Title: A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization

Author: Matthew O. Ward

Address: Computer Science Department, Worcester Polytechnic Institute, 100 Institute Rd., Worcester, MA 01609 USA

Contact Phone: (508) 831-5671

Contact Fax: (508) 831-5776

Contact E-mail: matt@cs.wpi.edu

Short Title: Glyph Placement Strategies

Acknowledgments: This work was partially supported under NSF Grants IIS-9732897, IIS-0119276, and EIA-9729878

Abstract – Glyphs (also referred to as icons) are graphical entities that convey one or more data values via attributes such as shape, size, color, and position. They have been widely used in the visualization of data and information, and are especially well suited for displaying complex, multivariate data sets. The placement or layout of glyphs on a display can communicate significant information regarding the data values themselves as well as relationships between data points, and a wide assortment of placement strategies have been developed to date. Methods range from simply using data dimensions as positional attributes to basing placement on implicit or explicit structure within the data set. This paper presents an overview of multivariate glyphs, a list of issues regarding the layout of glyphs, and a comprehensive taxonomy of placement strategies to assist the visualization designer in selecting the technique most suitable to his or her data and task. Examples, strengths, weaknesses, and design considerations are given for each category of technique. We conclude with some general guidelines for selecting a placement strategy, along with a brief description of some of our future research directions.

Index Terms – Multivariate data visualization, information visualization, glyphs, layout algorithms.

1 Introduction

It is well known that we are experiencing an explosion in the volume and complexity of data to which we have ready access. Scientific measurements, business transactions, surveys, simulations, and a host of other activities contribute to this growing mass, and our ability to effectively analyze the data has become increasingly strained. Much of this data is multivariate in nature, whether it be from tables, spreadsheets, multispectral sensors, or complex computations. Each row or data entry may have anywhere from two to several thousand entries, and the larger the number and size of the entries, the harder it is to detect, classify, and measure features and relations of interest.

Visualization has emerged as a powerful tool for the exploration of such large and complex data sets. While algorithmic analysis can be used to quickly and accurately process data to identify patterns and outliers, it is dependent on having a computational model of the phenomena of interest. The problem is that one may not know what one is looking for, or may not be able to set fixed parameters and thresholds to effectively guide the analysis. Visualization, on the other hand, uses the human perceptual system to extract meaning from the data, focus attention, and reveal structure and patterns. The author's Visual Information Seeking Mantra, "I'll Know it When I See it (IKIWISI)" (a follow up to Shneiderman's Mantra [72] "overview first, zoom and filter, then details on demand"), is a reflection of the notion that we often don't know what we are looking for when presented with a data set, and we rely on our visual pattern recognition system to help us in gaining knowledge of the data. It is also relevant to the concept that visualization is a powerful mechanism for confirming hypotheses about data phenomena; quantitative analysis may indicate that some relation exists, but a supporting visualization can improve our confidence in the results and our ability to remember them.

Glyphs (also referred to as icons) are a popular method for conveying information visually. Individual dimensions (variables) for a given data point are mapped to attributes of a particular shape or symbol, and variations, clusterings, and anomalies among these graphical entities may be readily perceived. Glyphs are a powerful communication mechanism in that a large number of data dimensions can be incorporated into the attributes of a single shape or symbol. The particular mappings may also be customized to reflect semantics relevant to specific domains, which can greatly facilitate the interpretation process.

Once a glyph has been designed and generated from a data entry, it must be placed at a location in display space (2-D or 3-D). The position attribute can be very effective in communicating data attributes or improving the detection of similarities, differences, clustering, outliers, or relations in the data. Many strategies exist for setting the position attribute; some are based on the raw data (or information derived from the data), while others are based on the structure of the data set (e.g., ordered or hierarchical). Methods also vary as to whether they allow overlapping glyphs, whether they are space-filling, or whether they employ empty space to highlight differences.

The goal of this paper is to present a comprehensive taxonomy of glyph placement strategies to support the design of effective visualizations. We first present a formal definition of glyphs, how they are formed, examples of glyphs which have been used in the past, and some limitations of glyphs for visualizing multivariate data. We then describe a categorization of methods for positioning glyphs, present several examples, and describe limitations and strengths of each. We conclude with some general strategies for selecting a placement strategy, and point to some directions for future research.

2 Glyph Fundamentals

Multivariate data (also called multidimensional or n-dimensional) consists of some number of points, m, each of which is defined by an n-vector of values. Such data can be viewed as an mxn matrix, where each row represents a data point and each column represents an observation (also called a variable or dimension). An observation may be scalar or vector, nominal or ordinal, and may or may not have a distance metric, ordering relation, or absolute zero. Each variable/dimension may be independent or dependent. For this paper, we assume that each dimension is an independent bounded numeric scalar value that we can normalize to the range $[0. \rightarrow 1.]$.

A *glyph* consists of a graphical entity with p components, each of which may have r geometric attributes and s appearance attributes. Typical geometric attributes include shape, size, orientation, position, and direction/magnitude of motion, while appearance attributes include color, texture, and transparency. Attributes can be discrete or continuous, scalar or vector, and may or may not have a distance metric, ordering relation, or absolute zero.

The process of creating a glyph thus becomes one of *mapping* one or more data dimensions for a data point to one or more geometric and/or appearance attributes of one or more components of a graphical entity. A glyph may also contain components or attributes that are independent of the data being used in its formation. For example, in a scatterplot the size and color of the plotting symbol may be data-independent.

Figure 1 shows a variety of examples of glyphs that have been proposed and used in the past. The list below describes these and other glyphs from the literature in terms of the graphical entities and attributes controlled by the multivariate data point.

- profiles [25]: height and color of bars.
- stars [73]: length of evenly spaced rays emanating from center.
- Anderson/metroglyphs [2, 33]: length of rays.
- stick figure icons [61]: length, angle, color of limbs.
- trees [48]: length, thickness, angles of branches; branch structure derived from analyzing relations between dimensions.
- autoglyph [8]: color of boxes.
- boxes [35]: height, width, depth of first box; height of successive boxes.
- hedgehogs [47] : spikes on a vector field, with variation in orientation, thickness, and taper.
- faces [15]: size and position of eyes, nose, mouth; curvature of mouth; angle of eyebrows.
- arrows[81]: length, width, taper, and color of base and head.

- polygons [71] : conveying local deformation in a vector field via orientation and shape changes.
- dashtubes [30] : texture and opacity to convey vector field data.
- weathervanes [28]: level in bulb, length of flags.
- circular profiles [58]: distance from center to vertices at equal angles.
- color icons [54] : colored lines across a box.
- bugs [19]: wing shapes controlled by time series; length of head spikes (antennae); size and color of tail; size of body markings.
- wheels [19]: time wheels create ring of time series plots, value controls distance from base ring; 3D wheel maps time to height, variable value to radius.
- boids [49] : shape and orientation of primitives moving through a time-varying field.
- procedural shapes [66, 26] : blobby objects controlled by up to 14 dimensions.
- Glyphmaker [64]: user-controlled mappings.
- Icon Modeling Language [62] : attributes of a 2D contour and the parameters that extrude it to 3D and further transform/deform it.

2.1 Glyph Limitations

Glyphs are not without limitations in the communication of multivariate data. Most, if not all, mappings introduce biases in the process of interpreting relationships between dimensions. Some relations are much easier to perceive (e.g., data dimensions mapped to adjacent components) than others. Some recent research [4] has focused on clustering and reordering of data dimensions to improve perception of relationships. The accuracy with which humans perceive different graphical attributes varies tremendously as well [20]. For example, our ability to accurately measure length is superior to attributes such as orientation and color. In addition, the accuracy can vary significantly between individuals and even for a single observer in different contexts. Color perception has been shown to be extremely sensitive to context [55, 31]; background color as well as the color of adjacent entities can modify our judgment of the color of a glyph.

There are also limitations based on the media being used to communicate the information. Screen space and resolution are limited, and displaying too many glyphs at once can lead to either overlaps (which can hinder accurate discernment of individual dimensions) or very small glyphs (though, as we shall see, dense packing can form texture patterns for global analysis). Dynamic scaling control for glyphs can thus be a critical component for an effective interface to glyph displays. Finally, having too many data dimensions in the mapping can make it hard to discriminate or interpret the individual dimensions.

2.2 Glyph Placement Issues

The first consideration when selecting a placement strategy is whether the placement will be *data-driven* (e.g., based on two or more data dimensions) or structure-driven (e.g., based on an explicit or implicit order or other relationship between data points).

A second consideration is whether overlaps between glyphs will be allowed. This can have a significant impact on the size of the data set that can be displayed, the size of the glyphs used, and the interpretability of the resulting images.

A third consideration is the trade-off between optimized screen utilization (e.g., space-filling algorithms) versus the use of white space to reinforce distances between data points.

A fourth consideration is whether the glyph positions can be adjusted after initial placement to improve visibility at the cost of distorting the computed position (how accurately must the position reflect a value?).

2.3 Glyph Placement Strategies

Figure 2 shows a taxonomy of glyph placement strategies for multivariate data sets. In the next two sections we describe each class of strategies and give examples of their use. To formalize the different algorithms we assume the following. Let $V = (v_1, v_2, ..., v_N)$ correspond to the N variables or dimensions of the data set, and $D = (d_1, d_2, ..., d_M)$ correspond to the M data records. Each record $d_i = (o_{i1}, o_{i2}, ..., o_{iN})$ contains N numeric observations normalized to the range $[0.0 \rightarrow 1.0]$. Finally, the set $L = (l_1, l_2, ..., l_M)$ contains the locations in normalized display space for the M data points. For the examples and algorithms in this paper, we assume the display space to be two dimensional, though many of the techniques and concepts readily extend to other display dimensions.

3 Data-Driven Glyph Placement

In *data-driven* placement, the data are used to compute or specify the location parameters for the glyph. The two categories of this strategy class are *raw* and *derived*. The placement algorithm can be characterized as a function that maps $D \rightarrow L$, and may involve either local knowledge (i.e., l_i is computed based solely on d_i) or global knowledge (l_i is computed using the entirety of D or both D and L). Each category of data-driven placement strategies can be further divided based on whether or not the computed positions are subjected to a modification/distortion process in order to reduce visual clutter or overlap. Examples of strategies, both for original placement and distortion, are described below.

3.1 Raw Data Strategies

In raw data placement strategies, one, two or three of the data dimensions are used as positional components (these dimensions may also be redundantly represented in the glyph itself). Thus for two dimensional display space, $l_i = (o_{ia}, o_{ib})$, where $0 < a, b \leq N$. This method conveys detailed relationships between the dimensions selected, as our ability to detect positional (as opposed to shape) patterns is quite strong. Many systems and techniques have been developed using this strategy. IVEE (now known commercially as Spotfire), allows users to select the mappings of data dimensions to both position and graphical attribute[1]. Becker et al [7] used geographically located glyphs to convey communication network traffic attributes. Many researchers [17, 68] have employed the cityscape metaphor to place 3D box glyphs at data driven positions. Figure 3 shows all pairwise mappings of the Iris data set, which consists of 150 points and 4 dimensions (petal and sepal length and width). Note that while each view separates one iris family (sailboat shape) from the other two (kite shape), varying degrees of separation can be seen within the large cluster. Also, some views reveal a number of outliers.

Since for N-dimensional data and a 2-dimensional display there are $N(N-1)/2$ possible mappings (we consider only orthogonal views in the N-dimensional space, and don't include rotational variations), it is critical that this mapping be done carefully. An ineffective mapping can result in substantial cluttering and poor screen utilization. For example, if one of the dimensions used for the mapping only takes on a small number of distinct values, a significant amount of overlap is possible. Also, depending on the semantics of the data, some mappings may be more meaningful than others to the person interpreting the display. Analytic techniques, such as the measures of "interestingness" used in projection pursuit algorithms [29, 40], can be used to identify potentially informative views, such as those exhibiting clustering or significant separation.

The major strengths of this placement strategy are its ease of interpretation, low computational overhead, and effectiveness at uncovering correlations between the dimensions controlling the position (as in a scatterplot). Among the limitations of this approach is the fact that bias is given to the dimensions involved in the mapping, and thus conveys only pairwise (or three-way, for 3-D) relations between the selected dimensions. Structure that is not orthogonal to the viewing direction may be difficult to discern. The issue of occlusion is also a significant concern when using this strategy (see the section below on distortion methods).

3.2 Derived Data Strategies

A derived data placement technique uses an analytic process to generate positions using the data values, and perhaps other information, as input. Thus instead of a location reflecting only one, two, or three of the data dimensions, it reflects some combination of all the dimensions in an attempt to convey N-dimensional relational information in a smaller number of display dimensions.

Some commonly used dimensionality reduction techniques include Principal Component Analysis (PCA) [43], Multidimensional Scaling (MDS) [51, 9, 11, 82], Self-Organizing Maps (SOMs) [50, 69], simulated annealing [5], spring-based models [34], and other optimization techniques[27]. PCA attempts to find linear combinations of the dimensions that explain the largest variation in the multivariate data set. The first two principal components can be used to specify the position of a glyph. SOMs and MDS are iterative refinement/optimization processes that attempt to adjust weights or positions until a certain criteria is met (in our case, the distances or similarities between points in 2-D is a good approximation of the N-D distances/similarities). Variations in the starting conditions and distance or similarity measure can have significant impact on the results of these methods.

As an example of this sort of derived placement, Figure 4 displays the iris and cars data sets using PCA to determine glyph positions. One can see the first two principal components were reasonably successful at separating the three classes of iris (depicted by the small "sailboat" shapes and large and medium size "kite" shapes). Likewise, clusters of similar shapes can be seen in the cars data set.

Another type of derived placement algorithm places objects in terms of their relationships with one or more reference objects, which may or may not be part of the original set. These techniques, often referred to as barycentric methods [38], start with B reference objects $(r_1, r_2, ..., r_B)$ located at the vertices of a convex shape. All data points are then positioned within the shape based on their relations to these reference points. The barycentric coordinates of point P form a linear combination with the reference points, such that $P = a_1r_1 + a_2r_2 + ... + a_Br_B$, where $a_1 + a_2 + ... + a_B = 1$. The coordinates $(a_1, a_2, ..., a_B)$ define a unique location within the shape that can be derived from a data set. Each choice regarding the size and content of the reference set generates a different view; thus the setting of these parameters is crucial in creating an informative visualization. A variation on this approach, described in [39], uses A equally spaced anchors around a circle, with each data point attached to the anchors by springs with constants proportional to the data values corresponding to the dimensions/anchors. Another related technique, the *bull's eye* display [59], places objects in concentric rings relative to their distance from a selected node of interest. Yet another technique based on relations to anchor positions was presented by Spoerri [74], where the corners of a polygon represent concepts and distinct locations within the polygon represent all the possible boolean relations among those concepts. The *Sunflower* metaphor presented by Rose [67] has a similar goal, using overlapping circles to convey all possible relationships between one or more documents and a set of domains.

In general, the techniques described above reinforce the perception of similar shapes by (often) placing them close to each other. However, the techniques do have their limitations. The resulting display coordinates, unlike in the raw data-driven techniques, have no semantic meaning. Instead they can be viewed as supplemental dimensions to the original data set. PCA assumes that the majority of the variation in a data set will be well embodied in the first few principal components, which is not always the case. MDS and SOMs, like all iterative optimization procedures, are not guaranteed to be optimal, and the results are generally not unique. Iterative strategies can also be computationally demanding, although several efforts have been recently made to improve the processing time [14, 9].

A category of data-driven placement strategies that can be perceived as a hybrid of raw and derived techniques can be found in 2-D and 3-D field data, where uniform or non-uniform samples are intended to represent a continuous field of scalar, vector, or tensor values. In these strategies, positions are first computed, and then the data are resampled to generate the values corresponding to the positions. These positions are often specified on a user-controlled regular grid, but significant research has been performed on using the data to compute positions that are most effective for conveying information. For example, while most flow visualizations place arrow glyphs on a regular grid, Globus et al [32] and Helman and Hesselink [36] describe experiments in placing glyphs at critical points in the flow field. This means that significant structures are less likely to be overlooked. Another approach to placing glyphs on a vector field, described by Dovey [24], allows resampling either in physical or parameter space, thus enabling the technique to be applied to curvilinear and unstructured grids. Finally, Fuhrmann and Groller [30] describe a method to place 3-D textured streamlines (called *dashtubes*) to optimize uniformity in spacing. Because these methods compute positions based on the data, we have grouped them with the data-driven, derived techniques. However, one could argue that they are not so much placement strategies as filtering strategies, as a continuum of possible glyphs is being filtered to select the subset to be shown.

3.3 Distortion Techniques for Data-Driven Placement

Once the positions for the glyphs have been computed, a possible post-processing step for data-driven techniques would involve distorting these positions to reduce clutter and overlap. For example, random jitter has been employed in statistical graphics when data-driven positioning is being used with discrete (as opposed to continuous) dimensions [21]. Other approaches try to shift positions to minimize or avoid overlaps. Keim and Hermann use a quadtree structure to relocate points to maintain proximity to their original positions while eliminating overlaps [46]. One concern of this approach is the level of distortion that is introduced. While a minor shifting probably will not significantly change the interpretation of the display, in cases where the data density is extremely high, a point's final position may be far from its initially assigned location. Another variation on position shifting was presented by Chuah et al [17], where elements of interest on a planar city-scape could be elevated above the plane and brought closer to the viewer for inspection. Woodruff et al [83] employ several strategies commonly used in map generalization to maintain a constant density of glyphs on the screen. These strategies include *aggregation* (combining low-level glyphs into high-level ones), *deletion* (removing glyphs based on proximity and similarity to their neighbors), and *replacement* (using smaller, less informative glyphs in areas of high density).

Distortion techniques have also been used to selectively vary the level of detail shown in the visualization [53]. In this way, regions of dense glyphs can be assigned a larger segment of the display space, with either position or position and scaling modified to emphasize or de-emphasize subsets of the data. These methods generally act on a local, rather than global, scale, requiring one or more focus points and the specification of a region of influence for the distortion process. Thus it is critical to the effectiveness of these methods to provide users interactive control of the transformation to facilitate variation of the focus and maintenance of context. Smooth animation between the original and distorted views is also a powerful mechanism in this regard.

A simple distortion technique for reducing glyph overlap is outlined below. Note that no constraint is placed on how far a glyph can move from its original position. Also, if the maximum distance a glyph can move per iteration is set too high, the procedure may increase, rather than decrease, the number of overlaps. Finally, no provision is made for oscillating behaviors (a glyph moving back and forth between two or more other glyphs). A small random perturbation to the glyph offsets could solve this problem. Figure 5 shows the breakfast cereal data set with data-driven positioning, without and with overlap reduction. Note that some data features that had been occluded in the original placement strategy are now visible.

```
Let Tolerance = the maximum percentage of overlapping glyphs.
Thus MaxOverlaps = M * Tolerance.

Let Size = the maximum size for a glyph.
Let D(i,j) = the distance between the centers of glyphs i and j.
Thus NumberOverlaps = number of distinct glyph pairs for which D(i,j) < Size.

Let Speed = the maximum distance a glyph can move per iteration (we suggest this
be a fraction of Size).

Let Iterations = the maximum number of iterations of adjusting positions.

Main Loop:
   For i = 0; i < Iterations AND NumberOverlaps < MaxOverlaps; i++ {
      for j = 0;j < M;j++ {
         for k = j+1;k < M;k++ {
            Overlaps = 0;
            PushVector = (0, 0)
            if D(j, k) < Size   {   /* j and k overlap */
               Direction  = normalized vector from k to j
               if Direction = (0, 0), set Direction to a random displacement
               Magnitude  = 1 - D(j,k)/Size
               PushVector = PushVector + Magnitude * Direction
               Overlaps = Overlaps + 1
               }
             }
          if Overlaps > 0 {
             if PushVector = (0, 0), set PushVector to a random displacement
             PushVector = PushVector * Speed / Overlaps
             Offset for glyph j = PushVector
             |
           }
       Offset all glyphs
      }
```

Algorithm 1: distorting glyph positions in derived placement strategies to reduce overlaps.

Distortion techniques can be very effective at reducing the level of overlap while maintaining a reasonable degree of data integrity, a quality metric based on the difference between the perceived and actual data values and interrelationships. The permissible variation in glyph position is often domain and context dependent. For example, the placement of a glyph representing a particular city should reside on the appropriate side of a state border. Indeed, the more constraints placed on the type and degree of distortion allowed, the more complex the algorithm becomes. The major trade-off thus becomes one of ease-of-implementation (e.g., random jitter and lens techniques) versus accuracy of the results (e.g., through constraint optimization).

4 Structure-Driven Glyph Placement

Structure-driven algorithms assume there exists some implicit or explicit connectivity or relationship between data points, which may or may not be derivable from the data values. For example, there might be a temporal ordering or hierarchical relationship which might not be included as one of the data fields. We categorize strategies of this form as either ordered, hierarchical, or generalized network/graph structures. While ordered and hierarchical structures can be considered special cases of the generalized graph structure, it is possible to take advantage of the constraints imposed on certain types of structure to derive layout strategies not applicable to the more general structure. Algorithms for structure-driven placement are characterized as functions that map $D \rightarrow S$, where S is some generic structure space, followed by a mapping of $S \rightarrow L$. Each category of structure-driven technique can be further divided based on whether overlaps are permitted or not, and whether extra space is added to emphasize relationships. Overlap reduction and space padding will be discussed under distortion techniques for structure-driven methods.

4.1 Ordered Structure

Ordered structure may be linear (1-D) or grid-based (N-D). By sorting the data on one or more dimensions and using this ordering to specify the glyph placement, we facilitate the detection of changes in the dimensions used in the sorting as well as provide clues into potential dependencies in the other variables (detecting if the patterns of change coincide with those of the sorted dimensions). Several placement algorithms have been developed in the past based on linear orderings, including raster scan [76], circular [44, 56], and recursive space-filling patterns [45]. These are illustrated in Figure 6.

Figure 7 shows a time series of economic data (Dow Jones index, Standard and Poors 500 index, retail sales, and unemployment over a three year period) positioned in a circular pattern. Data for December radiate straight up (the 12 o'clock orientation). Expected patterns include high sales and low unemployment in January and the inverse in June. Abnormally high unemployment is visible through much of the first year shown, which improves as other market indicators rise in value.

Figure 8 displays 5-dimensional remote sensing data (Spot, magnetics, and three channels of radiometrics) using star glyphs and a 2-D raster ordering. Because of the number of glyphs involved, individual data points are difficult to interpret. However, clearly discernable regions of uniform texture are visible, including the region highlighted in the zoomed in image identifying an area with particularly high uranium content (data courtesy of CSIRO Division of Exploration and Mining).

The algorithm for a raster-based placement of a 1-D sorted glyph set is as follows: Let $C = \sqrt{M} + .5$ be the number of glyphs one can fit into a row of the display canvas. Thus the maximum size of a glyph is given by the equation $S_{max} = W/C$, where W is the display canvas width. Finally the location of the i^{th} glyph (l_i) is computed as $l_{ix} = S_{max} * (i\ modulo\ C)$ and $l_{iy} = S_{max} * (i/C)$. Note that if $\sqrt{M}$ has a fractional component, the final row will contain some empty space.

The strengths of order-based positioning are its ease of implementation, ease of interpretation, and effective utilization of screen space. It also provides a number of potentially revealing views of the data, as significant clusterings and discontinuities within a given dimension can become visually apparent. The main limitation of the strategy is that it assumes significant features in N-D will be perceptible in 1-D or 2-D snapshots, which often is not the case. It could be difficult, if not impossible, to correlate the changes seen in one ordering with those seen in another to create a composite description of an overall pattern.

4.2 Hierarchical Structure

Hierarchical structure in data sets can be explicit or implicit. Explicit hierarchies are those in which each level of the hierarchy is associated with a single data dimension, and the branches emanating from this level correspond to some number of distinct ranges for that data dimension. For example, sales data may have dimensions associated with particular time periods, geographical locations, sales personnel, and products. Different hierarchies are generated depending on the order in which the dimensions are processed. Other examples of explicit hierarchies are file systems and organizational charts. Implicit hierarchies are generated algorithmically using clustering or partitioning algorithms in conjunction with some N-dimensional distance or similarity metric [25]. These methods differ from the derived data-driven techniques in that the goal is not to generate screen positions, but a hierarchical relationship structure that then can be conveyed to a placement algorithm customized to display this relational information.

Given a hierarchical structure, the task is to position glyphs on the display in such a way as to convey the relationships inherent in the structure. In most methods proposed to date, these relationships are reinforced through the use of additional graphics, using lines to connect parent and child nodes and/or to separate generations. White space can also be used to support this task. Many algorithms have been developed for drawing node-link graphs of trees (see [23] for an excellent survey). These algorithms vary according to features such as where the root node is relative to the rest of the tree (e.g., centered, top-most) and the relative direction between a node and its children (e.g., radially outward, horizontal, vertical, or alternating horizontal and vertical). Different criteria for layout can result in configurations that reveal different aspects of the data. For example, Wilson and Bergeron [80] describe a system for visualizing hierarchies that implements four layout algorithms (leaf-based, subnode-based, range-based, and density-based). Tree layout algorithms also vary depending on whether they are 2-D or 3-D (as in Cone Trees [65], Disc Trees [41], and others [13]). Each of these methods can be augmented to display glyphs as opposed to simple circles, boxes, or spheres. Since the multivariate data may only represent the terminal nodes of the tree, one could either represent non-terminal nodes with aggregation-based glyphs (e.g., the average of the values in the subgraph for each dimension) or simply use the original node representations above.

A simple algorithm for placement and drawing of hierarchically structured glyphs is given below. To create a more anesthetically pleasing graph, extra space on rows could be used to evenly separate all nodes at a particular level. Additionally, as will be described later, free space can be used to further separate non-sibling nodes at the same level. Finally, for unbalanced trees, space could be used beneath branches that terminate prior to other branches to reduce edges crossing over glyphs. Figure 10 shows the result of this technique on a hierarchical representation of the iris data, where the non-terminal glyphs show the average values for the glyphs below them. Glyphs at a given level have been evenly spaced.

```
Let H = the height of the tree.
Let W = the maximum number of nodes in one level of the hierarchy.
Let C = the canvas size (assumed a square)
Compute S = C / max(H, W) as the maximum size of a glyph

Assume glyph i is at position (l, j) of the hierarchy, where l is the level
number and j is the offspring number (i is the jth child of its parent)

For each node in the hierarchy
   Set its y position (row) to be l * S
   Compute its offset o in the row as the sum of the child counts for the nodes to
      left of its parent
   Set its x position (column) to be (j + o) * S
   Draw the node
   If it isn't the root node, draw a link to its parent
```

Algorithm 2: Hierarchical glyph placement.

An alternative to node-link based visualizations are the filled region methods for conveying (mostly univariate) data. One of the earliest of these, the *Tree-Map* [42], recursively divides the screen space (alternating horizontal and vertical) based on the number and populations of the subtrees, filling the resulting rectangular region with a color based on the dependent variable. A radial version, with aggregations also being depicted, was described in [18]. Wills [79] describes a hierarchical clustering mechanism that allows varying levels of detail to be conveyed with a tree-map structure, thus making it amenable to very large hierarchies. Ward and Keim proposed a variant on tree-maps for positioning glyphs [77]. This method, instead of being space-filling as in the original tree-map, allocates fixed size regions for each glyph, and thus may produce significant amounts of white space. A high-level description of the placement algorithm is outlined below.

If we assume a unit-sized glyph, the space required for a 2-level hierarchy consisting of a root node $R_{0,0}$ with $C_{0,0}$ children (all terminal nodes) is either $(1\ X\ C_{0,0})$ or $(C_{0,0}\ X\ 1)$. The two element subscript indicates the level of the hierarchy and the index of the node at the level specified. For a 3-level hierarchy, where the root has $C_{0,0}$ non-terminal nodes $(R_{1,0}, R_{1,1}, ..., R_{1,C_{0,0}-1})$, each of which has child counts given by $(C_{1,0}, C_{1,1}, ..., C_{1,C_{0,0}-1})$, the space required will be $(C_{0,0}\ X\ max(C_{1,i}))$, where $0 \leq i < C_{0,0}$, or alternately the same dimensions rotated 90 degrees. Thus we can envision the screen first being divided in one direction according to the number of branches off of the root, followed by the terminal nodes laid out in the opposite direction. The space needed for the terminal nodes, assuming we maintain a rectangular area, is the maximum of the number of children of the level one nodes.

Continuing this process, we discover the width and height needed for an arbitrary hierarchy can be computed by alternately multiplying the sum or the maximum of the children counts for nodes as we recurse down the tree. At any given level, either the height or the width will use the sum, while the other uses the maximum. To draw the glyphs then requires each node to be placed relative to its parent's base location (e.g., the lower left corner), using offsets based on its earlier siblings' space requirements. To visually emphasize the family relationships, we draw rectangles to separate siblings, with the width of the rectangle sides decreasing as we get deeper into the tree. Finally, glyphs are drawn as the terminal nodes, with their bounding rectangles encompassing their immediate siblings. An example using the iris data set is shown in Figure 10. From the figure it is clear that the clustering algorithm performed relatively well, as most glyphs within a given block have clear shape and size similarities. There are, however, a few misplaced glyphs, such as the sibling pair in the third row from the bottom of the lower left region.

Another filled region method, called *dimensional stacking* [52], recursively divides the display space based on binning two dimensions at a time, with inner subimages being divided by subsequent pairs of dimensions until all dimensions are used. The final rectangle is colored according to the dependent variable or an aggregation of all points mapping to the corresponding bin. Again, this technique, in conjunction with the constraint of using a uniform-sized region, could be employed to convey multivariate data with glyphs.

Placement strategies for hierarchical structures can be evaluated on a number of aspects, including the limitations on the width and depth of trees that can be effectively accommodated, the screen utilization, and the ease at which relationships between nodes can be perceived. This in turn depends on the degree of clutter caused by the display of links and the amount of control the user has on the view. Most of the techniques work best for balanced trees; however, most real data does not behave in this fashion. Thus evaluation must also include tests with trees of varying degrees of complexity and balance. Also, since most hierarchy visualization techniques concentrate predominantly on the display of the relations, rather than the data at the node, tests must be performed to determine how easy it is to interpret the values depicted at the glyphs positioned at the nodes of the trees. Clearly, link placement becomes more problematic in the presence of glyphs on the display.

4.3 Graph/Network Structure

A generalization of hierarchical structure is that of a graph or network, which consists of a set of nodes (in our case, the data points) and a finite set of links or connections (directed or undirected), each of which represents a relationship between a pair of nodes. As with hierarchical techniques, we can replace the simple node representation from an existing method with our glyph to convey the data. However, unlike hierarchical techniques it is not possible, in general, to avoid drawing some representation of the links, as implying connectivity via positioning can lead to errors in interpretation.

Network visualization has become a very popular research area in recent years. Reichenberger et al [63] developed a theory of diagram design and showed how different relationships can be mapped to different graph styles and the attributes of the nodes and links. Force-based techniques, such as described in [37], are commonly used to automatically place nodes on the screen to convey relations and their strengths. However, these methods can lead to cluttered displays, depending on the data characteristics.

Determining positions for graph structured data is a complex process, as many, often conflicting, factors must be considered [23, 38]. For example, rules of aesthetics, such as minimizing edge crossings or maintaining a relatively uniform distribution of nodes, are common [22]. Drawing conventions, such as using only straight lines or ninety degree bends in the links, also must be adhered to for consistency and ease of interpretation. Constraints may also be imposed on subsets of the graph, such as centering or clustering the elements of the subgraph or ensuring that a particular node is on the outer edge of the subgraph.

As in hierarchically-structured glyphs, perhaps the greatest concern when designing graph-structured placement algorithms is the scalability to large and complex networks. Since the links themselves may convey other information besides the connectivity of the nodes (e.g., traffic volume), it is important to allocate sufficient screen space and user control to these information-rich components of the display. Indeed, the links can be considered a secondary form of glyph, as their style, size, color, and shape can all be controlled by different aspects of the relationship. Graph-structured glyphs also have a disadvantage over hierarchies in that there is not a clear method to navigate through the structure. Thus there must be support for elevating arbitrary nodes to the center of focus and examining neighborhoods of this focus, rather than simply providing drill-down and roll-up mechanisms as found in hierarchical structures.

4.4 Distortion Techniques for Structure-Driven Layouts

Given an initial position assignment for structure-based techniques, we can choose to distort the results to attain other desirable properties. Distortion techniques have long been used in visualization to emphasize subsets of data while maintaining context and to deal with clutter resulting from large data sets. Examples include fish-eye and other lens techniques [53, 70, 75], pliable surfaces [12], and mapping to hyperbolic space [60]. Shape distortion has also been used to convey additional information, such as the area-based methods found in some cartographic algorithms. Position distortion has been employed in cartography to avoid overlaps and congestion, with the understanding that in many circumstances, the degree of accuracy needed varies considerably. As mentioned in the discussion of data-driven techniques, random jitter and algorithms to remove overlaps may also be applied. Luders and Ernst used a combination of node scaling and layout reorganization to improve the readability of graphs[57]. Beaudoin et al [6] use controlled overlapping to compress large hierarchies, with a variety of tools to support navigation and exploration of the stacked space.

An alternate goal of position distortion is to emphasize differences. Space padding can be used to separate glyphs which have data-specific differences which are not sufficiently reflected in the screen positions. For example, Figure 11 shows an ordered raster placement with spacing between glyphs in a given row proportional to their n-dimensional distance. The algorithm for this is given below.

```
Let G be the largest gap the user wishes between glyphs, with gap units equal to
  the maximum glyph size

Compute Di as the N-D distance from glyph i to glyph i+1
Compute Dm as the maximum N-D distance between adjacent glyphs

For each glyph i, compute the spacing to the next glyph as 1 + G * Di / Dm

Compute the sum S of all spacings, which includes the space for drawing the glyphs
Compute R, the number of glyph spaces per row, which is sqrt(s) + .5
Compute the maximum size for a glyph, which is W / R (W is canvas width)

position = 0
For i = 0; i < M; i++ {
```

```
draw glyph i at row = (position/R) and column = (position modulo R)
add spacing i to position
}
```

Algorithm 3: inserting white space to emphasize differences in adjacent glyphs.

As another example, a tree drawing algorithm often must put nodes from different subtrees adjacent to each other. By adjusting positions such that inter-sibling distances are significantly smaller than between less-related nodes sharing a level of the tree, node relationships become clearer (see Figure 12). However, this reduces the number of glyphs one may be able to display without overlap. Other ways to emphasize these differences would be via small vertical offsets (e.g., alternating between two y-values at each non-sibling transition along a row of the hierarchy) or coloring the nodes, links, or background between non-sibling adjacencies.

The concept of an acceptable level of overlapping is a crucial one; in many cases, minor, or even major, overlaps don't detract from the message being conveyed in the visualization. The ideal solution might be to allow user control of the overlapping, with smooth animation of positions between overlapped and non-overlapped states to allow easy tracking of relationships between the glyphs. This has been demonstrated in several 3-D tree visualization systems, such as reconfigurable disk trees[41]. Distortion techniques that permit compression and expansion of arbitrary sections of the structure, while maintaining context via animated transitions, can allow users to build a mental model of the regions of interest within the data without being overly constrained by the display resolution. While ideally the system itself would analyze the congestion on the screen and attempt to reduce overlap via distortion, to enable the system to understand the user's task in sufficient detail to automatically perform this process reliably has not yet been achieved.

5 Summary of Placement Strategies

In this section we attempt to get a feel for the number of possible layout algorithms that can be applied to a given data set, and then give some guidelines on how one might go about selecting one or more algorithms to test. It should be noted that, to date, no formal, extensive evaluations and/or user studies have been performed on this problem, although it is clearly a desirable future goal. Some studies focusing on subsets of the techniques have been reported (e.g., [41, 75]), and evaluation metrics for information visualization have been proposed that involve issues such as density, dimensionality, occlusion, and number of identifiable points [10]. However, we have not found a comprehensive study of glyph layout strategies in the literature.

5.1 The Size of the Design Space

One might wonder what the total number of distinct ways a set of N-D glyphs can be positioned on a 2-D data space. The rather simplistic computations below cover a significant subset of the methods described in this paper. Clearly adding the third display dimension (not to mention the time dimension) would increase this number considerably, and visually exploring the entire space of alternatives would be a daunting task.

data-driven (raw): $(1 + D)(N^2 - N)$, where D is the number of distortion techniques (e.g., random jitter, relaxation)

data-driven (derived): $(1 + D)\sum_{j=1}^{j\leq k} p_j$, where k is the number of distinct derivation algorithms and p_j is the number of variations within algorithm j (e.g., different distance metrics).

structure-driven (linear ordered): $N \times FP(1 + SP + OL)$, where FP is the number of filling patterns, SP is the number of methods which introduce white space for separation (space-padding), and OL is the number of methods which distort the placement to allow overlaps.

5.2 Selecting a Placement Algorithm

In selecting an appropriate method, one must consider several factors, including the characteristics of the data (size, distribution), the purpose of the visualization (presentation, confirmation, exploration) and the specific task(s) at hand (e.g., detection, classification, or measurement of patterns or outliers). An additional critical component in the selection process is the skills of the prospective user of the visualization. Some visualization methods require significant training to interpret correctly.

Domain knowledge also plays a significant role in selecting meaningful strategies. In data driven approaches, there are usually mappings of data dimensions to position that make more intuitive sense than others. Similarly, the dimensions used to control structure-driven techniques, whether it be the key for sorting or the order of dimensions used for layering a hierarchy, are usually best left to selection by the user.

Placement techniques often force the user to trade off between efficient screen utilization, the degree of occlusion, and distortion of the values being used to position the glyphs. Ideally, the user should be able to dynamically adjust the mapping to strengthen or weaken the relative importance of each of these factors. Thus rather than selecting a single visualization technique, he or she would have a continuum of interrelated methods. This is a relatively unexplored area in the development of glyph placement strategies, though static constraint satisfaction algorithms have been applied to graph drawing tasks in the past.

Another question the user must ask is whether to impose a structure on the data in the event that no explicit structure already exists. While the structure provides relational information that might not be readily perceivable in data-driven approaches, the results are often not unique and may convey misleading or erroneous information. For example, clustering algorithms require the specification of a distance or similarity metric to process the data, and it is often difficult to capture the relative interdependencies between data dimensions or their contextual importance within this metric. Again, user control of the process is vital, and this in turn requires users to be knowledgeable not only of the semantics of their data but also of the structuring algorithm being utilized.

As an initial, though not exhaustive or definitive, set of guidelines for selecting a layout strategy, we look at two of the important design criteria: data characteristics and user task. The table in Figure 13 displays several points within this design space, and a recommendation for a starting layout. Each can be augmented with distortion depending on the degree of overlap present and amount of empty space on the screen, as well as the desired accuracy of placement for data-driven methods. Note that these recommendations are not infallible, but reasonable starting points. Other methods, such as imposing structure on unstructured data or ignoring the structure in a structured set, may also prove insightful. The main point is that no single placement strategy is good for all tasks and data characteristics, which implies that for other than very limited circumstances, support should be provided for a variety of layout methods.

6 Conclusions

The visualization of multivariate data is a task of growing importance in a wide range of disciplines, and numerous techniques have been proposed and employed. In this paper we have focused on N-dimensional glyphs as a visualization component, and have presented a taxonomy of techniques for positioning the glyphs on the screen. Methods were divided between data-driven and structure-driven approaches, and a variety of post-processing distortions were described.

Our future work will involve consolidating the techniques described in this paper into a single application to facilitate a comprehensive analysis of the applicability and limitations of each placement algorithm. A wide assortment of customizable glyphs will be supported, similar to the Glyphmaker program [64]. Usability and perceptual testing will also be conducted. Finally, we are interested in exploring the use of animation to help overcome some of the deficiencies found in the static presentation of multivariate data with glyphs. For example, smoothly animating the transformation of one placement strategy to another might convey relational information not readily discernible in either static display.

Acknowledgments

This work was partially supported under NSF Grants IIS-9732897, IIS-0119276, and EIA-9729878. Special thanks to Jing Yang for her help in implementing the algorithms used to generate the figures in this paper.

References

[1] Ahlberg C, Wistrand E. *IVEE: an information visualization and exploration environment.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 66–73.

[2] Anderson E. A semigraphical method for the analysis of complex problems. *Proceedings of the National Academy of Science* 1957; **13:**923–927.

[3] Andreae P, Dawkins B, O'Connor P, DySect: an incremental clustering algorithm. [Document included with public-domain software] retrieved from StatLib at CMU (`http://lib.stat.cmu.edu/general`)(accessed 21 September 2002).

[4] Ankerst M, Berchtold S, Keim D. *Similarity clustering of dimensions for an enhanced visualization of multidimensional data.* Proc. Information Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 52–60.

[5] Assa J, Kohen-Or D, Milo T. *Displaying data in multidimensional relevance space with 2D visualization maps.* Proc. Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 127–134.

[6] Beaudoin L, Parent M-A, Vroomen L. *Cheops: a compact explorer for complex hierarchies.* Proc. Visualization '96 (San Francisco, CA, 1996). IEEE Computer Society Press: Los Alamitos, CA; 87–92.

[7] Becker R, Eick S, Wilks A. Visualizing network data. *IEEE Transactions on Visualization and Computer Graphics* 1995; **1:**16–28.

[8] Beddow J. *Shape coding of multidimensional data on a microcomputer display.* Proc. Visualization '90 (San Francisco, CA, 1990). IEEE Computer Society Press: Los Alamitos, CA; 238–246.

[9] Bentley C, Ward M. *Animating multidimensional scaling to visualize n-dimensional data sets.* Proc. Information Visualization '96 (San Francisco, CA, 1996). IEEE Computer Society Press: Los Alamitos, CA; 72–73.

[10] Brach R. *Metrics for effective information visualization.* Proc. Information Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 108–111.

[11] Brodbeck D, Chalmers M, Lunzer A, Cotture P. *Domesticating Bead: adapting an information visualization system to a financial institution.* Proc. Information Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 73–80.

[12] Carpendale T, Cowperthwaite D, Fracchia F. *3 dimensional pliable surfaces for the effective presentation of visual information.* Proc. ACM Symposium on User interface Software and Technology (1995); 217–226.

[13] Carriere J, Kazman R. *Interacting with huge hierarchies: beyond cone trees.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 74-81.

[14] Chalmers M. *A linear iteration time layout algorithm for visualizing high-dimensional data.* Proc. Visualization '96 (San Francisco, CA, 1996). IEEE Computer Society Press: Los Alamitos, CA; 127–132.

[15] Chernoff H. The use of faces to represent points in k-dimensional space graphically. *Journal of the American Statistical Association* 1973;**68:** 361–368.

[16] Christensen J, Marks J, Shieber S. An empirical study of algorithms for point-feature name placement. *ACM Transactions on Graphics* 1995; **14:** 203-232.

[17] Chuah M, Roth S, Mattis J, Kolojejchick J. *SDM: malleable information graphics.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 36–42.

[18] Chuah M. *Dynamic aggregation with circular visual designs.* Proc. Information Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 35–43.

[19] Chuah M, Eick S. Information rich glyphs for software management data. *IEEE Computer Graphics and Applications* 1998; **18:** 24–29.

[20] Cleveland W, McGill R. Graphical perception: theory, experimentation and application to the development of graphical methods. *Journal of the American Statistical Association* 1984; **79:** 531–554.

[21] Cleveland W. *Visualizing Data.* Hobart Press: Summit, NJ; 1993.

[22] Davidson R, Harel D. Drawing graphs nicely using simulated annealing. *ACM Transactions on Graphics* 1996; **15:** 301–331.

[23] Di Battista G, Eades P, Tamassia R, Tollis I. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall: Upper Saddle River, NJ; 1999. 397pp.

[24] Dovey D. *Vector plots for irregular grids.* Proc. Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 248–253.

[25] du Toit S, Steyn A, Stumpf R. *Graphical Exploratory Data Analysis.* Springer-Verlag: Berlin; 1986.

[26] Ebert D, Rohrer R, Shaw C, Panda P, Kukla J, Roberts D. *Procedural shape generation for multi-dimensional data visualization.* Proc. Data Visualization '99 (1999), Springer-Verlag: Berlin; 3-12.

[27] Eick S, Wills G. *Navigating large networks with hierarchies.* Proc. Visualization '93 (San Jose, CA, 1993). IEEE Computer Society Press: Los Alamitos, CA; 204–210.

[28] Friedman J, Farrell E, Goldwyn R, Miller M, Sigel J. *A graphic way of describing changing multivariate patterns.* Proc. Sixth Interface Symposium on Computer Science and Statistics (1972); 56-59.

[29] Friedman J, Tukey J. A projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers* 1974; **C-23:** 881-889.

[30] Fuhrmann A, Groller E. *Real-time techniques for 3D flow visualization.* Proc. Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 305–312.

[31] Gershon N. From perception to visualization. In: Rosenblum L, et. al. (Eds). *Scientific Visualization: Advances and Challenges.* Academic Press: San Diego, CA; 1994.

[32] Globus A, Levit C, Lasinski T. *A tool for visualizing the topology of three-dimensional vector fields.* Proc. Visualization '91 (San Diego, CA, 1991). IEEE Computer Society Press: Los Alamitos, CA; 33-40.

[33] Gnanadesikan R. *Methods of Statistical Data Analysis of Multivariate Observations.* John Wiley and Sons: New York; 1977.

[34] Gross M, Sprenger T, Finger J. *Visualizing information on a sphere.* Proc. Information Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 11–16.

[35] Hartigan J. Printer graphics for clustering. *Journal of Statistical Computing and Simulation* 1975; **4:** 187-213.

[36] Helman J, Hesselink L. Visualizing vector field topology in fluid flows. *Computer Graphics and Applications* 1991; **11:** 36-46.

[37] Hendley R, Drew N, Wood A, Beale R. *Narcissus: visualizing information.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 90–96.

[38] Herman I, Melancon G, Marshall M. Graph visualization and navigation in information visualization. *IEEE Transactions on Visualization and Computer Graphics* 2000; **6:** 24-43.

[39] Hoffman P, Grinstein G, Marx K, Grosse I, Stanley E. *DNA visual and analytic data mining.* Proc. Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 437–441.

[40] Huber P. Projection pursuit. *Annals of Statistics* 1985; **13:** 435-475.

[41] Jeong C, Pang A. *Reconfigurable disc trees for visualizing large hierarchical information space.* Proc. Information Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 19–25.

[42] Johnson B, Shneiderman B. *Tree-maps: a space-filling approach to the visualization of hierarchical information structures.* Proc. Visualization '91 (San Diego, CA, 1991). IEEE Computer Society Press: Los Alamitos, CA; 189–194.

[43] Jolliffe J. *Principal Component Analysis.* Springer Series on Statistics, Springer: Berlin; 1986.

[44] Keim D, Kriegel H. VisDB: database exploration using multidimensional visualization. *Computer Graphics and Applications* 1994; 40-49.

[45] Keim D, Kriegel H, Ankerst M. *Recursive pattern: a technique for visualizing very large amounts of data.* Proc. Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 279–286.

[46] Keim D, Hermann A. *The Gridfit algorithm: an efficient and effective approach to visualizing large amounts of spatial data.* Proc. Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 181–188.

[47] Klassen R, Harrington S. *Shadowed hedgehogs: a technique for visualizing 2D slices of 3D vector fields.* Proc. Visualization '91 (San Diego, CA, 1991). IEEE Computer Society Press: Los Alamitos, CA; 148–153.

[48] Kleiner B, Hartigan J. Representing points in many dimension by trees and castles. *Journal of the American Statistical Association* 1981; **76:** 260-269.

[49] Kerlick G. *Moving iconic objects in scientific visualization.* Proc. Visualization '90 (San Francisco, CA, 1990). IEEE Computer Society Press: Los Alamitos, CA; 124–130.

[50] Kohonen T. *Self-Organizing Maps.* Springer Series on Information Sciences, Vol. 30, Springer: Berlin; 1995.

[51] Kruskal J, Wish M. *Multidimensional Scaling.* Sage Publications: London; 1978.

[52] LeBlanc J, Ward M, Wittels N. *Exploring n-dimensional databases.* Proc. Visualization '90 (San Francisco, CA, 1990). IEEE Computer Society Press: Los Alamitos, CA; 230–237.

[53] Leung Y, Apperley M. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction* 1994; **1:** 126-160.

[54] Levkowitz H. *Color icons: merging color and texture perception for integrated visualization of multiple parameters.* Proc. Visualization '91 (San Diego, CA, 1991). IEEE Computer Society Press: Los Alamitos, CA; 164–170.

[55] Levkowitz H. *Color Theory and Modeling for Computer Graphics, Visualization, and Multimedia Applications.* Kluwer: Boston; 1997.

[56] Lipchak B, Ward M. *Visualization of cyclic multivariate data.* Proc. Visualization '97 Late Breaking Hot Topics (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 61–64.

[57] Luders P, Ernst R. *Improving browsing in information by the automatic display layout.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 26–33.

[58] Mezzich J, Worthington D. A comparison of graphical representations of multidimensional psychiatric diagnostic data. In: Wang P (Ed.) *Graphical Representation of Multivariate Data.* Academic Press: New York; 1978.

[59] Mukherjea S, Hara Y. *Visualizing world wide web search engine results.* Proc. International Conference on Information Visualization (IV '99) (1999); 400–405.

[60] Munzner T. *H3: laying out large directed graphs in 3D hyperbolic space.* Proc. Information Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 2–10.

[61] Pickett R, Grinstein G. *Iconographic displays for visualizing multidimensional data.* Proc. IEEE Conference on Systems, Man, and Cybernetics (1988); 164-170.

[62] Post F, Walsum T, Post F, Silver D. *Iconic techniques for feature visualization.* Proc. Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 288–295.

[63] Reichenberger K, Kamps T, Golovchinsky G. *Towards a generative theory of diagram design.* Proc. Information Visualization '95 (Atlanta, GA, 1995), IEEE Computer Society Press: Los Alamitos, CA; 11–18.

[64] Ribarsky W, Ayers E, Eble J, Mukherjea S. Glyphmaker: creating customized visualizations of complex data. *Computer* 1994; **27:** 57-64.

[65] Robertson G, Mackinlay J, Card S. *Cone trees: animated 3D visualizations of hierarchical information.* Proc. ACM SIGCHI Conference on Human Factors in Computing Systems (1991); 331-337.

[66] Rohrer R, Ebert D, Sibert J. *The shape of Shakespeare: visualizing text using implicit surfaces.* Proc. Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 121–129.

[67] Rose S. *The Sunflower visual metaphor, a new paradigm for dimensional compression.* Proc. Information Visualization '99 (San Francisco, CA, 1999). IEEE Computer Society Press: Los Alamitos, CA; 128–131.

[68] Rossi A, Varga M. *Visualization of massive retrieved newsfeeds in interactive 3D.* Proc. IEEE International Conference on Information Visualization (IV '99) (1999); 12–17.

[69] Rushmeier H, Lawrence R, Almasi G. *Visualizing customer segmentation produced by self organizing maps.* Proc. Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 463–466.

[70] Schaffer D, Zuo Z, Greenberg S, Bartram L, Dill J, Dubs S, Roseman M. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human Interactions* 1996; **3:**162-188.

[71] Schroeder W, Volpe C, Lorensen W. *The Stream Polygon: a technique for 3D vector field visualization.* Proc. Visualization '91 (San Diego, CA, 1991). IEEE Computer Society Press: Los Alamitos, CA; 126–132.

[72] Shneiderman B. *The eyes have it: a task by data type taxonomy for information visualization.* Proc. 1996 IEEE Symposium on Visual Languages (1996); 336-343.

[73] Siegel J, Farrell E, Goldwyn R, Friedman H. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery* 1972;**72:** 126–141.

[74] Spoerri A. *InfoCrystal: a visual tool for information retrieval.* Proc. Visualization '93 (San Jose, CA, 1993). IEEE Computer Society Press: Los Alamitos, CA; 150–157.

[75] Storey M, Wong L, Fracchia F, Muller H. *On integrating visualization techniques for effective software exploration.* Proc. Information Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 38–45.

[76] Ward M. *XmdvTool: integrating multiple methods for visualizing multivariate data.* Proc. Visualization '94 (Washington, DC, 1994). IEEE Computer Society Press: Los Alamitos, CA; 326–333.

[77] Ward M, Keim D. *Screen layout methods for multidimensional visualization.* Proc. CODATA Euro-American Workshop on Visualization of Information and Data (Paris, France, 1997); 2pp.

[78] Ward M, Lipchak B. A visualization tool for exploratory analysis of cyclic multivariate data. *Metrika* 2000; **51:** 27-37.

[79] Wills G. *An interactive view for hierarchical clustering.* Proc. Information Visualization '98 (Research Triangle Park, NC, 1998), IEEE Computer Society Press: Los Alamitos, CA; 26–31.

[80] Wilson R, Bergeron R. *Dynamic hierarchy specification and visualization.* Proc. Information Visualization '99 (San Francisco, CA, 1999). IEEE Computer Society Press: Los Alamitos, CA; 65–72.

[81] Wittenbrink C, Pang A, Lodha S. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics* 1996; **2:** 266-279.

[82] Wong P, Bergeron R. *Multivariate visualization using metric scaling.* Proc. Visualization '97 (Phoenix, AZ, 1997). IEEE Computer Society Press: Los Alamitos, CA; 111–118.

[83] Woodruff A, Landay J, Stonebraker M. *Constant density visualization of non-uniform distributions of data.* Proc. ACM Symposium on User Interface Software and Technology (1998); 19–28.

Figure 1: Examples of glyphs: variations on profiles; stars/metroglyphs; stick figure icons and trees; autoglyphs and boxes; faces; arrows and weathervanes.

Figure 2: Taxonomy of glyph placement strategies.

Figure 3: All pairwise raw data-driven views (star glyphs) of the Iris data set: (a) sepal length vs. sepal width, (b) sepal length vs. petal length, (c) sepal length vs. petal width, (d) sepal width vs. petal length, (e) sepal width vs. petal width, and (f) petal length vs. petal width.

Figure 4: Star glyphs of the Iris and cars data sets with position based on the first two principal components.

Figure 5: Star glyphs of 7 dimensions of the breakfast cereal data set with position based on fiber and sugar levels. The left image is the original mapping, and the right was distorted to reduce the level of overlap.

Figure 6: Various placement patterns for linearly structured data (from [44, 45]), including raster, radial, and recursive raster.

Figure 7: Profile glyphs of monthly economic time series data using linear ordering and spiral layout (from SpiralGlyphics[56, 78]). Dimensions are (from left to right) Dow Jones average, Standard and Poors 500 index, retail sales, and unemployment.

Figure 8: Star glyphs of remote sensing data from the Grant's Pass region of Western Australia (5 channels) using 2-D raster ordering(from XmdvTool[76]). Second image shows a zoomed in region highlighting an area with high Uranium values.

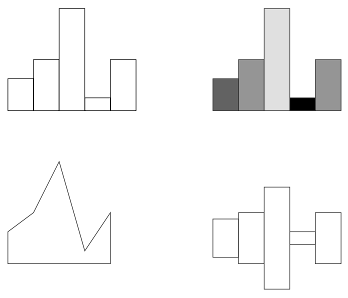
Figure 9: Profile glyphs of average values from hierarchical partitioning of the Iris data using the DySect partitioning algorithm [3].

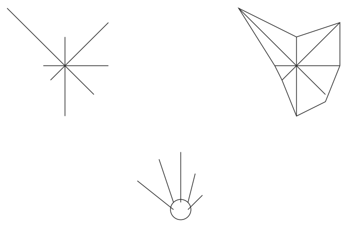
Figure 10: Star glyphs of Iris data set using recursive hierarchical placement based on the DySect [3] data partitioning algorithm and the Tree-map [42] recursive space partitioning technique[77].

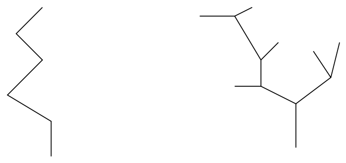
Figure 11: Star glyphs of the iris data set, sorted by sepal length, with spacing inserted at positions of significant variation between adjacent glyphs.

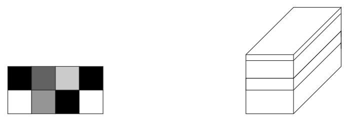
Figure 12: Profile glyphs of average values from hierarchical partitioning of the Iris data using the DySect partitioning algorithm [3], with added space placed between adjacent non-siblings to emphasize sibling relationships.

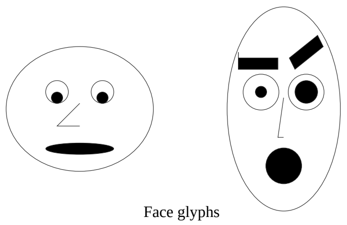
Figure 13: Sample points in a design space of glyph layout strategies.

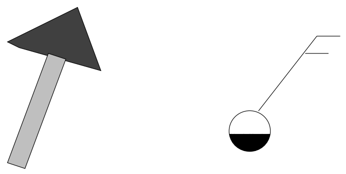

Variations on Profile glyphs

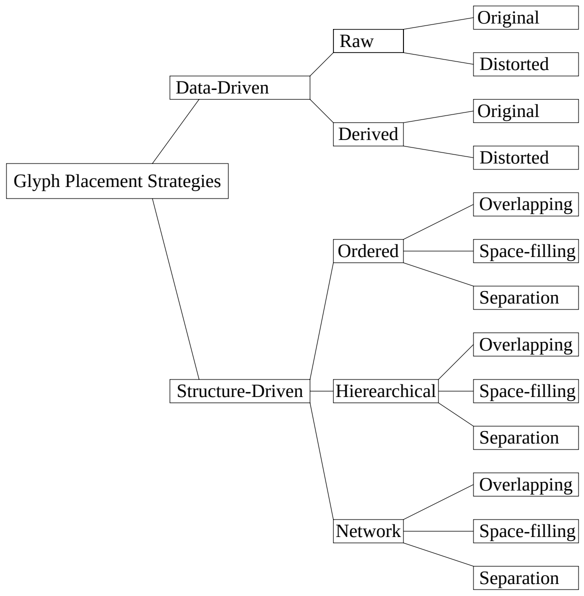

Stars and Anderson/metroglyphs

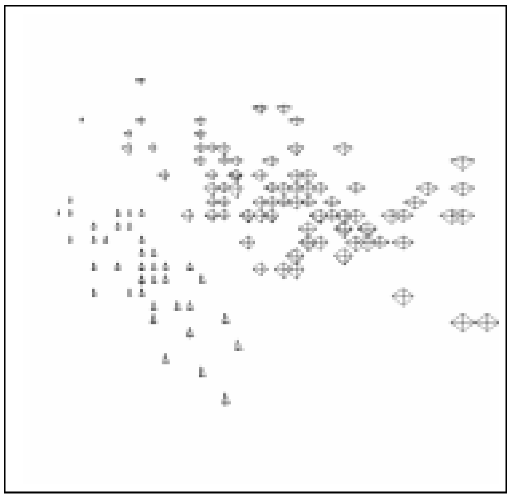

Sticks and Trees

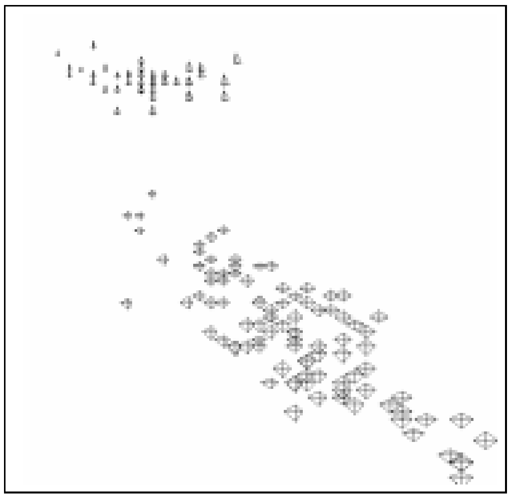

Autoglyph and box glyph

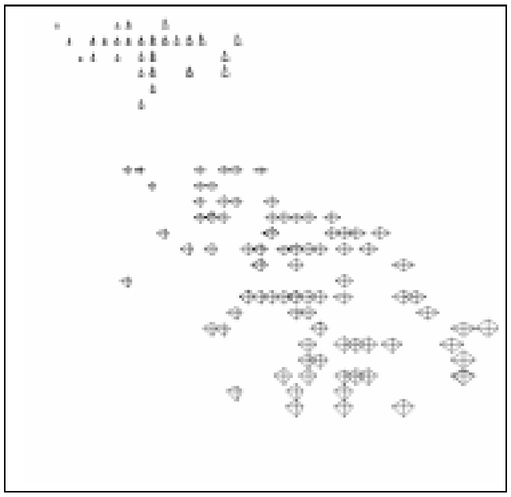

Face glyphs

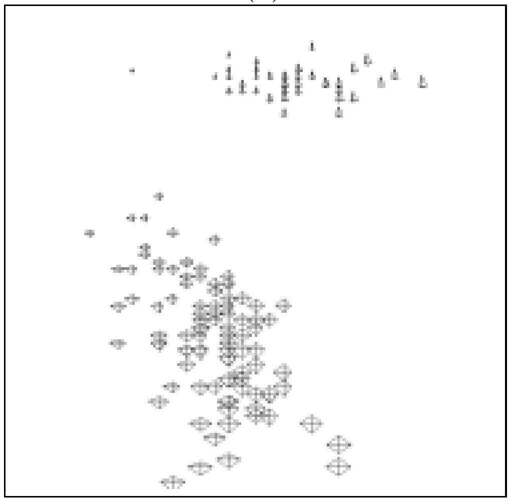

Arrows and Weathervanes

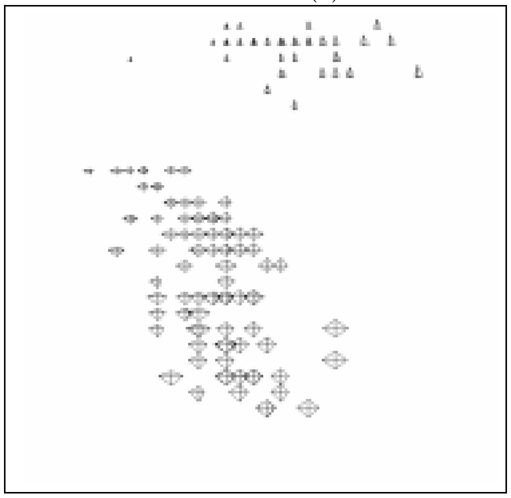


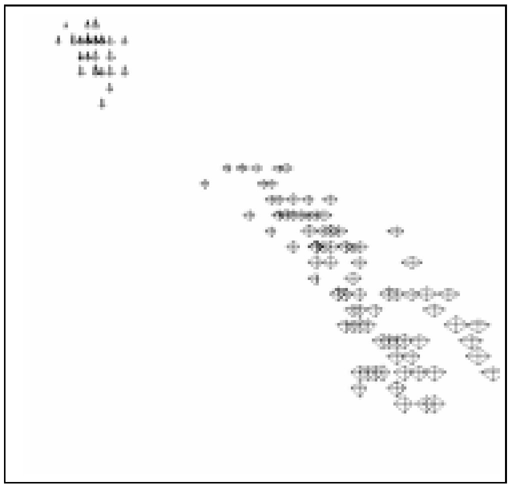

(a)

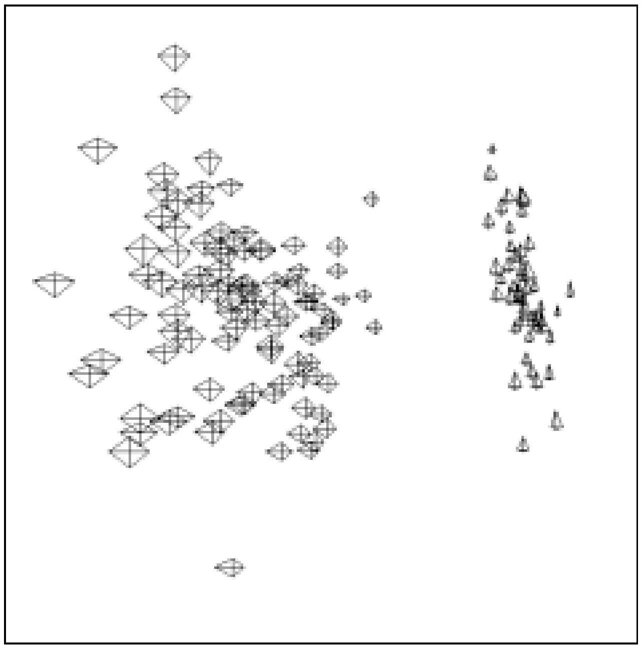

(b)

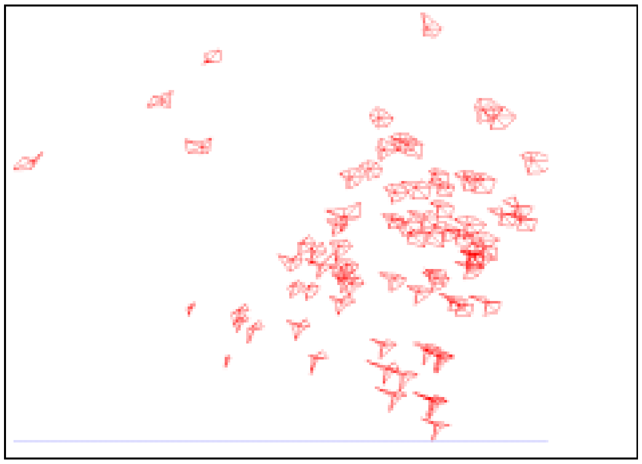

(c)

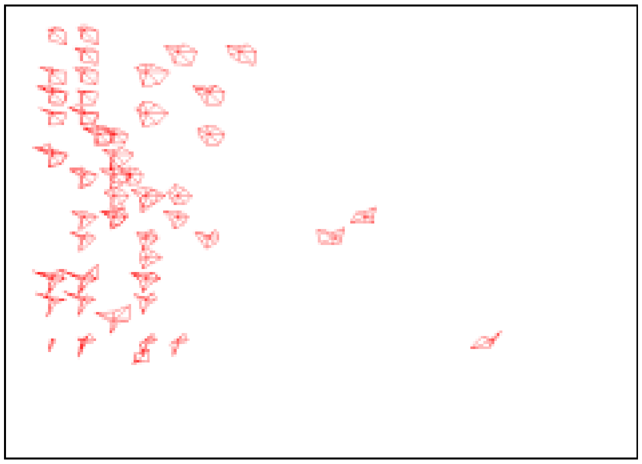

(d)

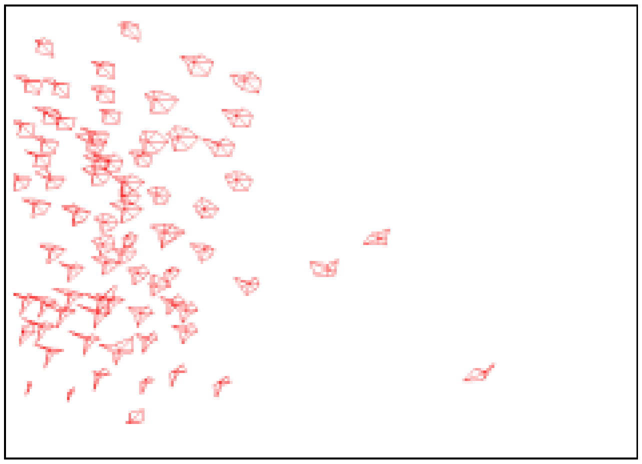

(e)

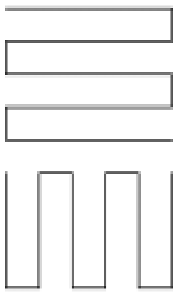

(f)

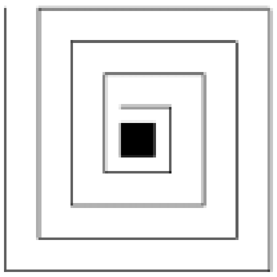


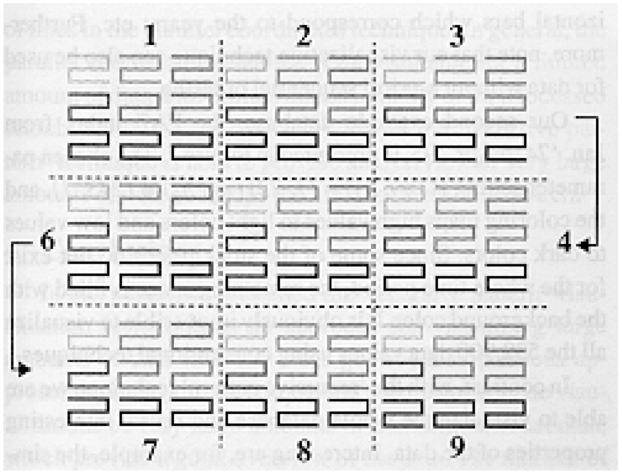


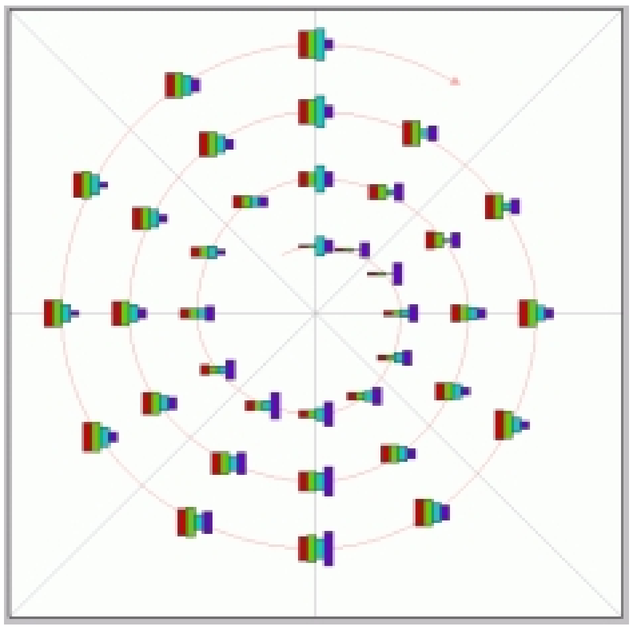


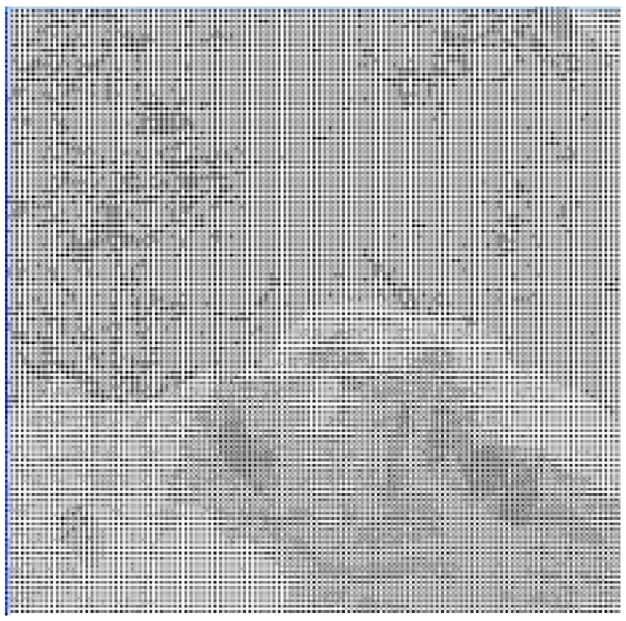


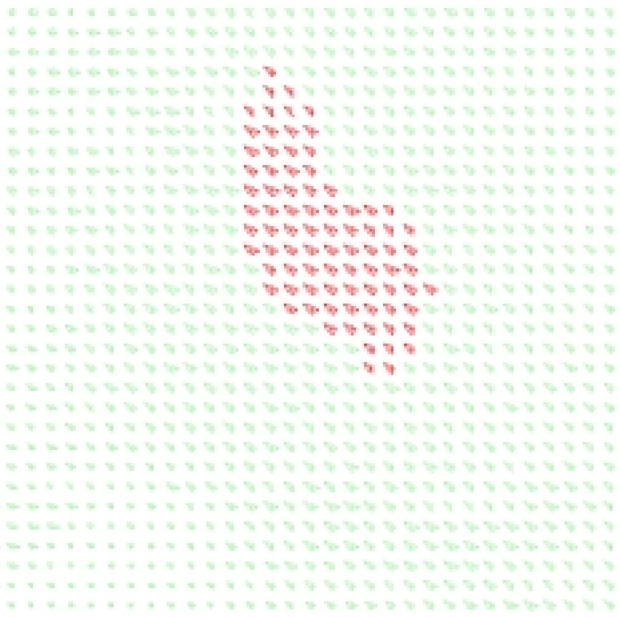


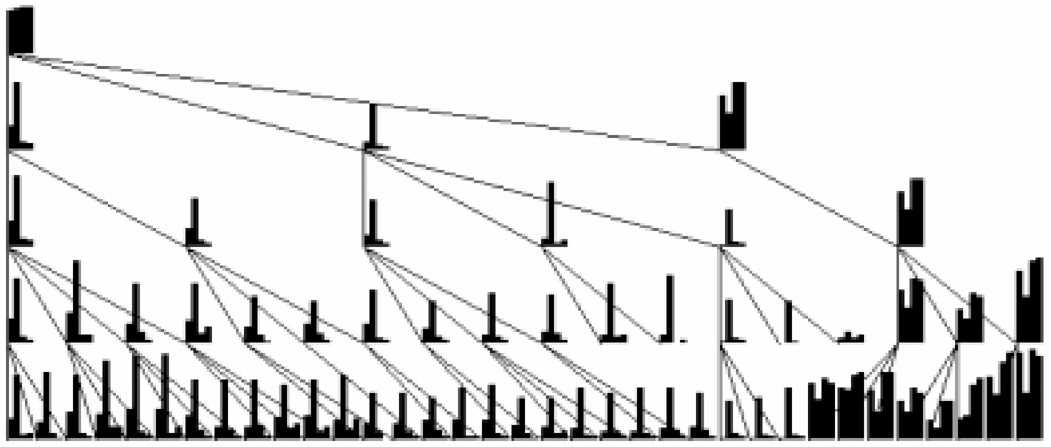


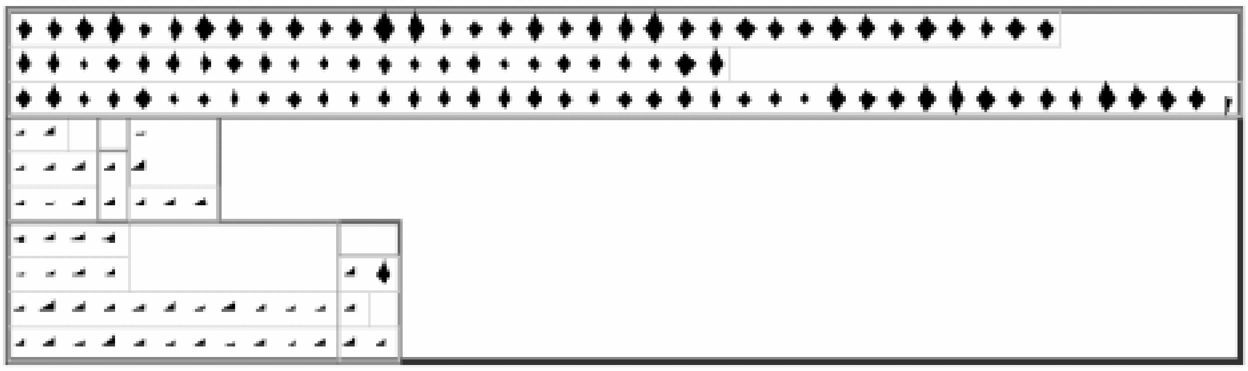

1
2
3
6
4
7
8
9

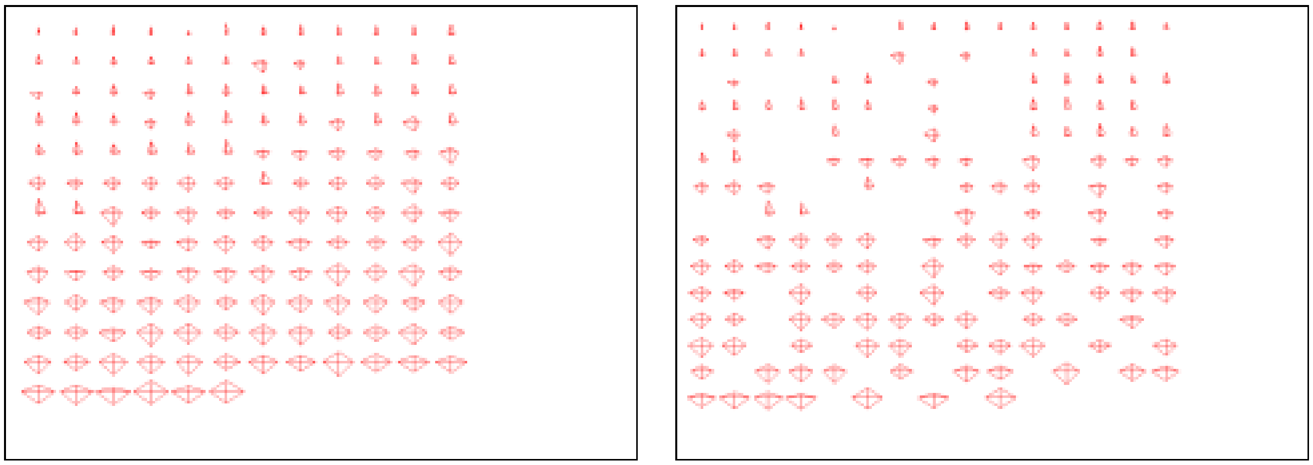


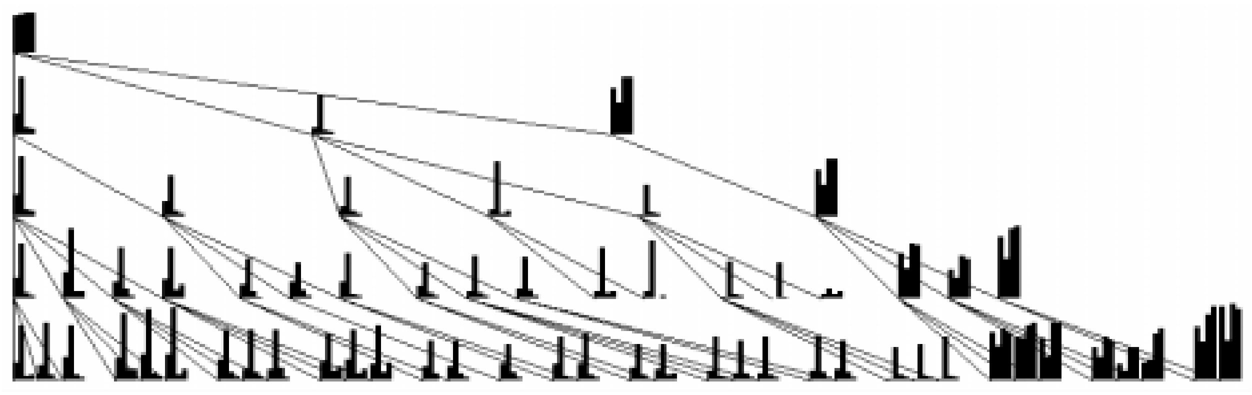












Data Characteristics	User Task	Recommended Strategy
Small to moderate size	univariate analysis	sorted structure-driven
Small to moderate size	bivariate analysis	raw data-driven
Small to large size	outlier detection	raw or derived data-driven
Moderate to large size	cluster analysis	derived data-driven, e.g., MDS
Small to moderate size	cluster analysis	hierarchical structure-driven after imposing a hierarchy
Small to moderate size, relational	link analysis	network structure-driven