Hierarchical Aggregation for Information Visualization: Overview, Techniques and Design Guidelines

Niklas Elmqvist, *Member, IEEE*, and Jean-Daniel Fekete, *Member, IEEE*

Abstract—We present a model for building, visualizing, and interacting with multiscale representations of information visualization techniques using hierarchical aggregation. The motivation for this work is to make visual representations more visually scalable and less cluttered. The model allows for augmenting existing techniques with multiscale functionality, as well as for designing new visualization and interaction techniques that conform to this new class of visual representations. We give some examples of how to use the model for standard information visualization techniques such as scatterplots, parallel coordinates, and node-link diagrams, and discuss existing techniques that are based on hierarchical aggregation. This yields a set of design guidelines for aggregated visualizations. We also present a basic vocabulary of interaction techniques suitable for navigating these multiscale visualizations.

Index Terms—Aggregation, clustering, clutter reduction, massive datasets, visual exploration, visual analytics.

1 Introduction

Overview is one of the most basic user tasks for information visualization [65] and it is a vital building block for the more complex process of visual exploration—using visualization to form, test, and validate hypotheses about complex or large datasets [53]. However, overview is becoming increasingly difficult to effectively achieve with the ever-increasing size of real-world datasets. For most basic visualization techniques that endeavor to show each item in a dataset—such as scatterplots [19], parallel coordinates [49], and treemaps [63]—a massive number of items will overload the visualization, resulting in clutter that both causes technical scalability problems [33] as well as hinders the user's understanding of its structure and contents [22, 57].

New visualization techniques, such as dense pixel displays [52], have been proposed for dealing with large datasets, but most of these approaches still attempt to draw each item in the dataset. This is not practical for massive datasets. Another solution to this data overload problem, be it technical or perceptual in nature, is to introduce abstraction that reduces the amount of items to display, either in data space or in visual space [22]; examples include hierarchical parallel coordinates [34], color histograms [29, 30], and clustered time-series data [75].

In this article, we draw from this existing work on data abstraction to present a model for transforming virtually any visualization technique into a multiscale structure using hierarchical aggregation. The model is based on coupling aggregation in data space with a corresponding visual representation of the aggregation as a visual entity in the graphical space. This *visual aggregate* can convey additional information about the underlying contents, such as an average value, minima and maxima, or even its data distribution. Any visualization technique that implements this basic model becomes a multiscale structure, and can be interacted with using generic interaction techniques for multiscale navigation. In this article, we attempt to unify the design space for this class of visualization techniques, showing implementations for a number of visual representations such as scatterplots, parallel coordinates, and node-link diagrams. We also give a basic vocabulary of interaction techniques for aggregated visualizations.

There are several benefits to having this kind of model: (*i*) it represents a way to turn **existing** visualization techniques into multiresolution versions that scale better to massive datasets; (*ii*) it suggests a method for designing **new** visualization techniques that directly support multiresolution functionality; and (*iii*) it gives users a unified way of interacting with this class of multiscale visualizations.

- *Niklas Elmqvist is with Purdue University in West Lafayette, IN, USA, E-mail: elm@purdue.edu.*
- *Jean-Daniel Fekete is with INRIA in Paris, France, E-mail: jean-daniel.fekete@inria.fr.*

Submitted to IEEE Transactions on Visualization and Computer Graphics. Do not redistribute.

This article is organized as follows: we first present the related work on classifying information visualization techniques. We then describe the idea behind visual aggregation as a method of data abstraction. We go on to study some examples of hierarchical aggregation in visualization techniques to see how they fit our model. This yields a set of guidelines for how to design and implement hierarchical aggregated visualizations. We exemplify the model and the design guidelines by showing how to add aggregation to a standard zoomable treemap [14, 63] as well as simple line graphs. We close the article with our conclusions and ideas on how to extend this work in the future.

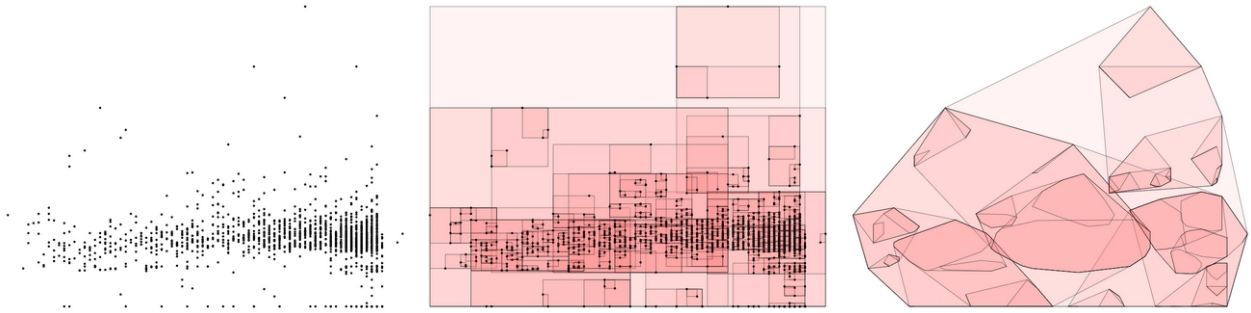
2 Related Work

Data abstraction for the purposes of reducing visual clutter and dataset size is not a novel idea. Shneiderman's taxonomy of information visualization tasks [65] touches upon important data abstraction operations such as filter and zoom as well as the importance of an overview. Oliveira and Levkowitz [23] survey the use of visualization for data mining and discuss some standard data mining techniques for abstracting large-scale datasets, including dimension reduction (e.g., Principal Component Analysis [3]), subsetting (e.g., random sampling [25, 56]), segmentation (e.g., cluster analysis [50, 51]), and aggregation. Our work in the present article is based on the latter technique—hierarchical aggregation of data items, where the original data items are used to create new and aggregated items.

Andrienko and Andrienko [4] give a coherent treatment of data aggregation and its statistical foundations in exploratory data analysis [73], including a wide range of examples and techniques. On a related note, Billard and Diday [13] introduce symbolic data analysis for statistical analysis of very large datasets. Many of the techniques described in the present article draw inspiration from this large body of knowledge, but our focus here is on hierarchical aggregation that support more dynamic visual exploration as opposed to the static aggregations traditionally employed in statistics. These techniques can often be adapted for hierarchical aggregation, both in data and in visual space.

Most recently, Ellis and Dix [27] present a taxonomy that captures virtually all aspects and strategies of data abstraction for visualization. They identify 11 different strategies for data abstraction and give examples of each, classifying existing work into an annotated taxonomy using a set of criteria derived from the literature. Their approach involves three rough groupings of strategies: appearance (affecting the visual representation of individual data items), spatial distortion (displacing visual data items), and temporal (animation).

However, the work of Ellis and Dix ignores the potentially useful distinction between performing abstraction in data space and in visual space that is highlighted by Cui et al. [22]. More specifically, some abstraction methods—including filtering [64], clustering [50, 51], and



(a) 2D scatterplot visualization. (b) 2D bounding box aggregation. (c) 2D convex hull aggregation.

Fig. 1. Hierarchical visual aggregation of a 2D scatterplot visualization. Note that scatterplots are overlapping, so occlusion can occur.

sampling [25]—operate on the actual items in the dataset, whereas others—such as distortion [55] and zooming [58]—affect the visual presentation of the items. However, many authors fail to acknowledge the benefit of providing a link between abstraction methods in data space and their representation in visual space.

Drawing visual representations of abstractions performed in data space support creating simplified versions of a visualization while still retaining the general overview [34]. By dynamically changing the abstraction parameters, the user can also retrieve details-on-demand [65]. Some existing work is based on this kind of visual aggregates; many of these techniques are presented in the body of this article in the relevant sections. To give a representative sampling, much of the inspiration for this article comes from the opacity representation of cluster results in parallel coordinates and starglyphs by Fua et al. [34], edge aggregation in adjacency matrices by Elmqvist et al. [28], and hierarchical edge bundles for node-link diagrams by Holten [47] (although the latter is not strictly a hierarchical aggregation technique).

Yang et al. [82] present a framework for interactive hierarchical displays that bears much resemblance to our work. In particular, they present hierarchical implementations for parallel coordinates [49], starplots [66], scatterplot matrices [19], and dimensional stacking [54]. However, unlike our model, their framework is designed solely for multivariate datasets and specifies a concrete visual representation and color coding for the aggregates. Therefore, our model is slightly more general, although by virtue of being more general, we also provide less guidance in actually implementing new multiscale visualizations.

As for the technical challenges of large-scale datasets, Fekete and Plaisant [33] show how to overcome them for datasets on the order of 10^6 items using scatterplots and treemaps. However, their approach is to preserve all visible items in the visualizations without any data abstraction, so while their findings—as well as similar findings on the scalability of visualization—are useful for tool architects, they are not directly relevant to the more conceptual nature of this work.

3 Hierarchical Aggregation for Visualization

Our model for hierarchical aggregation in visualization is based on aggregation in data space and corresponding simplified visual representations of the aggregates in visual space. Essentially, the aggregation process turns any visualization into a multiscale structure that can be rendered at any desired level-of-detail. This provides the user with a manageable overview that hides any clutter arising from details in the dataset while still giving a reasonable indication of the data size, extents, or distribution through the visual aggregates. A set of basic interaction techniques support navigating this structure. The visual aggregates can convey different information about the underlying data items, such as their average, extents, or even their distribution. Furthermore, the interaction allows the user to drill down and retrieve details on-demand. In this way, hierarchically aggregated visualization techniques directly support the visual information seeking mantra: *"overview first, zoom and filter, then details on demand."* [65]

In this section, we present two main types of information visualizations, describe the basic aggregation model, discuss suitable visual representations, explain the rendering process of aggregate hierarchies, and give a basic set of interaction techniques for controlling this class of visualization techniques. Note that while this is a generalized model, there exist many special cases for specific visualizations. We will discuss these in the visualization examples following this section.

3.1 Terminology

For the purposes of hierarchical aggregation, it is useful to distinguish between two main types of visualizations: *overlapping* versus *space-filling* visualizations [33]:

- *Overlapping visualizations.* A visualization type that enforces no layout restrictions on visual items, and items may thus overlap on the display (potentially causing occlusion). Overlapping visualizations include scatterplots, node-link diagrams, and parallel coordinates.
- *Space-filling visualizations.* A visualization that restricts layout to fill the available space and to avoid overlap. This means that individual visual data items do not occlude each other. Representative techniques include treemaps, most 2D and 3D geometric visualizations, and adjacency matrices.

In the following text, we will also use the term *aggregate hierarchy* (or *aggregate tree*) to mean a grouping of the original *data items* into a hierarchical structure of *data aggregates*, each representing their children. *Visual aggregates* are the graphical depictions of data aggregates, whereas *visual items* are graphical representations of data items. As a common term for both aggregates and items, we use *visual entity*.

Finally, given the above terms, we define *visual entity budget* as an upper limit of the number of visual entities to render for a particular hierarchically aggregated visualization. By controlling this limit, we can either use it to ensure a minimum frame rate by capping the amount of visual entities to draw, or to control the amount of visual clutter for the purposes of efficiently perceiving the visualized data.

3.2 Aggregation

Given a set of *data items*, hierarchical aggregation is based on iteratively building a tree of *aggregate items* either bottom-up or top-down. Each aggregate item consists of one or more children; either the original data items (leaves) or aggregate items (nodes). The root of the tree is an aggregate item that represents the whole dataset.

Bottom-up aggregation [50, 51] starts with treating each item as its own aggregate, then iteratively combines similar aggregates into a single aggregate until only one remains. Top-down aggregation, on the

Data structure	Visualization	Type	Aggregation	Visual aggregate	Metadata visualized
multidimensional	scatterplot	O/L	hierarchical clustering	points [19, 74]	average
multidimensional	scatterplot	O/L	hierarchical clustering	boxes [82]	extents (axis-aligned), average
multidimensional	scatterplot	O/L	space-filling subdivision	boxes [80]	extents (axis-aligned), average
multidimensional	scatterplot	O/L	hierarchical clustering	hulls [4]	extents (convex hull), average
multidimensional	scatterplot	O/L	hierarchical clustering	blobs [6, 15, 20, 44]	extents
multidimensional	parallel coordinates	O/L	hierarchical clustering	lines [75]	average
multidimensional	parallel coordinates	O/L	hierarchical clustering	bands [34, 82]	extents, average
multidimensional	parallel coordinates	O/L	hierarchical clustering	color histograms [29, 30]	distribution, extents
multidimensional	parallel coordinates	O/L	hierarchical clustering	beads [4]	distribution, extents
multidimensional	starglyphs	O/L	hierarchical clustering	lines [75]	average
multidimensional	starglyphs	O/L	hierarchical clustering	bands [34, 82]	extents, average
multidimensional	starglyphs	O/L	hierarchical clustering	color histograms [29, 30]	distribution, extents, average
tree	treemap	S/F	existing tree hierarchy	treemap nodes [63]	extents, average
tree	node-link diagram	O/L	existing tree hierarchy	thumbnails [17, 59]	extents, count, depth
graph	node-link diagram	O/L	hierarchical clustering	metanodes [2, 8]	extents, average
graph	node-link diagram	O/L	—	edge bundles [47]	link extents, average
graph	node-link diagram	O/L	data cube aggregation	metanodes [78]	node and link counts
graph	adjacency matrix	S/F	recursive edge merging	edge blocks [1, 28]	distribution, average
spatial	2D/3D geometric	—	recursive data merging	quad/octree blocks [4]	extents, average

Table 1. Visual aggregation strategies for a set of basic information visualization techniques (O/L = overlapping, S/F = space-filling).

other hand, starts with one aggregate containing all the items and repeatedly splits aggregates until a specific level of granularity has been reached, or all items belong to only one aggregate. The common denominator between these two approaches is a similarity measure defined using a data-specific distance function.

There are several specific algorithms to perform aggregation. The most common are clustering approaches, such as graph-based or k-means clustering [50, 51], but other examples include quadtree- and octree-based methods, such as the recursive merging of four adjacent edges into a single edge as done by Elmqvist et al. [28]. For cases where no suitable clustering method can be found, a good fallback may be to use data cube aggregation [70] (similar to PivotGraph [78]) to easily turn a multivariate dataset into a hierarchical structure.

In order to make use of this aggregate tree, visualization techniques that support hierarchical aggregation provide not only a visual representation for the actual data items, but also for the aggregate items. The aggregate tree becomes a multiscale structure for controlling the current level-of-detail of the visualization on the screen. Depending on the visualization technique, the visual aggregate can also convey information about the underlying data items (see the next section).

3.3 Visual Representation

Standard visualization techniques define a visual representation for individual data items (henceforth called *visual data items*), but a hierarchical aggregated visualization must also define a visual representation for data aggregates. This *visual aggregate* should convey something about the underlying data aggregation that the entity captures. It should also be distinguishable from visual data items (although this is not always the case). The type and amount of information conveyed depends on the visualization technique, but as a rule of thumb, the information should be simplified so as not to give rise to the same kind of visual clutter that we are trying to avoid.

Here are some examples of information that can be conveyed through visual aggregates (see [4, 13] for in-depth details):

- **Count/sum:** The number or the sum of the aggregated data items.
- **Average:** The arithmetic mean of the underlying data items (the color shading in Figure 8 shows the average point of each hull).
- **Mode:** The most frequent value in the underlying data set (most useful for discrete values).
- **Extents:** The extrema (minimum and maximum) of the underlying data items (Figure 1(b) shows the bounding boxes for each visual aggregate).
- **Median:** The value that divides the underlying data into two, equal-sized subsets. The median, and other positional measures, is often more robust against outliers than the average.
- **Percentiles:** The 25th, 50th (median), and 75th percentiles for the underlying data (such as the Tukey box in Figure 16).
- **Distribution:** The full distribution of the underlying data, often visualized using a histogram (Figure 11).

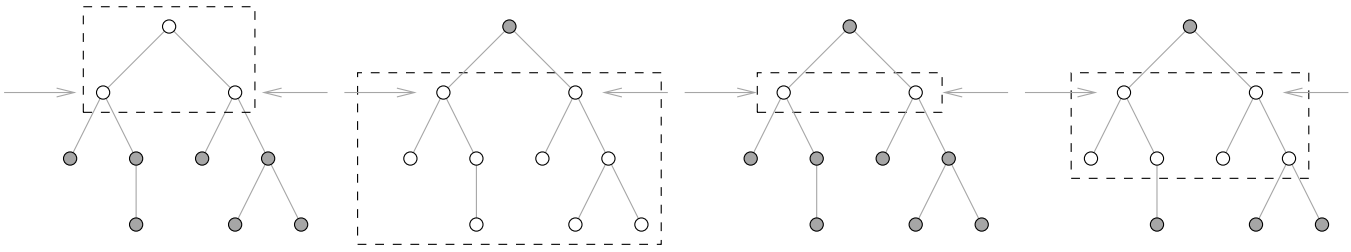
For non-numeric data items, we can still use several of the above symbolic metrics, including count, mode, and frequency distribution.

3.4 Rendering

Rendering a hierarchical aggregated visualization amounts to traversing the visual aggregate hierarchy. The traversal should typically be breadth-first to resolve depth ordering so that higher-level aggregates are correctly occluded by lower (more detailed) levels. This also allows for aborting the rendering at any stage, such as for maintaining a fixed frame rate, while still retaining a consistent visual output.

The visualization keeps track of the current detail level it is being viewed at, corresponding to the height in the tree to traverse to while rendering. Depending on the current state of the visualization, there are four main types of rendering traversals (Figure 2):

- **Above traversal:** All nodes **above** (and including) the current height are rendered. Here, data items are abstracted by high-level visual aggregates, hiding details to avoid perceptual and technical overload (Figure 2(a)).
- **Below traversal:** All nodes **below** (and including) the current height are rendered. Less common, this gives an indication of how data items are grouped together (Figure 2(b)).
- **Level traversal:** All nodes on the **same level** as the current height are rendered. This gives a snapshot of the data abstraction at any specific detail level (Figure 2(c)).
- **Range traversal:** All nodes in an **interval** of levels are rendered. As for level traversal, range traversals give a snapshot of the data abstraction at a specific level, as well as some additional detail on the underlying aggregate hierarchy (Figure 2(d)).



(a) Above traversal. (b) Below traversal. (c) Level traversal. (d) Range traversal.

Fig. 2. Rendering traversal types for hierarchical aggregated visualizations. Arrows indicate the current aggregation height. White nodes are rendered to the screen while gray nodes are omitted from rendering.

Beyond this, it is always possible to perform unbalanced traversals of the aggregate tree, e.g., to give details-on-demand to certain branches of the tree without traversing into neighboring branches. In particular, we could let the underlying variability of each aggregate automatically guide the rendering traversal [4]. This would devote more detail to the "interesting" parts of a dataset, where there is a lot of varying features in the data, and save the entity budget in areas with low variance.

Unbalanced rendering traversals can also be used to support generalized fisheye views [36]. This is done by letting the budget take a level of detail weight [36, 59] associated with the structure into account.

Clearly, the benefit of a data aggregate hierarchy and a corresponding visual aggregate hierarchy is that the resulting visualization can be adapted to the requirements of the human user as well as the technical limitations of the visualization platform. Instead of drawing an entire dataset as individual data items, a visualization technique supporting hierarchical aggregation can maintain a visual entity budget and only do a breadth-first traversal into the aggregate hierarchy until the budget has been expended. Since any visual aggregate at any level in the tree is guaranteed to serve as a faithful abstraction of underlying levels, the resulting image will remain consistent with the actual data.

3.5 Interaction

Beyond uniform methods for aggregation and rendering, one of the main benefits of hierarchical aggregated visualizations is that they can be made to support the same set of basic interaction techniques for navigating and manipulating the aggregate hierarchy. This allows users to apply the same reasoning and interaction strategies to different visualization techniques that support hierarchical aggregation. Here we will discuss a basic set of common interaction techniques for this purpose.

Note that while this article mainly focuses on developing new visualization techniques that support hierarchical aggregation, the space is also open for designing new interaction techniques for interacting with them. Hopefully, the presentation here will also serve as a useful framework for such activities.

For each of the interaction techniques in this section, we will use a combination of space-scale diagrams [38] and the aggregate hierarchy to describe the mechanics of the interaction. We will also give concrete examples of each interaction in existing systems and techniques. Because these interaction techniques mainly deal with various ways of navigating hierarchies (which is what an aggregate tree actually is), many of these techniques can also be found in tree visualization tools like treemaps [63], Sunburst [69], and InterRing [81].

3.5.1 Zoom and Pan

Zooming and panning changes the current view of a visualization, respectively changing the size and position of the viewport window on the visual substrate [38, 58] (Figure 3). In a way, zoom and pan operations can be seen as spatial filters that control which visual data entities to display on the screen as well as the screen space allocated to each entity. These operations are typically used to take a closer look at parts of a visualization without actually revealing more detail in the data.

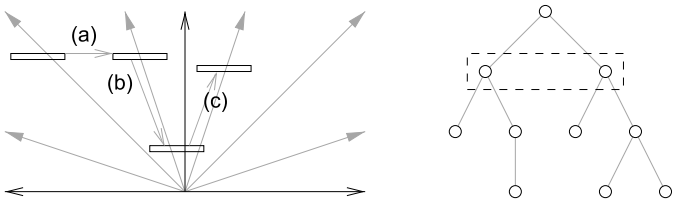


Fig. 3. Zoom and pan interaction. (a) Pure pan. (b) Zoom out to show two items. (c) Combined zoom in and pan.

Navigation in multiscale spaces has recently been a focus in the human-computer interaction community because of its applicability to high-precision selection [41, 42]. Many of these interaction techniques—such as speed-dependent automatic zooming (SDAZ) [48], Zliding [60], and OrthoZoom [5]—could be used to navigate the multiscale structures defined by the hierarchical aggregated visualizations discussed in this article.

Fua et al. [34] describe a specialization of zooming for parallel coordinates that they call *dimension zooming* where each dimension axis is zoomed independently; this approach can easily be employed for other visualization techniques, such as using different axis scales in scatterplots or trivially for space-filling techniques such as treemaps. On the other hand, it does not make sense for visualizations where spatial location has no intrinsic meaning, such as for node-link diagrams.

On a related note, the *snap-zoom* [14] technique is a variant of axis-independent zooming for continuous treemap navigation where the scale factor for the axes depends on a view-dependent (e.g. dynamically changing) layout criteria. This is a good example of how to customize a general interaction technique to the characteristics of a specific visualization while still retaining the same interaction pattern.

For the purpose of balancing zoom and pan operations, the work of van Wijk and Nuij [76] on smooth and efficient zooming and panning is particularly relevant. They present a computational model that can be used to control the speed and behavior of animated transitions from different viewports on a large visual space. However, this is clearly not relevant when the viewer has direct control over zooming and panning.

3.5.2 Drill-down and Roll-up

While zoom and pan merely control the geometric properties of the display, the *drill-down* and *roll-up* operations govern the level of detail at which the data is displayed [34] (Figure 4). In other words, drill-down moves the rendering traversal deeper into the aggregate hierarchy, showing increasing amounts of detail, whereas roll-up moves up in the hierarchy, showing less detail. These operations are usually coupled with above rendering traversals of the aggregate hierarchy,

although level traversals can be used to show a specific level in the hierarchy, and below traversals for showing the aggregation structure.

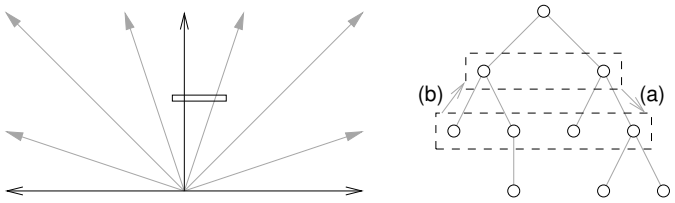


Fig. 4. Drill-down and roll-up interaction. (a) Drill-down from level 1 to level 2. (b) Roll-up from level 2 to level 1.

In some cases, drilling and rolling should be restricted to the current geometric zoom and pan, or coupled with geometric zoom (see Section 3.5.5): drilling down further may not make sense if individual visual entities become smaller than a single pixel, and rolling may analogously not be useful if the entities become larger than the screen.

While geometric zoom is typically implemented as continuous [38], the level-of-detail of an aggregated visualization is clearly discrete. To facilitate user perception of the interaction technique changing the detail level, animated transitions between aggregation levels, like in the Matrix Zoom [1] system, can come in useful.

Many times, the viewer may want to perform drilling and rolling on a specific subtree of the aggregation hierarchy instead of on the whole tree. See the next section for more details.

3.5.3 Local Aggregation Control

As discussed in Section 3.4, it is sometimes useful to perform unbalanced rendering of the aggregate hierarchy to reveal different amounts of detail for different regions of the visualization in a focus+context [36, 37] fashion (Figure 5). Local aggregation control gives the user control over this process in different ways.

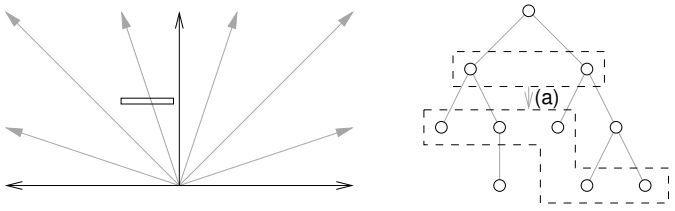


Fig. 5. Local aggregation control. (a) Unbalanced drill-down.

For example, one form of local aggregation control may be to allow the user to directly select which part of the aggregate hierarchy to expand or collapse by, for example, simply clicking on the visual aggregates in the display (left-click for expand, right-click for collapse). Another instance of this type of interaction may include directly controlling the type of rendering traversal (Section 3.4) to perform, or parameters of the current traversal (cutoffs, start points, current level, etc).

In this case, selecting the region of interest becomes part of the problem. While rubberband selection in screen space is the most common approach, structure-based brushes [35] support filtering whole branches of the aggregate hierarchy based on its structure. Chuah [18] describe the use of Magic Lenses [12] for local aggregation control. On a related note, the *dynamic masking* [34] technique interactively fades out unselected regions in favor of selected regions.

3.5.4 Flipping

If drill-down and roll-up are the aggregate hierarchy equivalents of geometric zoom, then *flipping* [14] is the equivalent of geometric pan (Figure 6). A discrete navigation operation, flipping lets the user visit neighboring siblings in the aggregate hierarchy (as opposed to nodes above or below in the hierarchy).

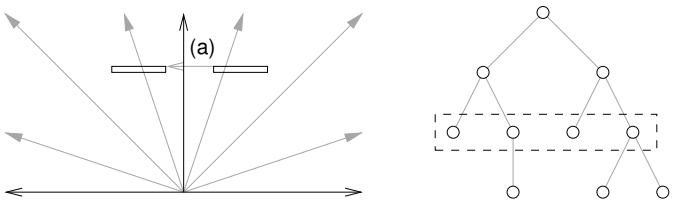


Fig. 6. Flipping interaction. (a) Flip between adjacent siblings.

Note that the effect of a flip operation *is* a geometrical pan because the operation changes the view of the visualization and not the aggregation level. However, as opposed to a normal pan, flipping is informed by the aggregate hierarchy. To make the transition between siblings easier for the user to follow, it is typically performed using an animation.

This interaction makes the most sense for visualizations (typically space-filling) where spatial containment and adjacency is well-defined, as opposed to visualizations—such as scatterplots or parallel coordinates—where adjacency and containment criteria are undefined.

3.5.5 Coupled Zooming and Drilling

Zooming and panning in a visualization can be interpreted as the shifting focus of attention of the viewer. This information can be utilized for also controlling the level of detail of the visualization. In other words, the *coupled zooming and drilling* operation ensures that zooming into a specific part of a visualization will cause the rendering to traverse deeper into the aggregate hierarchy, yielding more detail (Figure 7). Zooming out will have the opposite effect.

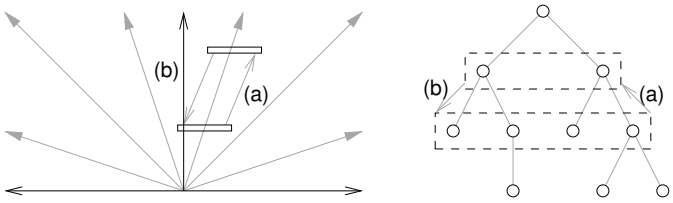


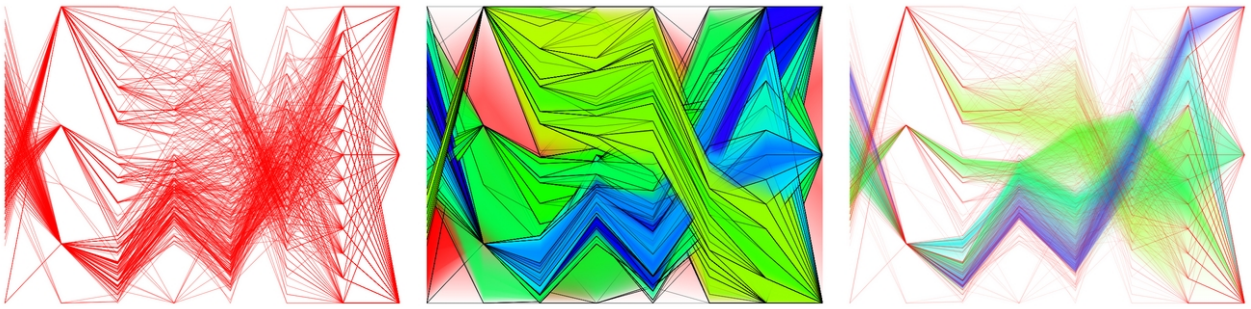
Fig. 7. Coupled zoom and drill interaction. (a) Drill-down and zoom in. (b) Roll-up and zoom out.

Another, perhaps more general, term for coupled zooming and drilling is *semantic zooming* [58], where the graphical representation dynamically depends on the geometrical properties of the view through which the user is looking at the data. Our use of the term is similar, yet explicitly refers to parameters of the underlying hierarchical visualization.

Coupled zooming and drilling is typically implemented either using a visual entity budget or an entity size tolerance. For the former case, the user or the system may specify the maximum number of visual entities to display, and will at any given moment traverse deeply enough into the aggregate hierarchy to expend this budget. This means that visually filtering out parts of a visualization by zooming or panning into certain regions will allocate more detail to this part. For the latter case, the user or the system maintains a size tolerance for visual entities, and traverses deeper into the tree (i.e. adds more detail) until each visual entity is within this tolerance. Again, this causes geometric zoom and pan to reveal more or less of the details of the aggregate hierarchy.

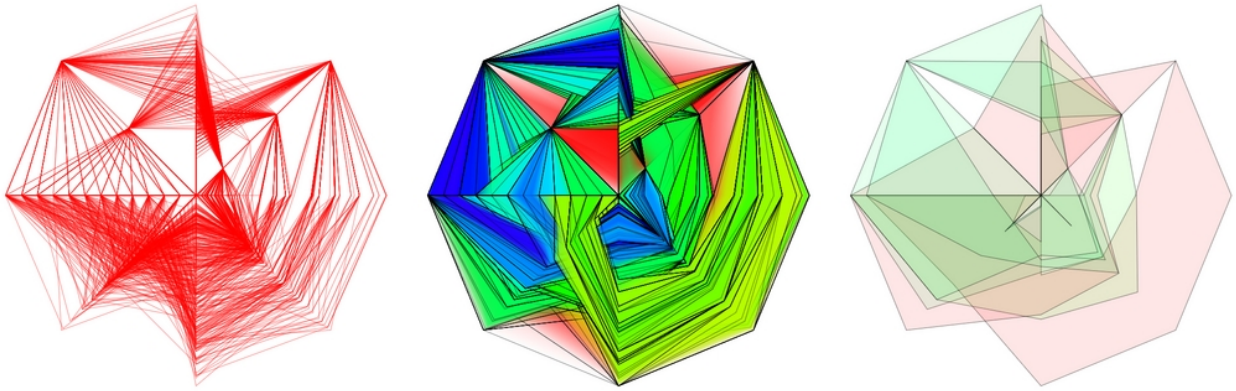
4 Visualization Examples

Given the above model of hierarchical visual aggregation, we now discuss a number of existing visual representations and see how we can



(a) Parallel coordinate visualization. (b) Opacity band aggregation (above-traversal). (c) Opacity band aggregation (below-traversal).

Fig. 9. Parallel coordinate visualization of an 8-dimensional dataset with opacity band [34] visual aggregations. Transparency is used to indicate the extrema and the averages for each visual cluster band.



(a) Starglyph visualization. (b) Opacity band aggregation (above-traversal). (c) Opacity band aggregation (level-traversal).

Fig. 10. Starglyph visualization of an 8-dimensional dataset with opacity band [34] visual aggregations. Transparency indicates the extents and the averages for each opacity band.

implement hierarchical aggregation to turn them into multiscale structures that are more scalable for massive datasets. Table 1 shows a summary of visual aggregation strategies for the different information visualization techniques discussed here.

4.1 Scatterplots

Scatterplots [19, 74] visualize multidimensional data by mapping data cases to graphical points in a two- or three-dimensional Cartesian space defined by two or three orthogonal axes. The position of each point representing a data case depends on the data dimension assigned to each axis. Visualization tools supporting scatterplots, such as XmdvTool [77], Spotfire, and GGobi [71], typically allow additional data dimensions to be mapped onto graphical attributes such as the size, color, and shape of the points. Because of their relative simplicity, familiarity with users, and high visual clarity, scatterplots are part of the standard vocabulary of data graphics.

Scatterplots are overlapping visualizations, and rendering a large dataset (or even a dataset that is locally dense) will cause a lot of item overlap in the output, making estimation of effects difficult. Fekete and Plaisant [33], while not focusing on aggregation per se, present a scatterplot implementation capable of visualizing a million separate items. Their examples clearly show the overlap problem (they even measure the amount of overlap), and they discuss methods involving transparency, stereovision, and interaction to reduce its impact.

As outlined in Table 1, the main visual aggregations for scatterplots involve hierarchical clustering of data points into points, boxes, hulls, and blobs. Point aggregates show only the average for each cluster, and could be seen as a sampling operation [25]. Originally introduced by Yang et al. [82], boxes are axis-aligned 2D or 3D bounding boxes that give an indication of the minimum and maximum values for the displayed dimensions (Figure 1(b)). They can be augmented to show averages as points or using opacity where the center point of the cluster is fully opaque and the extrema are fully transparent. Wills [80] present a scatterplot-based aggregation method using recursive subdivision of space with boxes.

Hulls are variations on boxes in that they show the extents of the displayed dimensions, but using 2D or 3D convex hulls instead of axis-aligned bounding boxes as a tighter visual metric (Figure 1(c)). Just like for bounding boxes, center points or opacity can be used to indicate the cluster average (Figure 8). Bivariate boxplots [79] and their simplification, bagplots [62], are statistical versions of this technique where concentric hulls are constructed to contain quartiles of the data.

Taking extent refinement even further, the blobs technique proposed by Heine and Scheuermann [44] provide implicit surfaces for conveying and interacting with fine-grained clustering, and can be employed for scatterplot diagrams. However, the authors point out the problem of color blending for overlapping blobs, a problem that generalizes also to boxes and hulls. The weaving approach proposed by Hagh-Shenas et al. [43] might come in handy for resolving this problem. Similar work includes the multivariate set visualization technique proposed by Byelas and Telea [15], the automated overlapping set technique introduced by Simonetto et al. [68], and the bubble set overlay mechanism presented by Collins et al. [20].

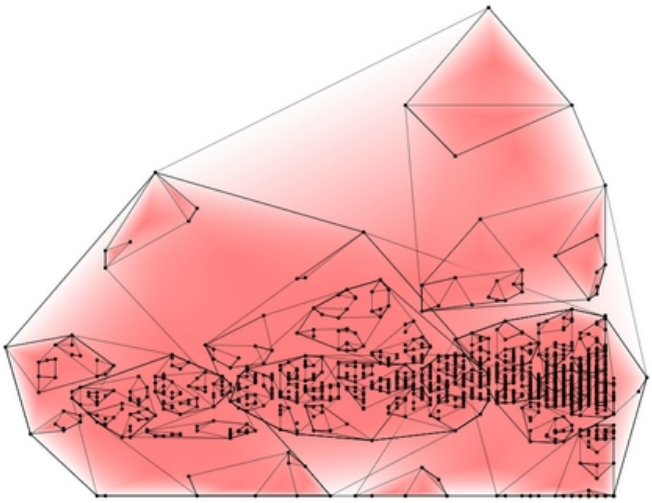


Fig. 8. Convex hull visual aggregation of a 2D scatterplot. Transparency is used to indicate the centers of each hull aggregate.

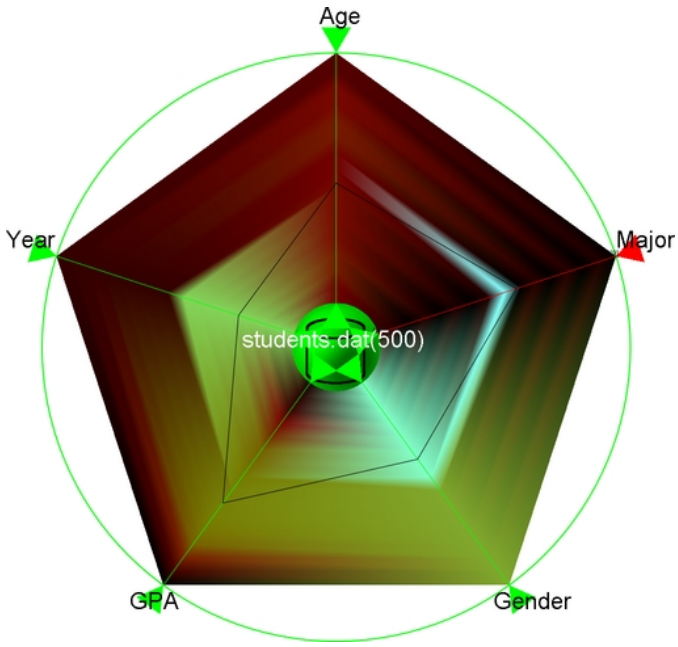


Fig. 11. Color histogram representation of a multidimensional university student dataset in a starglyph visualization from the DataMeadow [29, 30]. Brightness indicates high density.

4.2 Parallel Coordinates and Starplots

As opposed to the orthographic axes of scatterplots, parallel coordinates [49] stack the dimension axes of a multivariate dataset in parallel and visualize items in the dataset as polylines that connect the item's value for each dimension. Starplots [66] extend parallel coordinates by arranging the axes in a polar instead of a linear coordinate space, turning the open polylines representing data items into closed ones.

The simplest approach to visualize aggregated parallel coordinates is to just use polylines as visual aggregates for the average of the aggregated data items. van Wijk and van Selow [75] describe the use of this representation for hierarchically clustered time-series data. However, the information conveyed through line aggregates is low (only the average value). Also, the line aggregates have no visual affordance that indicates that they are actually aggregates of underlying data items.

To address this problem, Fua et al. [34], and later Yang et al. [82], describe a hierarchical visual aggregation approach for parallel coordinate diagrams based on bands that aggregate several polylines (visual data items) into a single band (visual aggregates) as wide as the extrema of the underlying data. This is an n-dimensional generalization of scatterplot boxes and hulls in 2D space. Bands can also convey the average value using opacity just like scatterplot hulls and boxes. See Figure 9 for examples of how to apply hierarchical aggregation to a standard parallel coordinate diagram.

Both lines and bands can be trivially extended to starglyph diagrams [34, 82] (Figure 10). Recent work on starglyph diagrams introduced the notion of *color histograms* [29, 30] showing the data distribution on the surface of the opacity bands (Figure 11). This method could also be employed also for standard parallel coordinate diagrams; Andrienko and Andrienko [4] show approaches to displaying quartiles and distributions in parallel coordinates using bands and beads, respectively.

Along similar lines, the parallel sets [11] technique extends parallel coordinate displays with frequency-based visualizations of categorical data (Figure 12). Sifer [67] develops the idea further by removing the bands connecting data categories and relying merely on color for identification. However, these approaches do not maintain an aggregate hierarchy and are thus not true hierarchical visualization techniques.

4.3 Tree Visualizations

When the data structure is a tree, it is generally used as its own aggregation hierarchy. For this reason, we discuss all tree visualizations together in one section. Using the tree structure itself as the aggregation hierarchy is problematic when the tree is unbalanced or when it is either very deep, such as phylogenetic trees, or very wide, such as the Java SDK class hierarchy. In this case, the aggregation hierarchy typically merges or splits some of the original tree nodes but remains congruent with the original tree structure.

Treemaps [63] visualize tree structures through enclosure using a space-filling layout algorithm based on recursive subdivision of space. The enclosure feature of treemap nodes conveys the extents of underlying nodes in the hierarchy. Furthermore, the surface of an internal tree node serving as a visual aggregate could conceivably be used to show additional information about the node's children. However, because the size of visual entities depend on a specific data attribute, some branches of a tree may disappear (i.e. be allocated display size less than a pixel) [63], a problem addressed by context treemaps [21] that allocate space even to zero-sized nodes.

Fekete and Plaisant [33] give a number of technical solutions for visualizing treemaps on the order of 10^6 items, cutting off rendering for data items that are smaller than one pixel in size.

The zoomable treemaps [14] introduced by Blanch and Lecolinet combine many of the interaction techniques described in this article with the use of treemaps as a multiscale space for navigation. Their work is instructive in how to extend an existing visualization technique with support for aggregation and navigation in the aggregate hierarchy.

SpaceTrees [59] use standard node-link representations of trees, but support interactive expansion and contraction of tree branches as well as screen-optimized dynamic layout of nodes and automatic camera movement to fit the current display. The technique supports many of the general interaction techniques for hierarchical aggregation described in this article, including local aggregation control, roll-up, and drill-down. Visual aggregates of unexpanded branches are shown using thumbnails that indicate the underlying node topology, or as simple triangle glyphs that show the count and depth of the contracted subtree.

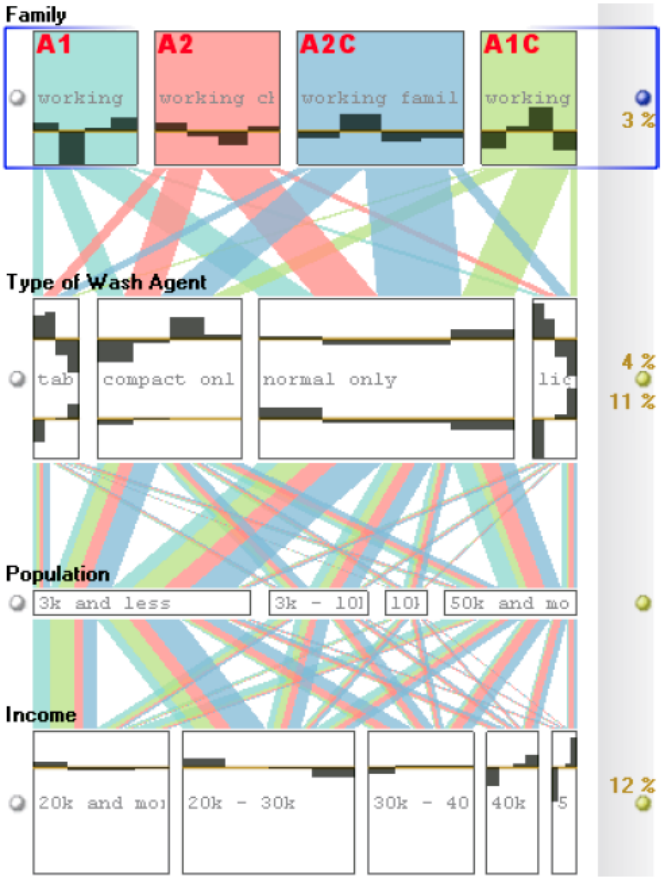


Fig. 12. Parallel set visualization of categorical customer data [11]. Histograms for each axis show the distribution of data points.

Degree-of-Interest trees [17] provide a focus+context [36] view of node-link tree representations that are optimized for the screen. As with SpaceTrees, DOITrees define several different visual representations for aggregated subtrees, including elided and glyph representations of their node count and depth.

4.4 Node-Link Diagrams

Node-link diagrams are perhaps the most straightforward graph visualizations, drawing a graph as a collection of visual nodes arranged on a 2D or 3D visual substrate using some graph layout algorithm [24] and connected using visual links. However, large numbers of edges or nodes may cause a highly congested visualization despite the use of efficient layout algorithms, prompting the need for hierarchical aggregation [26], either on edges, nodes, or both.

For node-link diagrams (as opposed to matrix visualizations, see below), edge aggregation is the least common of these approaches and only a few techniques take this route. Hierarchical edge bundles [47] group links together into thick bundles, akin to how electrical or network wires are tied together to avoid excessive tangling. However, despite its name, this is not strictly a hierarchical visualization technique because the aggregation is performed in visual space using implicit curves and not in data space.

As for node aggregation, there are essentially two approaches to conducting hierarchical aggregation on graphs visualized using node-link diagrams: *top-down* or *bottom-up*. In the bottom-up approach, a layout is computed for the whole graph, and the nodes are then hierarchically grouped using geometrical means [39]. However, by requiring a complete graph layout prior to aggregation, this approach may not scale well to extremely large graphs.

In the top-down approach, on the other hand, we instead start with a hierarchical aggregation and use this to dynamically compute layout as the user is drilling down into the dataset. This strategy is used by the TugGraph [6] system. Because this approach does not require an expensive global graph layout but instead amortizes this cost online as the user is exploring, it scales better than bottom-up approaches. It is therefore more promising for future work on aggregating large-scale graphs.

In general, Auber et al. [9] show how to use hierarchical node clustering for node-link diagrams to turn a graph into a multiscale structure based on its small-world structure. Their Tulip [8] system uses metanodes to cluster several nodes into a visual aggregate hierarchy. Metanodes may have a standard node visual representation with the union of all incoming and outgoing edges of the aggregated subnodes, or they may also show simplified contents, such as the clustered small-world network as miniature thumbnails for each node.

Building on the previous work, Archambault [7] presents a feature-based approach to visualizing graphs using node-link diagrams, and discusses a range of graph drawing techniques for this purpose. These techniques are designed to draw hierarchical representations of graphs; some of the proposed techniques support local aggregation control using distinct visual aggregates for metanodes such as nested circles (open) and hexagons (closed). The work also presents many specializations of the general principles introduced in this article that are specific to graphs and node-link diagrams; thus, this is a good example of how to adapt hierarchical aggregation to a specific subdomain.

ASK-GraphView [2] combines clustering functionality with navigation in hierarchically aggregated node-link diagrams of large-scale externalized graphs. The system supports interactive expansion and collapse of aggregate branches in an unbalanced fashion, and also manages visual clutter through transparency and masking of edges, similar to the interaction techniques catalogued in this article.

The NodeTrix [46] technique combines node-link diagrams with adjacency matrices for representing communities in social networks. This method utilizes a fixed two-level aggregation where subsets of a graph are clustered and rendered as adjacency matrices connected to other matrices using edge bundles similar to Holten's hierarchical edge bundles [47]. Because the aggregation is fixed, however, many of the interaction techniques described in Section 3.5 do not make sense for NodeTrix visualizations.

In an extension [32] of the growing polygons [31] dependency visualization, Elmqvist and Tsigas show how to augment the technique with hierarchical subsets that group together nodes in a directed acyclic graph. Their visual aggregate representation shows the cumulative dependencies for each subset, and additionally adds a drop-shadow to indicate that the subsets can be expanded (Figure 13).

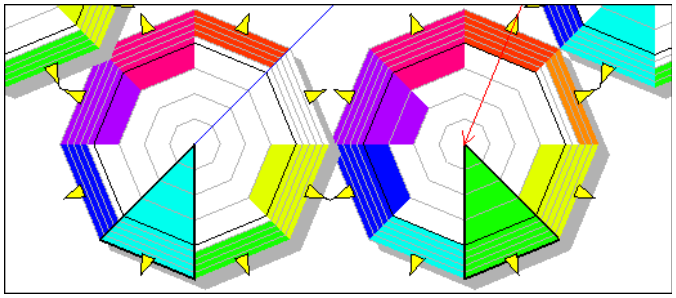


Fig. 13. Visual aggregate representation for the hierarchical growing polygons visualization technique [32]. Dependencies for the aggregate is the sum of its children's dependencies and the shadows give a visual affordance that the aggregates can be expanded.

Finally, pivot graphs [78] exploit data cube aggregation [70] to roll-up nodes into metanodes based on attribute values for categorical data. A specialization of hierarchical aggregation, this allows for the creation of simplified node-link graphs showing primarily the counts of aggregated nodes and edges. However, because the aggregation again is not hierarchical in nature, the interaction techniques described in this article do not apply to pivot graphs.

4.5 Adjacency Matrices

Instead of explicitly drawing nodes and edges as graphical entries laid out on a visual substrate, adjacency matrices arrange nodes along the rows and columns of a matrix and represent edges between nodes as cells in the contents of the matrix. Adjacency matrices have been shown to be more efficient than node-link diagrams for large and/or dense graphs [40], and are thus often employed for visualizing large graphs, either alone or in tandem [45] with node-link diagrams.

The NodeTrix [46] technique mentioned in the previous section is an example of an aggregated hybrid between node-link and matrix representations. However, as noted above, the technique is not a true aggregated visualization due to its fixed aggregation structure.

The Matrix Zoom [1] system, on the other hand, takes edge aggregation to its extreme by displaying large-scale graphs as a hierarchy of adjacency matrices given a predefined clustering hierarchy. Matrix Zoom uses animated transitions between levels in the aggregate hierarchy instead of supporting continuous zoom and pan interactions.

Similarly, the Zoomable Adjacency Matrix Explorer (ZAME) [28] defines a pyramid of recursive edge merging where each level is half the dimension of the level below. Hence, each visual edge aggregate clusters four visual aggregates in the level below (Figure 14). This way, ZAME is able to visualize very large graphs on the order of 10^6 nodes and edges (Figure 15). Instead of fixed animations, like Matrix Zoom, the system implements many of the interaction techniques outlined in Section 3.5 of this article such as zoom and pan, drill-down/roll-up, and coupled zoom and drill. In addition, the system also provides a rich set of visual representations for edge aggregates (Figure 16) drawn using small programs (shaders) running on the graphics hardware.

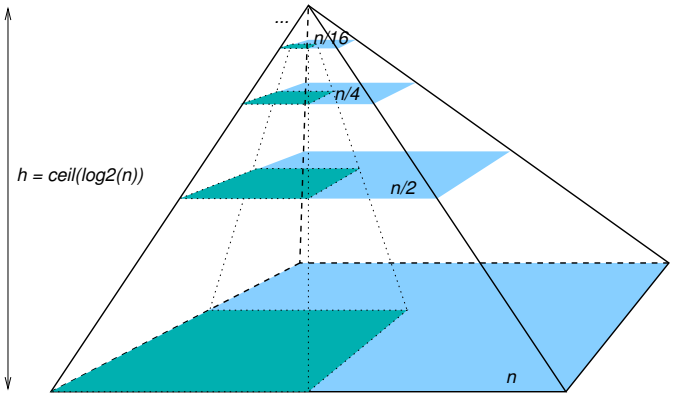


Fig. 14. Pyramid aggregation hierarchy for the ZAME [28] system. Each aggregate node represents four nodes in the level below.

4.6 Geometric Visualizations

Two-dimensional or three-dimensional geometric visualizations for scientific data are routinely turned into aggregated visualizations using spatial aggregation techniques such as quadtrees, octrees, and other spatial hierarchies (Figure 20). The same interaction techniques as for other aggregated visualizations can be used for these visualization techniques, allowing for geometric zooming, drilling down, and rolling up the data. Coupled zooming and drilling is often used to show more details in regions the user is zooming into.

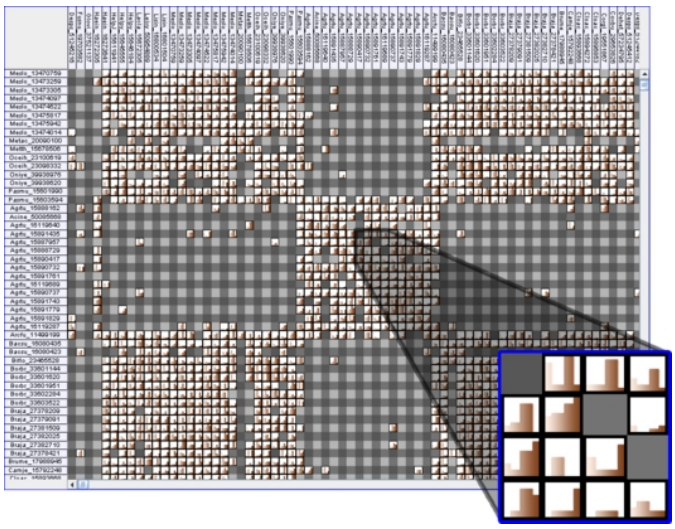


Fig. 15. ZAME [28] visualizing a protein-protein interaction dataset of 100,000 nodes and 1,000,000 edges. Inset shows a magnified view of step histogram edge aggregates in the matrix.

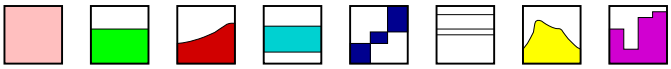


Fig. 16. Visual edge aggregate representations for ZAME [28] (average color shade, average, min/max curve, min/max range, min/max tribox, Tukey box, smooth histogram, step histogram).

The standard visual representation of aggregates for geometric visualizations is to simply use the average color shade of the children to represent an aggregate. Andrienko and Andrienko [4] show how to aggregate geospatial data while retaining metadata, such as geographical outlines and labels (like the continent outlines in Figure 20).

While geometric visualization is a very large topic, most geometric visualization techniques fall within the realm of scientific visualization. Therefore, geometric visualization are beyond the scope of this article where we concern ourselves mainly with information visualization.

5 Design Guidelines

Having formalized a model of hierarchical aggregation for visualization and surveyed existing examples for supporting aggregation in visualization techniques, we now derive a set of general guidelines for designing and implementing such techniques:

- G1 *Entity budget.* Maintain an entity budget;
- G2 *Visual summary.* Aggregates should convey information about underlying data;
- G3 *Visual simplicity.* Aggregates should be clean and simple;
- G4 *Discriminability.* Aggregates should be distinguishable from data items;
- G5 *Fidelity.* Beware that abstractions may lie; and
- G6 *Interpretability.* Aggregate items only so much so that the aggregation is still correctly interpretable within the visual mapping.

We describe each of these guidelines in greater depth below.

5.1 Entity Budget (G1)

Principle: Maintain a visual entity budget when rendering hierarchical aggregated visualizations.

Discussion: Aggregated visualizations automatically support the use of a budget of the number of visual entities to draw. If for no other reason than technical, it makes sense to cut off rendering for entities smaller than a pixel. In addition, it makes even more sense to control the amount of data rendered on the screen to avoid visually overloading the viewer.

The added benefit of a visual item budget is that it limits time spent on rendering and guarantees a lower bound on the framerate.

5.2 Visual Summary (G2)

Principle: Visual aggregates should convey information about the underlying data.

Discussion: We should design our visual aggregates to convey information about the underlying data that is currently not visible. This is one of the most basic features of a hierarchically aggregated visualization: the viewer can gain a reasonably detailed overview of a dataset while not being overloaded by the full data.

In some cases, it may even be useful for visual summary reasons to adopt an adaptive rendering scheme as described in Section 3.4 so that areas of the visualization with interesting features are automatically rendered at higher detail level than those with less variance.

5.3 Visual Simplicity (G3)

Principle: Design visual aggregates to have a clean and simple visual appearance.

Discussion: The purpose of aggregated visualizations is to reduce clutter in order to improve overview. Remember that even visual aggregates may be rendered in large quantities, so choosing a visual representation that is too complex or attempts to convey too much information may cause clutter in the visualization all over again.

The simplest possible choice of visual representation for the visual aggregate may often be the one used for visual data items. However, as argued by G4, this is often a bad idea, so a more complex (or at least different) representation may be necessary.

5.4 Discriminability (G4)

Principle: Design visual aggregates to be easily distinguishable from visual data items.

Discussion: Visual aggregates and visual data items must be easy to differentiate in order to not convey the wrong information to the viewer. While using the same visual representation for both aggregates as for items is often easiest, it may give the viewer the wrong idea about the data.

For example, the aggregate of two points in a scatterplot might be a single point, or the aggregate of two polylines in a parallel coordinate display might be a single polyline on their average [75]. A better choice is to introduce a novel representation for the visual aggregate, or to decorate the aggregate with some extra graphical feature, like the shadows for the growing polygons technique [32] (Figure 13).

5.5 Fidelity (G5)

Principle: Counteract fidelity problems in visual aggregates.

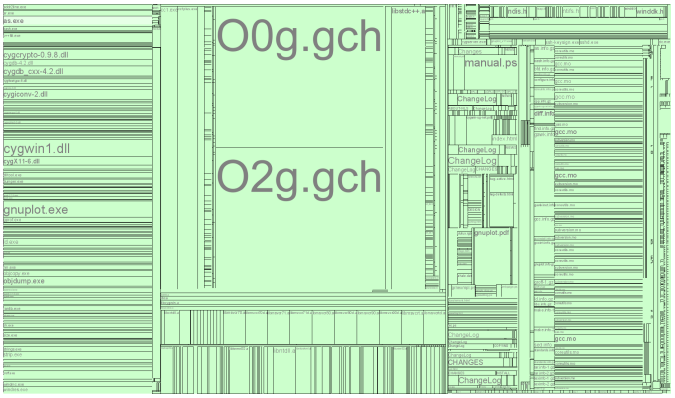


Fig. 17. Slice-and-dice treemap visualization showing a typical Cygwin distribution for Windows (648 directories, 6552 files, 181 MB).

Discussion: Data abstraction operations simplify datasets, but inherent in this process is also a loss of fidelity—the visualization may, in effect, lie about the size of the effect [61, 72]. This should be made clear to viewers using a visualization, either explicitly as done by Cui et al. [22], or by indications in the visual aggregates (i.e. deliberately choosing rough or imprecise visual indications of the aggregated data).

The use of averaging for data aggregation can be particularly problematic; Hans Rosling's TED 2006 talk about global health highlights how misleading averages can be when aggregating groups of countries together. Or, as Andrienko and Andrienko remark, "the mean surface temperature on the Moon may seem quite comfortable, but the actual temperature ranges from -230°C to $+130^{\circ}$C, and this should be taken into account in designing clothes for astronauts." [4]

Fortunately, the very tools provided by this article—interactive aggregation control—are exactly what is needed to counteract this problem. Drilling down into the dataset, if only a few levels down, will invariably uncover inconsistencies compared to higher-level aggregations.

5.6 Interpretability (G6)

Principle: Aggregate items only so much so that the aggregation is still correctly interpretable within the visual mapping.

Discussion: Related to guideline G5, this guideline reinforces the basic message of this article: aggregates convey meaning to the viewer, and if an aggregate is degenerate, it will cause the viewer to easily become misinformed. Items should not be aggregated merely for the sake of reducing visual complexity, but the aggregation should also be *interpretable* to the viewer in visual space.

For example, aggregates in a space-filling visualization based on containment, such as treemaps [63] or pivot graphs [78], carry high interpretability. However, an overlapping visualization such as a scatterplot, for example, will have low interpretability if the items are aggregated using dimensions not mapped to the visual attributes (i.e., the vertical and horizontal axes for a 2D scatterplot).

Also relevant in the interpretability discussion is the fact that visual aggregates should be designed so that they communicate to the viewer that there is more to see by expanding aggregated nodes (this goes hand in hand with guideline G4, which states that visual aggregates and visual data items should be distinguishable from each other).

6 Examples

In this section, we illustrate our model and design guidelines for hierarchical aggregation by studying how we can apply it to two different visualizations: a treemap [63] and a line graph.

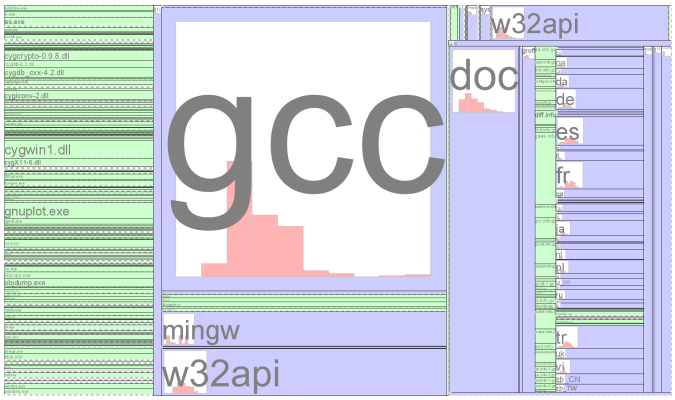


Fig. 18. Aggregate treemap visualization of the above file structure. Directories (aggregates) are blue, files (items) are green. Histogram glyphs show the file size distribution for each unexpanded directory, and nodes too small to render are drawn using a checkered pattern.

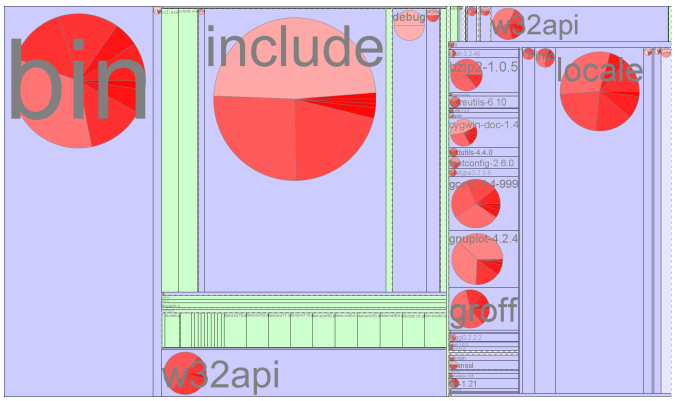


Fig. 19. Aggregate treemap of the cygwin file distribution. Pie chart glyphs are drawn for file size distribution to avoid the visual complexity of bar chart histogram glyphs.

6.1 Aggregated Treemaps

In this example, we study how to augment an existing zoomable treemap visualization [63, 14] with hierarchical aggregation support. In this way, we can turn the visualization into a multiscale structure suitable for supporting both overview as well as detail tasks.

We start our design from a basic slice-and-dice treemap [63] implemented in Piccolo [10] (Figure 17). The visualization already supports zoom and pan (Section 3.5.1) in geometric space, as well as flipping (Section 3.5.4) for visiting sibling nodes. However, the hierarchies that we want to represent using this visualization, such as a filesystem directory structure, may become arbitrarily large (easily more than 1 million entities [33]). This immediately gives rise to the two basic problems of large-scale datasets, performance and perception:

- **Performance:** Rendering performance will suffer for very large hierarchies.
- **Perception:** It is difficult, if not impossible, to get an overview of the dataset due to the large number of visual entities.

To help remedy this situation, we can apply design guideline G1 (entity budget) to begin introducing hierarchical aggregation to the visualization. In this case, we can utilize the tree structure itself as the aggregation hierarchy, saving us some work. Furthermore, G1 tells us that we can address both of the above problems by maintaining a visual entity budget and not propagate deeper into the aggregation hierarchy than necessary. In particular, there is clearly no need to draw any nodes that are smaller than a pixel for a given level of zoom. For this situation, we replace the visual representation of the aggregate and its children with a checkered pattern to communicate to the viewers that there is additional underlying data to be seen if they zoom in (G2). While this pattern may conflict with G3 (visual simplicity), it is better than the alternative, i.e., to attempt to draw all of the nodes, even though they are smaller than a pixel each. Furthermore, we believe that the checkered pattern still gives a clear visual indication that there is more to see in the checkered area.

However, while our entity budget solves our performance problems by capping the amount of entities to draw at any time, it has not fully solved the perception problem because this budget may still be larger than the information processing capacity of the viewer. To better support overview, we introduce local aggregation control (Section 3.5.3), where the user can selectively expand and collapse branches by rolling the mouse wheel or clicking (left-click for expand and right-click for collapse) individual entities on the display. This gives the user direct control of which detail level to view the hierarchy, while simultaneously supporting drill-down and roll-up operations (Section 3.5.2).

According to guideline G4, we must now make sure that aggregates and items can be easily distinguished. We achieve this by coloring aggregates blue, while items remain green. This also gives a straightforward indication that there is more information to be gleaned if the user simply expands a specific aggregate. Other possibilities to communicate this may have been to use a special border for the aggregate nodes, or to add a drop shadow or a "stack" graphic to the node (like for the processes in the growing polygons [31] technique).

We can now make the observation that by cutting down on the visual complexity as described above, we have also reduced the amount of information conveyed through the visualization. Guideline G2 tells us that aggregates should communicate information about the underlying (aggregated) data, while guideline G5 admonishes us to be conservative in doing so, lest we end up with an even more visually complex display than before. Our solution is to introduce a family of simple glyphs (similar to the glyphs in the ZAME [28] system) drawn in the corner of each treemap node that show various metrics of the contained subtree. In particular, we use a small bar histogram to show the distribution of file sizes in a directory structure (Figure 18). When designing the glyphs, we must also be mindful of guideline G5, which tells us to indicate the decreased fidelity of the glyph representations; interactive aggregation control also helps in this.

However, at this point, we note that the rectilinear appearance of a histogram rendered as a bar chart may be easily confused with the rectilinear visual representation of the treemap. This conflicts with guideline G4, which states that visual aggregates should be easily distinguishable, as well as to some extent G3, which argues for a clean visual appearance of visual aggregates. To avoid this conflict, we must find a visual appearance for the histogram glyph that is easily distinguishable from the treemap representation. One option may be to round the edges of the bar chart glyph, but this may not be sufficient for accurate disambiguation. Instead, we choose to add a new type of glyph based on pie charts to make the distinction perfectly clear.

Another important point to consider is guideline G6, which states that taking the aggregation too far may cause misinterpretation. Fortunately, treemaps generally have high interpretability due to their parent-child layout. However, for our aggregated treemap, initializing the visualization to show only the root of the tree as a single visual aggregate may cause the viewer to misunderstand the data, or even think that there is nothing more to see in the visualization. A simple solution is to expand the first few levels of the tree upon initialization, as well as to explicitly show that there is underlying data to explore, for example by giving the number of children in a tooltip or label.

Figure 19 shows the final redesigned version of our treemap visualization, now with support for hierarchical aggregation. The visualization has been implemented in Java using the Piccolo [10] toolkit,

and can be executed using a Web browser from the following URL: https://engineering.purdue.edu/ elm/projects/aggtreemap/

6.2 Aggregated Line Graphs

Basic line graphs are a ubiquituous form of data graphics [19] used to show all manners of data ranging from sequences of sensor readings such as seismographic, biological, and scientific data, as well as general data such as bandwidth usage, stock market data, and popularity ratings, to name a few. Recent work by Byron and Wattenberg [16] also studied aesthetics and layout techniques for stacked graphs, a specialization of line graphs.

Without going into the detail of the previous example for aggregated treemaps, adding hierarchical aggregation to a line graph can be done in essentially two ways: either by (1) aggregating the dimensional axes (X or Y), or by (2) aggregating multiple data series shown in the same graph. The approach taken will result in radically different designs.

For the first approach, aggregating the axes, Andrienko and Andrienko [4] show how to aggregate dimensions into higher-level intervals—e.g., aggregating 60 measurements at one-second intervals into a single measurement for a one-minute interval. A suitable visual aggregate (obeying G2, G3, and G4) could be to show the envelope (i.e., the extents) for each aggregated unit, as well as additional data such as average, median, or even distribution. By controlling the interval size, we can easily maintain a visual entity budget (G1).

For the other approach, van Wijk and van Seelow [75] use hierarchical clustering—thus supporting an entity budget (G1)—to combine multiple time series into averages of the underlying data (G2) and then represent them as normal line graphs in the display. However, while this is certainly simple (G3), the visual aggregates are not distinguishable from the visual items (violating G4). Also, care must be taken to control the level of clustering so that the resulting line graph does not misrepresent the underlying data (G6). A perhaps better approach would be to adapt the opacity bands [34, 82] representation to show the extents and averages of several aggregated data series.

7 Conclusions and Future Work

Our ambition with this article has been to present a model for hierarchical aggregation in information visualization for the purpose of improving overview and scalability of large-scale visualization. While the focus of our work is primarily on supporting the limited perceptual capabilities of the human viewer, many of these techniques will also come in useful for overcoming the technical challenges of rendering massive datasets. We have surveyed a set of standard visualization techniques and seen how they can be extended with hierarchical aggregation functionality. We have also described a set of general interaction techniques for manipulating aggregated visualizations. From this work, we have been able to formulate a set of common design guidelines for building new aggregated visualization techniques as well as extending existing ones with hierarchical aggregation.

Our future work will include refining the model further and deriving new interaction techniques that are common to this whole class of aggregated visualizations. We are also interested in experimentally examining the trade-off between simplification and accuracy that is inherent in all data abstraction models. Furthermore, coupling geometric zoom and level-of-detail expands the model of space-scale diagrams by providing an additional parameter: the granularity/density of the visualized items, an aspect that we want to study further. Finally, this article so far only discussed tree-shaped hierarchical aggregation schemes—a more powerful concept to explore in the future would be to use DAGs for hierarchical aggregation of certain data structures.

Acknowledgment

This work was supported in part by a grant from the joint Microsoft Research/INRIA ReActivity project.

References

[1] J. Abello and F. van Ham. Matrix Zoom: A visual interface to semi-external graphs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 183–190, 2004.

[2] J. Abello, F. van Ham, and N. Krishnan. ASK-GraphView: A large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):669–676, 2006.

[3] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. John Wiley & Sons, New York, second edition, 1984.

[4] N. Andrienko and G. Andrienko. *Exploratory Analysis of Spatial and Temporal Data: A Systematic Approach*. Springer-Verlag, 2006.

[5] C. Appert and J.-D. Fekete. OrthoZoom scroller: 1D multi-scale navigation. In *Proceedings of the ACM CHI 2006 Conference on Human Factors in Computing Systems*, pages 21–30, 2006.

[6] D. Archambault, T. Munzner, and D. Auber. Tuggraph: Path-preserving hierarchies for browsing proximity and paths in graphs. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 113–121, 2009.

[7] D. W. Archambault. *Feature-Based Graph Visualization*. PhD thesis, University of British Columbia, 2008.

[8] D. Auber. Tulip — A huge graph visualization framework. In M. Jünger and P. Mutzel, editors, *Graph Drawing Software*, Mathematics and Visualization, pages 105–126. Springer-Verlag, 2004.

[9] D. Auber, Y. Chiricota, F. Jourdan, and G. Melançon. Multiscale visualization of small world networks. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 75–81, 2003.

[10] B. B. Bederson, J. Grosjean, and J. Meyer. Toolkit design for interactive structured graphics. *IEEE Transactions on Software Engineering*, 30(8):535–546, 2004.

[11] F. Bendix, R. Kosara, and H. Hauser. Parallel sets: A visual analysis of categorical data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 133–140, 2005.

[12] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. DeRose. Toolglass and Magic Lenses: The see-through interface. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, volume 27, pages 73–80, Aug. 1993.

[13] L. Billard and E. Diday. *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. John Wiley & Sons, 2007.

[14] R. Blanch and É. Lecolinet. Browsing zoomable treemaps: Structure-aware multi-scale navigation techniques. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1248–1253, 2007.

[15] H. Byelas and A. Telea. Visualizing metrics on areas of interest in software architecture diagrams. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 33–40, 2009.

[16] L. Byron and M. Wattenberg. Stacked graphs - geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.

[17] S. K. Card and D. Nation. Degree-of-interest trees: A component of an attention-reactive user interface. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 231–245, 2002.

[18] M. Chuah. Dynamic aggregation with circular visual designs. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 35–43, 1998.

[19] W. S. Cleveland and M. E. McGill, editors. *Dynamic Graphics for Statistics*. Statistics/Probability Series. Wadsworth & Brooks/Cole, Pacific Grove, CA, USA, 1988.

[20] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2009. to appear.

[21] C. Csallner, M. Handte, O. Lehmann, and J. T. Stasko. FundExplorer: Supporting the diversification of mutual fund portfolios using context treemaps. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 203–208, 2003.

[22] Q. Cui, M. O. Ward, E. A. Rundensteiner, and J. Yang. Measuring data abstraction quality in multiresolution visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):709–716, 2006.

[23] M. C. F. de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):378–394, July/Sept. 2003.

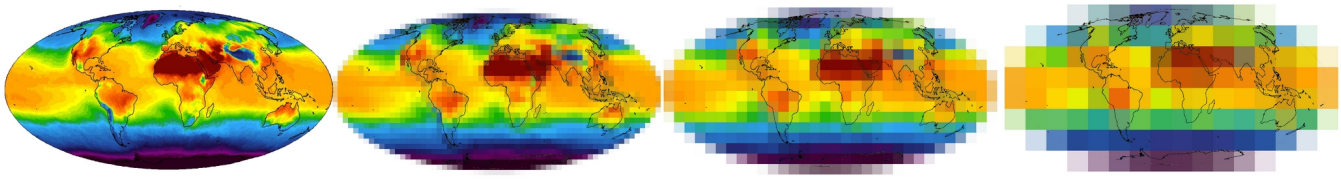


Fig. 20. NASA AIRS surface air temperature map for July 2003 (left) and spatially aggregated at three different levels of detail. Base data source: *http://www-airs.jpl.nasa.gov/*

[24] G. DiBattista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[25] A. Dix and G. Ellis. By chance - enhancing interaction with large data sets through statistical sampling. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pages 167–176, 2002.

[26] P. Eades and Q.-W. Feng. Multilevel visualization of clustered graphs. In *Proceedings of the Symposium on Graph Drawing*, number 1190 in Lecture Notes in Computer Science, pages 101–112, 1996.

[27] G. Ellis and A. J. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007.

[28] N. Elmqvist, T.-N. Do, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive large-scale graph visualization. In *Proceedings of the IEEE Pacific Visualization Symposium*, pages 215–222, 2008.

[29] N. Elmqvist, J. Stasko, and P. Tsigas. DataMeadow: A visual canvas for analysis of large-scale multivariate data. In *Proceedings of IEEE Symposium on Visual Analytics Science & Technology*, pages 187–194, 2007.

[30] N. Elmqvist, J. Stasko, and P. Tsigas. DataMeadow: a visual canvas for analysis of large-scale multivariate data. *Information Visualization*, 7(1):18–33, 2008.

[31] N. Elmqvist and P. Tsigas. Causality visualization using animated growing polygons. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 189–196, 2003.

[32] N. Elmqvist and P. Tsigas. CiteWiz: a tool for the visualization of scientific citation networks. *Information Visualization*, 6(3):215–232, 2007.

[33] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 117–124, 2002.

[34] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the IEEE Conference on Visualization*, pages 43–50, 1999.

[35] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Navigating hierarchies with structure-based brushes. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 58–64, 1999.

[36] G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM CHI'86 Conference on Human Factors in Computer Systems*, pages 16–23, 1986.

[37] G. W. Furnas. A fisheye follow-up: further reflections on focus + context. In *Proceedings of the ACM CHI 2006 Conference on Human Factors in Computing Systems*, pages 999–1008, 2006.

[38] G. W. Furnas and B. B. Bederson. Space-scale diagrams: Understanding multiscale interfaces. In *Proceedings of the ACM CHI'95 Conference on Human Factors in Computing Systems*, pages 234–241, 1995.

[39] E. R. Gansner, Y. Koren, and S. C. North. Topological fisheye views for visualizing large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):457–468, July/Aug. 2005.

[40] M. Ghoniem, J.-D. Fekete, and P. Castagliola. On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. *Information Visualization*, 4(2):114–135, 2005.

[41] Y. Guiard and M. Beaudouin-Lafon. Target acquisition in multiscale electronic worlds. *International Journal of Human-Computer Studies*, 61(6):875–905, 2004.

[42] Y. Guiard, M. Beaudouin-Lafon, and D. Mottet. Navigation as multiscale pointing: Extending fitts' model to very high precision tasks. In *Proceedings of the ACM CHI'99 Conference on Human Factors in Computing Systems*, pages 450–457, 1999.

[43] H. Hagh-Shenas, S. Kim, V. Interrante, and C. G. Healey. Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1270–1277, 2007.

[44] C. Heine and G. Scheuermann. Manual clustering refinement using interaction with blobs. In *Proceedings of the Eurographics - IEEE VGTC Symposium on Visualization*, pages 59–66, 2007.

[45] N. Henry and J.-D. Fekete. MatrixExplorer: a dual-representation system to explore social networks. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of Visualization/Information Visualization 2006)*, 12(5):677–684, 2006.

[46] N. Henry, J.-D. Fekete, and M. J. McGuffin. NodeTrix: a hybrid visualization of social networks. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1302–1309, 2007.

[47] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[48] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 139–148, 2000.

[49] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[51] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[52] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, Jan./Mar. 2000.

[53] D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[54] J. LeBlanc, M. O. Ward, and N. Wittels. Exploring N-dimensional databases. In *Proceedings of the IEEE Conference on Visualization*, pages 230–237. IEEE Computer Society Press, 1990.

[55] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, June 1994.

[56] F. Olken and D. Rotem. Simple random sampling from relational databases. In *Proceedings of the Conference on Very Large Data Bases*, pages 160–169, Los Altos, CA 94022, USA, 1986.

[57] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 89–96, 2004.

[58] K. Perlin and D. Fox. Pad: An alternative approach to the computer interface. In *Computer Graphics (ACM SIGGRAPH '93 Proceedings)*, pages 57–64, 1993.

[59] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 57–64, 2002.

[60] G. Ramos and R. Balakrishnan. Zliding: fluid zooming and sliding for high precision parameter manipulation. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 143–152, 2005.

[61] B. E. Rogowitz, L. A. Treinish, and S. Bryson. How not to lie with visualization. *Computational Physics*, 10(3):268–273, 1996.

[62] P. J. Rousseeuw, I. Ruts, and J. W. Tukey. The bagplot: A bivariate boxplot. *The American Statistician*, 53(4):382–387, November 1999.

[63] B. Shneiderman. Tree visualization with tree-maps: A 2-D space-filling approach. *ACM Transactions on Graphics*, 11(1):92–99, Jan. 1992.

[64] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE*

Software, 11(6):70–77, Nov. 1994.
[65] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343, 1996.
[66] J. H. Siegel, E. J. Farrell, R. M. Goldwyn, and H. P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, 72(1):126–141, July 1972.
[67] M. Sifer. User interfaces for the exploration of hierarchical multidimensional data. In *Proceedings of the IEEE Symposium on Visual Analytics Science & Technology*, pages 175–182, 2006.
[68] P. Simonetto, D. Auber, and D. Archambault. Fully automatic visualisation of overlapping sets. *Computer Graphics Forums (Proceedings of EuroVis 2009)*, 8(3):967–974, 2009.
[69] J. Stasko, R. Catrambone, M. Guzdial, and K. Mcdonald. An evaluation of space-filling information visualizations for depicting hierarchical structures. *International Journal of Human-Computer Studies*, 53(5):663–694, 2000.
[70] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):176–187, 2003.
[71] D. F. Swayne, D. T. Lang, A. Buja, and D. Cook. GGobi: evolving from XGobi into an extensible framework for interactive data visualization. *Computational Statistics & Data Analysis*, 43(4):423–444, 2003.
[72] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, 1983.
[73] J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.
[74] J. M. Utts. *Seeing Through Statistics*. Brooks/Cole, 3rd edition, 2005.
[75] J. van Wijk and E. R. van Seelow. Cluster and calendar based visualization of time series data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 4–9, 1999.
[76] J. J. van Wijk and W. A. A. Nuij. Smooth and efficient zooming and panning. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 15–22, 2003.
[77] M. O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proceedings of the IEEE Conference on Visualization*, pages 326–333, 1994.
[78] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of the ACM CHI 2006 Conference on Human Factors in Computing Systems*, pages 811–819, 2006.
[79] L. Wilkinson. *The Grammar of Graphics*. Springer-Verlag, 1999.
[80] G. J. Wills. An interactive view for hierarchical clustering. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 26–32, 1998.
[81] J. Yang, M. O. Ward, and E. A. Rundensteiner. InterRing: An interactive tool for visually navigating and manipulating hierarchical structures. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 77–84, 2002.
[82] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate data sets. *Computers & Graphics*, 27(2):265–283, Apr. 2003.



Niklas Elmqvist is an assistant professor in the School of Electrical and Computer Engineering at Purdue University in West Lafayette, IN, USA. Having joined Purdue in fall 2008, he was previously a Microsoft Research postdoctoral researcher in the AVIZ team of INRIA Saclay - Île-de-France located at Université Paris-Sud in Paris, France. He received his Ph.D. in December 2006 from the Department of Computer Science and Engineering at Chalmers University of Technology in Göteborg, Sweden. His research specialization is in information visualization, human-computer interaction, and visual analytics. He is a member of the IEEE and the IEEE Computer Society.



Jean-Daniel Fekete is a Senior Research Scientist (DR2) at INRIA Saclay - Île-de-France, one of the leading French national research centers, in Orsay, south of Paris. He leads the AVIZ team since 2007, which focuses on data analysis and visualization research. The AVIZ team is located in and collaborates with the Computer Science Department (LRI) at Université Paris-Sud. AVIZ studies analysis and visualization of large datasets, combining machine learning approaches with information visualization and multiscale interaction techniques to help analysts explore and understand massive data. Jean-Daniel's research topics include network visualization, evaluation of information visualization systems, and toolkits for user interfaces and information visualization. His research is applied in several fields such as biology, history, sociology, digital libraries, and business intelligence. He is a member of the IEEE.