Constant Density Visualizations of Non-uniform Distributions of Data

*Allison Woodruff *, James Landay, Michael Stonebraker*
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720 USA
E-mail: {woodruff,landay,mike}@cs.berkeley.edu

* Author's current address: Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304 USA

ABSTRACT

The cartographic Principle of Constant Information Density suggests that the amount of information in an interactive visualization should remain constant as the user pans and zooms. In previous work, we presented a system, VIDA (Visual Information Density Adjuster), which helps users manually construct applications in which overall display density remains constant. In the context of semantic zoom systems, this approach ensures uniformity in the *z* dimension, but does not extend naturally to ensuring uniformity in the *x* and *y* dimensions.

In this paper, we present a new approach that automatically creates displays that are uniform in the *x*, *y*, and *z* dimensions. In the new system, users express constraints about visual representations that should appear in the display. The system applies these constraints to subdivisions of the display such that each subdivision meets a target density value.

We have implemented our technique in the DataSplash/VIDA database visualization environment. We describe our algorithm, implementation, and the advantages and disadvantages of our approach.

KEYWORDS: Clutter, constant information density, multiscale interfaces, non-uniform distributions, visualization.

1. INTRODUCTION

Clutter and sparsity commonly occur in visualizations of data. They are undesirable for a number of reasons. For example, clutter can result in overplotting, in which certain objects are not visible because they are occluded by other objects. Sparsity can result in inefficient use of the available display space.

Many visualization systems address these problems by supporting multiple graphical representations of objects. The user is provided with an interactive visualization, and a representation with appropriate visual complexity for the current view appears in the display.

More concretely, many such systems use a spatial metaphor in which objects are displayed in a two-dimensional canvas over which the user can pan and zoom. The user's distance from the canvas is known as their *elevation*. Note that when the user views an object from a high elevation, it may appear quite small. However, when they zoom in on it, it gets larger, occupying a larger percentage of the display. As the object gets larger, it is appropriate to make its graphical representation more complex. Therefore, different graphical representations are appropriate for different elevations. For example, suppose a city is represented as a dot when the user is at a high elevation. When the user zooms in on the city, a text label or a city map may be added to the representation.

In systems that use this spatial metaphor, a common technique is to change automatically the representation of objects as the user zooms. This functionality is described as *semantic zoom*, *e.g.*, [7,4,17]. Programming the behavior of objects during zooming is a challenging task. As a result, objects in semantic zoom applications may have inappropriate visual complexity. Additionally, the amount of information in the display can vary significantly as the user pans and zooms.

We have identified a cartographic precept that defines appropriate transition points between different graphical representations. The Principle of Constant Information Density states that the number of objects per area should remain constant. More generally, the amount of information should remain constant as the user pans and zooms above a display [20,9]. We apply this principle to interactive visualizations of cartographic and non-

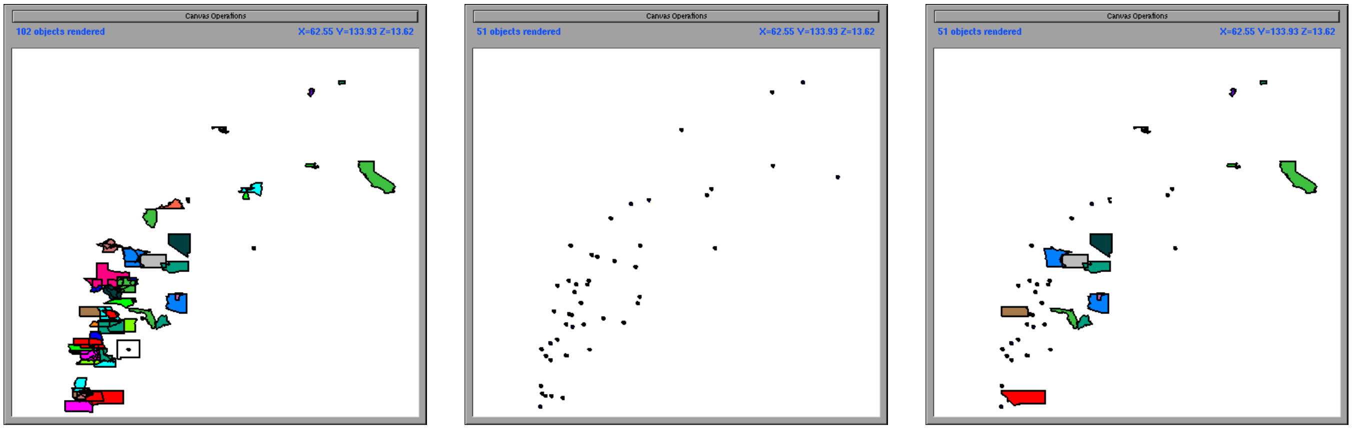

Figure 1a: DataSplash visualization. Figure 1b: VIDA_0 visualization. Figure 1c: VIDA visualization.

Figure 1: A visualization of states in the United States, displayed according to housing cost (*x* axis) and income (*y* axis). In Figure 1a, the visualization is obviously cluttered in many regions. In Figure 1b, objects are shown with a less detailed representation, as recommended by VIDA_0. Note that within Figures 1a and 1b, all objects are displayed with a single graphical representation. In Figure 1c, VIDA displays dots for objects that appear in dense regions; objects in less dense regions are displayed as polygonal outlines.

cartographic data.

In previous work, we presented a system, VIDA (Visual Information Density Adjuster), which helps users manually construct applications in which overall display density remains constant [26]. However, although the total amount of information in the display remains constant as the user zooms, the information within the display may not be distributed uniformly. In other words, this approach ensures uniformity in the *z* dimension, but does not ensure uniformity in the *x* and *y* dimensions, nor does it extend naturally to providing such uniformity. Because many naturally occurring data sets have non-uniform distributions, clutter and sparsity commonly occur in subdivisions of VIDA visualizations.

In this paper, we present a new version of VIDA that automatically creates displays that are uniform in the *x*, *y*, and *z* dimensions. In the new system, users express constraints about visual representations that should appear in the display. The system applies these constraints to subdivisions of the display such that each subdivision meets a target density value. Henceforth, we refer to the original version of VIDA as presented in [26] as VIDA_0 and the new version simply as VIDA.

Figure 1 illustrates the different approaches. It contains interactive scatterplots of selected states in the United States. On the *x* and *y* axes are housing cost and income, respectively. Each state may be graphically represented by a dot or by its polygonal outline. Figure 1a shows a visualization of this data that was created in the DataSplash database visualization environment [7]. The visualization is plainly cluttered.

Figure 1b shows a visualization of this data that was manually constructed based on VIDA_0's feedback about the density of the visualization. In this example, the number of vertices is used as the density metric; density metrics are discussed further in Section 2.2. Because the polygonal representation has high density, VIDA_0 recommends changing the display. In response, the user employs the graphical mechanism described in [26] to replace the polygonal representation with the dot representation. Note that in both Figures 1a and 1b, all objects in a given display are shown with a single graphical representation (a few of the smaller polygons in Figure 1a appear to be dots in this reproduction, but they are actually small state outlines).

Figure 1c shows VIDA's automatically generated constant information density visualization of this data. Note that different representations are chosen for different objects based on local density.

The technique shown in Figure 1c has several advantages. It minimizes both clutter and sparsity in the display. Further, observe that it provides additional detail about outliers. This additional information can be useful in the identification of exceptions, which Williamson and Shneiderman identify as a common task [25].

It can be argued that VIDA's technique distorts the user's view of the underlying data. However, the usefulness of such distortion in the cartographic domain is well-accepted. For example, hardcopy maps frequently show small cities in sparse areas while omitting cities of the same size in dense areas, *e.g.*, Rand McNally's world map [18]. Similarly, some map tools on the World-Wide Web remove labels from some cities or counties to reduce clutter, while leaving labels on other cities or counties; users can zoom in on the maps to see the labels that have been removed [23,24]. Because users understand these common cartographic conventions, we believe they can learn to understand the same techniques when they are applied in non-cartographic visualizations. We further discuss this issue in Section 3.1.

The remainder of this paper is organized as follows. In Section 2, we explain our approach in more detail, and we further discuss its advantages and disadvantages in Section 3. In Section 4, we describe related work. In Section 5 we discuss conclusions and future work.

2. TECHNIQUE

Recall that the Principle of Constant Information Density states that the amount of information per area should remain constant. Our general approach is to break the screen into subdivisions and fill each subdivision with graphical information such that some target information density value is met.

In this section, we describe the organization and creation of graphical objects in our system. We then define information density. Next, we discuss general processes by which density can be changed. We then describe our specific algorithm for creating visualizations and discuss its computational complexity. Finally, we discuss our implementation and provide a number of illustrative examples.

2.1 Organization and Creation of Graphical Objects

As described in further detail below, we have implemented our work in the DataSplash database visualization environment [7]. DataSplash objects appear in a two-dimensional canvas over which the user can pan and zoom.

In DataSplash, all objects in a canvas are organized into *layers*. Each object is a member of exactly one layer. Each layer is associated with exactly one database table. Each row in the table is assigned an x,y location in the canvas, *i.e.*, the rows are scattered across the canvas, giving an effect similar to a scatter plot. The x,y locations are derived from data values in the rows. For example, if the user has a table of United States cities with latitude and longitude columns, x and y can be assigned to the longitude and latitude values of each city.

At any point, the user can create an object in DataSplash's paint program interface and duplicate that object for every row in the database table. As a result of this duplication operation, a copy of the object appears at the x,y location of every row in the table. The effect is like splashing paint across the canvas, coating every scattered row. The user may also associate display properties of objects with columns in the table, *e.g.*, height, width, color, and rotation of each splash object can be derived from values in the columns of its row. Continuing our example of a visualization of United States cities, the user may specify that a circle is to be drawn at the x,y location of each city. The user may further specify that the radius of each circle be proportional to the population of that city. Such a visualization appears in the detailed example presented in Figure 3 in Section 2.6.

2.2 Density Metrics

As discussed in [26], we have designed a software framework in which we can explore generalizations of the Principle of Constant Information Density. Since we are unlikely to anticipate the needs of all applications, we express all of our notions of information density in terms of extension and configuration interfaces.

Information density metrics are expressed using density functions. Expert users may define new density functions to supplement those already included in the system. (These functions are currently compile-time extensions.) Density functions return the associated density metric value for a given layer at a given elevation.

Our system currently supports two density metrics, number of objects and number of vertices. There are a number of other metrics that could be used, *e.g.*, Tufte's data density [21]. For a thorough review of possible density metrics see [16]. Because the focus of our work is on maintaining constant information density for a given metric rather than on determining good density metrics, we have not yet implemented additional metrics.

2.3 Processes for Modifying Density

There are two general processes by which density may be modified. These approaches may be used individually or in combination.

First, different graphical representations of objects appropriate for different elevations, known as *multiscale representations* [11], can be selected according to their density. There are three interesting ways multiscale representations can decrease density (corresponding actions exist to increase density):

1. The glyph used to represent an object can be replaced with one of lower density. As one example, a river represented by a polyline with many vertices can be replaced by a simplified representation with fewer vertices.

2. Part of the graphical representation of an object may be omitted, *e.g.*, the text label for a city can be removed, leaving only a dot to represent the city.

3. A number of objects can be aggregated. For example, a number of dots in a scatterplot can be replaced by a single larger dot.

Second, objects may be omitted to decrease density. (Correspondingly, objects may be included to increase density.) Note that this can be considered a special case of number 1 above, in which a glyph is replaced with a glyph of 0 density. However, because this type of replacement has a dramatically different visual effect, we consider it separately.

In previous work, we applied these processes uniformly to all objects of a given type. For example, all cities might appear as dots in the display. Alternatively, all cities might be excluded from the display.

However, if we are to provide a uniform display of non-uniform data, objects that differ only in their location in the display must be displayed differently. As an example of non-uniformly applied multiscale representations, a large number of dots in one part of a scatterplot can be replaced with a large circle, while dots in other parts of the same scatterplot can be left in the display. As an example of non-uniformly applied omission, in a traditional map, small cities might not be shown in areas with large populations. However, cities of the same size might appear in less populous areas in the same map. We call this technique *selective omission*.

2.4 Algorithm

We have developed an algorithm that controls the display of layers in subdivisions of the screen based on density. (Recall that a layer consists of a data set and a specification of the graphical representation of that data set.) The algorithm works as follows. First, it divides the visible screen into a regular n x n grid. Every cell in the grid has a goal density. Goal density is the same for every cell in the grid and is configurable. For a given cell, our algorithm calculates the density of each layer. We assess the grid approach and discuss alternatives in Section 3.2 below.

We began our experiments with a fairly naïve algorithm that finds the combination of layers yielding the density value closest to the goal density for a given cell. It renders those layers within the boundaries of that cell. While this approach is effective for some applications, it is highly inappropriate for many others. For example, in one application, in some parts of the screen state outlines are rendered without cities while in other parts of the screen cities are rendered without state outlines (the former is acceptable while the latter is not).

Based on our experience with the naïve algorithm, we decided to allow the user to specify the combinations of layers that are semantically meaningful. We identified two possible approaches. First, we could allow the user to specify the valid combinations of layers extensionally, *i.e.*, we could provide an interface in which the user could explicitly identify all valid combinations. Second, we could allow the user to specify the valid combinations of layers intensionally, *i.e.*, we could allow the user to specify certain constraints from which the system could infer all the valid combinations. Because typical DataSplash applications range from 5 to 15 layers, we decided extensional specification would be prohibitively time-consuming for the user.

To simplify the user's task, we developed intuitive and composable constraints with which the user can express the relationships between layers. We designate each unit to which constraints may be applied a *bundle*; the simplest bundle is a single layer. When a constraint is applied to two bundles, it may be considered to generate a new bundle to which additional constraints may be applied. The constraints are a lower-level specification mechanism than the processes described in Section 2.3 and can be used to simulate these processes.

Constraint 1 (mutual exclusivity): The user can state that two bundles are mutually exclusive. In this case, only one of the two bundles may be displayed within a given subdivision of the screen. This can be useful when one layer is an alternative representation of another. For example, suppose two layers exist for representing cities, one of which represents cities as a dot and one of which represents cities as a circle. For a given city, only one such representation should be displayed. The user specifies a density ordering of the two bundles, *i.e.*, they specify which has higher density.

Constraint 2 (additivity): The user can state that one bundle may be added to another. In this case, the first bundle may appear alone or with the second bundle in a given subdivision of the screen. For example, the state outline layer may appear alone, or with the city layer. Note that the relationship is not symmetric, *e.g.*, cities may not appear without state outlines.

We illustrate our algorithm with the following example.

> **Example A**: Constraint 1 can be used to generate a bundle that specifies that cities may be represented as dots or circles. It can also be used to generate a bundle that specifies that city labels may be represented with a small font or with a large font. Constraint 2 can then be used to specify that the city dots/circles bundle can appear alone or with the city labels bundle.

We call a set of layers chosen for display a *configuration*. Applying our constraints to Example A results in six configurations: (1) a dot, (2) a circle, (3) a dot plus a small-font text label, (4) a dot plus a large-font text label, (5) a circle plus a small-font text label, and (6) a circle plus a large-font text label.

Our algorithm chooses a list of valid configurations such that the density of each of the configurations in the list is guaranteed to be non-decreasing. This is desirable because we want to make the choice of representation as consistent and stable as possible (see the discussion of flickering representations in Section 3.2). Additionally, we wish to limit the number of configurations to avoid unnecessary visual complexity.

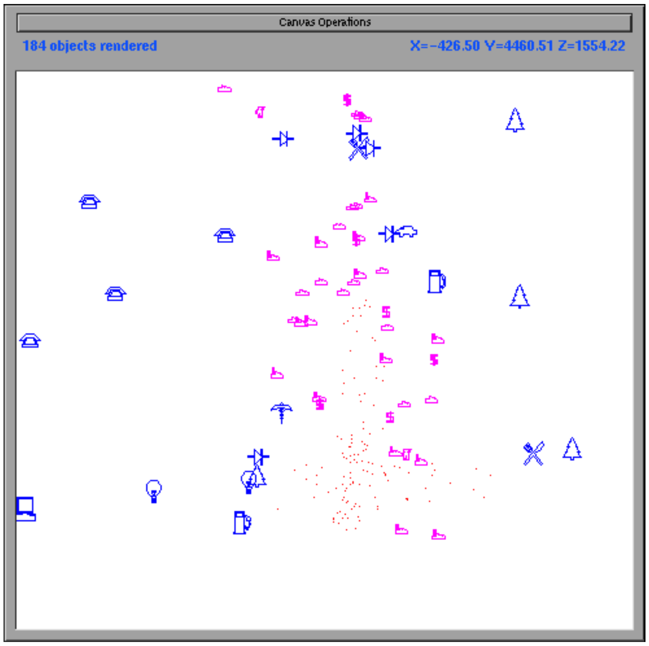

Figure 2a: VIDA visualization at a high elevation.

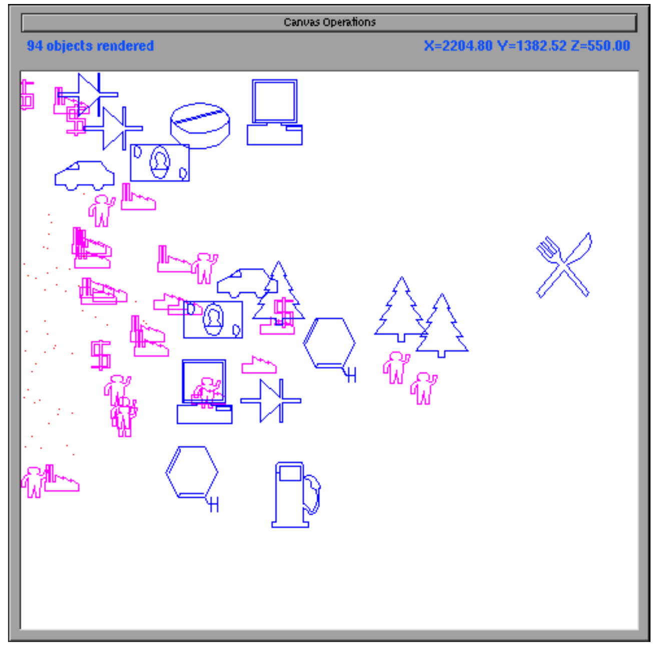

Figure 2b: Zoomed-in view.

Figure 2: A visualization of Fortune 500 data, displayed according to % profit growth (*x* axis) and income (*y* axis). In Figure 2a, VIDA displays different objects at different levels of detail, ranging from dots in the most dense regions to detailed icons for the outliers. In Figure 2b, the user has zoomed in. VIDA dynamically adjusts the display so that more detailed representations are visible, since they are more appropriate to the lower elevation.

Generation of a density-ordered list is simplified by two observations. First, recall that the density ordering of a mutually exclusive bundle is known from the user's specification. Second, observe that the density ordering of an additive bundle is obvious; a bundle appearing alone has density lower than or equal to that of the same bundle appearing with another bundle.

These observations immediately result in a partial ordering of the set of potential configurations listed above. In Example A, we can see that (1) has lower density than (2). However, the relative density of, for example, (2) and (3) is not known. VIDA's algorithm proceeds in a depth-first fashion, producing a fully ordered list consistent with the partial ordering. In this example, it chooses (1), (2), (5), and (6).

2.5 Computational Complexity

Our current density computations are fast enough that performance is not affected visibly when rendering our current main-memory-resident data sets (typical sizes range from hundreds to tens of thousands of objects). In fact, in many cases our techniques improve performance, since they significantly reduce the number of objects that must be rendered.

Our current algorithm for computing the density of an individual layer runs in linear time (using number of objects or number of vertices as the density metric). The algorithm that selects the layers to display is more computationally intensive. (In fact, the naïve algorithm that does not consider constraints is provably NP-Complete. This can be shown by a polynomial reduction from Subset-sum [13] to our problem of choosing a set of layers that sum to a given density.) In practice, we have found that the number of layers is generally low enough that performance is not affected noticeably.

There are at least two facts that can allow computation costs to become prohibitive. First, user-defined density metrics can be arbitrarily expensive to compute. Second, larger data sets can increase computation time significantly (albeit linearly). However, if necessary, the following three techniques can be used to reduce computation time:

- The density values can be stored during or across browsing sessions (currently they are dynamically computed each time the user moves).
- Heuristics can be used to estimate density. Because the density metrics are used to choose representations, approximations are acceptable. This can be particularly useful when updates are frequent.
- More advanced data structures can be used to compute or estimate density values. For example, R-trees can be extended to include density estimates as in [3].

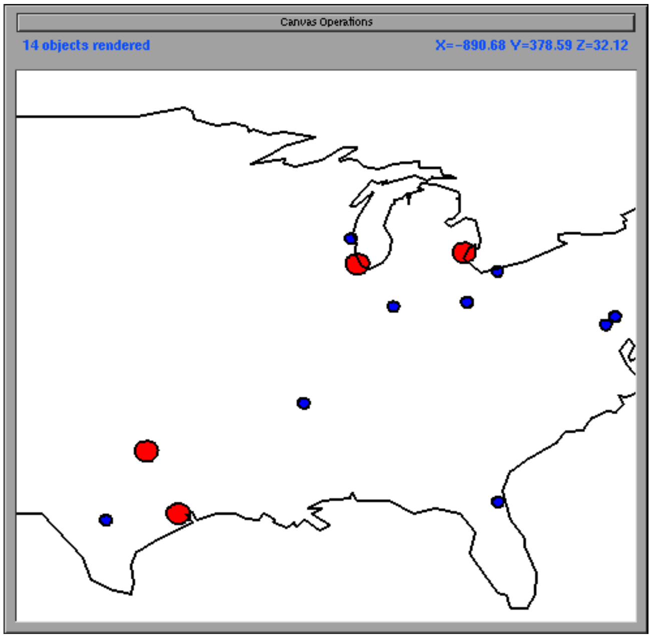

Figure 3a: Naïve DataSplash visualization.

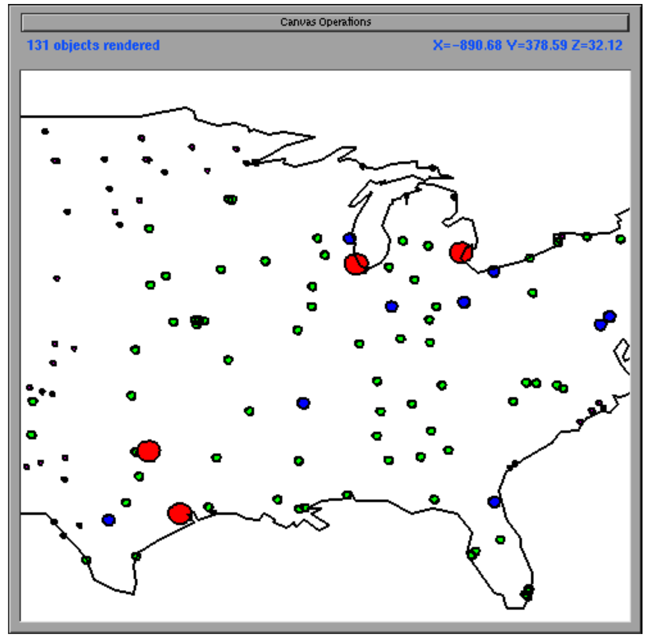

Figure 3b: VIDA visualization.

Figure 3: A visualization of population data, displayed according to latitude and longitude. In the naïve DataSplash visualization shown in Figure 3a, many regions of the visualization are too sparse. In Figure 3b, VIDA has made the display more uniform by showing cities of lesser populations in regions where no large cities exist.

2.6 Implementation and Examples

We have implemented our algorithm in VIDA_0, which is an extension to the DataSplash database visualization environment [2,7]. VIDA_0, VIDA, and DataSplash are implemented in C++, the Mesa graphics library [15], XForms [27], and POSTGRES [19], and run on most UNIX platforms.

In this section, we present three sample visualizations of non-uniform distributions of data. These examples illustrate a number of advantages and disadvantages of VIDA's technique for displaying non-uniform data. These issues are discussed in Section 3.

We begin by revisiting Figure 1, an interactive scatterplot of selected states in the United States. On the *x* and *y* axes are housing cost and income, respectively. Each state may be graphically represented by a dot or by its polygonal outline (each representation is associated with a layer). Previously we discussed Figures 1a and 1b. At this time, we provide more detail about Figure 1c, which shows VIDA's automatically generated constant information density visualization of this data. In this example, VIDA uses the number of vertices as the density metric. There is a mutually exclusive relationship between the two representations (dot and polygonal outline), so that within a grid cell all states are represented by either a dot or a polygonal outline, but not both. In this visualization, the general trend of the data is still visible (housing cost and income appear to be correlated), but additional data about the outliers is visible. For example, California is very expensive to live in, but also has a high income level.

The second example, shown in Figure 2, is an interactive scatterplot of selected Fortune 500 companies. On the *x* and *y* axes are % profit growth and number of employees, respectively. Each company has three potential representations (each associated with one layer): (1) a dot; (2) an icon of the general category of industry to which the company belongs; and (3) an icon of the specific type of industry to which the company belongs. Figure 2a shows VIDA's constant information density visualization of this data. As in Figure 1, VIDA uses the number of vertices as the density metric. There is a mutually exclusive relationship between the three representations (dot, general category icon, and specific category icon), so that within a grid cell all companies are represented by exactly one of the three possible representations. Because the outliers are shown with more detail, we can observe that many of the companies with a high % profit growth are forestry companies (the tree icons on the right), while many of the companies with a low % profit growth and a large number of employees are telecommunications companies (the telephone icons in the upper-left).

Suppose the user is browsing the visualization and wants more information about companies with high profit growth and a small number of employees. Then the user may zoom in to the area in the lower right of the visualization; Figure 2b shows the resulting zoomed-in view. Note that if the user zooms in on a fixed set of objects, each object that remains in the display occupies more screen space. Therefore, when the user zooms in, each object is shown with a more detailed representation. From this vantage point, the user learns that, unexpectedly, only a small number of the high growth companies with few employees are high-technology companies (many of them are forestry, service, or heavy industry companies).

The third example, shown in Figure 3, is a map of cities in the United States. On the *x* and *y* axes are longitude and latitude, respectively. One layer contains the outline of the United States. Four additional layers contain cities partitioned by population. In this example, VIDA uses the number of objects as the density metric. Additive relationships pertain to these layers, so that any given grid cell may contain only the outline of the United States, the outline of the United States plus the largest cities, the outline of the United States plus the largest and second largest groups of cities, *etc.*

Multiscale representations are illustrated in Figures 1 and 2. Selective omission is illustrated in Figure 3. In these examples, it happens to be the case that multiscale representations are supported by mutual exclusivity and selective omission is supported by additivity. However, each of these processes can be generated by either constraint.

3. DISCUSSION

In this section, we discuss the effectiveness of constant information density displays in non-cartographic domains. We then assess our methods for choosing representations to create uniform density displays.

3.1 Effectiveness in Non-cartographic Domains

Historically, the Principle of Constant Information Density has been used in the cartographic domain. In VIDA, we have applied this principle to non-uniform data in both cartographic and non-cartographic domains. In this subsection, we discuss our informal observations of its effectiveness for different tasks.

We begin by discussing the utility of constant information density displays. We then assess individual characteristics of the two processes by which constant information density displays are created, multiscale representation and selective omission. For each technique, we discuss advantages and disadvantages, as well as potential improvements.

Constant Information Density Displays. Constant information density displays have a number of advantages. If the target density value is chosen appropriately, overplotting is minimized and use of the available display space is maximized. Further, the displays are visually appealing. Additional advantages are discussed below.

A potential disadvantage of the technique is that it might give users a distorted perception of the actual density of the underlying data space. We have three responses to this argument in general; more specific issues are discussed in the subsections on multiscale representations and selective omission below.

First, we mentioned above that we believe users can learn to interpret VIDA visualizations, just as they have learned to interpret maps which distort information.

Second, we observe that because cluttered visualizations contain overplotting, they do not give an accurate representation of the distribution of the data. Therefore, for tasks specifically involving density distributions, we recommend developing visualizations compatible with the Principle of Constant Information Density. For example, a visualization in which each pixel is colored to show density at a given location has constant information density and provides a good sense of the distribution of data. As another example, a scatterplot can use aggregation in the following manner: when a region of the screen contains more than a given number of dots, these dots can replaced by a single, larger dot of a different color.

Third, we observe that one can apply the Principle of Constant Information Density only to those dimensions in which the user is not explicitly studying distribution. For example, if the user is looking for high-density areas in the *x* and *y* dimensions, it might be appropriate to apply the Principle of Constant Information Density only in the *z* dimension. More concretely, suppose that at a given elevation, all visible objects are shown with the same representation. This specific representation can be chosen such that the total density of the screen approximates a target density value. Techniques to ensure uniformity in the *z* dimension are discussed further in [26].

Multiscale Representation. Note that multiscale representation of data according to density shows outliers with more detail (and therefore often with more screen space). We argue that showing more detail for outliers assists in the detection and investigation of anomalies. However, because this technique makes outliers disproportionately prominent, there is a danger that this technique will overemphasize outliers, which may be inappropriate for some applications.

As an additional advantage of the variable-detail representation technique, we have observed that, rather than obscuring trends, the technique actually makes the distribution of the data more apparent. For example, in Figure 2a, it can be seen that the data distribution roughly follows the shape of a bell curve. (In the color version of the visualization, this is strongly apparent, since each representation has a different color.)

Selective Omission. While omission does achieve constant information density displays, it can be fairly misleading. Consider Figure 3b. This visualization is consistent with the common cartographic convention of omission in paper maps. However, if this same technique were applied to a different type of visualization, *e.g.*, a scatterplot, the user might incorrectly infer that no data existed in a specific region when in fact such data did exist.

It is our belief the display could be significantly improved by the addition of visual cues that indicate that objects have been removed. Such cues fall into two general categories. First, cues can indicate the regions from which objects have been omitted. For example, Magic Lenses might be placed over regions in the display to indicate that they are being shown at a different level of detail [5,8]. Second, cues can encode information about the distortion that has been applied. For example, the background of the visualization can be colored to indicate density in various regions (note that it would be interesting to color the backgrounds of a set of Magic Lenses in this way). Alternately, objects can be blurred to indicate the accuracy of the representation; blurrier areas can represent areas in which higher distortion has occurred.

3.2 Choice of Representations for Display

In general, the gridding mechanism and simple constraints we provide are effective. We typically use a 10 x 10 regular grid, and find that boundaries are not perceptible unless the data distribution is highly uniform (see Figure 4). In cases in which it is desirable to use a less perceptible grid, alternative subdivisions such as hexagonal and non-regular grids could be used.

We find the constraints are quite powerful, supporting a number of diverse applications. Much of this utility is evident in the examples provided above. In this subsection, we discuss the limitations we have identified with the gridding and constraint mechanisms and suggest potential improvements.

Flickering Representations in the Grid. Despite the fact that the grid boundaries are not readily apparent, there is a significant drawback to the grid-based approach. In our current implementation, the grid boundaries are relative to the screen (one can imagine a transparent grid that moves independently above the two-dimensional canvas). In this model, the contents of a given grid cell change as the user pans or zooms.

Consider the case when the user pans. Suppose an object is in a grid cell that has high density. In this position, it is displayed with a low-density representation. If the user pans slightly, the grid shifts. In the new subdivision of the screen, the object may be in a grid cell that has low density and therefore may be displayed with a high-density representation. A similar problem exists when the user zooms. For example, during one continuous inward zoom, a low density representation might be replaced by a high density representation and then be replaced by the original low density representation.

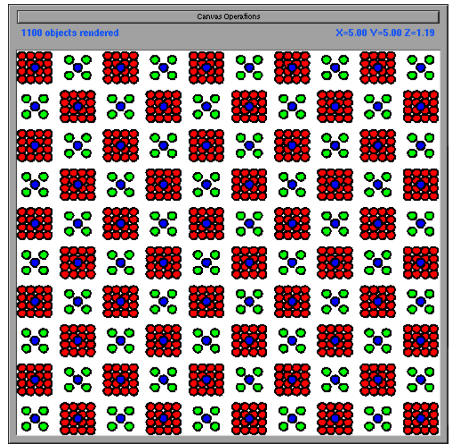

Figure 4: Visualization of an artificially-created data set with uniform distribution. When the grid-based algorithm is applied to uniformly-distributed data, grid cell boundaries are immediately discernible (*c.f.* Figures 1-3).

Such flickering representations are highly distracting and create undesirable visual effects. Panning flicker can be solved easily by registering the grid to the canvas, *i.e.*, by embedding it in the underlying *x*,*y* space. Unfortunately, zooming flicker can not be solved easily because no single size for a grid cell is appropriate for all elevations. A spatial data structure such a quad-tree that subdivides the space might seem appropriate, but does not in fact entirely solve the problem. Possible solutions include choosing representation on a per-object rather than a per-grid-cell basis. Fortunately, the zooming case is not as visually obtrusive as the panning case.

Aggregated Objects in the Grid. Our current implementation chooses layers to render within each grid cell. It does not explicitly take into account the semantic relationship between objects in different layers. For example, it does not explicitly consider that the United States is a single object that contains each of the individual states. We would like the system to render either the United States, or all the individual states. However, these semantics are not enforced in the current implementation; in some situations in which the objects are in multiple grid cells, states are rendered in some cells but not in others. An object-based model for display would ameliorate this problem.

Allocation. In some situations, the user may wish to specify the relative visual resources to be allocated to given bundles. For example, suppose an application has two bundles, one for political boundaries and one for cities. Each of these bundles may contain a number of different representations with widely varying density; the political boundaries bundle may range from a country outline to county outlines, while the city bundle may contain a number of layers with different numbers of cities. It would be useful to provide a mechanism with which the user could specify that at any given time, *n*% of the visual density should be allocated to the political boundaries and the remainder should be allocated to the cities. This mechanism could be used, for example, to ensure that a certain type of feature is always present in the display.

4. RELATED WORK

Indirectly-related work has been done in a number of areas. In our own work, we have conducted a preliminary pilot study of user response to constant information density visualizations of uniform data [26]. Other researchers have examined appropriate amounts of information density for specific character displays, user interface screens, or images (the equivalent of a fixed elevation in our system). Useful summaries appear in [12,22]. Related work on creating visually appealing displays has considered the layout of objects at multiple granularities [14]. However, the layout problem detailed in this work has different objectives, requiring that no objects overlap and that the minimal amount of space be wasted.

Other systems consider ways to construct displays with constant information density. In our own work, we have proposed changing underlying data and graphical representations to modify density [26]. Additionally, as discussed above, $VIDA_0$ utilizes the Principle of Constant Information Density to interactively guide users in the construction of applications with constant information density [26]. This system works best on uniformly distributed data. Further, researchers in the area of map generalization have studied automated generation of maps of given scales, although with limited success [6]. The multi-scale tree algorithm takes a different approach [9]. Given a set of maps produced (manually or with a computer-assisted tool) at different scales, the algorithm is designed to automatically produce different views of the data as the user zooms. These views would have a constant number of active pixels. However, the algorithm described in this approach has several major limitations. It gives the user no control over which representations are chosen for display (VIDA provides a constraint mechanism for this purpose). It supports only one density metric (ink, or the number of live pixels), in contrast to $VIDA_0$ and VIDA, which have a general infrastructure for supporting multiple density metrics. Finally, examples are limited to the cartographic domain, and it is not clear the technique would generalize to other applications.

A number of interactive visualization systems provide clutter reduction mechanisms. For example, Ahlberg and Shneiderman's work allows the user to filter objects in the display dynamically [1]. This technique does not differentially filter subdivisions of the display. Therefore, the resulting visualizations can have regions that are too dense and/or too sparse. Fishkin and Stone extend the dynamic query model by providing movable filters which the user can position manually in the display [8]. This technique is highly appropriate for non-uniform data. However, unlike VIDA, the movable filter technique as described in this work requires the user to choose explicitly the areas to which filtering is applied. One can imagine using VIDA's density techniques to place movable filters appropriately, as mentioned in Section 3.1.

Non-linear magnification schemes can also be used to minimize clutter in the display [10]. In fisheye views, for example, objects are viewed as though through a fisheye lens so that objects in the center of the screen are given more display space than objects on the periphery. Objects on the periphery can be displayed with reduced detail or can be omitted. However, such displays can be disorienting for the user. For applications in which they are desirable, the Principle of Constant Information Density can be applied to them in interesting ways. For example, representations could be chosen or objects could be placed to guarantee constant information density. In some situations, this would create concentric circles containing representations with decreasing detail.

Another interactive visualization technique, semantic zoom, reduces clutter by controlling the representation of objects according to elevation [17,4]. Our work uses the notion of density to control the representation of objects. However, the general approach can be simulated in semantic zoom systems. This can be achieved by using density metrics to assign the transition points between different representations on a per-object basis.

In theoretical work related to interactive visualization, Furnas and Bederson present a framework for multi-scale viewing [11]. This framework seems to extend naturally to some density metrics, *e.g.*, the number of objects [26]. However, it is not clear that it extends to all such metrics.

5. CONCLUSIONS AND FUTURE WORK

We have presented VIDA, a system that creates visualizations that have uniform information density in the *x*, *y*, and *z* dimensions. VIDA visualizations have this uniformity even when the data sets being displayed are non-uniformly distributed. We have discussed the advantages and disadvantages of the visualizations created by VIDA, particularly in non-cartographic domains. VIDA visualizations are automatically constructed using constraints specified by the user. We have described these constraints and assessed their utility.

Further studies of user response to applications with constant information density are plainly warranted. Additionally, although we are currently focusing on preserving constant information density for a given metric rather than comparing density metrics and studying appropriate values for such metrics, such comparative studies would be useful. A formal taxonomy of density metrics would also be of significant interest.

Finally, we are extending our display and constraint mechanisms. Planned improvements include replacing the grid-based model with an object-based model, adding an allocation mechanism, and developing a graphical mechanism with which users can specify constraints.

ACKNOWLEDGMENTS

We are grateful to Paul Aoki and Chris Olston for many helpful discussions and suggestions. Chris Olston suggested extending VIDA to use blurring and variable color to indicate density as discussed in Section 3.1. This research was sponsored by NSF under grants IRI-9400773 and IRI-9411334.

REFERENCES

1. Ahlberg, C., and Shneiderman, B., “Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays,” *Proc. ACM SIGCHI ’94*, Boston, Apr. 1994, pp. 313-317.

2. Aiken, A., *et al.*, “Tioga-2: A Direct Manipulation Database Visualization Environment,” *Proc. 12th Int’l Conf. on Data Engineering*, New Orleans, Feb. 1996, pp. 208-217.

3. Avnur, R., *et al.*, “CONTROL: Continuous Output and Navigation Technology with Refinement On-Line,” demonstration description, *Proc. ACM SIGMOD ’98*, Seattle, June 1998, pp. 567-569.

4. Bederson, B.B., and Hollan, J.D., “Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics,” *Proc. ACM UIST ’94*, Marina del Rey, CA, Nov. 1994, pp. 17-26.

5. Bier, E., *et al.*, “Toolglass and Magic Lenses: The See-Through Interface,” *Proc. ACM SIGGRAPH ’93*, Anaheim, CA, Aug. 1993, pp. 73-80.

6. Buttenfield, B.P. and McMaster, R.B. (eds.), *Map Generalization: Making Rules for Knowledge Representation*, Longman, London, 1991.

7. DataSplash software. Copyright © 1997, the Regents of the University of California. Available at http://datasplash.cs.berkeley.edu/.

8. Fishkin, K., and Stone, M.C., “Enhanced Dynamic Queries via Movable Filters,” *Proc. ACM SIGCHI ’95*, Denver, May 1995, pp. 415-420.

9. Frank, A., and Timpf, S., “Multiple Representations for Cartographic Objects in a Multi-Scale Tree - An Intelligent Graphical Zoom,” *Computers & Graphics*, Nov.-Dec. 1994, 18(6):823-829.

10. Furnas, G. W., “Generalized Fisheye Views,” *Proc. ACM SIGCHI ’86*, Boston, Apr. 1986, pp. 16-23.

11. Furnas, G.W. and Bederson, B.B., “Space-Scale Diagrams: Understanding Multiscale Interfaces,” *Proc. ACM SIGCHI ’95*, Denver, May 1995, pp. 234-241.

12. Galitz, W.O., *User-Interface Screen Design*, QED Pub. Group, Boston, 1993.

13. Garey, M., and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York, 1979.

14. Ioannidis, Y., *et al.*, “User-Oriented Visual Layout at Multiple Granularities,” *Proc. AVI ’96*, Gubbio, Italy, May 1996, pp. 184-193.

15. Mesa software. Copyright © 1995-1996 Brian Paul. Available at ftp://iris.ssec.wisc.edu/pub/Mesa/.

16. Nickerson, J.V., *Visual Programming*. Ph.D. dissertation, New York Univ., New York, 1994.

17. Perlin, K., and Fox, D., “Pad: An Alternative Approach to the Computer Interface,” *Proc. ACM SIGGRAPH ’93*, Anaheim, CA, Aug. 1993, pp. 57-64.

18. Rand McNally & Co., *Rand McNally Signature Series World Map*.

19. Stonebraker, M. and Kemnitz, G., “The POSTGRES Next-Generation Database Management System,” *Communications of the ACM*, Oct. 1991, 4(10):78-92.

20. Töpfer, F., and Pillewizer, W., “The Principles of Selection, A Means of Cartographic Generalization,” *Cartographic J.*, 1966, 3(1):10-16.

21. Tufte, E.R., *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, 1983.

22. Tullis, T.S., “Screen Design,” *Handbook of Human-Computer Interaction*, M. Helander (ed.), Elsevier Science Publishers B.V., Amsterdam, 1988.

23. United States Geological Survey, *MapFinder*, http://edcwww.cr.usgs.gov/webglis/.

24. Washington State Employment Security Department, Labor Market and Economic Analysis Branch, *Washington State Labor Market Information*, http://www.wa.gov/esd/lmea/sprepts/leg/STATLEG.HTM.

25. Williamson, C., and Shneiderman, B., “The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System,” *Proc. ACM SIGIR ’92*, Copenhagen, Denmark, June 1992, pp. 338-346.

26. Woodruff, A., Landay, J., and Stonebraker, M., “Constant Information Density in Zoomable Interfaces,” *Proc. AVI ’98*, L’Aquila, Italy, May 1998, pp. 57-65.

27. XForms software. Copyright © 1996 T.C. Zhao and Mark Overmars. Available at http://bragg.phys.uwm.edu/xforms/.