departure time
United_652	ORD	EWR	5:32 a.m.	2:30 a.m.
NW_826	MEM	ATL	10:13 a.m.	8:00 a.m.
Trans_235	MDW	LGA	9:00 a.m.	5:55 a.m.
United_244	LAX	ORD	3:37 p.m.	9:45 a.m.
United_342	ORD	MCO	12:19 a.m.	8:55 a.m.
American_494	ORD	EWR	7:48 a.m.	4:30 a.m.
American_491	BOS	DFW	7:29 a.m.	4:15 a.m.
		• • •		
United_7282	JFK	BOS	8:31 a.m.	7:00 a.m.
NW_5875	MEM	MOB	9:45 a.m.	8:20 a.m.
Delta_725	DFW	SFO	10:54 a.m.	9:00 a.m.
TWA_7762	PHL	JFK	7:53 a.m.	7:04 a.m.
American_346	ORD	LGA	10:04 a.m.	7:00 a.m.
American_340	ORD	LGA	11:36 a.m.	8:30 a.m.
Continental_1943	AH	LAS	9:46 a.m.	8:39 a.m.

Figure I-1: Cognitive design for the airline-scheduling task
(Note that the flights are not all shown here because the table is very large)

Performing this task cognitively using the raw data arranged in a table format (Figure I-1) would take up to approximately 4 minutes (based on a GOMS evaluation of the visualization). Previous automatic systems explored how this cognitive design can be appropriately mapped to graphics (Figure I-2), which lowers the total task time to only 30 seconds (assuming no occlusion). In our work we allow automatic preprocessing of data before presentation and the design generated by our system allows the task to be quickly solved in 3 seconds (Figure I-3). A summary of the GOMS estimated time for all three designs is shown in Figure I-4. We briefly describe the perceptual and cognitive steps for these displays in section I-2. The detailed GOMS sequences for these three visualizations can also be found in appendix C-1.



(a) Full data set
This visualization shows all the elements in the data set (i.e. all 135 flights).



(b) Truncated data set.
This example visualization shows the ideal case where there is little occlusion among the different flight lines. This data set was chosen so that it contains some flights that fulfill the task constraints as well as some other random flights that do not occlude one another.

Figure I-2: Perceptual design for the airline-scheduling task
Each line represents a flight with origin and destination city mapped onto the *y-axis* and arrival and departure time mapped onto the *x-axis*. This is the best design that gets generated when ONLY mapping operations are considered by the automatic system. I.e. this is the best possible design from current state of the art systems.

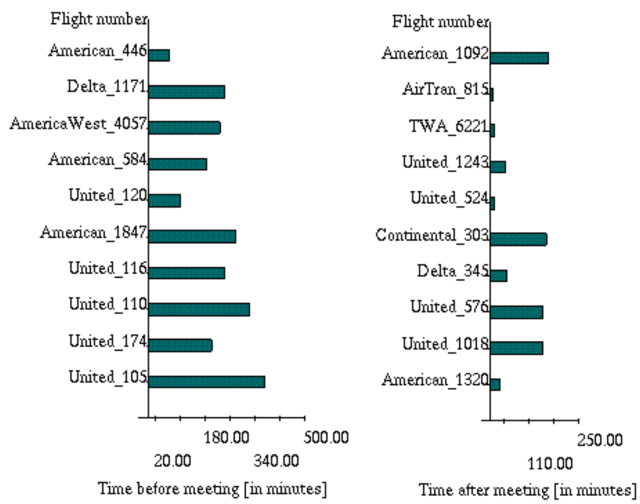


Figure I-3: Design generated when data processing operations are integrated into the automatic visualization system. The full data set is considered here but data transforms are applied by the automatic system to filter the data set so that only relevant flights are shown. The total downtime before the meeting for the flights from LAX to ORD is shown on the left chart and the total downtime after the meeting for the flights from ORD to BOS is shown on the right chart.

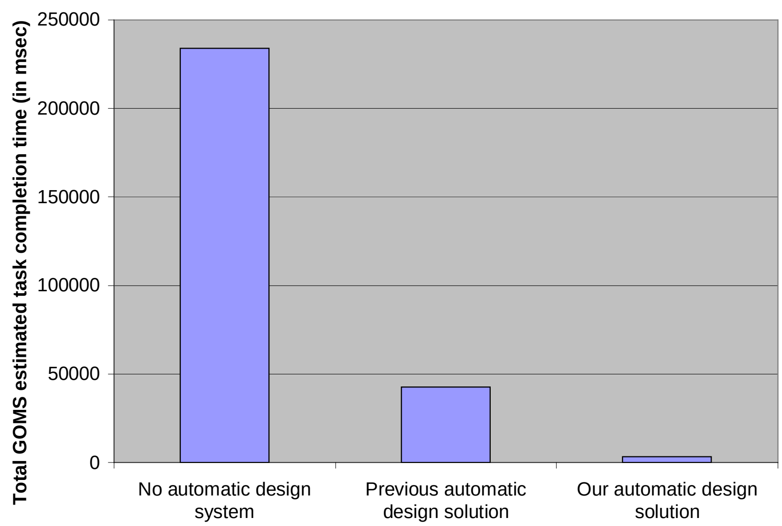


Figure I-4: GOMS estimated total time for solving the airline-scheduling task for a data set of 135 flights. Detailed GOMS sequences for each design are presented in appendix C-1.

2. <u>*Address a larger range of tasks*</u>

Adding data processing functions into the automatic design process also allows us to address a larger range of tasks than was possible with previous systems. Some tasks do not have a purely perceptual solution, including processing of large data sets, abstract mathematical operations (e.g. *log*, *exp*) or complex calculations that contain multiple related mathematical operations (e.g. *(a+b) / (c+d)*). In this thesis we enable automatic design systems to deal with these problems by automatically computing the tasks fully or partially so that users are presented with combined computation (data transforms) and perceptual (mapping transforms) solutions. Note that while it is possible to always compute a task fully before conveying it to the automatic designer it is also undesirable and restrictive to do so. As we will show in chapter IV, pre-computing entire tasks may severely constrain the flexibility of users and may not produce the most effective design solution. To generate "good" design solutions we must integrate data processing with mapping decisions because design decisions made in the data processing phase affects the mapping phase and vice versa. And just as we cannot anticipate all combinations of data and information analysis tasks that may be demanded by users, we cannot anticipate all combinations of data processing and mapping operations that are appropriate and useful in our designs.

3. Produce a larger range of interesting alternative designs

Expanding automatic design to include data processing operations also allows us to generate a larger range of interesting alternative designs compared to previous systems. Designs generated may be purely perceptual, purely computed, or hybrid computed and perceptual designs. This larger range of choices is important because the data analysis process is an iterative process where users first construct mental models of their current tasks and based on these mental models, pose design requests to an automatic system. Depending on the results of the design request, users may then update their mental models and then repeat the process. The ability of users to arrive at useful answer(s) to their data analysis problems depends on the range and quality of design solutions returned by the automatic system as this will facilitate the next iterative cycle. The wider range of design alternatives provided by our system enable users to better match design solutions to their data analysis goals as well as personal preferences.

4. *Usefully integrate the application of input-devices into the design process*

Input-devices are very effective for allowing end-users to flexibly change a visualization design interactively. This allows large data spaces to be represented because we can interactively focus in on different subsets of data elements at different times. Previous automatic systems did not consider the use of input-devices. This is primarily because these systems always showed all of the available data (i.e. there was no data summarization, computation, or culling). As a result there was no need for users to navigate through the visual representation by using input-devices. Unfortunately, the lack of interactivity and data summarization operations also constrained these systems so that they can only address problems with relatively small data sets (< 20 elements). This was clearly illustrated in Figure I-2b where showing the entire data set of 135 flights produced a visual display that was too occluded to be of any use. In this thesis we add input-devices and data transforms into the automatic design process so that our system can deal with larger data sets (> 100 elements) through data culling, summarization, and interactivity.

To integrate data processing functions into automatic design we develop three core technologies in this thesis:

1. *Methods*: We develop a way to characterize data and mapping functions within a visualization design, how these methods may be combined with each other, with the output media and with available input-devices.
2. *Principles*: We also develop a set of design dimensions and strategies that can help us gauge the quality of different design alternatives. These dimensions and strategies determine when and how data and mapping methods should be used based on user tasks, data, and preferences.
3. *Systems*: We show that the methods and principles developed are complete and applicable by using them to implement an automatic visualization design system. We then evaluate the system through a series of GOMS analyses to show that the results generated by our prototype designer are correct (i.e. they support the input tasks) and are ordered based on cognitive, perceptual, and motoric complexity. I.e., the most effective or least complex design is generated first and the most complex design is generated last.

I-1 Methods: Visualization Techniques Framework

In this thesis we develop a framework that characterizes the function and structure of visualization techniques that are used to create and modify visual designs. This framework provides our automatic design system with the necessary constructs to build visualization interfaces that may contain data processing functions, mapping functions, as well as input-devices.

Each visualization technique within our framework is defined to have a selection component and a transformation component. Selection can be achieved through enumeration or through a constraint function (functional description). Transformation can be achieved using the four different functions within the visualization creation process shown in Figure I-5. To build richer visualization techniques, we can combine multiple techniques together through a set of composition functions.

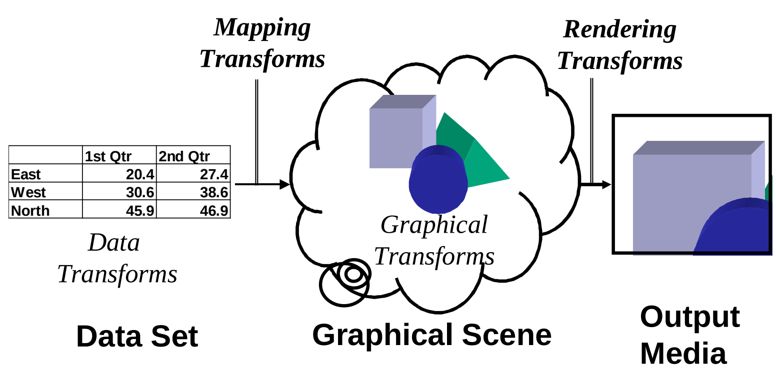


Figure I-5: The four phases of the visualization creation process

The visualization creation process consists of four primary phases: data, mapping, graphical, and rendering. Initially in the *data* phase the task data is processed and a portion of it is chosen for subsequent mapping. In the *mapping* phase the chosen data elements from the previous phase are mapped onto *graphical properties* (e.g. *color, position, shape*) and *graphical objects* (e.g. *marks, bars, lines*). In the *graphical* phase, the graphical scene constructed from data mapping is further processed to accommodate changes that may not be reflected in the data set. For example objects in dense areas may be made smaller to avoid occlusion. Finally in the *rendering* phase the graphical scene is transferred onto an output media. There are currently many different media types available (e.g. PalmPilot™, CRT screens, image projection screens) with different constraints on visualization size, number of colors, resolution, mobility, etc. All these constraints affect the way with which the visualization design may be displayed and explored.

Previous automatic design systems only considered the use of mapping functions. In this thesis we expand automatic visualization design to include both data and mapping functions. Even though our automatic design system only uses functions from the data and mapping phases we decided to lay out all four transformation classes in Figure I-5 in our framework because it helps us better understand the roles that data and mapping functions can play in the design process, it allows graphical and rendering functions to be easily integrated into the automatic design process in the future, and it increases the applicability of our framework, allowing us to categorize and analyze current visualization systems and techniques.

Below we show how a very simple visualization technique can be specified based on the primitives and composition rules in our framework. The technique is a simple highlighting technique that allows users to select a set of objects using a *bounding-box* and then subsequently highlights the selected objects *red.*

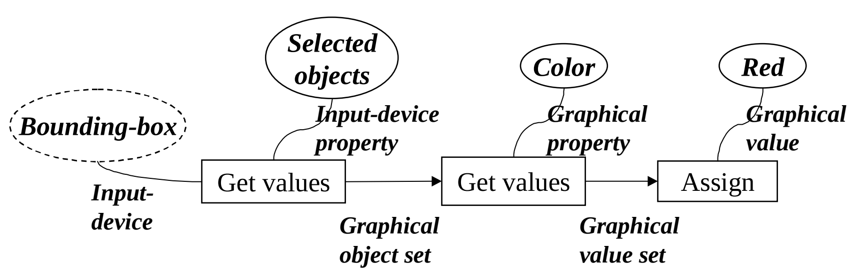


Figure I-6: Highlight technique specification. The input-device bounding-box is used to select a set of objects. These objects (*selected-objects*) are extracted from the bounding-box device using the *get-values* function. We then get the *color* values from all of the selected objects using a subsequent *get-values* function. Finally we change all of the *color* graphical values to *red* using the *assign* function.

All function primitives are shown with normal *Times-Roman* font within rectangles and all inputs to the primitive functions are shown as ***italicized bold*** text within ovals. Inputs provided by users are shown with dotted ovals and those provided as designer defaults are shown with regular unbroken ovals. The directed arrows (→) connecting one primitive function to another indicate a flow of objects or values from a source function to a destination function. The ***italicized bold*** labels next to the connecting links indicate the types of objects or values that are being passed through that link.

Aside from being a crucial component to our automatic design system, this framework also provides multiple other contributions to the field:

1. *Prototype and Tailor Visualization Interfaces*

Our framework helps designers prototype and tailor visualization interfaces. For example, a visualization designer can very easily adjust the technique in Figure I-6 so that it highlights objects *blue* instead of *red* by changing the oval marked *red* to *blue*. Alternatively we can let end-users specify the highlight color through an input-device by replacing the oval marked *red* with an input-device. The high-level visualization techniques description language provided by our framework enables designers to create and adjust visualization techniques without resorting to writing code.

We show more examples of our visualization techniques language and how they can be created and varied in chapters II and III. For now however, it is important to stress that this visualization language is not meant for end-use. A user-friendly interface should be built on top of the specification language before it can be readily accessed by end users. For example the SAGE system [*Roth, 1994*] has an underlying language for describing the data and graphical elements within a visualization as well as the mapping relationships among them. However it is also attached to a graphical user interface, SageBrush, that provides end users with simple drag and drop techniques for utilizing this language. The same situation applies for the data, mapping, graphical, and rendering functions considered in this work. A friendlier interface is needed for end-users but this interface must be based on an underlying language that captures the functionality and structure of visualization techniques. Our framework can serve as this basis.

2. *New Two-level Design Methodology*

Our framework also presents a new design methodology for creating visualization techniques. This methodology divides the design process into two different levels of abstraction: a *functional* level, and an *instantiation* level. At the *functional* level designers focus on providing users with the proper operations to serve their current goals. I.e. focus is on choosing appropriate functions from each of the four visualization phases and combining these functions. The *instantiation* level, on the other hand, is more concerned with the general appearance and usage of the visualization technique. At this level focus is on choosing appropriate devices for input entry, choosing effective graphical attributes for visual feedback, as well as general layout of the visualization interface. This division helps designers separate the two different aspects of visualization techniques, *function* and *form*, so as to decrease the likelihood of falsely constraining functionality based on appearance concerns. This two level methodology is an advance over previous work that only considers either the *functional* [*Tweedie, 1997; Card, 1999*] or *instantiation* [*Brodlie, 1991*] levels in isolation.

3. <u>Exploration of the Visualization Techniques Design Space</u>

Finally the framework helps to scope out a large part of the visualization techniques design space, and allows for more systematic exploration within that space. We show at the end of chapter II the space of current visualization techniques and how they may be combined to form new methods. A description of the current space of techniques is important because it shows us the areas we have explored and points to new and future areas of exploration. For example we found that most visualization techniques that are used to search for data objects utilize simple feedback methods to show their results. Feedback for these techniques usually involve changing a single graphical property (e.g. *color*) to different constants (e.g. *red*). Thus one new area of exploration could be in developing useful object search techniques with richer feedback methods that change multiple graphical properties simultaneously in meaningful ways.

I-2 Principles: Design Dimensions and Strategies for Measuring the Goodness of Visualization Designs

The visualization techniques framework described in the previous section provides an automatic design system with the proper language for expressing a wide variety of visualization designs. However there are many possible different alternative designs that fulfill a particular data analysis task. For example, the airline-scheduling task presented earlier can be solved using any of the three alternative designs shown in Figure I-1, Figure I-2 and Figure I-3. Thus in addition to a visualization techniques language, an automatic system must also be equipped with design rules and strategies that help guide it down more promising design paths and prevent it from generating ineffective designs. For this purpose we develop a set of design dimensions for measuring the goodness of different design alternatives as well as a set of design strategies that help our system first generate designs that are deemed more effective based on our design dimensions.

Our design dimensions are built upon previous work by Abowd and Beale for measuring the effectiveness of user interfaces. This framework calculates the overall "goodness" of a visualization design or its "*semantic distance*" by using four distances: *articulatory distance, functional distance, expressive distance,* and *observational distance*. *Semantic distance* refers to the degree with which user goals are fulfilled by the visualization. A large semantic distance means that the goals are not achieved well and a small semantic distance means that the goals have been satisfied acceptably. *Articulatory distance* measures the amount of input-device manipulation required from users. *Functional distance* refers to whether the system possesses software functions or procedures capable of achieving user tasks. *Expressive* *distance* determines whether sufficient feedback or information is provided to users to solve the input tasks. Finally, *observational distance* refers to the ease with which a user can interpret system feedback. Specifically, *observational* distance measures the effectiveness of the visual objects, visual properties, and visual compositions used to fulfill the input analysis tasks. Based on these dimensions we develop a set of design strategies that help minimize the *overall semantic distance* of a visualization.

The data processing operations added by this work can improve the *semantic distance* of a visualization design by offloading difficult cognitive operations onto the computer system in addition to offloading them onto the user's perceptual system with mapping transforms as was done previously. For example consider the airline-scheduling task presented at the start of this chapter. The pure mapping design (Figure I-2) encodes each flight with a *line* graphical object. The *origin* and *destination* cities are mapped to the *y-axis* and the *arrival* and *departure* times are mapped to the *x-axis*. To perform the task, users must first search for all lines that originate from *Los Angeles* (LAX) and flies to *Chicago* (ORD) as well as originate from *Chicago* (ORD) and flies to *Boston* (BOS). Next the set must also be narrowed down to only those flights that arrive before the *2 p.m.* meeting time in *Chicago* and leaves after *4 p.m.* (i.e. end-point of LAX-ORD flight is to the left of *2 p.m.* on the *x-axis* and starting-point of ORD-BOS flight is to the right of *4 p.m.*). In contrast, Figure I-3 uses data transforms to offload these cognitive search tasks to the computer system instead of to the perceptual system. Specifically, the computer system performs the *city* and *time* search and only presents those flights that fulfill both the city and time constraints in the task. As a result the design is less cluttered and easier to interpret compared to the pure mapping design (Figure I-2).

However, a significant portion of the information from the original data set has been filtered out in Figure I-3 so that if we changed our meeting time or our meeting venue the data transform design would no longer be usable and we would have to generate a new visualization. In contrast the pure mapping design (Figure I-2) is more flexible and can better accommodate changes in user goals (i.e. the mapping design can still be used to solve the modified scheduling task). Thus depending on the demands of current tasks, an automatic design system may choose to apply different blends of data and mapping transforms. In chapter III we explore these issues and develop design strategies that can help our automatic system decide when it is more appropriate to use data transforms to offload a task onto the computer system and when it is more appropriate to use mapping transforms to offload a task onto the user's perceptual system.

In appendix F we outline how graphical and rendering functions can also be integrated into automatic design in the future. Specifically, graphical and rendering functions improve the semantic distance of a visualization design by addressing readability issues. By readability we mean problems arising from constraints of the output media and its interactions with our perceptual system that impede the optimal use of a visual design (e.g. *object occlusion*). It is crucial to address these readability issues because they may cause an otherwise valid design to become unusable because of extreme clutter, or overly small graphical representations. In appendix F we identify four important readability issues: *occlusion, density, dwarfing* and *information proximity* and discuss how these issues can be addressed through the use of graphical and rendering functions. Readability problems can sometimes also be avoided through judicious use of data transform functions or by mapping the data to a larger graphical representation. We discuss some of these data and mapping readability enhancements in chapter IV. Previous automatic systems did not consider readability issues because it is difficult to address these issues with only mapping transform functions..

I-3 Systems: AVID – Automatic Visualization Interface Designer

Finally to show that our theoretical concepts are sound, practical, and sufficiently complete, we implement an automatic design system, called AVID, based on our framework as well as our design dimensions and strategies. AVID accepts a task specification like the one shown in Figure I-7 as input. Based on this task specification it will generate a series of design alternatives ranked based on their effectiveness with respect to the input task(s).

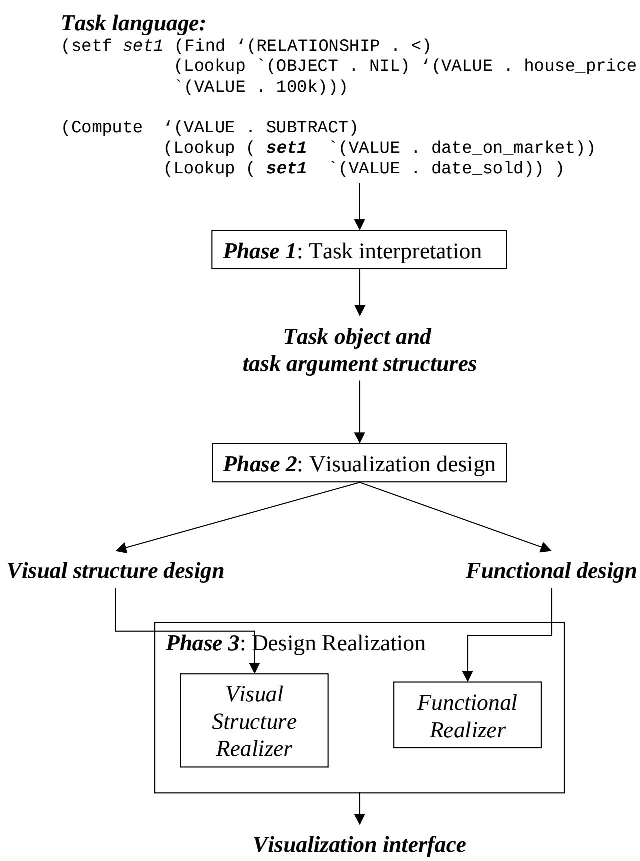


Figure I-7: Three components within AVID

AVID, consists of three components corresponding to the three stages of the automatic design process (Figure I-7):

1. <u>The task interpretation phase</u>

Initially, a higher level agent (user or a domain specific system) that has a deeper understanding of the problem domain generates a set of tasks for AVID. Tasks are expressed using a simple language based on the EDA (Exploratory Data Analysis) task model first developed by Tukey [*Tuket, 1977*] and later refined by Casner [*Casner, 1991*] for automatic design. This language is relatively low-level and its purpose is to capture important components of a task that may affect the visual design process. We do not expect typical end users to specify tasks in this language; rather, specifications will most likely be generated by domain specific systems that use graphics to present and summarize their results to users, such as automatic planning systems, automatic information analysis systems, agent based information gatherers, etc. The task interpreter within AVID evaluates the input task language and generates a set of task objects and argument structures.

2. <u>The design phase</u>

AVID's design component parses the task objects and argument structures generated from the task interpretation phase and converts them to design constraints and cost preferences. These design constraints and cost preferences are generated based on the design dimensions and strategies we discussed in section I-2 (detailed descriptions are in chapter IV). Based on these constraints, AVID explores the design space for the input tasks and automatically generates a set of visualizations ordered from best to worst. These output designs are expressed in a language that captures the visual structure of a visualization interface as well as any underlying transform functions and active interactive components. Visual structure descriptions have been developed in previous work [*Mackinlay 1986a, 1986b; Roth, 1990*]. As was discussed in section I-1, this thesis develops a language for capturing the functions and active components within a visualization (detailed descriptions are in chapters II and III).

3. <u>The realization phase</u>

AVID's "realizer" component interprets design specifications generated by the design component and renders an active visualization interface. This component makes layout decisions and assigns default values to visual components that are left unspecified or unconstrained in the design specifications. Currently, AVID's realizer is capable of interpreting most of the visualization technique primitives described in this thesis (e.g. computations, set-operations, threshold operations, etc). By combining these primitives it can generate a wide range of interactive behaviors such as aggregation, painting, dynamic queries, simple semantic zoom, SDM graphical manipulation operations, navigation operations, etc.

In chapter V we provide details on how our visualization techniques framework as well as design dimensions and strategies are codified within our automatic design system, AVID, and how the design search space is explored. In appendix E we perform a series of GOMS evaluations on the designs generated by our automatic system to ensure that its design rankings conform to cognitive, perceptual, and motoric complexity.

I-4 Previous Work

This thesis builds upon a wide variety of previous work. In the following sections we divide work related to this thesis into two classes: 1) work that pertains to the visualization techniques framework, and 2) work that pertains to automatic visualization design (including design dimensions and strategies, as well as system implementations).

I-4.1 Visualization Techniques Framework

There are two classes of visualization frameworks: *functional* frameworks and *instantiation* frameworks. Some example *functional* frameworks include Tweedie's DIVA research [*Tweedie, 1997*] and Card et al's framework [*Card, 1997, 1999*]. These frameworks are high-level and are used to analyze and classify existing techniques based on task, data, functionality, etc. Instantiation frameworks such as Data Explorer, IRIS Explorer, and AVS [*Brodlie, 1991*], on the other hand, establishes a concrete language for describing visualization techniques. Instantiation languages are very detailed and describe visualization techniques completely. Because they are much lower-level compared to functional languages, they are also less appropriate for the analyses and classification of techniques. However, instantiation descriptions, unlike functional descriptions, are realizable or renderable (i.e. these descriptions can be easily translated into an active visualization interface). Our framework differs from all previous frameworks because it encapsulates both the functional and instantiation levels of descriptions. In appendix B-1 we compare our framework to previous work.

The design of our framework is based on previous work in visual specification languages [*Jacob, 1986*], user interface languages [*Foley, 1990*; *Card, 1990; Mackinlay, 1990*] and visualization frameworks [*Card, 1997, 1999*].

I-4.2 Automatic Visualization Design

There are hundreds of rules that graphic designers use to generate visualizations based on their intended task and the data they represent. In addition, there are also a large number of rules that can be derived from psychophysical literature and from user testing of visualization systems. In the next sections we present some background on automatic visualization systems and their internal heuristics which primarily consists of graphic design rules. In addition we will briefly describe some of the work performed in perceptual theory and visualization system testing that can also be used to support automatic visualization design.

I-4.2.1 Automatic Visualization Systems

Tufte [*Tufte, 1983*] and Bertin [*Bertin, 1983*] started the initial work in laying out a set of useful graphic design rules and in characterizing the structure of visualization displays. In his book, *A Semiology of Data Graphics*[*Bertin, 1983*], Bertin identified some of the most important issues in visualization design and exposed many of the important artifacts in their structure. Bertin's work was later refined by Mackinlay [*Mackinlay, 1986a, 1986b*] who developed a syntax for expressing the components of visualizations, effectiveness criteria to decide when and how to use the different graphical components and rules of composition that specify how and under what conditions graphical elements can be combined. Mackinlay then used the expressiveness and effectiveness criteria that he developed to implement a system called APT that could automatically design a well-defined set of visualizations. Casner [*Casner, 1991*] later continued Mackinlay's work by taking a task centered approach to creating visualizations. While Mackinlay previously generated visualization designs solely based on the structure of the input data, Casner now also considered user goals. Casner proposed decomposing a user task into a series of logical operators. These logical operators were then replaced by more efficient perceptual operators where possible, and visualizations were then created based on these perceptual operators. The SAGE system [*Roth, 1990, 1994*] carried this area farther by developing a richer data representation for visualizations thus allowing a wider and more complex set of abstract visualizations to be generated. The SAGE system also allowed users to direct the automatic design system by entering in design preferences in the form of partial designs or a previous favorite design.

Our work builds on these previous systems in two primary areas. First of all, these previous systems only considered the issue of how data can be effectively mapped to graphics. In our work, we additionally consider how the input data can be effectively processed before it is mapped and shown to users. Secondly, these previous systems only generated static, non-manipulable visualizations. Our work allows automatic design systems to generate interactive visualization interfaces so that users can navigate through the visual representations and explore larger data sets than was previously possible.

I-4.2.2 Psychophysical Studies

Most of the previous work done on developing effectiveness and expressiveness criteria for automatic design are based more on well established graphic design rules [*Tufte, 1983, Bertin, 1983*] rather than on perceptual theory. This is because it is difficult to abstract from the low-level results contained in perceptual literature and apply them to higher level perceptual operations that occur in visualization analyses. A large cause of this complexity is due to the presence of a wide variety of graphical styles and graphical properties that may be used. This makes it difficult to isolate the effects of each element and even more difficult to determine the conflicts and relationships between the different graphical artifacts. Studies of perceptual theory are further complicated by the external knowledge of graph reading that is assumed of the user.

Thus it is not feasible to define effectiveness and expressiveness criteria solely based on psychophysical results. Doing so will produce an incomplete model of design and limit the generality of the system. However that is not to say that psychophysical results cannot be used. Green [*Green*] showed various instances in which psychophysical literature can be used to support some of Bertin's design strategies and other instances in which the literature showed errors in those strategies. Senay and Ignatius [*Senay, 1994*] are also beginning to apply psychophysical results to design decisions. In this work we will also use perceptual theory [*Livingstone, 1988*; *Treisman, 1982, 1988*] to enrich our design strategies.

I-4.2.3 User Studies on Visualization Interfaces

There have been many user tests conducted to show the effectiveness of new interactions and visual techniques [*Hollands, 1989*; *Ahlberg, 1992; Plaisant, 1996*]. These tests are usually conducted over a small set of specific tasks and are used to illustrate the usefulness of newly introduced techniques and visual representations. While such experiments are effective for demonstrating the utility of new ideas, they are usually not broad or general enough for us to derive general design rules and strategies.

I-5 Summary

The main contribution of our work is in adding data processing decisions into the automatic visualization design process. This is in contrast to previous work in automatic design that only considered mapping transforms. Our work expands the quality and breadth of designs that may be generated and allows automatic systems to address a larger range of tasks as well as larger data sets. In addition, our work also expands automatic design systems so that they may now begin to generate interactive interfaces. All of this enhances human computer communication because a greater, improved, visual vocabulary allows richer and more complex concepts to be conveyed. In addition, the effectiveness of AVID as a design assistant is also increased because it is able to provide a larger range of "good" design alternatives and choices to users. Our design system also culls out bad designs (i.e. task inexpressive designs or designs that do not support the input task(s)) as well as duplicate designs. This saves users from having to devote attention to these less appropriate visual representations while still having good coverage of the design space.

In order to integrate data processing operations into automatic design, we developed three technologies: 1) a visualization techniques framework, 2) design dimensions and strategies for measuring the goodness of various visualization designs, and 3) an automatic design system (AVID) that is able to automatically design visualizations based on a set of user input goals. Our visualization techniques framework provides our design system with a set of primitives and composition rules from which it may build and design visualization techniques. In addition to being a crucial component in our automatic design work, our framework also stands as a contribution in its own right. First of all the framework simplifies the creation and prototyping of visualization techniques by providing designers with a higher level API set.

Secondly the framework provides a new design methodology that separates the design process into two levels, functional and instantiation, and through these two levels promotes better functional design of techniques. Finally our framework allows a designer to systematically explore the visualization techniques design space and identify design holes within that space.

In addition to the framework, we also developed a set of design dimensions and strategies that help our automatic design system pick the best or most effective design alternatives for the current task(s). These dimensions and strategies can also be applied by human designers as a quick evaluation of their designs and as yardsticks of comparison among multiple current designs. Finally we implement an automatic design system based on our framework and design strategies. This automatic system shows that our theories are sound and complete enough to be actualized.

Concurrent research is also underway for combining the graphics generated by automatic visualization systems with text [*Kerpedjiev, 1997*]. Research in this area, while related, does not deal with the same issues that are relevant to automatic visualization design. Rather, work in this area assumes the existence of automatic text and visualization generation systems, and focuses instead on how best to integrate these two communication media. Therefore, the advancements made by our research to automatic visualization design will naturally feed into the combined text and graphical work as well.

I-6 Walkthrough

This thesis is divided based on the three main technologies presented above. Chapters II and III describes our visualization techniques framework including primitives, composition rules and how new techniques may be created by combining previous methods. Chapter IV contains a set of design dimensions for measuring the effectiveness of visualization designs as well as a set of guidelines that discuss when it is appropriate to use data transforms and mapping transforms. Chapter V shows how we integrated our framework and design rules into an automatic design system. Chapter VI presents some concluding thoughts on the work, discusses its scope, and presents a summary on its impact and how it can be expanded in future work. The first 4 appendix sections (appendix A, B, C, and D) are organized to provide additional information and examples on the material in chapters II, II, IV, and V respectively. Appendix E presents a series of GOMS evaluations on the designs generated by our automatic system and shows that our design output does indeed conform to cognitive, perceptual and motoric complexity. Finally appendix F discusses how we anticipate readability issues can be addressed using graphical and rendering functions.