Appendix F

Enhancing Readability with Graphical & Rendering Transforms

The American Heritage Dictionary defines *readable* as “capable of being read easily”. This is a broad definition and can refer to any factor that affects the ease of interpreting a visualization (this corresponds to the *observational* distance of a visualization). In this thesis however, we use *readability* in a narrower sense to refer to

> ***problems arising from constraints of the output media and its interactions with our perceptual system that impede optimal use of a visual design***[1].

F-1 Readability Problems

Different readability problems arise due to our choice of output media. In our work, we are interested in the CRT screen, which has two main restrictions, limited space and limited resolution. Readability problems are also affected by limitations of our perceptual system when interacting with the output media. Three primary limitations of our perceptual system include: 1) Single point of focus (we can only focus on one spatial area at a time), 2) Limited area of focus (our eye is only sensitive to a limited amount of space and is unable to capture visual elements beyond this area of sensitivity), and 3) Limited resolution (our eye can only pick up objects or features at a certain minimum resolution).Based on these limitations we identify four commonly occurring readability problems:

F-1.1 Occlusion

Occlusion occurs when one or more graphical objects in the display visually hinder access to other objects that are important to our task. Occlusion is one of the most commonly recognized readability problem and has been dealt with in many spatial layout algorithms for a variety of visual structures. This is because occlusion problems arise very frequently (especially with the large data sets that we have to deal with today) and cannot be easily avoided. We identify three principal occlusion classes: *line-of-sight occlusion, overlap occlusion,* and *overplotting.*

1. *Line of sight occlusion*: Line of sight occlusion occurs when we try to view a three dimensional scene or a three-dimensional glyph. It is difficult for us to access objects within a three dimensional scene when there are other objects are in front of them in our line of sight because we only have a single point of focus and we cannot see through opaque objects. Line of sight occlusion may occur as a result of: a) self-occlusion or b) inter-object occlusion.

[1] By “*optimal use of a visual design*” we mean the most effective perceptual, cognitive, and articulatory strategy that may be used with a visual design given a non-problematic or “good” data distribution and data set size.

2. *Overlap occlusion*: Overlap occlusion may occur in both three-dimensional and two-dimensional spaces. Unlike line of sight occlusion which occurs for objects along the same line of sight, overlap occlusion occur for objects that are very close to each other spatially. Objects may be partially or fully occluded. It is obvious when and where partial occlusion occurs but for fully occluded objects, we must supply strong cues indicating that there is hidden information. For example, we may provide users with an occlusion map overlay where color in the map represents the amount of occlusion at a spatial region. Another possibility is to move the occluded objects to the front and back automatically. Note that because overlap occlusion is not caused by viewing location but rather by object positions, it cannot be solved by changing the scene viewpoint as was done for *line of sight* occlusion.

3. *Overplotting*: We use overplotting here to refer to the extreme case of overlap occlusion where two objects occupy the exact same position[2]. Unlike overlap occlusion, overplotting cannot be addressed with spatial distortion techniques. Stretching the space between objects (i.e. allocating more space between objects) will not help in this case because the objects are on the exact same location. Thus, the only techniques that can be used in this situation are the object based graphical transform techniques.

F-1.2 Density

When designing visualization displays it is not only important that we avoid occlusion, but also that we reduce visual clutter, i.e. high ink density. High ink density can be distracting to users making it difficult to identify graphical patterns and trends. In general, visual elements should only be introduced if they help facilitate the task so that display density is not unnecessarily increased.

Display density was very carefully studied by Tufte in his book "*The Visual Display of Quantitative Information*" [*Tufte, 1983*]. Tufte presented the idea of *data ink*, which represents the non-erasable core of a graphic. Ink that did not express information required by the task (i.e. *chartjunk*) was wasted ink and only served to clutter up the display. Some example artifacts that commonly result in chartjunk include unintentional optical art (e.g. introducing hash marks and textures that are not needed), overly gridded displays, and art that may beautify the display but does not add to its effectiveness or information content.

Note, however, that removing chartjunk **does not** mean that we should only show the bare minimum amount of data required by the task. If a visual element increases the overall effectiveness of the visual design with respect to the current task(s) then it is **not** chartjunk. Well-rendered grids for example can help direct a user's eye to the positional axes so that value lookups can be performed more accurately. In addition, it may sometimes be advantageous to emboss objects with texture to make them more salient so that we can direct a user's attention to chosen objects in the display. Depending on the task, it may also be

[2] Note that we can reduce the amount of overplotting by using better layout and rendering schemes such as drawing the larger objects behind the smaller objects so that both are visible to the user. However, in those cases where good layout is insufficient to solve the problem, graphical and rendering transforms can be used as well.

effective to multiply encode data attributes (i.e. represent the same data attributes using different graphical properties). Different graphical properties and graphical representations are effective for different data types and tasks. For example, suppose we want to search for houses worth more than 100k as well as be able to accurately look up those sales figures once we find them. For the search task, it is much more effective to encode the values using position because then we can pre-attentively find the houses costing more than 100k. However, text allows for more accurate lookups. By multiply encoding the house *selling-price* attribute as text and bar length, we enable users to perform both the search and lookup tasks effectively.

F-1.3 Dwarfing

Dwarfed encoding scales arise when we have a small range of graphical values representing a large range of data values. As a result different data values may be mapped to similar or non-distinguishable graphical values. Because our perceptual system can only differentiate between graphical values at a certain minimum level of resolution, dwarfed scales often produce highly inaccurate information.

F-1.4 Spatial separation

Our eyes are only sensitive to a limited spatial area. When related data concepts are placed in spatially distant locations, we must not only perform more eye, head, or body movement, but also store information in short term memory from one eye fixation to the next. Thus while it is important to reduce occlusion and display density by separating out the visualization objects, we must balance that with the additional processing required for finding related objects over large spatially distinct locations.

Spatial separation problems arise because of representation constraints (e.g. a bar in a bar chart can only have two other bars next to it), spatial constraints (there is limited adjacent space around a two-dimensional object), and density constraints (areas that are too dense or cluttered are difficult to interpret) that force graphical objects to be spread farther apart.

F-2 Readability Solutions

Readability problems may be addressed in three ways:

1. *Changing the data to graphical mappings within the visualization*: Readability problems can often be avoided by changing the graphical properties and objects used to show the data. However, we must be careful that these changes do not reduce our ability to solve our target tasks (i.e. force a less effective perceptual strategy to be used) or lead to other readability problems.

2. *Changing the balance between data transforms and mapping transforms*: We can also solve readability problems by using data transforms to summarize our data so that fewer data values need to be shown.

Although this method produces visual displays that are less cluttered due to object filtering, the articulatory cost tends to be higher due to the need for user input in directing the filtering process.

3. *Using graphical and rendering transform methods*: Finally, we may also use graphical and rendering transforms to change the appearance of objects in a visualization to solve readability problems in the design. For example, we can allow users to interactively change the transparency of the occluding objects with graphical transform methods. Another possibility is to stretch the display with rendering transforms so that there is more white-space between the objects and less likelihood that they will occlude one another.

Which of the three methods described above is most appropriate for addressing current readability issues depends on how they change the overall *semantic distance*[3] of our visual designs with respect to our task(s). Solution 1 may increase observational distance because less effective data to graphical mappings may need to be used to avoid readability problems. Solution 2 generally increases articulatory load because users need to specify more task parameters when using a highly computed visual design. In addition, expressive distance may also increase because data transform operations often summarize or cull out portions of the source data. Finally, solution 3 adds to articulatory and observational distances because users must interact with the graphical and rendering transform methods as well as interpret their feedback. Nevertheless, solution 3 also allows for more flexibility in picking effective data to graphical mappings.

In chapter IV we considered how readability problems can be avoided by making good mapping and data transform choices (i.e. corresponds to solution 1 and solution 2 above). In this section, we focus on solution 3 and consider how a subset of the graphical and rendering functions described in chapters II and III can be used to solve readability problems. Specifically we consider five basic visualization functions: three graphical and two rendering. These methods were chosen because of their simplicity, and their ability to generate interesting and useful behaviors for addressing readability issues.

In general, rendering techniques differ from graphical techniques because they do not change the graphical objects within a visualization. Instead, they change the way in which these graphical objects are mapped onto an output media. Unlike rendering techniques, graphical techniques are constrained by the *graphical class* of their input objects. A *graphical class* description defines the visual appearance of graphical objects as well as their graphical properties. For example, in our framework, objects belonging to the *horizontal-bar graphical class* has properties *x-length, y-position, color* and *thickness*. Graphical techniques can only operate on these properties and none other. For example, graphical techniques cannot alter the rectangular shape of horizontal-bar objects because the *horizontal-bar graphical class* constrains the objects to be rectangular and do not provide any properties for changing the bars in a non-rectangular

[3] *Semantic distance* measures the goodness of a design with respect to a set of tasks. We described *semantic distance* in detail in chapter 4 and we expand our *semantic distance* model in this chapter, in section 2.

way. Thus graphical techniques guarantee perseverance of "object design" as defined by their *graphical class* (i.e. a horizontal bar will always appear as a horizontal bar before and after the graphical transform method). Rendering techniques on the other hand are not constrained by *graphical class* properties, as a result they may not preserve object appearance. For example rendering methods such as fisheye-lenses and bifocal lenses may distort horizontal-bars so they are no longer rectangular.

F-2.1 Constant Graphical Methods

Constant methods set the values of particular graphical properties to a user or designer declared constant. This is achieved with an *assign* graphical transform that takes a set of graphical property values and a constant as input.

Constant methods are often used to attach a common identifying feature to a set of objects so that users can recognize them as a group. A common application is in providing feedback for search tasks. For example in the *dynamic query slider* technique we search for a set of objects that fulfill certain data attribute constraints by marking that constraint on a slider input device. Objects that fulfill the search constraints will then have their *color* or *transparency* values changed to a common constant for easy perceptual identification. Constant methods can also be used to support occlusion problems by setting the *visibility* or *transparency* property of graphical objects.

F-2.2 Additive Graphical Methods

Additive methods add or subtract a constant to or from a set of graphical values. Additive graphical methods use a *binary graphical transform*[4] with the addition operator (+) as input. As with constant methods, additive methods can be used to provide feedback (e.g. used to change an encoding graphical parameter in a consistent way so that we can identify the objects as a set) or to address readability problems such as removing objects from occlusion or enlarging them so that they can be easily perceived.

The most important difference between additive methods and constant methods is that additive methods maintain the relative ordering among the transformed values while constant methods do not. I.e. if height *a* is less than height *b* before the transform, then this relationship will still hold after an additive transform but not after a constant transform. Therefore, when transforming data encoding graphical properties, it is preferable to use either *additive* or *multiplicative methods* because they both maintain relative ordering. If we use constant methods we lose all of the encoded data information. On the other hand, constant methods are more effective at perceptually grouping a set of objects because they only require users to detect absolute feature similarity while additive and multiplicative methods require users to identify trend similarity (i.e., have the objects changed in a similar way?) which is more difficult to perform.

[4] Binary graphical transforms are described in chapter III-1.2.2.2

F-2.3 Multiplicative Graphical Methods

Multiplicative methods change a set of data values by multiplying them with a constant. Like additive methods we use a *binary graphical transform* but with the multiplication (*) operator as input. Multiplicative methods primarily allow us to stretch or contract the encoding range for the set of chosen graphical values (i.e. representing the same data values with more/fewer graphical values).

As with additive methods, multiplicative methods maintain relative ordering among the transformed values, however, multiplicative methods also maintain ratio relationships. This means that if height *a* is two times greater than height *b* before transformation, it will still be two times greater after a multiplicative transform. Additive methods do not preserve such ratio relationships. For example, once we transform both heights *a* and *b* by adding a constant *c* (where $c \neq 0$), the ratio relationship between the two values no longer holds true, i.e. $a + c$ does not equal $2(b + c)$. An exception is when we apply additive methods to positional properties, for example, in the *SDM* positional shift operations [*Chuah, 1995*].

F-2.4 Linear Positional Rendering Methods (Point of View Navigation)

Linear rendering functions translate the graphical scene verbatim onto the output media. I.e. the relative position and surface properties of objects in the graphical scene remains the same on the output media. In this appendix section, we only consider rendering methods that are applied to positional properties (e.g. *x-position, y-position*) because they are most effective for solving the readability problems that we are interested in. We leave the treatment of non-spatial rendering techniques (e.g. lighting effects, wire-frame rendering) for future work.

Linear positional rendering techniques (Point of view navigation) allow users to view different sections of a visualization scene. This can be achieved by controlling the camera viewpoint on the scene or controlling the scene itself (i.e. moving or rotating the scene). Both classes of techniques transform desired portions of the graphical scene onto an output media and allow users to explore subsets of the graphical scene without introducing any spatial distortions.

F-2.5 Non-linear Positional Rendering Methods (Distortion)

Non-linear functions distort the graphical scene by changing/distorting the positional and/or surface property relationships among objects. Distortion methods unlike point of view methods do not change the viewpoint on the scene (i.e. the portion of the graphical scene being translated onto the output media remains constant). Instead, distortion techniques stretch certain sections of the graphical space while contracting other surrounding areas. This allows users to focus on particular sections of the visualization while maintaining surrounding context unlike the linear rendering methods described above.

F-3 Applying Graphical and Rendering Methods to Readability Problems

Table F-1 summarizes the expressiveness and effectiveness of the graphical and rendering methods described in section F-2 with respect to the four readability issues (occlusion, density, dwarfing, and spatial separation) discussed in section F-1.

		Occlusion			**Density**	**Dwarfing**	**Spatial separation**
		line of sight	overlap	overplot			
Graphical transforms:							
constant	*positional*	-	-	-	-		+
	spatial retinal	-	-	-	-		
	non-spatial retinal	+*	+*	+*	-*		
additive	*positional*	+	+	+	+		+
	spatial retinal	+	+	+	+		
	non-spatial retinal	+*	+*	+*			
multiplicative	*positional*	-	-	-	-	+	+
	spatial retinal	-	-	-	-	+	
	non-spatial retinal	-*	-*	-*		+	
Rendering transforms:							
Point of view	*positional*	+			+	-	+
Distortion	*positional*	-	-		+	-	+

Table F-1: Expressiveness and effectiveness of graphical and rendering transforms with respect to readability.
+ : Readability issue is supported reasonable well; - : Readability issue is not supported well;
"*empty*" : Readability issue is not supported; * indicates the transparency property only.

Each graphical transform class in Table F-1 is divided into the three classes of graphical properties that may be altered: 1. positional properties (*x-position*, *y-position*), 2. spatial retinal properties (*orientation, size*), and 3. Non-spatial retinal properties (*color, shape*))[5]. It is important to consider the range of graphical property classes because they affect the effectiveness of the graphical and rendering functions with respect to readability problems. For example, changing the *color* property is effective for drawing a user's attention but it is not too helpful for solving occlusion problems, which require a change in the *visibility* property. Note from Table F-1 that non-spatial retinal properties are not very effective for dealing with most of the readability issues (occlusion, density, and spatial separation) considered here except for the *transparency* retinal property that can be useful for addressing occlusion problems. Non spatial properties are less effective because the readability issues we address are inherently spatial in nature and cannot be easily resolved by changing non-spatial properties.

F-3.1 Occlusion

The expressiveness of the five graphical and rendering methods discussed in section F-2 with respect to the line of sight occlusion problem is summarized in column 2 of Table F-1. In this case all the techniques can be used to address the line of sight occlusion problem[6] however some methods are more effective (i.e. have a lower *expressive* or *observational distance*) than others. In particular, *constant* methods are not as effective as some of the other methods because it results in the greatest loss of information (i.e. both relative and ratio relationships are lost). Multiplicative methods are also not too effective here because they only allow us to control the positional scale for a set of objects and it is easier to remove occlusion when we have control of the absolute objects positions as is the case with additive transforms. Transparency graphical methods can also be used to address occlusion problems. A possible weakness here is that it is more difficult for us to access information from transparent or translucent objects (higher *observational distance*). However, if the objects we transform are not pertinent to the task then this has no effect on the overall *task semantic distance*. Of the two rendering methods, the *point of view* methods are rated higher here because they allows us to solve line-of-sight occlusion problems with less visual distortion than the *distortion* methods (lower *observational distance*).

From Table F-1 we see that the two rendering techniques, point of view navigation and distortion techniques can be used to solve line of sight occlusion and overlap occlusion. However, these rendering techniques are not expressive of the overplotting problem because they can only change the spatial distance between differently positioned objects, while overplotting is caused by identically positioned objects.

F-3.2 Density

Graphical transforms can be used to reduce ink density by creating *context sensitive displays*. Context sensitive displays allow users to make portions of data ink visible or non-visible depending on the current focus objects. In this way, we may turn on the axis grid lines when we need them for accurate lookups and then turn them off otherwise using a *transparency constant* graphical transform. Graphical transforms can also be used to move ancillary objects away from the focus regions or to minimize the size of those objects to reduce area density. Note that unlike occlusion problems, additive and multiplicative transparency techniques are not expressive of the density problem because simply making a set of objects more or less translucent does not change object density of an area.

---

[5] Note that another possible graphical property class is the temporal (time) dimension. Temporal techniques are not shown here because they are always paired with either a positional or a retinal property, and their expressiveness is dependent on the expressiveness of the paired/linked property.

Apart from the total number of graphical elements within the display, density is also dependent upon the total amount of display space available. Thus another alternative to lowering display density is to render a smaller portion of the graphical scene onto the given output space by dividing the scene into multiple segments and using p*oint of view* rendering techniques to view each of the scene segments separately. However, this lowers the expressiveness of the visualization (not all the information can be shown simultaneously) and may increase the spatial separation among related data concepts. We can also reduce the density of an area by stretching out the area using *distortion* rendering techniques.

F-3.3 Dwarfing

Dwarfing problems may be avoided if we use data transform techniques to summarize the results of our tasks as was shown in chapter IV. Another alternative is to use *multiplicative* graphical transforms on certain ranges of the dwarfed graphical property to expand the graphical value differences among the objects that are of current interest. Constant and additive graphical transform methods are not expressive of the dwarfing problem because they do not have any effect on graphical value encoding scales. When dwarfing occurs on positional graphical properties, we can also address the problem with either of the positional rendering methods. For example, we can divide our data set into several information segments and navigate from one segment to another using point of view navigation. This allows us to encode a smaller data range in each segment, and thus dwarfing is reduced. However, this separation makes it difficult for us to compare values that are in different segments and requires additional articulatory load for navigation. Distortion rendering techniques can also be used to expand the dwarfed areas. However, distortion techniques are less appropriate for the dwarfing problem because they may distort the positional encoding scale, making it difficult to accurately translate the distorted positions back to data values.

F-3.4 Spatial separation

A solution to the spatial separation problem is to use graphical transform methods to move graphical objects to different positions in the display so that the objects that must be compared are never too far from one another. Another alternative is to use rendering distortion techniques to map portions of the graphical scene onto a smaller vertical space, thereby reducing the vertical distance between graphical objects. We can also apply point of view rendering to map the different graphical scene segments we want to consider next to each other. When using these graphical and rendering techniques, however, we must be careful not to overly increase display density or cause occlusion among the elements of interest.

F-4 Graphical and Rendering Transform Guidelines

Design guidelines for selecting mapping transforms (i.e. selecting data to graphical mappings) were set forth in previous work on automatic visualization design. Design guidelines for selecting data transforms and contrasting their use with mapping transforms was discussed in chapter IV. Here we

---

[6] The only exception is that *positional* methods **cannot** be used to solve self-occlusion problems.

consider guidelines for using the last two transformation classes: graphical and rendering transforms. We focus on rules for selecting effective graphical and rendering methods for solving the four readability issues that commonly arise in visualization displays (occlusion, density, dwarfing, and spatial separation). These design guidelines are aimed at reducing the semantic distance measures described in chapter IV-2.

F-4.1 Relevance of Readability Problems with respect to Tasks

Many readability problems may arise within a visualization design, however, not all of these readability problems are as important or relevant to the current data analysis tasks. Solving all readability problems will require introducing many visualization techniques into the design and this will invariably result in resource conflicts (graphical property and input device conflicts) among the techniques. Thus the first step in choosing appropriate graphical and rendering techniques is to determine which readability problems are most relevant and important. The readability problems that are most important and relevant are those that most affect the end user's ability in solving their goals. Readability issues that do not affect the task or occur over objects that are unrelated to the task need not be addressed. Whether a readability problem significantly affects our task solution depends on the following three factors:

1. *Task specification (task operators and task arguments)*: The relevance of readability problems depend first and foremost on whether the data concepts they affect are important to our current task(s). Both the task operator and the task arguments affect the relevance of readability problems within a visual display. To effectively address readability problems we must first identify which ones are most relevant based on our task and task input arguments.

2. *Distribution of data values*: The relevance of readability problems also depends largely on the distribution of data values that are related to the task. For example finding the *max* value within a dwarfed data set that only has one clear high value is much easier than finding the *max* value within a dwarfed data set with many high almost equal values.

3. *The accuracy required of the task results*: Task accuracy is also an important factor in determining the relevance of readability problems. Readability problems often lower the accuracy of task results, thus depending on the level of accuracy required by the current task, different readability issues may take precedence. Accuracy also affects the effectiveness of graphical and rendering techniques. Some of these techniques distort or change the graphical representations in the visualization display so that it no longer shows the exact data values and relationships in the original data set. Depending on the task accuracy required, these distortions may affect task expressive and observational distance to different degrees.

In summary to design an effective visualization system, we must identify and order existing readability problems based on their importance and relevance to our current data analysis tasks. We then attend to the readability issues according to their order of importance, assigning better feedback properties and input controls to the problems that have greater significance on the overall semantic distance.

F-4.2 Continuity

Continuity in this context refers to gradual visual transformation. For a change to occur gradually we divide it into multiple smaller changes all occurring within an acceptable time period of the other as to give the appearance of animation or movement. Animated or continuous techniques provide users with better context on how a visualization technique changes the display and which objects are affected (i.e. improves *technique observational semantic distance*). Only continuous graphical properties (e.g. position, saturation, size, length) can be animated. Non continuous graphical properties (e.g. shape, texture) can only be changed discretely. It is however possible, although not desirable, to change continuous properties in a discrete way by making the entire change in a single non-animated step.

An example discrete technique is the painting technique [*Becker, 1987*], which changes the hue of objects discretely so that they appear more salient. Another example is the bifocal lens technique, which commonly changes the size and position of objects in a single visual step that can be quite jarring to users. Some example continuous techniques include the techniques within the SDM system that allows users to manually and continuously change particular graphical properties of objects by selecting and dragging on object handles. Some other examples of continuous techniques include the node rotations within a ConeTree [*Robertson, 1991*], the stretching techniques presented by Sarkar et. al. [*Sarkar, 1993*], or the tree reorganization operation in Hyperbolic trees [*Lamping, 1995*].

Continuous techniques reduce *technique expressive distance* because they provide users with good constant updates of the state and progress of a visualization technique. Continuous feedback of intermediate states also enables users to more easily detect technique errors and bugs. Continuous techniques also reduce *technique observational distance* because they provide users with better context on how a visual representation has changed from its initial state as a result of the visual transformation. In addition they also give users with more time to focus on the interesting objects while a change is occurring. On the other hand, continuous feedback is less useful when the visual changes are small and well understood (i.e. when the observational distance of the design is small to begin with) or if the time addition due to the continuous change is very significant (e.g. when the visual transform is repeated many number of times).

Another related issue is that of spatial continuity. This refers to whether the changes brought about by the graphical and rendering techniques create a spatial discontinuity in the display. The bifocal lens technique [*Leung, 1989*] for example creates a spatial discontinuity because the display is divided into two disjoint sections that are rendered at different levels of magnification. The same effect arises in the magic lens, and table lens techniques. Such spatial discontinuity may be jarring to users as was shown by Hollands et al. in a user test comparing bifocal lenses to fisheye lenses.

F-4.3 Individual vs. Group Readability

As was discussed in appendix C-3, there are two general tasks classes: simple value pair tasks or more complex group tasks. For value-pair tasks we must address readability issues for individual task related objects. Usually, single objects are selected for transformation many times, thus the frequency of executing the readability technique is high. Consequently, we want to keep the cost of each readability operation to a minimum even at the cost of greater initial learning time.

For group tasks we solve readability issues for an entire group of objects (not single objects as was in the previous case). For example when solving group occlusion issues we are more concerned with showing the general shape of the group, (i.e., we are more concerned with occlusion at the group edges) rather than with internal occlusion among individual objects within the group. Unlike value-pair tasks, group tasks require a readability technique to be repeated less frequently because operations are applied to groups rather than individual objects. Thus it is less important that we keep the input load (i.e., *technique semantic distance*) low and more important that we keep *task semantic distance* to a minimum.

F-4.4 Object spatial proximity

Rendering techniques are effective for addressing readability problems that occur among spatially proximal objects because rendering transforms operate on spatially contiguous regions and applying a single transform may remove readability problems from several objects simultaneously. To address the same readability problems using graphical transforms would require a higher articulatory distance because each of the task related object must be enumerated by the user[7]. In addition, more visual feedback must be provided to indicate the location and extent of changes introduced by the graphical transforms. This results in a higher *observational distance*.

However, if the target readability objects were spatially distant from one another, we would need to apply multiple rendering distortions to spatially disjoint areas thereby increasing the feedback complexity and the amount of distortion introduced into the display. Articulatory distance is also higher because now users must specify multiple focal points. On the other hand, graphical transforms become more effective in this case because they introduce less visual distortion compared to rendering transforms that are applied over multiple separate focal points.

F-4.5 Reversibility

Reversibility refers to how easily the changes made by a graphical or rendering transform may be removed from a visual display. Reversibility of graphical and rendering transforms is important for three reasons:

[7] The articulatory load may be reduced by using functional definition selections, (e.g. using a bounding box to set constraints on the position of the objects desired).

1. *Reduces conflict among different transformation techniques*: Some graphical and rendering techniques may introduce new readability problems into the visual display. For example, the fisheye technique increases density of the contextual areas and distorts the position of objects within the display. As a result, we may want to remove the effects of the fisheye transformation once we perform our immediate task so that the display distortions it introduced will not influence other tasks that need to be performed. The ability to reverse the transformation allows us to deal with the readability issues on a task by task basis and not have to worry about conflicts in readability operations across different tasks. This reduces the task expressive and observational distances of a visualization technique.

2. *Enables users to easily repeat visual changes that were missed*: Reversible transformations are also useful for repeating a graphical or rendering transform. Sometimes, we may not catch all the visual changes made by a technique. Being able to reverse and reapply the technique gives us the ability to peruse the effects of the technique repeatedly and more effectively capture the visual alterations that have occurred. I.e. this lowers technique observational distance.

3. *Reduces the cost of input and operation sequence errors*: Interactive visualization techniques allow users to control or manipulate part of its operation and effects through input devices. Sometimes however, users may make input errors and it is important to allow the effects of those error(s) to be reversed. User errors may also occur when we need to perform a series of graphical and rendering operations in sequence to get combined effects on a set of objects. In such an instance if we accidentally perform an operation out of sequence, we must be able to reverse it so that we need not repeat the entire sequence again from scratch. Reversible techniques reduce the articulatory cost of making input errors or operation sequence errors.

F-4.6 Learning

The effectiveness and usability of a technique is increased if it can be easily learned and if it is easy to use and remember once learnt. Some important learning issues include:

1. *Consistency* : Techniques within the same visualization should share consistent interfaces and controls. This helps users remember the techniques and enables them to transfer knowledge from one technique to another. To ensure technique consistency we can decompose the techniques based on the interactive framework presented in chapters II and III, and then give preference to techniques with similar structures.
2. *Vocabulary size*: The more techniques there are in a visualization system (i.e. the greater the technique vocabulary size), the harder it is for users to master and utilize these disparate techniques. Thus we should try to keep the number of controls and number of different techniques to a minimum.
3. *Affordances*: This refers to whether affordances or cues are provided to users for indicating how a visualization technique should be used and manipulated and what problems it can address. Such cues can take many forms. We can provide instructional support by integrating the automatic visualization

designer with an explanation system like AutoBrief, which can provide textual instructions. Effective affordances can also be derived by picking input devices whose appearance suggests the inputs it can generate and the way it should be manipulated.

Note that the three factors presented here are obviously not a complete list of factors that affect learning a technique. Because of the breadth and complexity of learning issues however, we leave a more complete treatment of it for future work.

F-5 Conclusion

In this section we identified four important readability issues (*occlusion, density, dwarfing* and *spatial separation*) by examining constraints of the CRT screen and out perceptual system. We then considered how the graphical and rendering transform techniques captured within our framework in chapter III can be used to address these four problems.

To add these readability decisions into our automatic design system, we must add several additional steps in our search algorithm as is shown in gray in Figure F-1. In particular, once we have finished constructing a design, we check to see if that design has any readability problems. If so, we consider all available graphical and rendering techniques and pick one that is most appropriate given the current task, data set size, data value distribution, and according to the guidelines and metrics discussed in chapter IV-2 and section F-4.

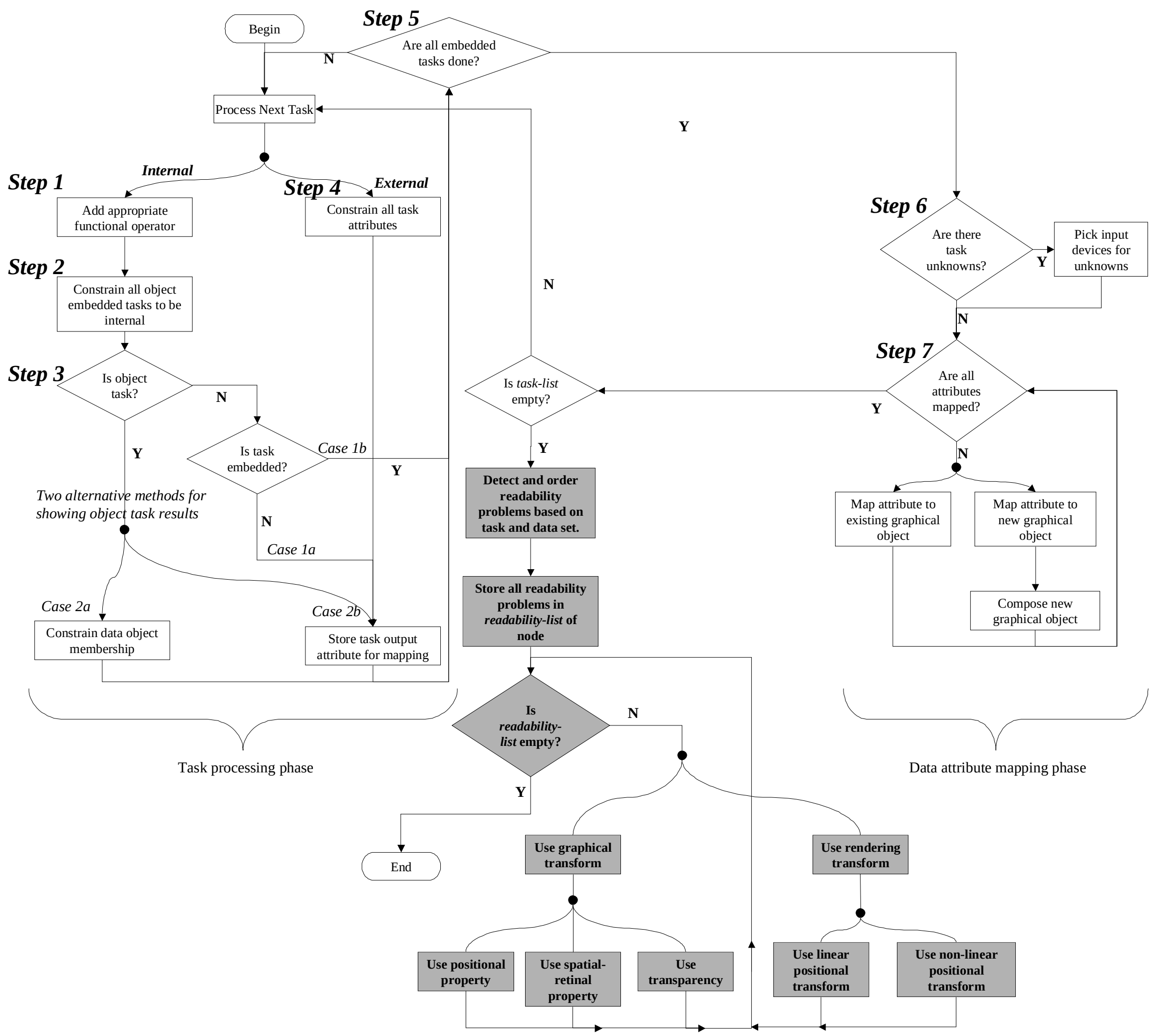

Figure F-1: Augmented search algorithm for our automatic design system AVID. Additional steps take into account readability issues and how to solve them.