ABSTRACT

WHITEHORNE, ANDREW E. Telescope: A Multivariate Visualization Framework in Support of the Development of a Perceptual Visualization Hierarchy. (Under the direction of Associate Professor Dr. Christopher G. Healey).

Scalability has become a major issue within the field of visualization as data gathering methodologies and display technologies diversify. To compensate for large data sets and display limitations, perceptual visualization techniques aim to optimize graphical data representations with human vision in mind. The following outlines the principles, motivations, and development behind the Telescope multivariate visualization framework. Telescope provides mechanisms for generating and manipulating visualizations in which the visual angle and display resolution of individual elements are of chief concern. This software utilizes ongoing research towards the development of comprehensive perceptual feature guidelines to provide a means of dynamically mapping several common visual features to a given data set. Specific element mappings may be disabled or enabled when the size or display resolution of the displayed data elements crosses a defined threshold. Visualization features of interest are: hue, density, direction, flicker, luminance, orientation, regularity, size, and velocity. The system also allows for the simulation of various viewing environments with concern to display size, viewing distance, and visual angle. Support is provided for the input of a common data format and the easy manipulation of system parameters by means of a rule system. The end result is a system which lays the foundation for the implementation of a comprehensive perceptual visualization hierarchy.

Telescope: A Multivariate Visualization Framework in
Support of the Development of a Perceptual Visualization Hierarchy

by

Andrew Ennis Whitehorne

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2007

Approved By:

Dr. Robert Fornaro

Dr. James C. Lester

Dr. Christopher G. Healey
Chair of Advisory Committee

Dedication

In Memoriam

Clarence R. Ennis

1924-2007

Biography

Andy Whitehorne was born July 16, 1984 in High Point, North Carolina to Robert and Jane Whitehorne. He entered North Carolina State University in Raleigh, North Carolina in 2002 and graduated Cum Laude with a BS in Computer Science and a minor in Cognitive Science in 2005. He is currently a graduate student in Computer Science at North Carolina State University.

Acknowledgements

Special Thanks To:

Dr. Christopher G. Healey, Academic Advisor and Research Supervisor

Dr. James C. Lester, Committee Member

Dr. Robert Fornaro, Committee Member

Amit Sawant, Collaborator

Ping-Lin Hsiao, Collaborator

The Members of the Knowledge Discovery Lab

Mom, Dad, Katie, and all my friends and family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There seems to be a trend in academia, and society as a whole for that matter, towards an increasingly data dependent existence. Alongside the propagation of computing technologies into various disciplines, so too has information analysis become a crucial part of many industries [GEC98]. While the task of accurate and efficient data evaluation is daunting enough on its own, this issue becomes compounded by the sheer glut of data generated due to the unprecedented ability to collect and store ever larger and more detailed data sets. Given these concerns, there exists a dangerous potential that human beings will be unable to analyze and comprehend vital information in a timely and competent manner. This could ultimately lead to situations where the failure to understand available information could result in a great material or financial loss.

1.1 Visualization

The field of visualization attempts to simplify the communication of information by presenting graphical depictions rather than relying solely on hard numerical data [BEJV95]. Often times, discerning variations or identifying specific cases within a data set can be challenging and time consuming when viewing textual representations. Visualizations aim to streamline the process by mapping data values to various graphical features allowing the user greater ease in identifying trends, statistical outliers, and unique cases at an improved rate [GEC98]. However, while visualization is often effective in this task, as the amount of information increases, achieving good perceptibility and scalability becomes more challenging [Che05, Joh04]. Aside from the obvious technical issues regarding managing and rendering massive data sets, as the amount of information increases, screen space becomes more valuable. For instance, if data elements were each allocated an area 32×32 pixels (points on the display), the size of say a desktop GUI icon, the maximum number of whole, non-overlapping 2D elements which could fully fit on an 800×600 pixel resolution display, without overlap, is 450, with rows containing 25 elements and columns containing just over 18 elements. This is grossly insufficient for many data sets, which might exceed thousands if not tens of thousands of data points or more.

Increasing the display size is certainly an option; however, at a certain point this becomes costly, technologically taxing, and impractical. Even if building increasingly larger displays was financially, computationally and physically possible, the screens would fairly quickly become so large that users wishing to see a significant portion of the image would have to view from a distance so far away that they would be unable to resolve the fine details of individual elements, unless each element was allotted more screen space. This negates the advantage of a large display. And while many studies advocate the advantages of utilizing large scale displays [BBSN07, YHN07, BN05], larger displays are not always preferred by users and may not have an significant impact on performance in all cases [Sim01]. Beyond this mere realization is the progressive miniaturization and mobilization of computing technologies, making the ever increasing display that much more impractical. The fact of the matter is that while large scale displays can be useful in sedentary and scientific viewing environments, large displays are not practical in all situations, particularly those with size and mobility constraints.

Given the limitations placed on the size of a display, it might be more advantageous to allow the user to manipulate what is actually presented. There is a large body of research into various sorts of visualization navigation techniques, whereby users can move the focus of the system to make different sets of data viewable [DH02] and focus+context or overview+detail systems that attempt to present a detailed view while maintaining the ability to compare elements [BGBS02, Fur86, LRP95, RMC91]. These methods are necessary due to the inability to accommodate all the information on a display at a single time. Although helpful, they are in many cases highly user driven, requiring the activity of the user or otherwise some intelligent agent. Also, even the best navigation or focus techniques typically take some time for the users or agents to perform and recognize the transition between states, with user input increasing the possibility of human error. Therefore, it would be desirable to minimize the amount of navigation and user manipulation required within a system while still providing as much detail as possible.

One of the more straightforward ways to do this would be to decrease the size of each data element, as long as it is still possible to represent and detect the feature detail. For instance, if it is possible to decrease the necessary pixel allotment per element by half without a significant reduction in user performance for a given task, then twice as many elements could be viewed at a time in the same amount of space. How though does a visualization creator know the best methods for this type of optimization? How can he or she determine how change in size will impact visualization features such as color or motion?

Accomplishing this sort of task properly requires a fundamental understanding of human visual processing as well as the ability to apply this knowledge. The field of perceptual visualization seeks to create graphics and visualization systems while considering the capabilities, limitations, and predispositions of the human perceptual systems. Realizing the strengths and weaknesses of human vision allows for systems to be engineered which avoid instances where data will be obfuscated by poor perceptual situations.

Often this involves designing visualizations such that information with high priority is represented by perceptually strong visual features. Determining the best feature for a given situation is difficult. Various work has been conducted to characterize the abilities of different types of features [HBE96, DEB+07, PGB07, Wun04] with prior research conducted within the North Carolina State University Knowledge Discovery Lab contributing to the development of basic guidelines with regards to relative feature strength and effectiveness [Hea99, Hea07]. Current research is focusing on the role that the display resolution and angular size of data elements have in the effectiveness of low level perceptual features and the impact this information will have on the development of a visualization feature hierarchy.

1.2 The Telescope Framework

The following outlines the implementation of Telescope, a multivariate visualization framework intended to aid in current and future research regarding low level visual processing of common visualization cues. The system will serve as a prototype for a comprehensive perceptual visualization hierarchy, implementing existing guidelines and providing dynamic feature mappings based upon changes in data element size and display resolution which influence changes to acuity in varying display environments. The Telescope framework allows for quick and easy manipulation of visual feature mappings across a standard data input format. Telescope is also capable of dynamically concealing mapped features from being displayed based upon changes to display resolution, subtended visual angle, viewing distance, and both known and experimental perceptual thresholds. Telescope can quickly be modified for testing various display environments and can also be used to simulate various displays and viewing distances on a standard workstation monitor.

The main goal in the creation of this software is provide a prototype which can perceptually optimize visualizations given a set of known viewing conditions. By adding or removing detail from the image at changes to viewing scale, distance, or resolution only those items which are distinguishable by a user will be presented. As the viewing environment is altered, different amounts of information will be available to the user. Also, by creating a system which can easily interpret datasets with different types of content and allow for quick mapping of information to graphical features, both existing and future research will have more time to spend with studying user performance and less time preparing test software. The framework also provides an easy means for testing across a varied set of displays and for simulating display devices of a different scale. This reduces the need for the collection or assembly of a large number of display devices for testing purposes and provides constant display conditions in order to eliminate display differences from testing focused on display resolution and visual acuity.

1.3 Thesis Organization

In order to facilitate an understanding of the ideology and motivations behind the construction of this software, the details of its design and implementation will be preceded by a brief overview of visualization principles and several applicable perceptual issues. The visualization background will describe the goals and practices of visualization design. Perceptual concerns will primarily concentrate on the aspects of human visual acuity and preattentive processing. In order to provide insight and justification for this project, prior relevant research will be discussed noting areas of deficiency the Telescope framework aims to assist in resolving.

The design section will elaborate on the basic functionalities the Telescope framework and the motivations behind their inclusion. In the implementation section, the general organization and flow of data throughout the software will be outlined, noting the technical concerns which affect the system. The resulting prototype will then be detailed and example data will be presented to display the resulting visualizations. Concluding commentary concerns the future work related to the system as well as discussion of the strengths and shortcomings of the project.

Chapter 2

Background

2.1 Visualization Basics

Visualization is first and foremost concerned with the accurate communication of information or concepts [War04]. This is accomplished by creating abstractions, or relationships, between graphical features and the data which the visualization is attempting to represent. These graphical features may be as simple as basic visual aspects such as the color of a glyph or its orientation, or may be a more complex entity comprised of multiple forms and features. That is to say, that for one data element D, there exists a set graphical features S such that D is represented by each of the items in the set, $D \Rightarrow S\{s_0, .., s_n\}$.

Variations of these features are used to represent variations in the data set, for instance using reddish hues to denote regions of high magnitude and bluish ones to represent low magnitudes. The primary visual features concerned with this particular research project can be found in Table 2.1.

Table 2.1: Visualization Features

Name	Description
Hue	The variation in the hue across some color range.
Density	The quantity and compactness of glyphs contained within the space allotted to an element.
Direction	The direction in which the glyph moves.
Flicker	The rate of at which the glyph cycles on and off.
Luminance	The variation of a glyph in luminance. Appears as grayscale without a combined color mapping.
Orientation	The angle of rotation of the glyph about the center.
Regularity	The amount to which a glyph conforms to its defined position. The amount it "strays" from center.
Size	The dimensions of the space a glyph occupies.
Velocity	The rate at which the glyph moves.

Within ongoing research, the primary focus in establishing feature guidelines is concerned with color and texture cues. Color cues refer to hue and luminance, while texture cues refer to density, size, orientation, and regularity. Motion cues are studied as a means of preparing for future experimentation, however current research efforts have not performed detailed analyses.

The term color refers to the variation in hue, saturation, and luminance across some defined color scale. A color scale allows for the definitions of different colors at points along a set of values. One such example would be the unit length RGB color scale, wherein as values move from 0 to 1, the colors proceed from blues in the low end, to greens in the median, and reds as values approach 1. Color abstractions may also define distinct colors to represent specific data cases, such as allotting individual colors to movie genres (romance = red, drama = blue, etc...). Hue represents the perception of different wavelengths of light each giving off shades of what many think of commonly as "color",(red, blue, green, and so on). Luminance refers to the luminance measure of the element; that is, the "lightness" quality of the color [CWE04]. When the hue aspect is absent, luminance is represented effectively as the grayscale from black to white, or the presence of light at a pixel. Determining the variation between elements requires determining the variation in contrast between elements [CW00].

Spatial representations can also be very useful. Density describes the concentration of glyphs within an area. Typically a higher glyph concentration would be used to denote greater magnitudes. Regularity describes the amount of deviation from an element's defined position within a some structured spatial organization. Elements which adhere strictly or with a high level of conformity to this structure are described as being regular, whereas those which deviate are described as irregular [HE98a]. Regularity is particularly useful when elements exist in a pattern or grid organization in which the user can identify a deviation from the spatial organization.

Motion can be split into three distinct properties; direction, velocity, the rate of flicker. Direction describes the vector towards which the element moves. The velocity is the rate at which an element traverses the image. Flicker defines the rate at which a glyph cycles from on to off.

Orientation is represented as the rotation of a glyph about its center. The perceptual characteristics of these features will be discussed in later sections. Size defines the physical dimensions of the glyph.

By varying these features dependent on a particular piece of data it is possible to identify differences in the data. In some cases this indicates correlating the magnitude of the data to the scale of the feature, such as making elements with larger magnitudes brighter in the image. However, other times the data may have a more direct correlation with a feature, such as data indicating an angle corresponding to an orientation. The decisions of which data to map are many times based upon the type of data as well as the purposes for which the visualization is being created.

The field of visualization is sometimes divided into two distinct subsections; information visualization and scientific visualization [RTM+03]. The key differences between the two disciplines are essentially the constraints placed upon the abstraction of data into graphical representations. In scientific visualization the information to be conveyed is heavily based in scientific principles and physical metrics, the key usually being an existing 2D or 3D spatial positioning of elements. This can limit the flexibility in which features can be used to represent certain data due to real world representations and organizations. For example, if there was some data on precipitation levels over a region and the key was to identify regions with the highest levels of rainfall, it might seem make sense to organize the position of the graphic elements in order of magnitude. The problem with this sort of rationale is that it ignores the spatial relationships of the data. It makes more sense to organize the data like a map, with elements representing the data positioned such that they maintain their physical location relationships. Scientific visualizations typically have a basis in the physical world and therefore often must be represented in a way that reflects actual common observations [Lab07].

Information visualization is typically much less constrained. Often the data comes from a less tangible origin, where there exists no corresponding existing default depiction aside from the raw data [GEC98]. A simple example would be representing the ratings of movies along with their revenues. Here there exist two quantities for which there are no well defined physical depictions in the real world. Therefore it would be just as feasible to use color to denote the revenue magnitude as say another feature such as the glyph density. While there is some debate to the level at which the two disciplines can share methodologies, the fact that the goals and information processing aspects remain fundamentally the same [RTM+03] indicates that many methodologies can be used or adapted for either discipline. Where perceptual issues are concerned, this work will not make any distinctions between the two disciplines.

It has been stated previously that the goal of visualization is to communicate information via graphical abstractions. But what advantages does this actually provide and why? What is the reason for why visualizations are effective? The typical reasoning is that data presented visually is less taxing on human beings than other means [War04]. By presenting data as a visual cues, it is easier to identify trends and patterns within a data set which may be lost in a large set of numerical data. While text representations may be better for exact quantitative analysis, the ability visualizations provide to quickly and accurately resolve relationships between a large number of data points is highly desirable. Compound this with the fact that human brains seem hardwired to recognize these variations in many common base visual features [CWE04] and the advantages of working with visualizations become more clear. Ultimately, understanding why visualization is effective requires an understanding of human perception.

2.2 Perceptual Issues

The construction of the Telescope framework is based upon research conducted on human perception. Of particular importance to the development of this software are the concepts of visual acuity and preattentive feature processing. This section provides some basic background information on these perceptual concepts and their relevance to current and future visualization research.

2.2.1 Visual Acuity

Visual acuity in the simplest sense is the ability to discern detail in the images presented to the eye [HGBE00, CWE04]. There are a number of different measures of acuity, each denoting the ability to discern some specific type of difference in form, color, texture, etc. However, to simplify terminology and analysis, the term acuity will be used generally for the ability of users to identify a target or region which varies in the magnitude or style of a visual feature from the background field of elements.

Many different factors can influence acuity. But as the key experimental value enabled by the Telescope framework is element size, the most important factors with respect to this research are the number of pixels and visual angle. Computer displays are composed of a discrete number of individual light sources referred to as pixels and combinations of pixels are used to form the images on a display. In certain cases, there are limits on what a specific number of pixels can actually display. For instance, variations in size cannot be represented by a single pixel. An individual pixel cannot change its own dimensions, only turn on or off, therefore only values of a sufficient enough magnitude will "turn on" the pixel and all others will be off. This creates a situation where a portion of the data points cannot be viewed. Even if a 2×2 square set of pixels where alloted to each element, there are only three possible size representations given this amount of space, 0 pixels per side, 1 pixel per size, and 2 pixels per side. As such, regardless of the actual size of the pixel, the number of pixels will impact the ability to represent variations within a data set.

The visual angle is "the angle subtended by the extremities of an object at the entrance to the pupil or other point of reference"[HGBE00], or in other words, an angular measurement of the size of an image when projected onto the retina. The retina is the membrane on which the visual receptors, the rods and cones, are located. Light energy stimulates these rods and cones which then signal the brain [CWE04]. The key here is that the resolution of the retina is both fixed and the receptor density varies across the field of view, the highest within a mere 2 degrees of central focus [YC06].

In order to be able to see more detail, the light reflecting off an object must stimulate a greater number of receptors and since the number of rods and cones cannot be increased or decreased for a given area, to cover more receptors the image must span a larger amount on the retina. Therefore the size of the image is paramount to the ability to resolve detail. If a person were able to see the whole of an image from two different distances but the image cast on the eye was twice the size from the closer distance, then the closer viewpoint would occupy more space of the retina and thus more receptors are stimulated across a greater range of angles. An example of this is presented in Figure 2.1.

To calculate the visual angle there are several pieces of information which need to be available. First is the size of the object being viewed. For a traditional computer display, this size is determined given the number of pixels tall (by convention) allotted to a data element, n multiplied by the size of each pixel, p, such that $s = n * p$. Second, there is the distance from the observer to the object. Given the size of the object in a dimension s, and a distance of d, then the visual angle can be computed via the formula $\Theta = 2 \arctan \frac{s}{(2d)}$ [Kai05].

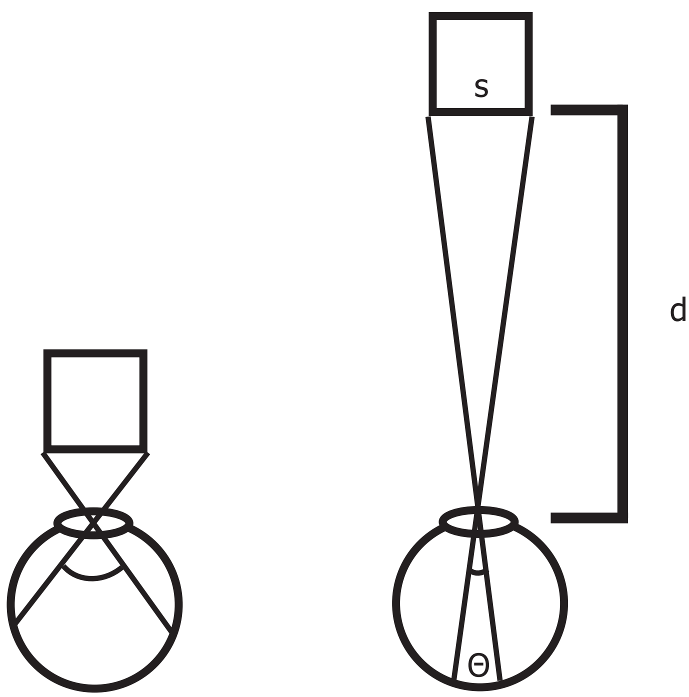


Figure 2.1: Visual angles of an object at different viewing distances.

Visual angle is a key measurement where display size is concerned. A large display communicating the same set of visualization elements and attribute mappings as a small display will have elements that cast a much larger image onto the retina from the same distance when pixel resolution is equivalent. This increases the resolution of the viewed object on the eye, allowing for more detail to be perceived. However, while visual angle is important, it is not the sole factor when scalability is concerned. There have been studies where even when maintaining equivalent visual angles, larger sized displays have had a significant variation in user performance for some tasks [YHN07, TGSP03]. Also, different visual features perform with varying degrees of performance [HBE95] and the combination of different features may aid, interfere, or otherwise influence data communication, depending on the pairings and implementation [Wei04, RLMJ05, UIM+03]. Therefore, there must be other processes which occur outside of mere visual angle and display resolution.

2.2.2 Preattentive Processing

Fairly frequently within images, certain aspects of the image seem to "pop out" at a viewer. This often occurs in cases where there exists a piece of data which is visually disjoint from the rest of the set. That is to say, that in a field of red hues a blue region would quickly draw the attention of a viewer, without a large amount of effort or time expended by the user. This type of viewer reaction is said to happen because of the "preattentive processing" of the brain [HBE95, HBE96]. These preattentive processes seem to draw the focus of the user without their active participation. While not absolutely attention independent as their name might apply, preattentive processes do appear to act in a manner intended to quickly motivate the concern of a viewer to a particular point of interest [DEB+07].

Preattentive processes are separated out from normal visual recognition processes because of their quick performance. Low-level actions which can be conducted in 200-500 milliseconds are generally considered to be preattentive in nature [HBE96, Hea07]. These are typically found to occur when there is the identification of a unique element in the field of vision. Such a mechanism seems to make sense as an evolved vision system would tend to prioritize the recognition of variances which may indicate a threat or means of survival.

Preattentive processes are especially noticable in tasks concerning:

1. The detection of unique target features in a field [HBE95, Hea07].
2. The detection of boundaries between groups of like elements [HBE95, Hea07]
3. Following the movement of targets or regions with similar elements [Hea07].
4. Quantification of differing elements within a field [HBE95, Hea07].

To provide an example of a preattentive task, Figures 2.2-2.3 present a visual search task where a user is asked to identify a different target region from the field. In Figure 2.2, the target is a region of red hued glyphs from the field of black glyphs. In Figure 2.3, the target is a region of square glyphs in a field of circular glyphs. In both cases, the target region is easily and quickly identifiable.

This is a fairly small data set though, and it utilizes reasonably large sized glyphs. Figures 2.4 and 2.5 present the same visual search tasks with a larger number of elements, at a smaller per element size, and a smaller target region. One of the key advantages of utilizing preattentive processing for the purposes of visualization is that preattentive features have thus far been shown to scale extremely well [HE99], with only small decreases in performance when size of the display and number of elements are increased.

The preattentive processes of the human visual system are not a magic key to solving the visualization situation though. As more features mappings are added and the various abstractions form conjunctive representations, features will begin becoming less rapidly identifiable within a landscape of highly cluttered features [RLMJ05] or influence the perception of other features [DEB+07, HBE96], such as the luminance impacting the perception of hue within an image.

Figure 2.6 shows an instance where the combination of features may hinder the speed at which a person identifies a preattentive target. The field consists of blue circles and red squares. The target is a red circle. In this case, there is no unique feature to cue on, both

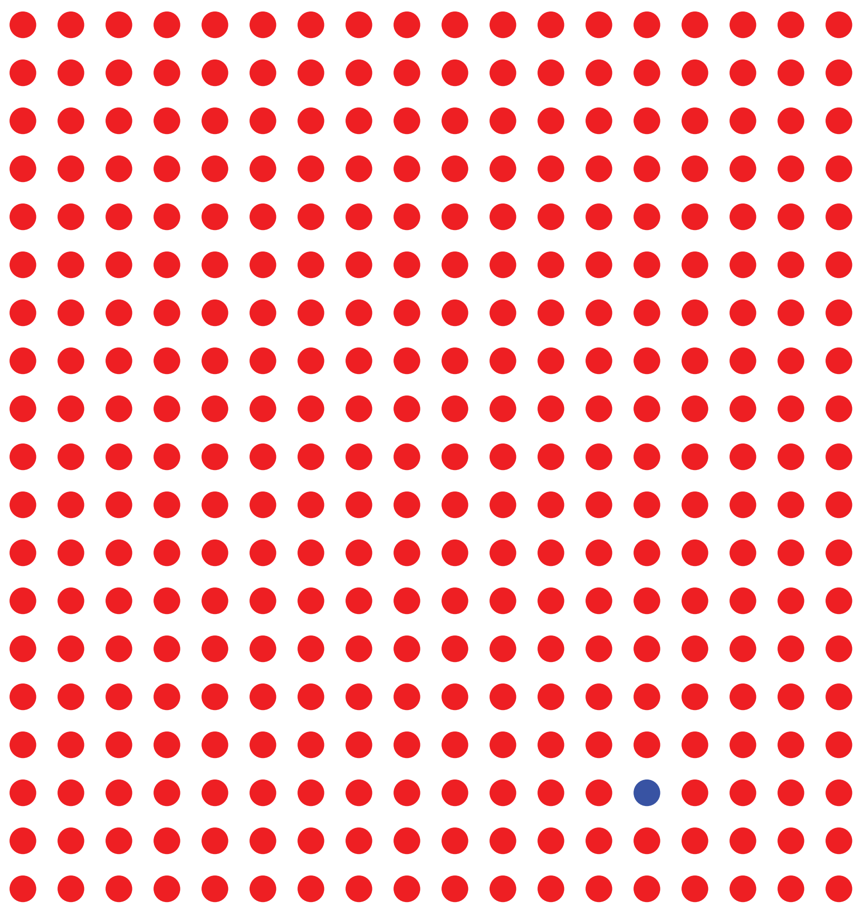


Figure 2.2: Hue in preattentive situations.

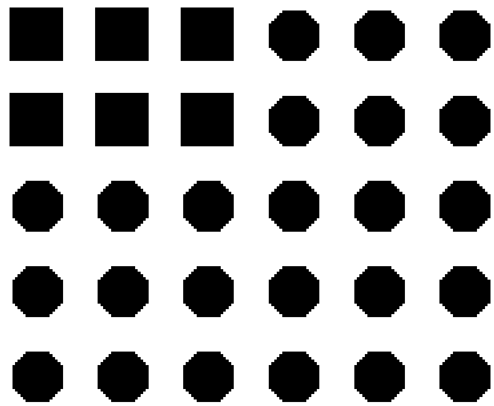


Figure 2.3: Form in preattentive situations.

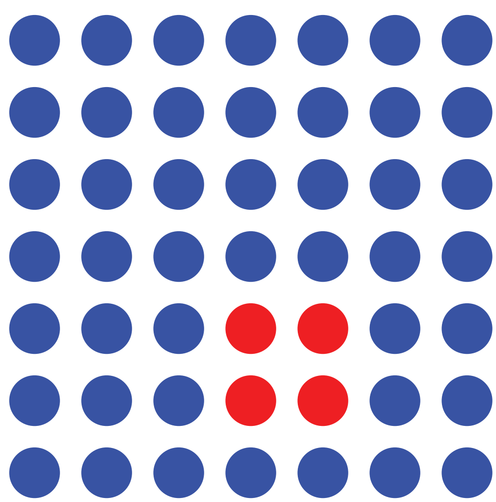


Figure 2.4: Hue with smaller targets and more elements.

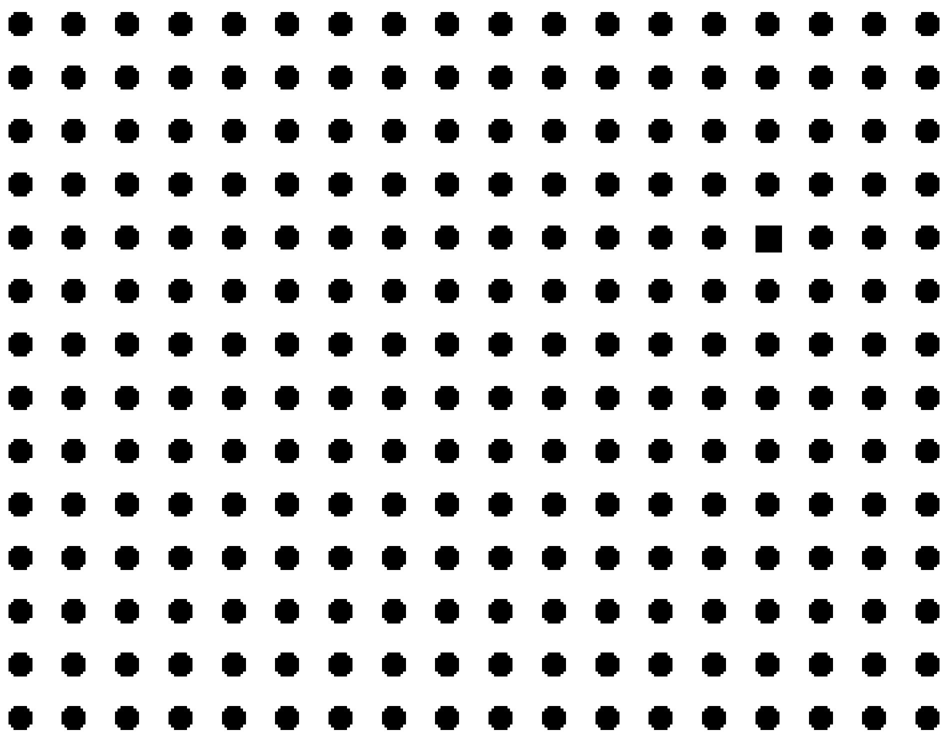


Figure 2.5: Form with smaller targets and more elements.

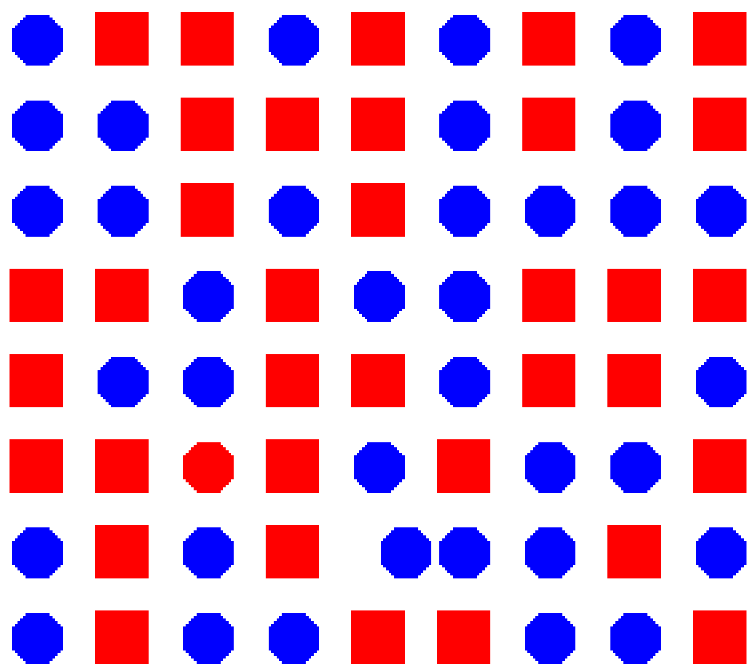


Figure 2.6: Hue and Form Mixed.

red and blue hues are found as well as square and circular glyphs. As more abstractions are added to each element, more effort is required to determine uniqueness.

The keys to effectively utilizing preattentive processing for the creation of visualizations include choosing abstractions that appropriately highlight important data and avoid conflicts with other abstractions. This requires that designers have a fundamental knowledge of which visual cues are more effective than others and how these features interact. Also, significant to this research, how does the size of a given feature or the display conditions under which it is viewed, influence the perception and comprehension of a given data abstraction.

2.3 Research Goals

The main focus of the current line of research is heavily steeped in the previously mentioned area of perceptual visualization. By obtaining a better understanding for the basis of how human beings process low-level visual features, it may be possible to gain a better understanding about which graphical features perform better or worse given certain situations. Also, by testing features against one another the amount of interference between elements can be better understood. Finally, studying the limitations of various feature abstractions given display and viewing conditions will aide in eliminating scenarios where data analysis is impacted by viewing size and distance.

The primary reasoning for this work is to facilitate the further understanding of the relationships between common visualization features and their effectiveness across differing viewing conditions and sizes. This will contribute to the development of a graphical feature hierarchy. This graphical feature hierarchy will provide the benefits of identifying the best feature mapping given particular perceptual situations and a logical mechanism for the dynamic adjustment of feature display when circumstances become less than ideal [SH07].

To these ends, the Telescope framework is intended to assist in the analysis of graphical features and viewing conditions as well as implementing a feature hierarchy prototype. The Telescope framework should provide mechanisms for representing data with various data mappings and allow the mappings to be disabled or enabled when the viewing conditions become less than ideal. The final product should provide a mechanism for analyzing and optimizing the representation of data with respect to viewing distance and angular element size.

Chapter 3

Related Work

There is no shortage of research into visualization. Much of the early work in computerized visualization was concerned with the technical challenges associated with rendering the graphics. More recently though, as many of the outstanding rendering issues have been conquered or sufficiently marginalized, researchers have been able to spend more time analyzing the human-factors issues behind graphical designs. Given the research goals the Telescope framework aims to assist, the areas of research with the most concern are those involving the study of performance for features and those concerning the scalability of visualizations and features.

3.1 Feature Studies

A considerable amount of work has been invested in determining the various impacts and performances of certain features. Color primarily has drawn a considerable amount of attention, most likely due to the general consensus and experimental results suggesting that it is one of the most highly salient visual features. Color can be seen to show a dominant effect on other features and is often utilized as the highest priority visual features [Hea07, Wei04]. Proper color selection is important to effective data communication [Hea96].Other features such as regularity and density have received less attention but appear to perform at a slightly lower level of performance on average and have a larger spatial requirements [Hea03, Hea99].

Healey and Enns performed extensive research into preattentive processing of basic features with concern to visualization [HBE92, HBE95, HBE96] based largely upon background work in human psychophysics by Triesman [Tre86] and others. Their analysis focused on the application of preattentive processing towards visualization tasks and expanded on known preattentive features as well as identifying several others. The work has been furthered to lead to the creation of perceptual texture elements, or pexels [HBE95, HE98b], capable of representing the aforementioned visual features.

Continued research has produced the foundations of a comprehensive visualization framework. By developing feature guidelines, as well as investigating assisted navigation techniques, and involving an intelligent visualization abstraction assistant, ViA, the research has developed several mechanisms which can be implemented to aid perceptually based visualization creation and usage [DKS+05].

3.2 Scalability Studies

Graphical scalability work has largely consisted of resolving the technical constraints created by large and small scale display hardware and graphical rendering. However, as these are being improved, more usability and human factors work has come into play [CRM+06]. Unfortunately, much of this work involves very specific high level tasks rather than low level processing and does little to compare performance at common visual angles.

Both North [YHN07, BN05] and Tan et. al. [TGSP04, TGSP03] have performed a number of scalability experiments on large displays in which visual angle is kept constant or factored into the results. Typically task performance tends to improve, even in cases when the visual angle remains constant. This seems to indicate that factors in addition to visual angle affect perception of onscreen features. The problem is that in most cases, the visual search and navigation tasks are high level oriented, requiring users to analyze and identify complex abstract relationships rather than simple visual properties.

Also, where the size of the data set is concerned, many studies tend to focus on aiding users in the navigation of systems. These systems either streamline the transition between data sets or provide a balance of detail and overview [DH02, BGBS02], though these are user driven processes.

3.3 Context versus Detail Studies

Paramount to the management of large data sets is achieving a balance of context and detail. That is, optimizing a system to show as much information about a single element without losing the ability to understand that element as part of the greater data set. In the simplest form panning and zooming allow a user to analyze separate parts of a data set at differing levels of detail; however, this essentially charges users with maintaining context while zoomed through their own recollection.

A large variety of focus+context or overview+detail systems have been implemented which attempt to use system mechanics to moderate detail. Of these there exist distortion techniques, which modify a portion of the screen to display greater detail when focus is applied to a region or element. Examples of such a mechanism would be a simple magnification viewport or a fish-eye lens effect. In either case, the portion under focus by the user or agent is magnified into a certain area, while the surrounding area remains at the original scale. They key difference in a fish-eye lens implementation is that the distortion is not even throughout the area of focus. The greatest magnification is at the center of the focus, and trails off as distance from the center increases, much like a fish-eye camera lens [Fur86].

Another methodology to handle context and detail would be to use a tree or organizational structure to guide the focus of the system. In this way, data is organized into a logical structure, with associated information located in close proximity and connected by some logical mechanism. In a tree structure, multivariate data is contained as the child nodes of higher level parent nodes. As the user navigates through the nodes of the tree, the children and parents of the currently selected node are more prominently featured and given more detail than nodes far away in the organizational structure. The actual representation of these trees may take a more traditional representation of nodes above and below one another [RMC91] or perhaps be organized according to some more complex display [LRP95].

More specific examples of this methodology include the usage of cone trees. Here, the concept of a tree is extended into three dimensions by creating cones of child nodes within the structure. This conical arrangement has the benefit of providing depth as a means of signifying focus between elements on the same level while still maintaining a tree like visual construction [RMC91].

Hyperbolic geometry can be used to focus on areas based on euclidian spatial organizations. By mapping a euclidian space into hyperbolic space, it is possible to generate distortion effects much like a magnifying lens in the center of focus while also reducing size and detail in the periphery [LRP95]. Here, not only is the section of focus being improved, but the rest of the scene is organized at a reduced scales based on the current selection.

Tree maps also use the principle of modifying the area of focus and allocating the overview differently. In a tree map each node within a tree is allocated a certain amount of space based upon factors of signifigance or focus. Then any child nodes are allocated space from the parent's allotment [Shn92]. The size of the area provided to each node signifies the weight of a node and its children.

3.4 Research Deficiencies

While there is a fair amount of visualization research in large scale viewing environments, there is considerably less featuring small displays. Although the manufacturing of large displays has become much easier and more cost effective, this runs counter to a large movement in miniaturization and mobility. Cell phones and mobile computers have become ubiquitous in society today and other handheld devices and instruments can be found in all sorts of environments. These small computers can still be utilized for visualization tasks, however their mobility is often dependent on a small size, including the display. Consider also that many of the scalability tests are related only to specific tasks and less to specific visual features, so their findings may not hold for all cases of low level visual processing.

Navigation systems may at times dynamically alter the level of detail within an image. Typically very little perceptual concern is given to this process though. By and far these over view and context systems are calibrated to provide the best balance of detail and context for a particular data set and representation.

The Telescope prototype will attempt to fill in several of these research gaps. The key experimental factors to be examined with regards to features are visual angle and number of pixels. Also, the system will consider the perceptual environment of the viewer when manipulating feature abstractions according to scale. This will create a system which manages the overview and detail of the system dynamically. The user will not have to directly control the level of focus, only the distance from which they view, and the number of pixels allotted to each element. In cases where the viewing distance is far or the pixel allocation small, features will automatically reduce. Then as the conditions improve more detail will be added to the scene.

Chapter 4

Design

The design of the Telescope framework is intended to supplement the previous work [DKS+05] towards a generalized perceptual visualization framework by creating a foundation for a software implementation of concepts presented by a graphical feature hierarchy [SH07]. The development of the framework was intended to provide key points of functionality as well as serving as an easily extensible entry point for future projects and applications.

4.1 Basic Requirements

The design of the Telescope framework is engineered to provide:

1. The ability to read and store a standard data set and to allow for easy mapping of feature attributes to each element.

2. The ability to turn features on or off when an element's display resolution or visual angle crosses a defined threshold.

3. The ability to easily modify the system parameters for a variety of displays and to provide display and viewing position simulation.

4.1.1 Data Management and Feature Mapping

In creating research visualizations, many times a new system must be implemented for each task. Also, at times, feature mappings are determined and established during the development process. Therefore the data input types and feature mappings can many times be difficult to manipulate without redeveloping aspects of the program. To reduce the time and effort spent on these redevelopments, the Telescope framework provides some basic I/O functionality to ease inputting new data and parameters and to allow these settings to be stored for future use. This will prevent the necessity for multiple different application frameworks all of which provide the same basic operations.

The data management module has two main features, to read in and map data to on-screen data elements, and to read in and apply a simple set of rules. The data is contained within a file and is read and stored into an appropriate data structure. Along with this process, relevant information will be taken from the data such as the data field name as well as minimum and maximum values from which a normalization will be calculated when appropriate. The rules, however, require more care.

The intent of the rules file is to initialize the system to a specific attribute mapping and viewing setting. Each rule defines an input parameter and a paired set of values. For instance, the user may wish to define the width of the window to be 800 pixels or to specify a viewing distance 22 inches. This also applies for feature mappings.

The user should be able to enter the name of the feature as it is listed within the data file and map it equivalent to one or potentially more of the aforementioned graphical representations (see Figure 2.1). All of the dynamic parameters of the system which are not actual data should be available within the rules file. These rules include defining attribute mappings, viewport size and scale, viewing distance information, and both visual angle and pixel cutoff values utilized by the feature occlusion functionality. If a rule is not defined it will then default to an preset, typically an off value or general condition.

4.1.2 Feature Occlusion

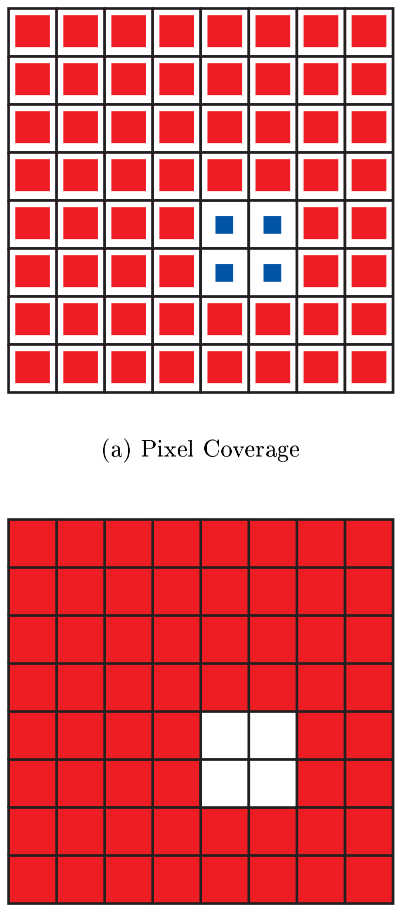
Imagine a scenario where a military commander is observing a fleet of ships at sea from a bluff. Looking out over the fleet, the commander can see the whole of the formations and general movements of the enemy ships. Unfortunately, at this distance he is unable to resolve the details of the various ships; their type, armaments, capabilities, etc. This sort of information is crucial to his plans to for a successful counterattack. He then extends a telescope to enhance his view of the distant vessels. Now, because of the magnified image presented to him, he can easily spot the various ships to determine their various capabilities and more adequately perceive the potential threat of each craft.

Now though, there is a conflict between focus and context. When the telescope is utilized, the commander can no longer see the whole of the battlefield, so while he can now discern the capabilities of several ships, he may miss important formation or movement trends of the fleet as a whole. The commanders' total field of view has essentially been narrowed in the magnification of a single area.

What the commander needs is adjustable telescope which will permit him to optimize his view such that he is able to see enough detail while retaining context and but not overwhelming him with more than he can handle at a given time. While this sort of metaphor might seem archaic, it is a fundamental representation of the motivation and goals of the feature occlusion functionality.

Take a data set with a set of values each with a corresponding feature mapping: temperature→hue, cloud cover→luminance, wind speed→size, etc. Displaying at a single pixel allotted to each element would possibly allow for a difference in hue or luminance to be observed, but regardless of the visual angle present to the viewer, one pixel alone is incapable of representing differences in orientation or size. Since pixels are discrete in size, a single pixel can be on or off, so a size representation would only vary if the size of the data element was sufficiently high enough to render the pixel on, or otherwise sufficiently low enough to render the pixel off. Likewise, orientation is created by activating different pixels to give a shape or outline with some defined orientation, as physical display pixels cannot themselves be rotated. Even after increasing to a 3×3 pixel element size, while the system may be able to render a few different sizes, variations in orientation will still be highly constrained.

This begs the question, if a data point is too small to allow a user to resolve the detail of a specific feature, why should this feature even be rendered by the system? In the best case the feature would do no harm to the visualization, however it is also possible that on top of being unable to resolve the feature, it could interfere with another visible feature. For example the color not being represented because the size of an element is not sufficiently large enough to raster into a lit pixel. This scenario is presented in Figure 4.1. Image



(a) Pixel Coverage

(b) Pixel Output

Figure 4.1: Size interferes with hue presentation.

(a) presents a grid of pixels within which size and color are represented, while image (b) represents the output. Here, the blue elements are not sufficiently large enough to render, interfering with the ability to notice a variation in color. A viewer might thus assume that size is the only varying factor at that position rather than both size and color. Therefore it would be both prudent and beneficial to remove this data mapping when it cannot contribute to the visualization. This is the key principle behind the feature occlusion functionality.

Recall the telescope metaphor from earlier. All of the information is available to the commander, however he cannot see portions of detail and therefore to aid he extends a telescope, allowing him to see a larger image and more detail. When he no longer needs this detail the telescope is contracted presenting him with less detail. This is conceptually the same idea used in the removal of features from the scene. Within the rules file the user can define thresholds for when features should be turned on or off based upon visual angle and the pixel allotments per element. In Figure 4.2, each item has a low number of pixels allocated to it. In setting the thresholds, luminance can be viewed with any pixel allotment, hue in medium or large allotments, and density in large allotments only. Therefore with a small number of pixels per element, only luminance, the lightness variation between elements, will be displayed by the system. In this instance, the telescope is compacted in order to save space. It does not provide any scaling assistance but the two features below the threshold are concealed within to prevent interference. As the device telescopes out, no longer is hue concealed within the device, since the defined threshold has been crossed, but it is now visible and there is a small amount of magnification. Figure 4.3 demonstrates that now both color and luminance are visible, though the number of elements possible to be seen have been reduced. Density remains concealed within the device, and therefore is not displayed to the viewer. Finally, if the telescope is extended out again, density is no longer concealed and the feature is turned on. An even greater pixel allotment is present and Figure 4.4 shows that each feature is represented with each portion of the telescope extended, providing a large amount of magnification with no features concealed.

The feature occlusion functionality tracks the current visual angle and pixel allotment given to each element and uses these to check against predefined thresholds to determine

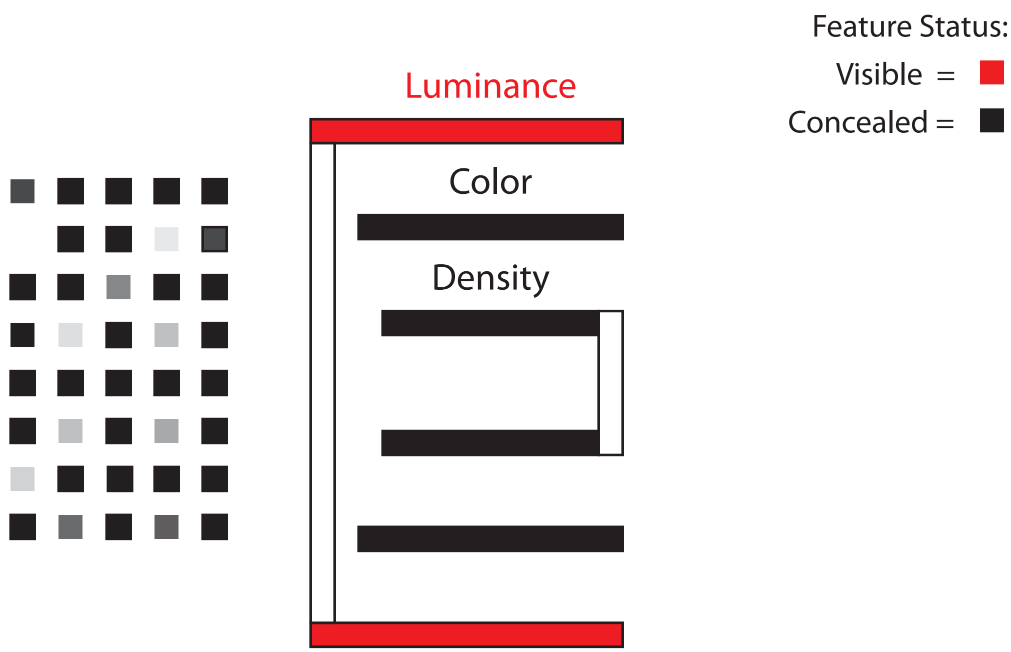


Figure 4.2: Small Target Size

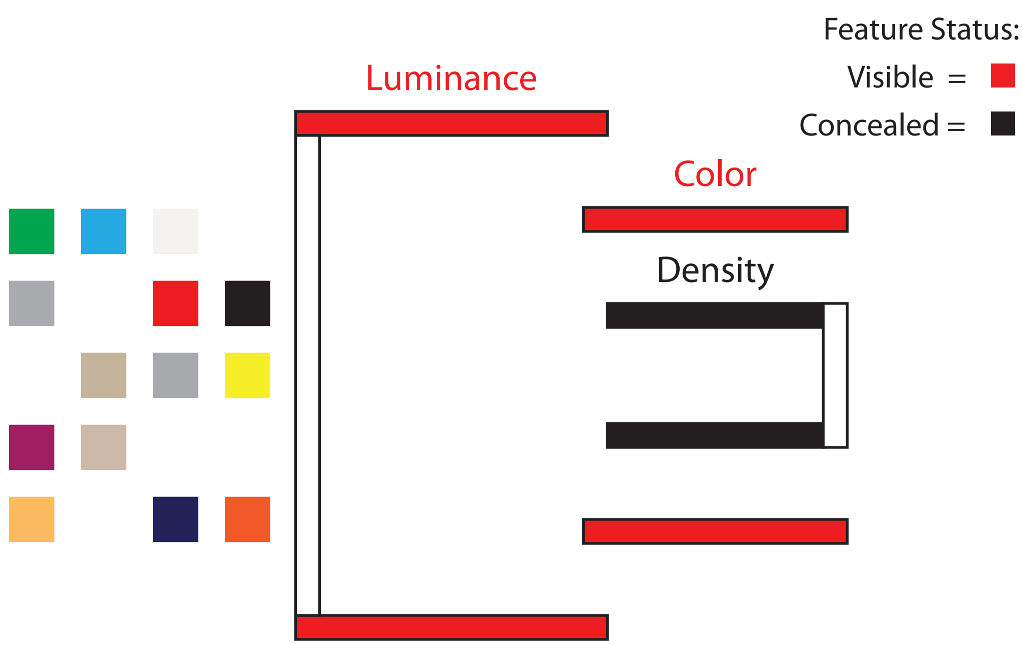


Figure 4.3: Medium Target Size

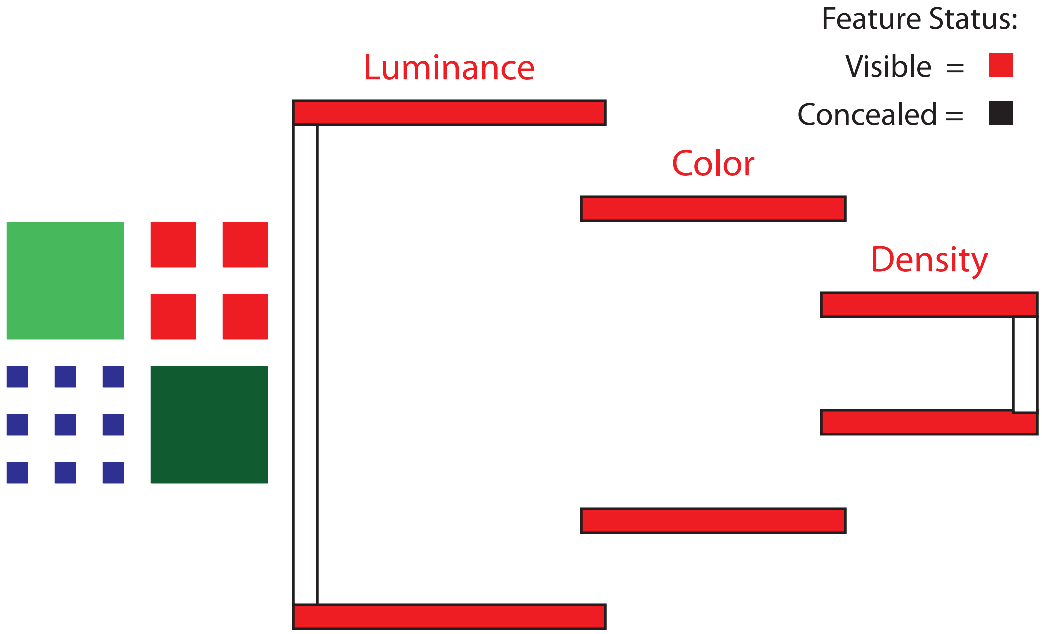


Figure 4.4: Large Target Size

if a feature can distinguished. In the case that visual angle or element resolution are not sufficient, then there exists a potential scenario for skewing other features and the features will thus be removed from the display range.

4.1.3 Simulating Display Environments

Once more consider the telescope metaphor. The commander wishes to keep his command center further from the enemy ships to reduce the chance of bombardment, however moving too far away will limit his ability to visualize the battle. Three sites are potentially viable but there is only time to move the equipment and men to one. What the commander needs is the ability see what his vantage point would be like from each site without actually having to move there.

Given the difficulty in obtaining and housing various display types and environments, it would be extremely useful to be able to simulate to a certain degree various display sizes. Given the constraints placed on facilities it may not always be practical to move a person far away or extremely close to a display. In order to do this the system allows for a simulated viewing distance to be defined in the rules file along with the actual view distance and size of elements. When a simulated viewing distance is defined the viewing angle from that distance is calculated and the screen is transformed to scale the window size and each element down to that visual angle.

The visual angle scaling presents the data elements at the size at which they would appear from another distance. The viewport size changes also because, if the viewport were to remain constant, while the data elements shrunk or grew, then a different number of data elements would be visible. This is useful because given spatial constraints, it may not always be practical to move the test subject a far enough distance away from the monitor and will permit testing different scenarios from a single workstation.

Chapter 5

Implementation

The implementation of the Telescope framework is primarily a straightforward software engineering challenge. Beyond the design phases, the implementation iterations all tread on fairly well established technical methodologies. The primary challenge is generating these concepts in an object oriented construct such that various elements can be included at ease within other applications. In the following sections, the key components of the system will be detailed, along with a general outline of the technical challenges and constraints faced during the course of construction.

5.1 Technical Background

The telescope framework is implemented utilizing the OpenGL graphics API in C++. The package primarily utilizes the C/C++ standard libraries. The operating system specific OpenGL graphics libraries are used for rendering the visualizations and are commonly found on most systems. Currently, a graphical user interface is constructed utilizing the GLUT utility package.

Data can currently only be input via one of two standard binary data formats. A bin file defines a set of multivariate data points and any applicable attribute mappings. In a grid file, a special case of the bin file, the special position of elements within the file are inherently defined by a regular grid structure of columns and rows. These data files also include information about the data set as well as existing attribute mappings, the acceptable value range for each element, and different frames of data each representing different subsets of the same format (eg. months within a years worth of data).

Development and testing were conducted primarily on a machine running Solaris based on the SPARC architecture with a 21 inch (19.8 viewable) flat CRT monitor at a resolution of 1280×1024 pixels via integrated graphics hardware. The overall development of the system is OS agnostic, however the encoding of the data files is dependent on architecture and therefore the system is not currently implemented for Windows or Mac based systems.

5.2 Organizational Structure

Four main components comprise the Telescope framework: the viewing information, a set of selective perceptual texture element glyphs that represent data elements (S-pexels), the data manager, and the graphics interface. Minimally, the basic feature occlusion and view distance simulation can be implemented by using only two modules, the viewing info and the selective pexels. The basic organization of the Framework is demonstrated in Figure 5.1.

5.2.1 Viewing Info

The viewing info acts a container for information regarding the viewers position, visual angle, display resolution, and virtual camera. Input into the viewing info are the dimensions of the display in pixels. Since display size is often represented by the diagonal measurement of the display, the display resolution width and height are used to calculate the diagonal pixel measurement of the display which then allows for a pixel per measurement metric. Therefore a 1600×1200 resolution, 19.8 inch viewable display has a 2000 pixel diagonal measurement and therefore $2000/19.8 = 101.01$ pixels per inch of screen space on the diagonal. For reference, resolving the width and height of the display to 15.84 inches and 11.88 inches respectively, there are $1600/15.84 = 101.01$ pixels per inch horizontally, and $1200/11.88 = 101.01$ pixels per inch vertically, providing the equivalent value. The

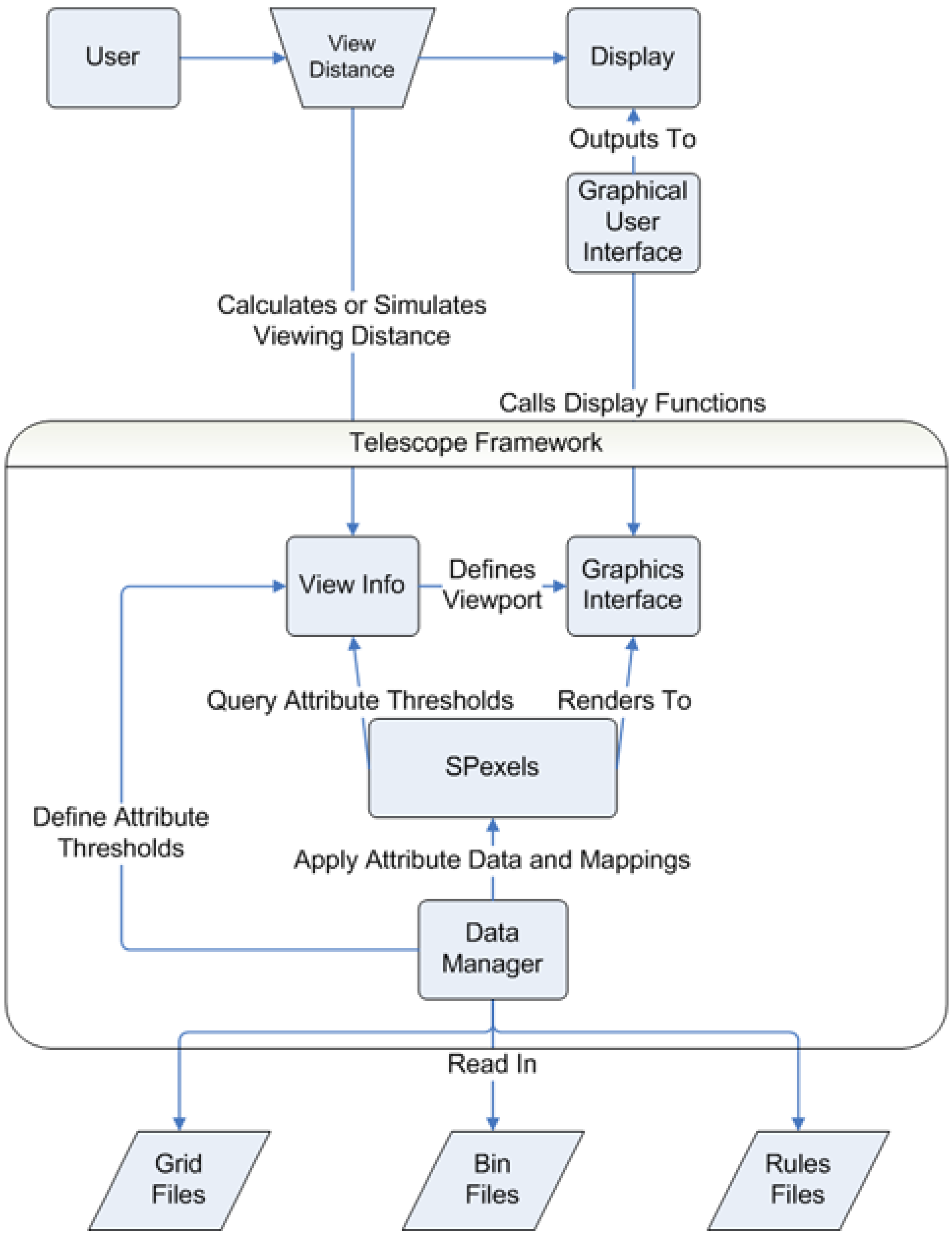


Figure 5.1: Generic Program Flow.

computation of this metric allows for the size of an element on screen to be determined given the number of pixels in either width or height. In an element with an aspect ratio of 1:1, width to height, this choice of measurement is insignificant, however should the aspect ratio differ, to retain a constant for experimentation and to maintain convention, the vertical size is taken. So, for an element with a height of 1 pixel, the resulting physical size is 1 divided by 101.01, or approximately .0099 inches.

Given that an element has a physical size measurement, the appropriate visual angle can be determined given the distance of the viewer from the display according to the previously defined formula. Therefore an element of 1 pixel, or .0099 inches, viewed from an distance of 22 inches would result in a visual angle of 0.02578 degrees.

Presented with this information the view info module makes several determinations. First, if the visual angle or pixel size falls below a predefined threshold for a given feature, it adjusts whether this feature is turned on or off. Secondly, in the event that the user wishes to simulate a viewing scenario based upon the current data, the view info will calculate the proportion between the viewing angle of the actual distance and the viewing angle of the simulated distance, applying this factor to viewport size and element size. Unfortunately this is not a perfect simulation, as the scaling factor may require a representation which does not result in an exact multiple of pixel size; that is to say, the boundaries of the elements may at this point not correspond exactly to pixel measurements and reside between pixels, resulting in an approximation by the rendering engine.

5.2.2 Selective Pexels

Pexels, or "perceptual texture elements", are graphical constructs which combine to form textures via the representation of one to many visual features. The pexels utilized for this framework are capable of representing the features previously mentioned and also listed in figure 2.1 [HE98a, Hea03]. Pexels are constructed to represent a single multivariate data element at a given position within the visualization.

The selective pexel extends the concept of a "perceptual texture element", by managing the rendering of the aforementioned visualization attributes dependent on number of pixels allocated to an element and angular size. The pexel is initialized based upon feature mappings and data values to render a single data point at a position on the screen. The reason this module is referred to as a selective pexel is because each pexel contains a reference to the system's view info module so that it can verify if a feature is to be drawn and either render it or not based on the predefined visual angle and pixel allotment cutoffs. As stated previously, the view info will update its own data based on the parameters it receives and the input of the users. Each S-pexel has a pointer to a view info object and can query it at each step along its rendering process to decide whether a feature is to be currently rendered.

While it is possible that each pexel could manage itself in this matter, given the fact that there may be large data sets involved, each pexel would have to update its information whenever a change was made, resulting in a frequent, computationally expensive action. This design approach also reduces the amount of data necessary to be stored within each pexel. When a feature is not to be represented, the S-pexel defaults to a base value. Default color may be defined to any color, while other features are typically either not represented or set to a 0 value(ie. no cycling for flicker, an upright orientation, no motion etc.).

The S-pexel glyphs are initially unit length, and are scaled dependent on the aspect ratio of an element and the current scaling factor contained in the view info object the S-pexel references. For example an S-pexel with an aspect ratio of 1:4 and a zoom factor of 3 will be 3 pixels wide and 12 tall. Also, each S-pexel may represent a size variation based upon data and attribute mappings, however the largest an element may become is limited by the aspect ratio and zoom factor. Orientation rotates the glyph around its center point, while regularity and motion/direction impact the positioning of a glyph within its drawing region. The drawing region does not define the physical size of the glyph but does specify an area in which the glyph may be expected to exist.

5.2.3 Data Management

The data manager serves as a utility module. It provides support for reading the data input files, as well as reading user defined rules. The data format is either a bin file or a grid file, both of which are simple binary data formats. From these files, a set of data values are assigned to each element. These values exist within some acceptably defined range from which a minimum and a maximum value are determined. Data falling outside of this acceptable range denotes the absence of value from data set.

The rules file allows the user to define attribute mappings as well as initial viewing parameters and various other settings. Rules can be used to represent data values paired with visual features, such as in the case of temperature $\rightarrow$ hue. A rules file may consist of few to many rules stored as plain text. This allows the users to manipulate settings quickly without the need for a separate application to generate program settings by merely entering in attribute and parameter pairs. The rule abstraction however is not limited to file I/O, though. There exists functionality within the data manager to accept a rule from any source that can provide the proper rule pairing. Although not implemented it should be possible to retrieve rules from sources such as databases or user input with ease.

The rules abstraction represents the key method by which the user can specify the attribute mappings and viewing parameters. Rules files give the advantage of allowing users to save multiple configurations in a small amount of space, easily modifiable by even novice users. Also, by maintaining this initialization there is no need for the users to change mappings or thresholds within the framework source.

When the data manager is initialized it is passed a data file and a set of rules files. The data file is read and each element is stored in memory. The view info is then initialized and the rules files are read. The display parameters are set within the view info module, while the attribute mappings are stored within the data manager as indexes of the location of the specific type of data within the set of elements. Should a rule conflict with a previously applied rule, such as two sets of data mapped to the same feature, the most recently read rule exerts dominance, changing what is necessary to apply its information.

5.2.4 Graphics Interface

The graphics interface is a set of function calls which implement OpenGL graphics rendering and viewing control mechanisms and which can be called via multiple user interface platforms. When initialized, the graphics interface allocated the S-pexels memory based on the data manager's stored values and attribute mappings. In instances where an element is lacking data for a particular mapping, the entire data element is discarded. This is due to the fact that to represent it could possibly result in a case of misinformation. Take the case of mapping temperature data and cloud cover to color and orientation respectively. Temperature data is available for all elements, however cloud cover data is not present over large bodies of water such as lakes. If both were to be represented, the elements representing areas of lakes and oceans could default their orientation to 0 degrees or mimic nearby elements, however since this is not the actual case, it would result in providing an inaccurate impression of actual events.

The graphics interface then has functions corresponding to the drawing of elements to the display, the reshaping of the viewport as well as common usability functions such as zooming and translation. The graphics interface also updates the S-pexels based upon the passage of time, at certain program state intervals.

5.3 Component Interaction

As previously mentioned, the framework can be effectively reduced to the functionality of only the view info and S-pexel modules. Each S-pexel has a reference to a view info object and performs a query for each stage of the rendering process. These checks determine whether a feature is to be mapped or ignored to a default value based upon display resolution and viewing angle.

Figure 6.1 details the flow of interaction between the components in a typical application. A program utilizing these modules would initialize a data manager, read the input data and apply the rules as necessary to the viewing info and attribute mappings. The graphics interface creates the S-pexels and will attempt to render each S-pexel when the functions are called by the user interface. However, before the S-pexels render a feature, they query their referenced view info module to determine whether or not they have a sufficient enough visual angle to represent that feature. If the user updates his viewing position and/or screen size, the display resolution and visual angle will be recalculated and with the view info updating against the defined cutoffs. The S-pexels will then adjust the feature mappings upon querying the view info on the next draw pass.

Minimally, the S-pexels could be initialized from some other data source and stored within a data structure. The view info would be initialized and retrieve data from whatever interface was being utilized. The interface would then call the draw event for each S-pexel. The graphics interface and data manager are both constructed to serve these tasks but are both tied to specific standards, the data input and the interface mechanisms respectively.

Chapter 6

Results

6.1 Feature Mappings and Scalings

The following figures use the Telescope framework to represent a set of meteorological data obtained in the United States. The data set contains various weather and atmospheric metrics such as temperature and precipitation for a set of regularly arranged collection points sampled at $\frac{1}{2}^{\circ}$ steps in latitude and longitude. In some cases, there may exist a value for one metric and not another at a given point. This is due to the fact that the data for the particular attribute was not collected for the given point, such as collecting information over a body of water, etc. In the event that a data element does not have a value for a mapped attribute, the entire element is by default omitted from the visualization. The rationale behind this is that in order to represent the other attribute mappings, some arbitrary or default value must be assigned to the missing attribute, potentially creating a situation that does not accurately reflect the actual data and its trends.

The display environment used for testing consisted of a 21 inch (19.8 inch viewable) diagonal CRT monitor running at a total resolution of 1280×1024. The visual angle is calculated from a viewing distance of 22 inches, a reasonable estimation of a possible viewing distance for a user operating a standard workstation computer. Given then these parameters, the display consists of approximately 82.788 pixels per inch. Therefore for one pixel is approximately, $1/82.788 = .0121$ inches in size resolving to a visual angle $.0315^{\circ}$ of at 22 inches from the display ,according to the previously defined formula.

The images in Figures 6.1-6.6 represent the mapping of the mean temperature for the month of January to 6 different visual features: hue, luminance, regularity, orientation, density, and size . Each data element has been allocated a square 2×2 region of pixels for the representation. In Figure 6.1, hue scales from blue through green to red hues as magnitude increases. For luminance in Figure 6.2, glyphs become lighter as magnitude increases. The patterns become more irregular as temperature increases for regularity in Figure 6.3. Orientation rotates from $0°$ to $90°$ for as temperature increases in Figure 6.4. Glyphs become more dense in as temperature escalates in Figure 6.5 and size increases for increases in temperature in Figure 6.6. Images have been scaled for ease of viewing.

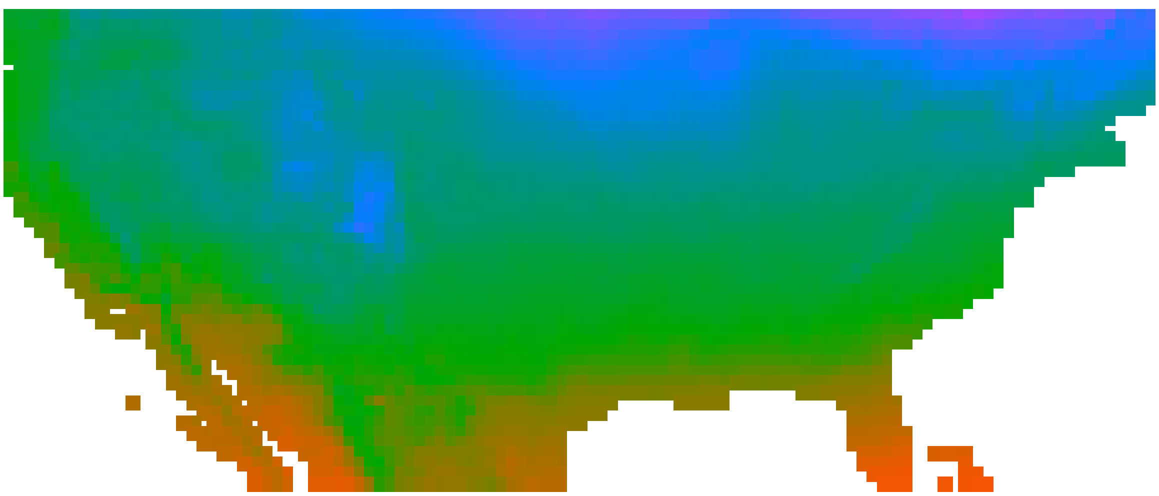


Figure 6.1: Mean Temperature for January Represented by Hue, 2×2 Pixels.

In small scales the color properties of hue and luminance tend to perform well. This is because very few pixels are actually needed to represent changes in the luminance and color especially when compared to features such as orientation and density. To represent a density range of 1 to 4 glyphs within the element area, at the very minimum 4 pixels are required (one for each glyph) and even then there will not be any space between glyphs to be able to discern that the element is in fact composed of separate glyphs. Compare this with

Figure 6.2: Mean Temperature for January Represented by Luminance, 2 × 2 Pixels.

Figure 6.3: Mean Temperature for January Represented by Regularity, 2 × 2 Pixels.

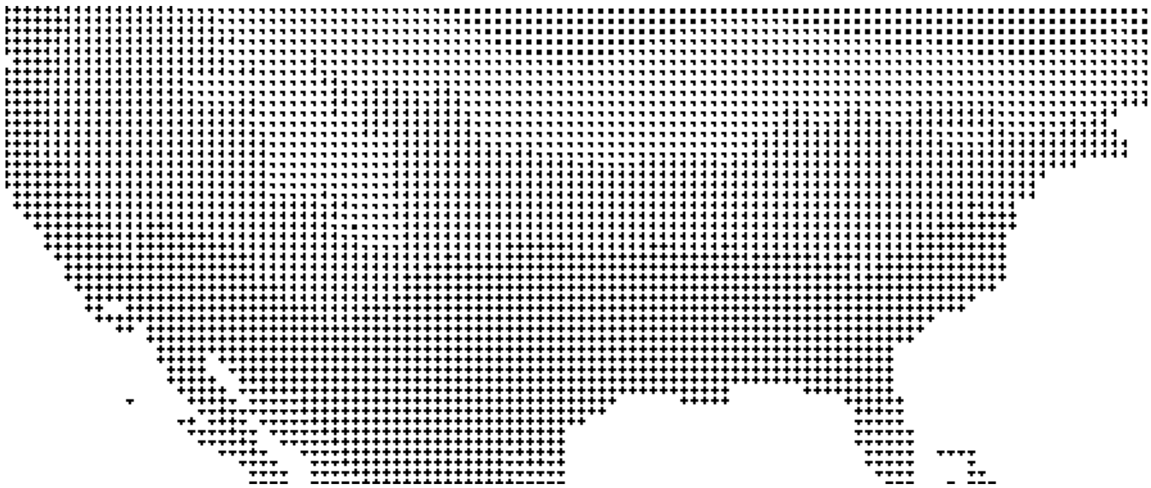


Figure 6.4: Mean Temperature for January Represented by Orientation, 2 × 2 Pixels.

Figure 6.5: Mean Temperature for January Represented by Density, 2 × 2 Pixels.

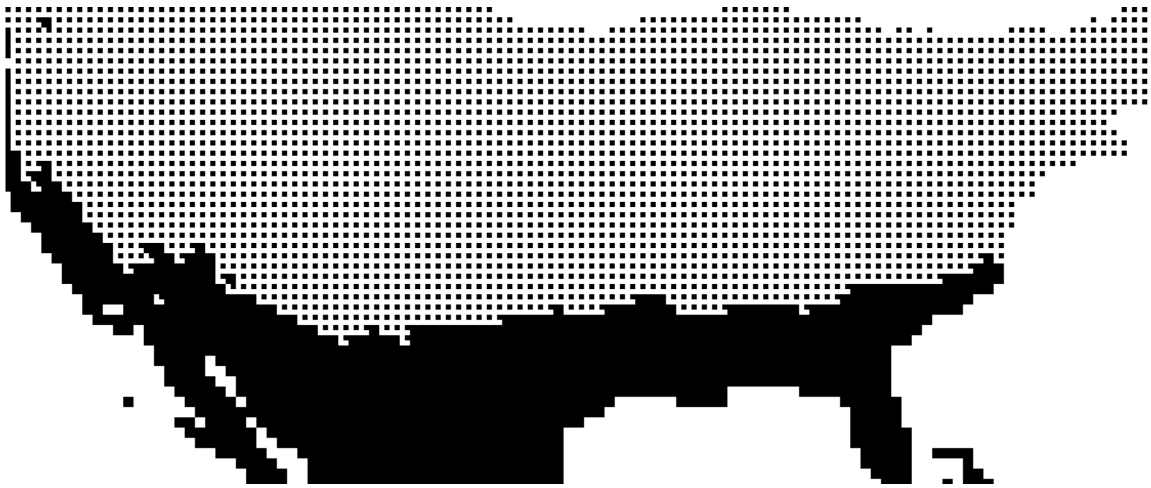


Figure 6.6: Mean Temperature for January Represented by Size, 2×2 Pixels.

the hue which can show a wide array of colors with only a single pixel and the advantage seems clear. While the sharp divisions in size and regularity regions might indicate a strong representation of data, the fact is that this is largely an artifact of having only a small range of variance which can actually be represented. By and large, these variations around these boundary regions are much less concrete and have more variation at a more gradual rate of change. In Figures 6.7-6.12 more pixels are alloted to each element in an 10×10 arrangement (64 per element). Here more differentiation can be made in features such as size, orientation, density, and regularity.

In this case, much better variations in size and orientation can be seen. Regularity also improves slightly. However, density is still difficult to discern exactly and may require more pixels given the viewing situation. As the amount of screen space alloted to each element increases, the number of elements which can be on screen at a given time decreases, however more detail can be resolved. In Figures 6.7 - 6.12 the increased pixel count has allowed for more variation between feature representations, but now the visible west coast data can not be compared with the east coast data.

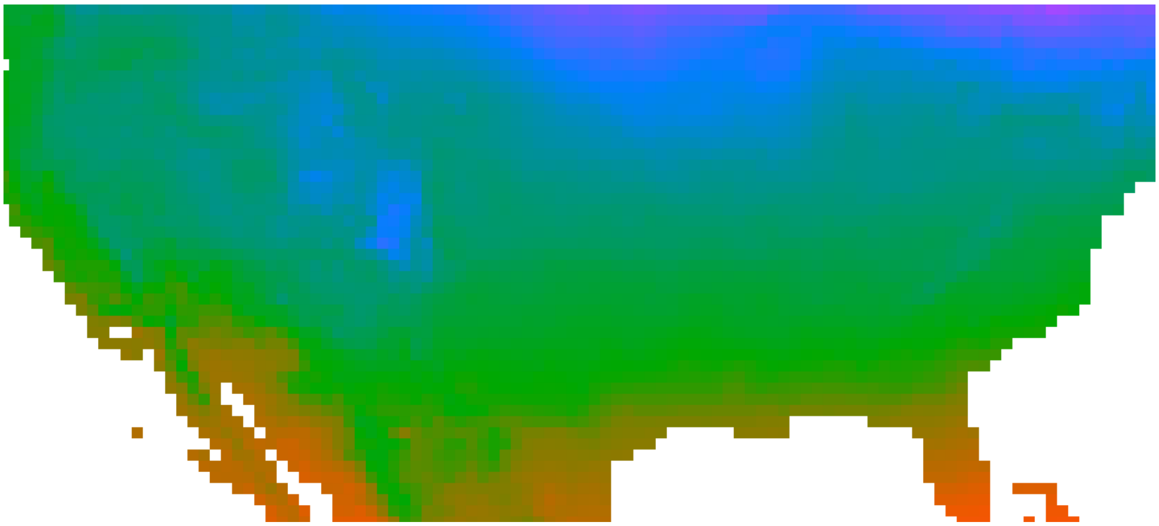


Figure 6.7: Mean Temperature for January Represented by Hue, 10×10 Pixels.

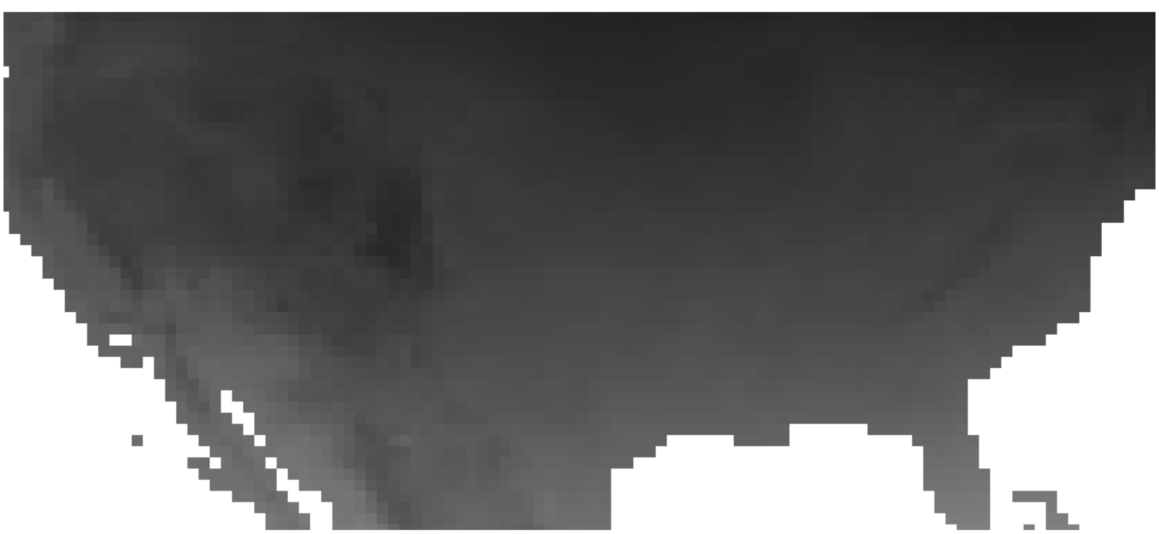


Figure 6.8: Mean Temperature for January Represented by Luminance, 10×10 Pixels.

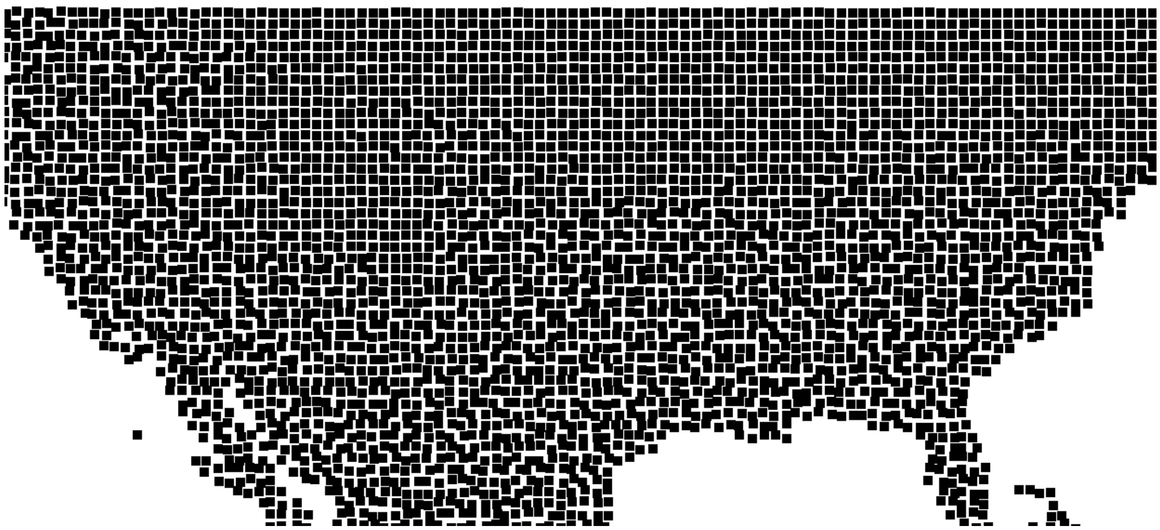


Figure 6.9: Mean Temperature for January Represented by Regularity, 10 × 10 Pixels.

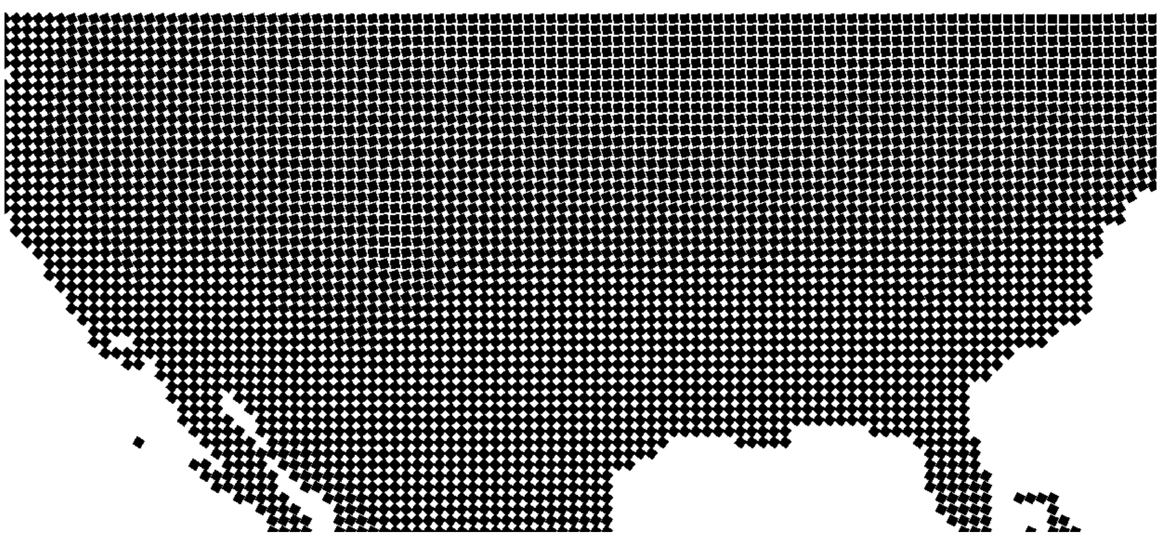


Figure 6.10: Mean Temperature for January March Represented by Orientation, 10 × 10 Pixels.

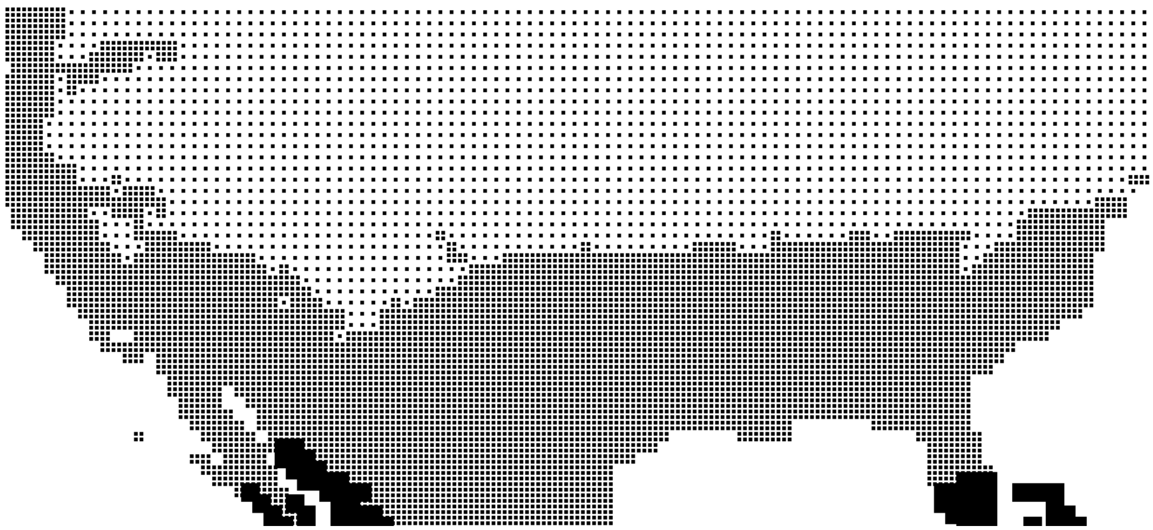


Figure 6.11: Mean Temperature for January Represented by Density, 10 × 10 Pixels.

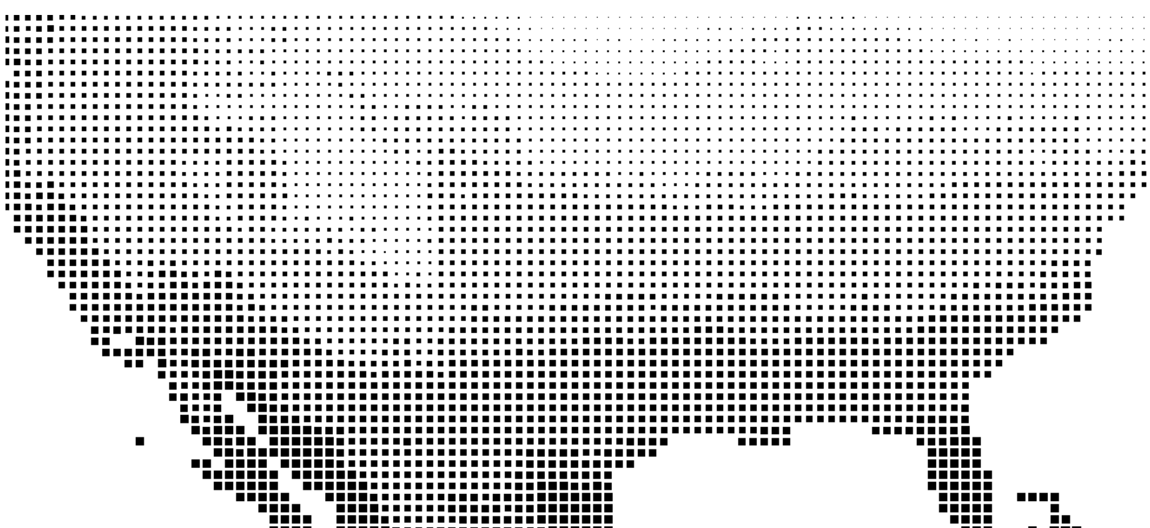


Figure 6.12: Mean Temperature for January Represented by Size, 10 × 10 Pixels.

Multiple data types can be represented at the same time, and with various mappings. Figure 6.13 represents all the following features at a 1×1 pixel allotment: temperature is mapped to color, cloud cover to luminance, vapor pressure to orientation, and wind speed to size.

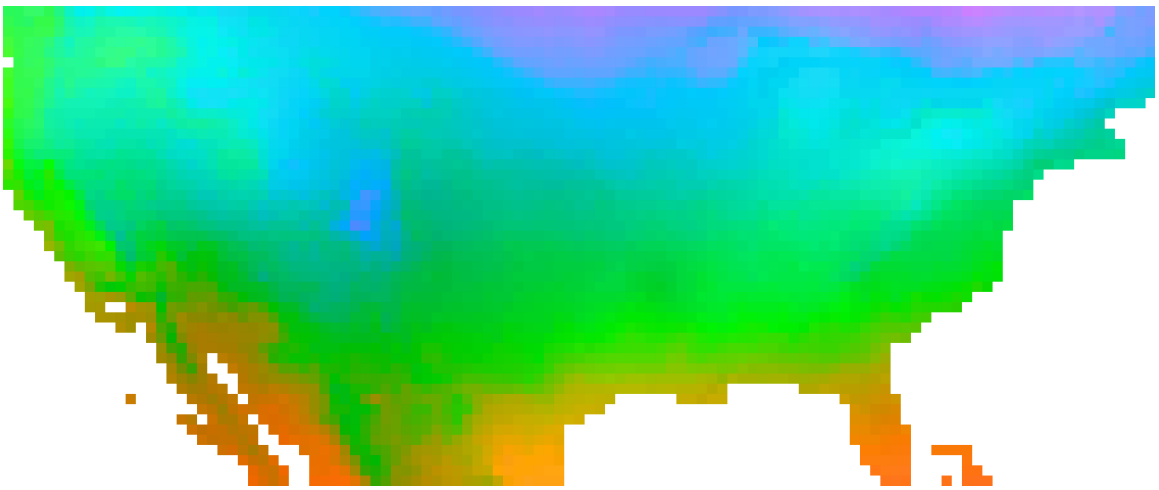


Figure 6.13: 4 way mapping, 1×1 Pixels.

When examining the image, hue and luminance can be viewed easily, however the other features suffer. Luminance is present and distinguishable, thought at times it can be challenging to detect precise boundaries with hue. With one pixel the only variation in size which can be determined is the presence or absence which would indicate a very low value. As mentioned earlier, if the size, orientation, or positioning of the element does not sufficiently cover the pixel, it will not be rendered. At 1×1, no orientation can be represented either as individual pixels do not inherently possess orientations.

Table 6.1: Feature Thresholds

Name	Display Resolution	Visual angle
Hue	4×4	.2480
Luminance	1×1	.1265
Orientation	4×4	.1265
Size	8×8	.1265

To resolve these issues, Telescope will implement the feature hierarchy guidelines which are presented in Table 6.1, to dynamically modify the images as display resolution changes, with thresholds of 1×1 pixels, 4×4 pixels, and 8×8. Figure 6.14 begins with a 2×2 representation of luminance. Once the pixel allotment reaches 4×4 in Figure 6.15, Telescope checks against a defined value and determines that hue and orientation can be rendered. Scaling further, Figure 6.16 represents the activation of size at 8×8 pixels.

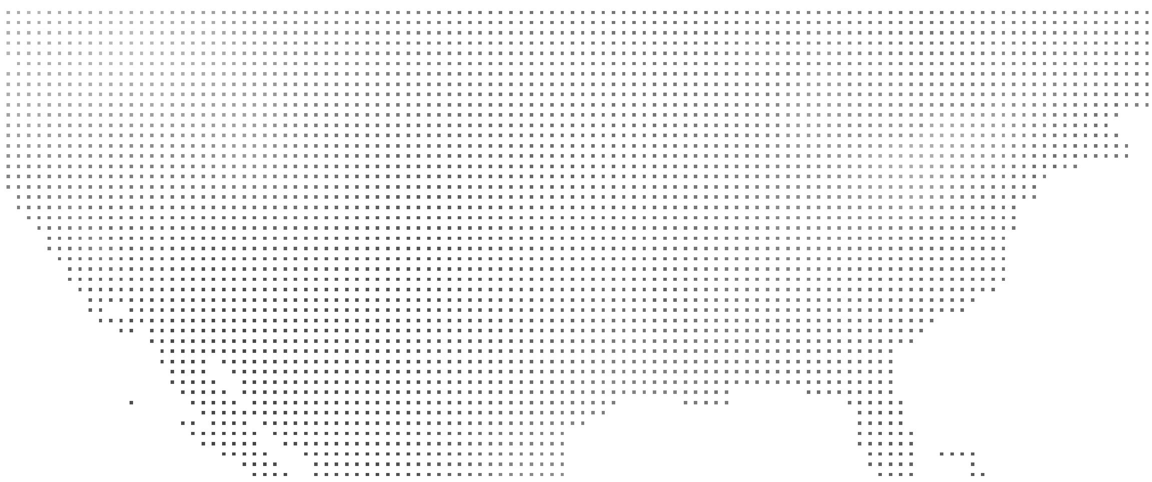


Figure 6.14: Luminance, 1×1 Pixels.

By occluding features at given scales, it is possible to prevent potential interference that may be caused by variations in one feature towards another which are magnified at small scale. Also, data elements which can provide only small variations in quality at low levels can be removed. Since size has only a small number of variations, it will not be included until a time such that more pixels can be used towards size representation. As the scale increases, these features are displayed at points which they will be more effective in Figures 6.15 and 6.16.

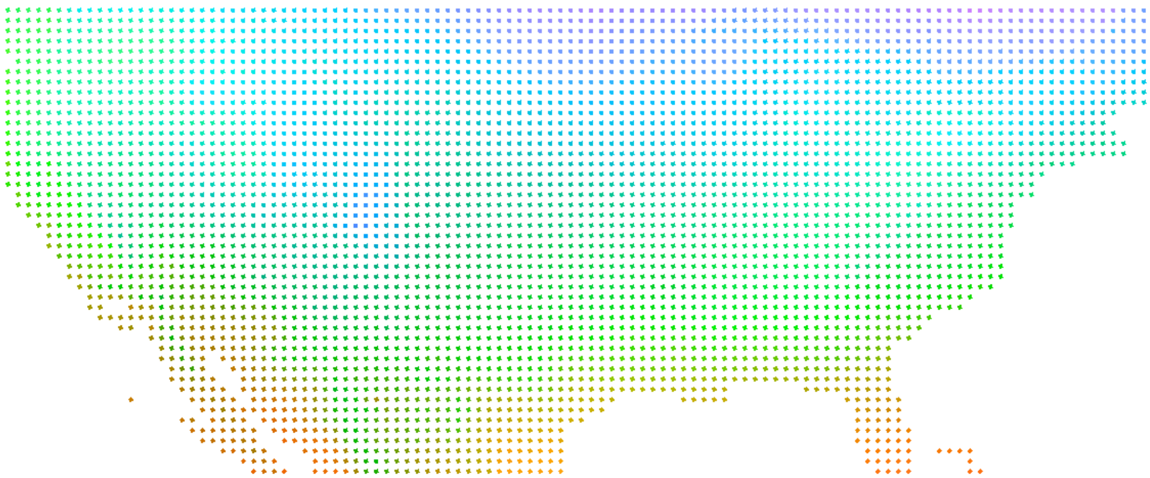


Figure 6.15: Hue, Luminance, and Size, 4×4 Pixels.

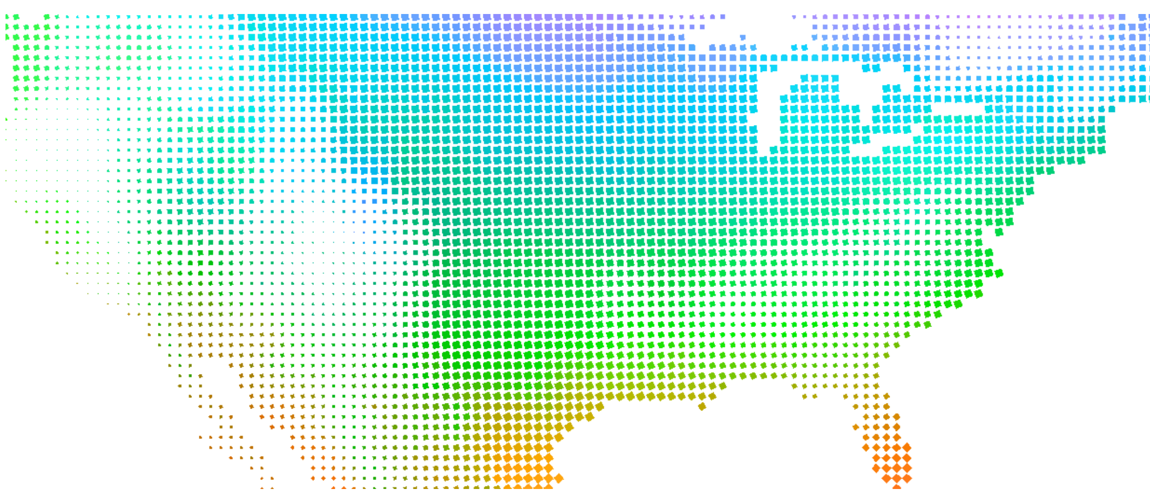


Figure 6.16: Hue, Luminance, Size, and Orientation, 8×8 Pixels.

As stated previously the ability to represent different element sizes to simulate various view distances and angles may be beneficial in certain scenarios. Visual angle is greatly influential on the perception of different visual cues. Current ongoing research has been used to develop experimental thresholds of visual angle which serve as working metrics for the size at which a cue cannot be consistently discerned. Figure 6.17 displays the four-way mapping as previously mentioned. The image is presented with the visual angle calculated from 22 inches away. Rather than moving a viewer physically a larger distance from the display, Telescope presents an altered image in order to provide a similar visual angle. Figure 6.18 simulates the same data as in 6.17, maintaining equivalent pixel allocation from a distance of 56 inches and Figure 6.19 moves the viewing distance even farther back to 113 inches. In both these cases, Telescope removes data from the display when the visual angle crosses the acuity threshold.

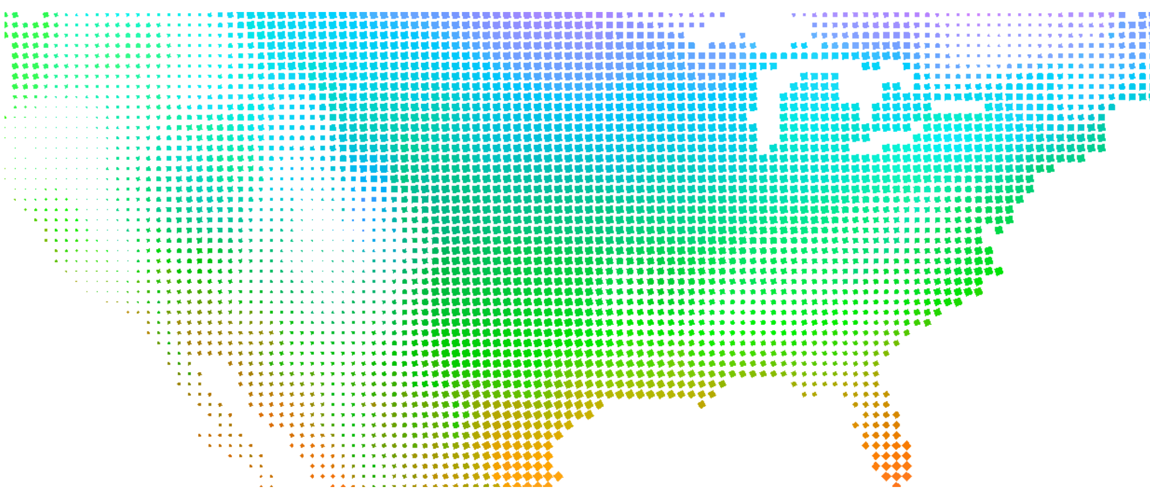


Figure 6.17: 4 way mapping from 22 inches.

The amount of space captured within each figure has also been scaled appropriately. In all cases, the data represented is the same. However for Figures 6.18 and 6.19, the angular size of each element has been scaled down due to the resulting simulated movement of the user backwards several feet. As such the viewport size also reduces. This provides an approximation of the viewing conditions from that position. The reason why this is only

Figure 6.18: Data from 56 inches.

Figure 6.19: Data from 113 inches.

an approximation is because pixels are discreet and therefore scaling the image by anything but a whole number will result in portions of elements falling between pixels.

In Figure 6.18, hue has been removed, leaving orientation, size and luminance. This seams counter intuitive, however the acuity research collected within laboratory studies has shown that performance of hue performance has degraded more than orientation, size, and luminance with respect to visual angle. While hue still possesses large degrees of variability at small scale, the actual visual size presents problems. In Figure 6.19, the orientation and size aspects have been removed. The resulting image presents luminance the only key factor. While, all three elements suffer performance at this angle, luminance is maintained arbitrarily so that some data can still be observed. Large regions can still be determined but smaller more detailed analysis are difficult.

Given another set of data, assume that a user was viewing a map of the world analyzing cloud cover. He or she might be interested in correlating the data with the average temperatures of the area. The user utilizes Telescope to provide a clear overview of cloud cover at a high level as in Figure 6.20 where cloud cover is mapped to luminance.

Here at a 1×1 bright areas denote a significant cloud cover. The user then notes

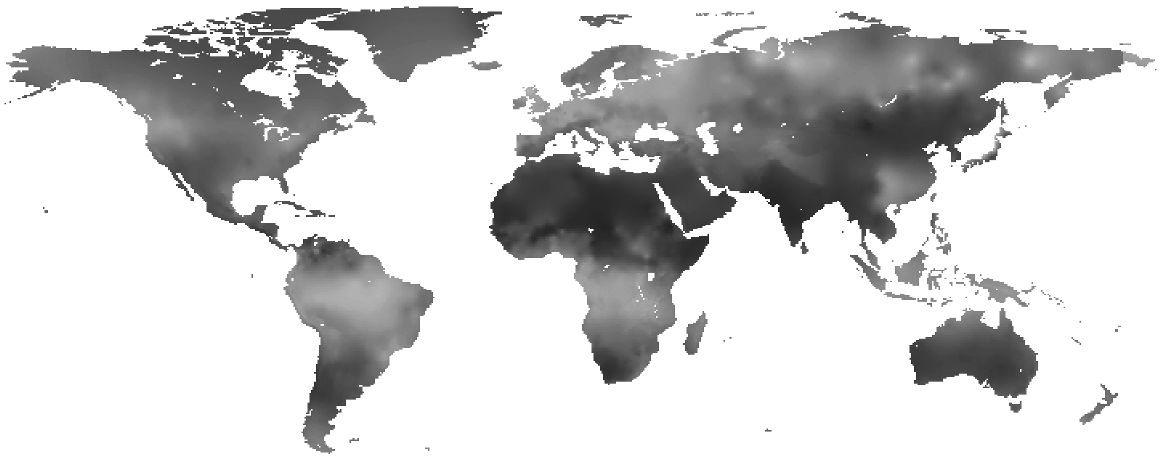


Figure 6.20: Cloud Cover mapped to Luminance.

that Europe has an interesting amount of cloud cover. Zooming in, Telescope turns on other features which had been absent. Experimentation had shown that hue and orientation were affected for viewing angles less than around .1265 and .2480 respectively, therefore the system threshold was set to only turn these features on when exceeding the appropriate visual angle. Now the user can analyze temperature and vapor pressure over Europe, in Figure 6.21.

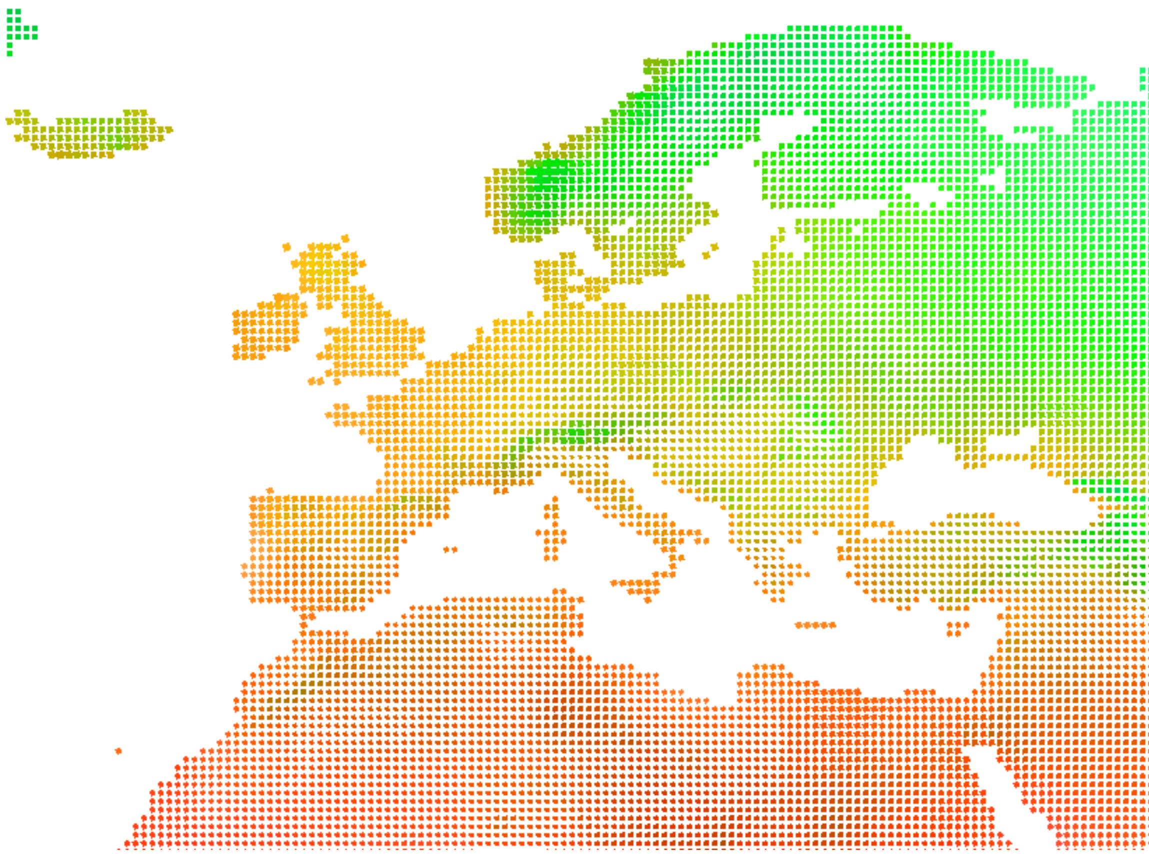


Figure 6.21: Four Way Mapping of Europe.

Chapter 7

Conclusion

7.1 Review

Visualization is dependent on vision. There is no escaping this fact. Therefore, by better understanding vision and perceptual processes, it may be possible to create more visually effective graphic data representations. Previous research in psychology and human-computer interaction has done a great deal to determine what makes an effective visualization and how the brain processes the imagery presented to it. However, as the means of creating and displaying these visualizations change, it would be naive and contrary to the evidence to assume that the perception of these systems would be unaffected.

Given the fact that visual acuity is dependent on the scale of objects and viewing distances, and the fact that display environments are becoming increasingly diverse and more cluttered, scalability becomes a concern for visualization engineers. By understanding how different visual features perform under different circumstances, it may be possible to develop a visualization hierarchy which will help define which features take precedence for important data given a particular viewing situation and how these attribute mappings should be modified given the ability to resolve their details at various scales.

To facilitate research into user performance across the aforementioned visual features, the Telescope framework provides functionality to quickly and easily output visualizations with adjustable feature mappings. By not rendering features which fall below visual angle or pixel allotment thresholds, clutter and interference effects can be reduced to allow features which can be observed at a given distance to be seen more clearly. Also, by providing a means to simulate visual angle, research will not be encumbered with the task of establishing display environments which may not be conducive to the facilities and equipment available. The system which has presented accomplishes these tasks, providing a means of developing dynamic visualizations.

7.2 Resulting Context and Future Work

The Telescope Framework is fairly straightforward in implementation but offers some strong functionality. Applications utilizing this work should be able to integrate its features by use of some or all modules of the package. This streamlines the process of creating a visualization system and provides several mechanisms for testing and development based around element size. Telescope is easily integrated into other applications, requiring only a small number of function calls from external applications and acts in a largely self contained manner. The rules mechanisms provides intuitive means of setting parameters and also a means for integrating other tools, such as intelligent agents.

The ability to compensate for widely varying data sets is not without its challenges though. The construction of the system assumes data will be more or less regularly distributed such that normalized values stretch across a large span of scales and representations. However, practically speaking data sets will many times be skewed and irregular in nature. Currently there is little in the way of compensating for this so there will be times when it is difficult to discern variations between elements because of a close grouping of values. The only way to manipulate the feature scaling is to set a base value for each feature defining its minimum or default value and maximum cutoffs. This leads to large amounts of calibration on the part of the user towards achieving the desirable output ranges and amount of feature contrast.

Also, the system needs further optimization to improve its memory and processor usage. When dealing with very large data sets, the system experiences lag in navigation. This is alleviated by the implementation of view frustum culling but the system is still fairly taxing on less than optimal systems.

The future of this framework will hopefully fulfill the plan to develop a robust visualization environment [DKS+05]. The system could potentially be integrated with ViA, an intelligent visualization agent developed and used within the research laboratory. ViA provides suggestions on appropriate attribute mappings for visualization environments. The system could have an interface which would be able to generate its own rules files based on ViA guidelines.

Currently, the system is only set to implement 2D representations. Since the pexel implementation is not limited to two dimensions, future work may extend functionality to 3D representations. This will complicate determinations of visual angle and pixel allocation, as the view switches from orthographic to perspective.

The dynamic feature occlusion could potentially be expanded from merely an on off switch to a dynamic mapping to a different available feature. This would allow for the use of a highly salient feature at low scales and less salient features at higher scales.

Ultimately, this system should aide current and future research and provide a basis for a system implementation of a graphical feature hierarchy. This feature heirarchy will provide a high level of dynamic visualization as well as aiding both visualization design and user interaction. The goal will be to further optimize visualization systems given perceptual concerns.

Bibliography

[BBSN07] John Booker, Timothy Buennemeyer, Andrew Sabri, and Chris North. High-resolution displays enhancing geo-temporal data visualizations. In *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*, pages 443–448, New York, NY, USA, 2007. ACM.

[BEJV95] Judith R. Brown, Rae Earnshaw, Mikael Jern, and John Vince. *Visualization: Using Computer Graphics to Explore Data and Present Information*, pages 1–10;125–147. John Wiley and Sons, 1st edition, 1995.

[BGBS02] Patrick Baudisch, Nathaniel Good, Victoria Bellotti, and Pamela Schraedley. Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 259–266, New York, NY, USA, 2002. ACM.

[BN05] Robert Ball and Chris North. Effects of tiled high-resolution display on basic visualization and navigation tasks. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1196–1199, New York, NY, USA, 2005. ACM.

[Che05] C. Chen. Top 10 unsolved information visualization problems. *Computer Graphics and Applications, IEEE*, 25(4):12–16, 2005.

[CRM+06] Mary Czerwinski, George Robertson, Brian Meyers, Greg Smith, Daniel Robbins, and Desney Tan. Large display research overview. In *CHI '06: CHI '06* *extended abstracts on Human factors in computing systems*, pages 69–74, New York, NY, USA, 2006. ACM.

[CW00] Aaron C. Clark and Eric N. Wiebe. Color principles - hue, saturation, and value. `www.ncsu.edu/scivis/lessons/colormodels/color_models2.html`, 5.8 2000.

[CWE04] Stanley Coren, Lawrence M. Ward, and James T. Enns. *Sensation and Perception*. John Wiley and Sons, 2004.

[DEB+07] Matthias Deller, Achim Ebert, Michael Bender, Stefan Agne, and Henning Barthel. Preattentive visualization of information relevance. In *HCM '07: Proceedings of the international workshop on Human-centered multimedia*, pages 47–56, New York, NY, USA, 2007. ACM.

[DH02] Brent M. Dennis and Christopher G. Healey. Assisted navigation for large information spaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 419–426, Washington, DC, USA, 2002. IEEE Computer Society.

[DKS+05] Brent Dennis, Sarat Kocherlakota, Amit Sawant, Laura Tateosian, and Christopher G. Healey. Designing a visualization framework for multidimensional data. *IEEE Comput.Graph.Appl.*, 25(6):10–15, 2005.

[Fur86] G. W. Furnas. Generalized fisheye views. In *CHI '86: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 16–23, New York, NY, USA, 1986. ACM.

[GEC98] Nahum Gershon, Stephen G. Eick, and Stuart Card. Information visualization. *interactions*, 5(2):9–15, 1998.

[HBE92] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Harnessing preattentive processes for multivariate data visualization. Technical report, 30, 1992.

[HBE95] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. Visualizing realtime multivariate data using preattentive processing. *ACM Transactions on Modeling and Computer Simulation*, 5(3):190–221, 1995.

[HBE96] Christopher G. Healey, Kellogg S. Booth, and James T. Enns. High-speed visual estimation using preattentive processing. *ACM Trans.Comput.-Hum.Interact.*, 3(2):107–135, 1996.

[HE98a] Christopher G. Healey and James T. Enns. Building perceptual textures to visualize multidimensional datasets. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization '98*, pages 111–118, 1998.

[HE98b] Christopher G. Healey and James T. Enns. Building perceptual textures to visualize multidimensional datasets. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 111–118, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[HE99] Christopher G. Healey and James T. Enns. Large datasets at a glance: Combining textures and colors in scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(2):145–167, 1999.

[Hea96] Christopher G. Healey. Choosing effective colours for data visualization. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 263–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.

[Hea99] Christopher G. Healey. Fundamental issues of visual perception for effective image generation. In *SIGGRAPH 99*, Course 6: Fundamental Issues of Visual Perception for Effective Image Generation, pages 1–42, 1999.

[Hea03] Christopher G. Healey. Pexels: Visualization with perceptual textures. `http://www.csc.ncsu.edu/faculty/healey/HTML_papers/pexel/pexel.html`, 2003.

[Hea07] Christopher G. Healey. Perception in visualization. `http://www.csc.ncsu.edu/faculty/healey/PP/`, 1.27 2007.

[HGBE00] Henry W. Hofstetter, John R. Griffin, Morris S. Berman, and Ronald W. Everson. *Dictionary of Visual Science and Related Medical Terms*, pages 7–8. Butterworth Heinemann, Boston, 5th edition, 2000.

[Joh04] Chris Johnson. Top scientific visualization research problems. *IEEE Comput.Graph.Appl.*, 24(4):13–17, 2004.

[Kai05] Peter K. Kaiser. *The Joy of Visual Perception: A Web Book.* 2005.

[Lab07] Scientific Visualization Laboratory. Scientific visualization tutorials. `http://www.cc.gatech.edu/scivis/tutorial/tutorial.html`, 2007.

[LRP95] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[PGB07] Nilma Perera, Albert Goodman, and Kathy Blashki. Preattentive processing: using low-level vision psychology to encode information in visualisations. In *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, pages 1090–1091, New York, NY, USA, 2007. ACM.

[RLMJ05] Ruth Rosenholtz, Yuanzhen Li, Jonathan Mansfield, and Zhenlan Jin. Feature congestion: a measure of display clutter. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 761–770, New York, NY, USA, 2005. ACM.

[RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: animated 3d visualizations of hierarchical information. In *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 189–194, New York, NY, USA, 1991. ACM.

[RTM+03] Theresa-Marie Rhyne, Melanie Tory, Tamara Munzner, Matt Ward, Chris Johnson, and David H. Laidlaw. Information and scientific visualization: Separate but equal or happy together at last. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 115, Washington, DC, USA, 2003. IEEE Computer Society.

[SH07] Amit Prakash Sawant and Christopher G. Healey. Need for perceptual display hierarchies in visualization. *Crossroads*, 13(3):6–6, 2007.

[Shn92] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans.Graph.*, 11(1):92–99, 1992.

[Sim01] T. Simmons. Whats the optimum computer display size? *Ergonomics in Design*, 9(4):19–24, 2001.

[TGSP03] Desney S. Tan, Darren Gergle, Peter Scupelli, and Randy Pausch. With similar visual angles, larger displays improve spatial performance. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 217–224, New York, NY, USA, 2003. ACM.

[TGSP04] Desney S. Tan, Darren Gergle, Peter G. Scupelli, and Randy Pausch. Physically large displays improve path integration in 3d virtual navigation tasks. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 439–446, New York, NY, USA, 2004. ACM.

[Tre86] Anne Treisman. Preattentive processing in vision. In *Papers from the second workshop Vol. 13 on Human and Machine Vision II*, pages 313–334, San Diego, CA, USA, 1986. Academic Press Professional, Inc.

[UIM+03] Timothy Urness, Victoria Interrante, Ivan Marusic, Ellen Longmire, and Bharathram Ganapathisubramani. Effectively visualizing multi-valued flow data using color and texture. In *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, page 16, Washington, DC, USA, 2003. IEEE Computer Society.

[War04] Colin Ware. *Information Visualization: Perception for Design*. Morgan Kaufmann, 2004.

[Wei04] Daniel Weiskopf. On the role of color in the perception of motion in animated visualizations. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 305–312, Washington, DC, USA, 2004. IEEE Computer Society.

[Wun04] Burkhard Wunsche. A survey, classification and analysis of perceptual concepts and their application for the effective visualisation of complex information. In *APVis '04: Proceedings of the 2004 Australasian symposium on Information Visualisation*, pages 17–24, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[YC06] Hee Yong Yoo and Suh Hyun Cheon. Visualization by information type on mobile device. In *APVIS '06: Proceedings of the Asia Pacific symposium on Information visualisation*, pages 143–146, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[YHN07] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond visual acuity: the perceptual scalability of information visualizations for large displays. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 101–110, New York, NY, USA, 2007. ACM.