# Using R-trees for Interactive Visualization of Large Multidimensional Datasets

Alfredo Giménez, René Rosenbaum, Mario Hlawitschka, and Bernd Hamann

Institute for Data Analysis and Visualization (IDAV), Department of Computer Science, University of California, Davis, CA 95616-8562

**Abstract.** Large, multidimensional datasets are difficult to visualize and analyze. Visualization interfaces are constrained in resolution and dimension, so cluttering and problems of projecting many dimensions into the available low dimensions are inherent. Methods of real-time interaction facilitate analysis, but often these are not available due to the computational complexity required to use them. By organizing the dataset into a level-of-detail (LOD) hierarchy, our proposed method solves problems of both inefficient interaction and visual cluttering. We do this by introducing an implementation of R-trees for large multidimensional datasets. We introduce several useful methods for interaction, by queries and refinement, to explain the relevance of interaction and show that it can be done efficiently with R-trees. We examine the applicability of *hierarchical parallel coordinates* to datasets organized within an R-tree, and build upon previous work in *hierarchical star coordinates* to introduce a novel method for visualizing bounding *hyperboxes* of internal R-tree nodes. Finally, we examine two datasets using our proposed method and present and discuss results.

## 1 Introduction

As measuring instruments advance technologically, datasets increase in both dimension and quantity. It becomes very difficult to interactively explore and analyze large, multidimensional datasets because of the high computational complexity required. Visualizing these dimensions also becomes a major problem for large dimensionalities and large datasets since standard visualization interfaces are constrained to a small number of dimensions and resolutions.

In fields of algorithms and complexity, efficient methods for data processing are often introduced through the use of hierarchical data structures. We have applied a hierarchical structure to large multidimensional datasets by generating an R-tree that contains the dataset. We utilized the efficiency of R-trees by implementing several interactive operations for analysis, and also used the hierarchical properties of R-trees to visualize the data at increasing levels-of-detail (LODs) in order to reduce visual clutter.

To achieve appropriate low-dimensional visualization of high-dimensional data within a hierarchy, we have implemented existing methods of hierarchical multidimensional visualization and have extended upon one of these methods in

order to accommodate it for more beneficial and efficient use within an R-tree structure. Specifically, we have examined *hierarchical parallel coordinates* and built on previous work to develop a new method for *hierarchical star coordinates*.

## 2 Previous Work

### 2.1 Multidimensional Visualization

Multidimensional visualization explicitly involves the problem of how to project $d$ dimensions onto the a small number of dimensions available on visualization interfaces. Popular methods to do so include using visual cues, multiple visualizations, and alternative coordinate systems.

Chernoff [1] introduced a method using *visual cues* which involved transforming individual features of a face geometrically, and visualizing each multidimensional element as the resulting face. However, he stated that this technique is constrained to a small number of dimensions. This is an inherent problem; *visual cues* must be explicitly defined for each dimension.

Wright [2] introduced the use of multiple visualizations, scatterplot matrices, where a matrix of two-dimensional scatterplots is displayed such that every dimension is plotted against every other dimension. Similar multiple visualization schemes have been developed as well; however, with all these techniques, either the number of dimensions is constrained by the screen space available for multiple plots, or the visualization cannot display all dimensions at once.

Alternative coordinate systems attempt to provide a visualization for any number of dimensions. We have implemented and built on two of these techniques, specifically *parallel coordinates* [3] and *star coordinates* [4].

### 2.2 Hierarchical Visualization of Multidimensional Data

In order to generate a visualizable hierarchy from a dataset, several proposed methods involve hierarchical clustering algorithms. Fua [5] presented one of these algorithms based on proximity information and Linsen [6] presented another one based on density functions. Though effective for generation of a hierarchy, they both involve an added preprocessing step to cluster the data. These approaches have high computational complexity for generation and interactive operations, since they are not guaranteed to be balanced trees.

## 3 Main Idea

We introduce a method to generate a hierarchical structure of data which allows for efficient interactive operations as well as methods for visualization of data within this hierarchical structure. In contrast to previous work, our method provides a great degree of efficiency and requires minimal data-specific information, while also adding functionality for analysis.

We propose using R-trees to generate this hierarchy and examine the benefits for doing so in section 4. R-trees provide functionally visualizable aggregate items within an LOD-hierarchy while also increasing efficiency, which improves upon the problems encountered in some previous proposals (see subsection 2.2).

We examine methods to visualize aggregate items as well as data items within the R-tree in section 5. Some of these methods are already well-known, and we introduce a new method to visualize multidimensional R-tree aggregate items based on some existing proposals. We examine two types of interactive operations, queries and refinement, in section 6. Finally, we apply our proposed methods on real datasets in section 7.

## 4 R-trees: An Effective Data Structure for Interactive Visualization of Large Multidimensional Datasets

In order to provide 1) a scalable hierarchy for large multidimensional datasets, 2) visualizable and accurately representative aggregate items within that hierarchy, and 3) efficient interactive operations on the structure, we propose organizing datasets into R-trees.

### 4.1 Generation of an LOD-Hierarchy

R-trees generate an LOD-hierarchy of aggregate and data items in a "bottom-up" fashion. All individual data elements are inserted into the bottom level, and nodes are split into two new ones when their respective number of children exceeds the maximum number of children, $m$. Whenever the root node is split, a new LOD in the hierarchy is introduced. Every internal node contains a number of children and a region which bounds all of its children. Node splitting in R-trees is a widely covered research topic, as the optimal solution requires factorial time complexity [7]. Our implementation uses linear splitting, a method which delivers accurate enough results for our application as well as linear time complexity.

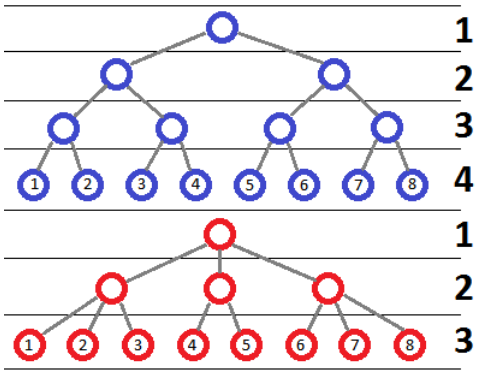


**Fig. 1.** Two R-trees with the same 8 elements inserted, but with different user-specified values for $m$ (maximum number of children per node). The top has an $m$ value of 2 and 4 levels-of-detail (LODs), while the bottom has an $m$ value of 3 and 3 LODs. The top requires storage of 7 *hyperboxes* (one per internal node), while the bottom requires storage of 4. The ability to define $m$ allows for dynamic LODs and storage space, and in turn, different quantities of refinable *hyperboxes* and regions-of-interest (ROIs) within each level.

R-trees allow for alteration of their internal tree depth, and different tree depths directly affect both user preferences and storage space. We can control the depth of the hierarchy by specifying different values for $m$. A larger value of $m$ corresponds to fewer splits and, therefore, fewer levels within the hierarchy. With more children per node, there are more available refinable nodes per level and less total aggregate items. This means that there are more detailed specification of ROIs within a level, and less total internal storage space is required. However, having too many children per node contributes to visual clutter and less LODs within the hierarchy. For smaller datasets, a small value of $m$ is useful, because more LODs organize the data more efficiently for interactive operations and introduce more LODs. Lower values do, however, increase the storage space. An illustration of the differences between high and low $m$ values is shown in figure 1.

### 4.2 Effective, Visualizable Aggregates

In order to allow for a scalable representation, we require a structure that not only organizes the data into an LOD-hierarchy, but allows for appropriate visualization of levels within it. For this reason, it is crucial that we generate aggregate items that are accurately representative to the actual data, as well as usable in various visualization schemes. An item that is accurately representative of the data is one which does not remove semantic information from the dataset.

The R-tree aggregate items are ranges of values for each of the total $d$ dimensions, which we will denote as *hyperboxes*. In one dimension, a *hyperbox* is a range of points, or an extension of a single point, which is a line segment. In two dimensions, a *hyperbox* is a range of lines, or an extension of a single line segment, which is a rectangle. We continue this process of extending lower-dimensional *hyperboxes* in order to generate *hyperboxes* of unlimited dimensionality.

These *hyperboxes* are accurately representative aggregate items for visualizing internal levels of a hierarchical data structure because they denote where the children of their respective nodes are as well as how sparse or dense the elements within that *hyperbox* are, due to their bounding property. These characteristics allow the user to draw conclusions about what values within the dimensions of the data are most common as well as how varied the dimensional values are in comparison with each other.

### 4.3 Efficiency for Real-Time Interaction

It is crucial for our application to interactively operate on datasets that are not only large in quantity of elements, but large in dimensionality as well; therefore, generation, queries, and refinement operations must be low in computational complexity. The use of R-trees allows us to execute hierarchical generation and interactive operations very quickly, even with large datasets of many dimensions.

R-trees are inherently balanced trees, which provides a great deal of efficiency. Every time a node is split, its children are distributed amongst the new nodes in order to maintain the same depth throughout the R-tree and avoid empty nodes. This property allows for insertions, deletions, and searches to be

made in worst-case $O(md \log_m n)$ time for $n$ data elements of $d$ dimensions. Our proposed method to execute queries requires even less time than searches, as we will explain in detail in section 6.2. This method for generation of a hierarchy improves upon Linsen's [6], which does not maintain tree balance and generates many empty nodes. Furthermore, while Linsen's [6] method applies a more accurate automatic generation of clusters, it necessitates specification of a density function and introduction of another preprocessing step to evaluate densities and quantities of clusters.

## 5 Visualization of Datasets within R-trees

The visualization of massive multidimensional datasets as organized within R-trees requires a transformation from the $d$ dimensions of the R-tree data into the two dimensions available on screen space, as well as effective methods of visualizing both aggregate items and individual data elements.

We examined and implemented two alternative coordinate systems for multidimensional visualization. We show that R-trees are visualizable using *hierarchical parallel coordinates*, and introduce a method which builds upon Kandogan's [4], which we denote as *hierarchical star coordinates*. In both cases, we describe how to represent multidimensional data elements as well as bounding *hyperboxes*.

### 5.1 Hierarchical Parallel Coordinates

Parallel coordinate visualization was defined by Inselberg [3], and has been extended to represent multidimensional value ranges, *hyperboxes*, by Fua [5]. In *parallel coordinates*, each dimension is denoted by a single line such that all lines are unique and parallel to each other, and points are represented as polygonal lines with values plotted on each respective dimensional line. To represent *hyperboxes*, we simply plot two data elements in this fashion, the maximum and minimum, and fill the area between both segments, so that we attain a polygon which covers all values within the range of the *hyperbox*.

### 5.2 Hierarchical Star Coordinates

For single elements within *star coordinates*, the technique is, again, explicitly defined [4], and we propose extending this idea to also represent *hyperboxes*, as Fua [5] did with *parallel coordinates*. The dimensional axes are represented by a set of lines which all emanate from a single point (the *star coordinate* origin). The data elements in *star coordinates* can either be represented as a polygonal line which connects dimensional values, or as a single point which is translated in the direction of each dimensional line by the magnitude of the value. We use the latter. In order to represent the *hyperboxes*, we cannot simply plot the minima and maxima of the range as with *parallel coordinates*, because the area between the minimum point and maximum point no longer accurately represents the

range. Instead we introduce a method that plots all possible combinations of the minimum and maximum values in each dimension–the corners of the *hyperbox*– and fills the area between those points. We fill the area by calculating the convex hull of these points and constructing its respective convex polygon.

## 6 Interactive Operations for Visualization and Analysis

### 6.1 Refinement Methods for Dynamic Removal of Clutter

The fact that R-trees are an LOD-hierarchy allows for several methods to remove clutter, both programmatically and interactively. Clutter is defined as the ratio of LOD to available screen resolution; thus, LOD corresponds directly to the amount of clutter in the visualization. In an LOD-hierarchy, it is possible to refine down the hierarchy and therefore alter the LOD of the visualization dynamically. Dynamic alteration of LODs, in turn, allows for dynamic removal of clutter.

To be more explicit, refinement means breaking down certain regions within the R-tree into their more detailed components. This is done by removing a *hyperbox* from the visualization and replacing it with its child *hyperboxes* or data elements. This provides us with a more accurately detailed visualization.

Refinement can be done uniformly or non-uniformly as well as programmatically or interactively, with different benefits for each.

**Uniform Programmatic** We introduce one uniform programmatic method for refinement: a simple breadth-first search (BFS). This method refines all *hyperboxes* of a single level within the hierarchy. In this way, it is possible to alter the LOD uniformly–all elements visualized have the same LOD at all times. In this way, the user can draw initial conclusions about the dataset as a whole and determine which areas are more of interest than others. When the user determines a region within the dataset that is particularly of interest, the ability to refine non-uniformly and interactively becomes crucial.

**Non-Uniform Interactive** To facilitate interactive non-uniform refinement, we introduce a method to execute queries. These queries allow the user to define which dimensions and regions are of interest, and then refine the corresponding *hyperboxes* as desired.

### 6.2 Interactive Queries for Real-time Analysis

The user may construct and execute two types of queries on the R-tree: 1) bounded and 2) overlap. Both query methods iterate over nodes of the R-tree and execute comparisons between the constructed query and each node processed. Both also require the same input: a set of 3-tuples, which each specify 1) a dimensional index, from 1 to $d$ inclusively, 2) a value within that dimension, and 3) a margin value.

**Bounded Queries** *Bounded queries* find nodes whose *hyperboxes* completely encompass the query values and margins in the specified dimensions.

This type of query facilitates interactive searching for programmatically generated clusters of data–because it tests for nodes that completely encompass the query region, this is an effective way for the user to find bounded clusters created by the R-tree generation.

**Overlap Queries** *Overlap queries* find all nodes which overlap any part of the query values and margins in the specified dimensions.

Because *overlap queries* allow searching for any data within the specified range, they are useful for drawing conclusions about the data regardless of the internal R-tree structure, and therefore is based on the data elements rather than the data structure.

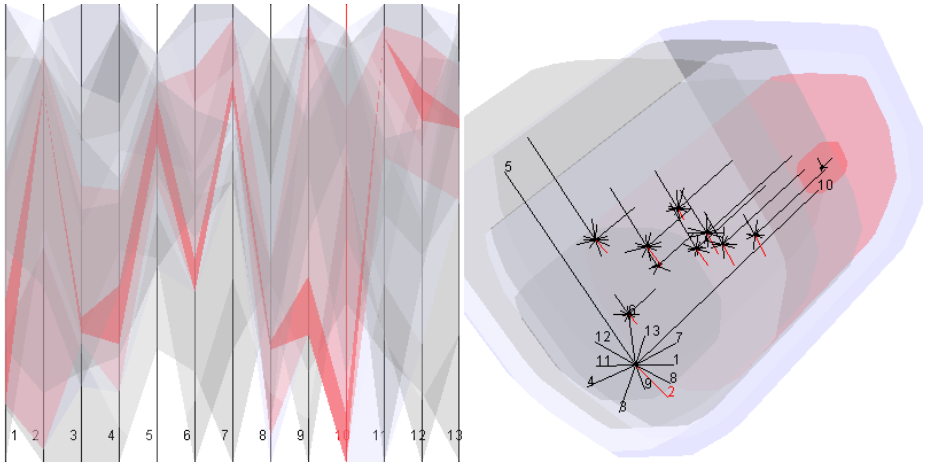


**Fig. 2.** We highlight visualized hyperboxes in parallel coordinates (left) and star coordinates (right). By increasing the LOD of a large hyperbox containing sparsely distributed data, we obtained detail of a small hyperbox (at the end of the axis numbered 10) within that region containing densely distributed data (the small red hyperbox).

## 7 Visual and Interactive Data Exploration and Analysis

### 7.1 Isolating Outliers

It is essential in visual data analysis to isolate and either extract more detail from or eliminate outliers. With our system, it is possible to discover outliers quickly and either decrease their significance or refine them in order to examine them more closely. We explain the process by example, with a dataset of regional wines [8] of 13 dimensions and 178 elements.

As 178 elements is a fairly small number of data, we choose a small value for $m$, 2, in order to increase LODs available. Next, we execute BFS refinements to draw initial conclusions about where outliers may lie. From this step, we can see distributions of values in each dimension. Some are densely packed around certain values, like dimension 10 and dimension 5. The outliers in each dimension are those values which lie outside of the densely packed regions. In order to show the efficacy of removal as well as examination, we remove the outliers in dimension 5 and examine in detail the outliers of dimension 10.

In order to remove an outlier, we construct a query which contains it. As explained in section 6, when we are looking for specific elements, like outliers, *overlap queries* are more effective. After running the *overlap query* and coloring the result white, the outliers in dimension 5 barely contribute to the visualizations.

To examine an outlier, we execute an *overlap query* in red followed by a number of overlap refinement operations until we obtain the LOD required. After just a few overlap refines, we achieve a very specific outlying region visualized in both the *parallel coordinates* and *star coordinates*, while avoiding clutter due to the region-specific refinement operations.

### 7.2 Examining Correlations

We can examine correlations between dimensions and between individual clusters/elements by performing refinement operations until we achieve the desired LOD in a ROI, and arranging the visualization to show correlations. As example we analyze a dataset of forest fires within the northeast region of Portugal [9].

After initial setup, we determine a good ROI and begin rearranging the visualization methods in order to analyze correlations. We rearrange the *parallel coordinate* axes to observe dimensional correlations: high values in dimension 4 correlate with low values of dimensions 11 and 12. In the *star coordinate* view, we increase the magnitude and vary the direction of certain dimensions, in this case 4, 7, 11, and 12, shown in figure 7.2. As we can manipulate these axes, we observe to what extent the shape of the aggregates is affected. The blue aggregates are fairly unaffected by manipulation of dimensional axis 11 and highly affected by manipulation of dimensional axis 4; therefore, these aggregates have low values in dimension 11 and high values in dimension 4. Furthermore, large hyperboxes represent very sparse distributions of data, observed in red, whereas small hyperboxes represent dense distributions, observed in blue. We conclude that a large quantity of the data has fairly high values in dimension 4 and extremely low values in dimension 11.

## 8 Possible Drawbacks

One principal drawback of our *hyperbox* visualization method is that it requires $O(2^d)$ complexity to calculate the *hyperbox* corners. The effects are rather detrimental if visualization of 20 or more dimensions is required, so improved methods

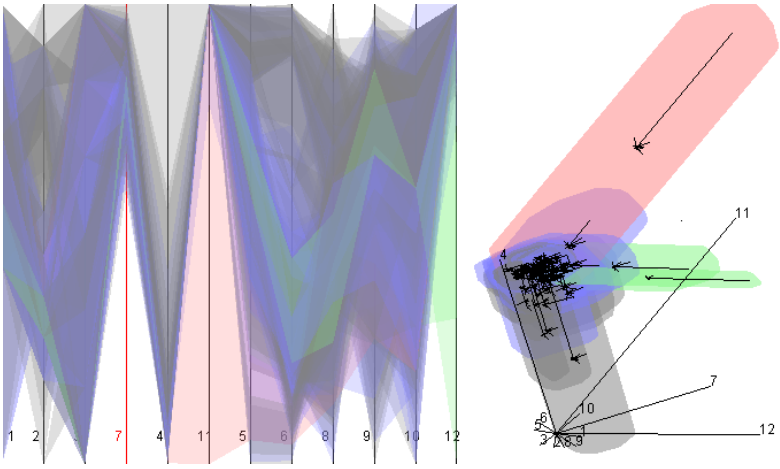


**Fig. 3.** We show hyperboxes as colored regions in parallel coordinates (left) and star coordinates (right) for a dataset of forest fires. Each numbered line in the star coordinate visualization corresponds to a dimension. These numbered lines can be manipulated in direction and length. If we extrude the line numbered 11, the red hyperbox is extruded more than any other hyperboxes. Therefore, the red hyperbox contains data with a high range of values in dimension 11.

would be necessary to provide real-time visualizations at this level of dimensionality. Note that this complexity applies to the visualization, rather than the interactive operations.

## 9 Conclusions and Possible Future Research

We have implemented and built upon several existing methods for multidimensional visualization and visualizable hierarchical structuring of multidimensional datasets. We have introduced a novel method to generate an efficient LOD-hierarchy for large, multidimensional datasets using R-trees, we have examined methods to visualize *hyperboxes* and elements within that LOD-hierarchy, and we have examined the use of interactive operations on the data to facilitate analysis. We have used existing visualization schemes, *parallel* and *star coordinates*, in order to introduce a new method for visualizing *hyperboxes*, while retaining the ability to use existing visualization methods as well. Our method for LOD-hierarchy generation provides a great deal of efficiency and functionality in contrast to previous ones, and in combination with the introduced visualization schemes and interactive operations, added benefits for analysis and exploration of data.

A possible improvement to the drawback of complexity mentioned in section 8 could be to apply Linsen's [6] splat-based ray-tracing method to these *hyperboxes*, in which case the complexity would be constrained by screen resolution, rather

than the data dimensionality. Another possible improvement, for more accurate hierarchical cluster generation, could be to develop new node-splitting algorithms based on factors other than proximity.

Future implementations of our method could significantly influence areas which use progressive refinement, such as Rosenbaum's [10] technique for device adaptation. As progressive refinement methods require generation of LOD-hierarchies for many types and sizes of multidimensional data, our method provides much of the necessary functionality.

## 10 Acknowledgements

René Rosenbaum was supported by the German Research Foundation Deutsche Forschungsgesellschaft (DFG), and Mario Hlawitschka was supported in part by NSF grant CCF-0702817. We thank our colleagues from the Institute of Data Analysis and Visualization (IDAV) at UC Davis.

## References

1. Chernoff, H.: The use of faces to represent points in k-dimensional space graphically. Journal of the American Statistical Association **68** (1973) 361–368
2. Wright, D.B.: Scatterplot matrices. Encyclopedia of Statistics in Behavioral Science **4** (2005) 1794–1795
3. Inselberg, A.: The plane with parallel coordinates. The Visual Computer **1** (1985) 69–91
4. Kandogan, E.: Visualizing multi-dimensional clusters, trends, and outliers using star coordinates. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '01, New York, NY, USA, ACM (2001) 107–116
5. Fua, Y.H., Ward, M.O., Rundensteiner, E.A.: Hierarchical parallel coordinates for exploration of large datasets. In: Proceedings of the conference on Visualization '99: celebrating ten years. VIS '99, Los Alamitos, CA, USA, IEEE Computer Society Press (1999) 43–50
6. Linsen, L., Long, T.V., Rosenthal, P., Rosswog, S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. IEEE Transactions on Visualization and Computer Graphics **14** (2008) 1483–1490
7. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, ACM (1984) 47–57
8. Forina, M.: An extendible package for data exploration, classification and correlation (2010)
9. Cortez, P., Morais, A.: A data mining approach to predict forest fires using meteorological data. In: Proceedings of the 13th EPIA 2007 - Portuguese Conference on Artificial Intelligence. (2007)
10. Rosenbaum, R., Hamann, B.: Progressive presentation of large hierarchies using treemaps. In: ISVC '09: Proceedings of the 5th International Symposium on Advances in Visual Computing, Berlin, Heidelberg, Springer-Verlag (2009) 71–80