Visualization of Cluster Hierarchies

Bjoern Heckel, Bernd Hamann

Center for Image Processing and Integrated Computing (CIPIC)/
Department of Computer Science
University of California, Davis
Davis, CA 95616-8562

ABSTRACT

Clustering is a powerful analysis technique used to detect structures in data sets. The output of a clustering process can be very large. However, if presented in a textual form, the amount of information that can be understood is limited. An alternative approach is to display the data in a graphical way. An advantage of visualization is that a larger amount of information can be perceived. Supporting user interaction and manipulation of the object space enables exploration of large data sets.

We present a technique for the visualization of cluster hierarchies. The input to our technique is a finite set of n-dimensional points. All points are initially placed in one cluster, which is recursively split, creating a hierarchy of clusters. Principal component analysis is used to determine how to optimally bisect a cluster. After splitting a cluster, a local reclassification scheme based on the Gabriel graph is applied to improve the quality of the classification. As a byproduct of the generation of the cluster hierarchy, we compute and store the eigendirections, eigenvalues and local centers for each cluster at each level of the hierarchy.

For the visualization of the cluster hierarchy, the user has to specify the three dimensions that are used for the rendering process. The local coordinate systems (centroids, eigenvalues, and eigendirections) of each cluster induce a local metric that can be utilized to define "density functions." These functions describe hyperellipsoids that we render in two different ways: (1) we generate a set of (transparent) contour surfaces (each cluster would appear as a translucent surface) or (2) we apply "ray casting" to simulate the behavior of X-rays penetrating the density fields implied by the clusters. In order to show the different levels in the hierarchy, one can either render only those clusters belonging to the same level or, alternatively, use transparency to be able to see through the 'outer shell' of a cluster and see the finer, more detailed structures inside.

Keywords: Data Analysis, Hierarchical Classification, Clustering, Visualization, Data Exploration, Data Reduction, Multiresolution Hierarchy, Ray Casting, Contouring.

1. INTRODUCTION

Modern computer and imaging technology allows the generation of larger and larger amounts of data. These massive data sets can be found in engineering applications (e.g., computational fluid dynamics), business applications (e.g., warehousing data about consumer behavior), as well as in other fields. However, the amount of data that has to be comprehended by humans has far surpassed our ability to "digest" and interpret this data. In order to gain some form of higher-level insight into massive data, it is crucial to develop technology that supports various means of analyzing and visualizing different degrees of abstraction within massive data sets.

Data analysis and visualization have a symbiotic relationship: Even in high-performance visualization environments, the number of polygons that can be displayed and manipulated in real-time is limited [2]. In addition, the perceivable amount of

Further author information -
Email: {heckel, hamann}@cs.ucdavis.edu

information displayed on a screen is bounded by resolution and by the cognitive abilities of the user. Therefore, data analysis is needed to obscure details with low informational value, showing only the essence of the data. When more detailed information is desired, data analysis can be used to generate a multiresolution hierarchy which can be explored interactively.

On the other hand, presenting the outcome of a data analysis process, normally in a textual form, allows only a limited understanding of the information. There, an order of tens of different elements is perceivable, while visualization allows the rendering of thousands of elements [1]. Visualization becomes very important with increasing complexity of the data.

The application of advanced visualization techniques coupled with user interaction during analysis provides a better delivery of information. We believe that a symbiosis of statistics-based approaches and modern visualization technology will greatly enhance our ability to comprehend quickly vast amounts of information.

2. OVERVIEW

We propose utilizing and generalizing techniques classically known in statistics and computer visualization. In particular, clustering techniques have been used to analyze multi-dimensional data sets. However, very little work has been done on the visualization of cluster hierarchies. A classical representation of a classification is the so-called dendogram [3]. A dendogram is a tree which shows how related sets of objects are nested in a hierarchical classification. This representation shows only the structure of a classification, and does not provide any information about either the domain of the objects nor about the characteristic of each set. Cone trees [15], recently developed in information visualization, can be used to lay out such a hierarchy in three dimensional space. However, it has similar properties as a dendogram and focuses on displaying structural information. Instead, a representation is needed that shows how sets of objects are characterized and how such sets are related. An initial approach is described [16] in which spheres are wrapped around groups of similar objects. Our approach considers the geometric shape of clusters and their locations, as well as their position in the hierarchy.

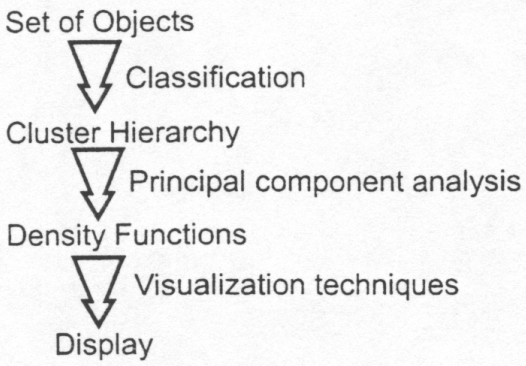

Figure 1: Overview of the cluster visualization process.

The input to our method can be a set of data tuples of arbitrary dimension. We perform a hierarchical clustering technique before doing any type of visualization. We determine a hierarchy of clusters, each one defined in terms of a cluster center and its (scaled) principal directions. We apply visualization techniques to user-specified three-dimensional subsets of our cluster hierarchy. Our current visualization module requires the user to identify those three dimensions that are most significant for subsequent data analysis. A cluster hierarchy, when restricted to three dimensions, is used to define density functions which can be superpositioned -- and reflect the extent and shape of the individual clusters. We apply three visualization methods for the rendering of a cluster hierarchy:
(1) slicing the domain of the density functions and color-coding the slices according to function value; (2) computing and rendering contours of the density functions; and (3) generating transparent images by simulating rays penetrating the three-dimensional domain of the density functions ("ray casting").

3. CLASSIFICATION

Classification is concerned with the analysis of a set of objects and their correlation. It is a classical data analysis technique that has been studied thoroughly over the last decades [11]. Its goal is to establish a set of groups such that objects assigned to the same group have certain similarities while they differ from objects in other groups. These groups are not known a-priori and must be determined by examining the characteristics of the given objects. Often, one is also interested in such data partitions at different levels of granularity. In this case, a hierarchy of nested set partitions is needed. Hierarchical classifications are very common in all fields of science, such as biology, psychology, and chemistry.

With the rapid growth of electronically stored data, hierarchical classification becomes increasingly important for data management and exploration. Classification has also been adopted as a standard technique in the emerging field of data

mining, where it is used to analyze very large data sets [10]. It is applied in an enormous variety of research areas, ranging from fields like earthquake prediction, whale monitoring, marketing, psychology, biophysics, criminology, information retrieval, image segmentation to phenetic taxonomy [4].

Referring to Tucker's terminology [4] the type of the data we use is "two-way two-mode": The set of objects to be examined is described by a two-dimensional profile matrix. The rows and columns refer to disjoint sets. The columns indicate attributes of the objects, while the rows describe the objects. M objects with n attributes are represented by an m-by-n matrix. The data set can also be understood as a sequence of n-dimensional vectors. Currently, we restrict the domain of the attributes to real numbers. Other domains (e.g., integer, complex, symbolic) could be mapped to real numbers by using appropriate transformation functions. However, the characteristics of the original domain and the transformation must be considered for measuring dissimilarities between objects and sets of similar objects, so-called clusters.

For the construction of a classification, a (dis-)similarity measure between pairs of objects and between objects and clusters is essential. The characteristic of the measure determines the output of the classification. Often, the dissimilarities of the objects are explicitly stored in a symmetric m-by-m matrix. In our case, we are interested in creating a cluster hierarchy for large data sets. Therefore, the size of the dissimilarity matrix could be very large and strongly vary during the classification process. To limit storage requirements, we consequently measure the pairwise dissimilarity implicitly as a function of the attributes of the objects/clusters. Currently, we use Euclidian distance to measure dissimilarity of two n-dimensional vectors. But for instance, converting the currency in a financial data set would possibly yield a different cluster hierarchy, if Euclidean distance were used as a measure. To avoid this effect, an affine invariant norm could be used instead [7]. Other important issues to be considered are inclusion, standardization, and weighting of the attributes describing an object [3]. Overall, choosing an appropriate dissimilarity measure requires knowledge about the nature of the data and the context of the classification.

Our method for creating the cluster hierarchy is based on an incremental and adaptive paradigm. We are using a top-down approach, since we are interested in finding clusters with non-trivial shapes, which therefore have a cardinality of more than one. So, the lowest levels of a complete hierarchy are, in general, not needed. Because of the exponential growth of clusters as a function of the depth of the hierarchy, using a bottom-up approach would result in a higher time complexity. It would be a function of the number of objects, instead of the number of desired depth of the hierarchy.

Initially, all data objects are placed in one cluster, which is recursively split until a termination criterion is met. The classification stops when a certain number of clusters have been created or the global error does not exceed a given limit. The global error is defined as the sum of the distance of each object to its associated cluster's center. Recursively, the cluster with the highest local error is split incrementally, reducing the global error of the classification. A cluster is either split in the direction of the dimension with the highest deviation or in the direction of the eigenvector with the highest eigenvalue. To determine the centers of the two new clusters that replace the split cluster, the direction vector is added to/subtracted from the center of the split cluster. In order to use the eigenvector for splitting a cluster, principal component analysis (PCA) must be conducted first. For a cluster, which consists of many associated objects, this can be very costly. Therefore, a threshold variable τ_{PCA} is used to control when each splitting mechanism should be used. If the number of objects associated with the cluster C_i to be split exceeds the value of τ_{PCA}, the cluster C_i is split in the direction of the dimension with the highest deviation. Otherwise, the eigenvector with the highest eigenvalue is used to determine the centers of the new clusters that result from splitting C_i. From our experience, using the eigenvector yields, on average, a classification with a slightly smaller global error. After splitting a cluster, one of three different reclassification strategies is used:

1. *Global reclassification*: All objects are reassigned to their closest cluster. The centers of the clusters are then updated by averaging the attributes of the associated objects.
2. *Partition of split candidate*: Only the objects associated with the split cluster are considered for the reclassification. These objects are assigned to either one of the two new clusters, which then is updated based on the assigned objects.
3. *Local reclassification*: The objects to be reclassified are given by the objects that are assigned to the split cluster or to the Gabriel neighbors of that cluster. Two clusters are Gabriel neighbors, if they are connected in the Gabriel graph constructed from the cluster centers. Two points p and q are connected in a Gabriel graph, if only p and q are contained in the hypersphere, which has the midpoint of p and q as the center and the distance from p to q as the diameter. The Gabriel graph of a point set is a subset of its Delaunay graph and a superset of its relative neighborhood graph [13][14]. After the reclassification, the neighbor list of affected cluster is updated.

After the classification is established, the cluster hierarchy must be created. If partitioning is used as a reclassification scheme, the cluster hierarchy follows directly from the construction of the classification. For global and local classification, the hierarchy must be explicitly created, since the inclusion property is not necessarily valid, for a spit cluster and its two children; due to the reclassification, an object assigned to a cluster might not be a member of its child clusters. Therefore, the hierarchy is created in a bottom-up fashion by recursively merging two clusters that have the lowest dissimilarity. While constructing the hierarchy, a *rank* is assigned to each cluster. The leaves of the hierarchy have rank 0. A cluster with rank 0 is called *leaf cluster*, while the cluster with the highest rank is considered *root cluster*. A root cluster contains all objects of the data set. After merging two clusters, the rank of the resulting cluster is the maximum of its child clusters plus one. Clusters with the same rank are considered to lie on the same *level* of the hierarchy.

The time complexity for the classification process using global reclassification is $O(c^2*n)$, where c is the number of clusters and n the number of objects. Therefore, global reclassification is hardly applicable to larger data sets. For local reclassification and partition, the time complexity is $O(c*n)$. However, partitioning is independent of the number of dimensions, while the execution time using local reclassification depends on the number of neighbors, which generally increases with the number of dimensions. (In the formula above, the number of dimensions is assumed to be constant.) A three-dimensional, randomly distributed data set using partitioning as a reclassification scheme is about eight times faster than local reclassification. The global error using local reclassification is about 5%-10% less than using partitioning, and 1%-3% higher in respect to global reclassification.

Alternatively, other heuristic reclassification schemes could be used. For example, the quality of the local classification scheme could be improved by recursively propagating changes throughout the neighbor clusters until no changes occur. However, we find that using the local reclassification scheme described above is a good compromise between output quality and execution time. It provides a tremendous speed-up compared to global classification, while the error is only slightly greater.

4. DENSITY FUNCTIONS

Each cluster in the cluster hierarchy has an associated center and a set of eigenvectors v_i and eigenvalues λ_i obtained by principal component analysis. This data describes a cluster's hyperelliptical shape. It can also be interpreted as a local, orthogonal coordinate system: the center of the cluster χ is the origin and the eigenvectors, scaled by the corresponding eigenvalues, are the unit vectors of the local coordinate system. Thus, each node in our tree representation of the cluster hierarchy has an associated coordinate system, which we use to define a cluster-specific density function.

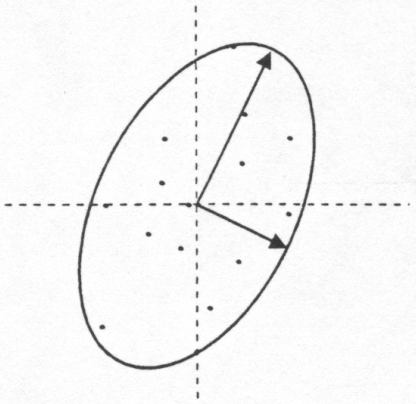

Figure 2: Local coordinate system.

To measure the distance δ of a given point ξ in n-dimensional space to a certain cluster, we use the metric implied by its local coordinate system. First, ξ is transformed to a point in the local coordinate system ξ' by projecting it onto the eigenvectors and scaling it by the corresponding eigenvalues:

$$(x_i) = \frac{(\xi - \chi) \cdot v_i}{\lambda_i} \qquad (1)$$

Then the distance to the center of cluster is:

$$\delta(x) = \sqrt{\sum_{i=1}^{n} x_i^2} \qquad (2)$$

We use these local distance measures to define *density functions* with compact support for each cluster. More specifically, the region over which the density function is non-zero contains all the points that are part of the cluster. Outside this region, the density function is zero. We use a Hermite-like approach: The density function has its maximum at the origin of the cluster's local coordinate system (and zero derivative at this point) and decreases smoothly to zero on the boundary of its compact support region. The boundary is determined by that point in the cluster that lies furthest away from the origin of the local

coordinate system. The distance α to this point is used to scale the parameter δ of the Hermite function. The value of the density function varies in a cubic fashion from its maximum value 1 at the origin to zero on the boundary, where its derivative is also zero. The independent variable of this distance function is distance from the local origin. Furthermore, this distance is measured in the metric induced by the local eigendirections and eigenvalues, i.e., points in the domain having the same distance to the origin lie on hyperellipsoids.

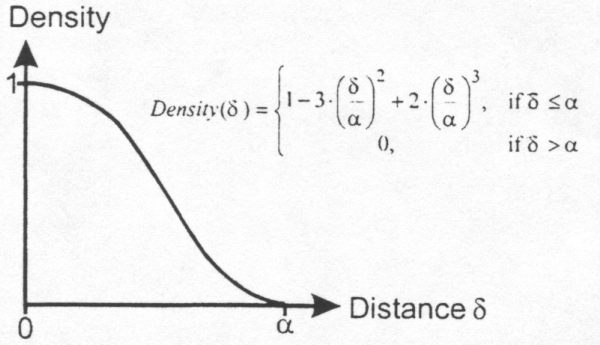

Figure 3: Density function.

We assign a color value to each cluster. Given a point P in n-dimensional space, the color contribution of a cluster to the color at P is calculated by using the density function to determine the intensity of the cluster's color in P.

We use the principle of superposition to define two overall functions for density and color at a given point. These are in some sense the sum of all the individual, cluster-specific functions. Later we will use these functions for the visualization of a cluster hierarchy. We use the following paradigm: Density functions associated with clusters with a low rank have relatively large density values associated with the origins of their local coordinate systems. Density functions associated with a high rank have relatively small density values associated with the origins of their local coordinate systems. The reason for this becomes more obvious in the context of the visualization process: In order to generate transparent images of the cluster hierarchy, smaller clusters have to shine through the larger parent clusters. This is achieved by weighting the individual density functions using individual or level-specific maximum values for the individual density functions. This also allows highlighting certain clusters or layers in the hierarchy. In order to support data exploration, these parameters can be changed interactively. Only a subset of the complete cluster hierarchy is used for the visualization process in order to provide real-time visualization using slicing and contouring. Clusters are only considered if their density function has a positive weight.

Another approach is to only consider the density of clusters with the lowest rank at a given point. In this case, if the point P lies in a cluster, the contribution of its parent clusters are simply ignored in the overall function. Because clusters with an equal rank may overlap, more than one cluster may contribute to the density or color at P.

For ray casting, instead of measuring the distance to the cluster center, we alternatively use the distance to the boundary of the cluster as input for the density function. We call the resulting function a cluster's *boundary density function*. Using this function in the visualizing process yields the shell of a cluster. This approach allows the user to better see through clusters and look at their interior. In order to control the thickness of the shell, the boundary distance is multiplied by a scaling factor. Choosing a thin shell results in very sharp contours of a cluster's shell.

5. SPECIAL CASES

It is possible that certain clusters consist of individual points, of points lying on a line, or points lying in a plane. Thus, the inherent dimensionalities of such clusters are 0-dimensional, 1-dimensional, and 2-dimensional, respectively. The principal component analysis will yield zero eigenvalues for the corresponding directions.

In the context of defining our density functions and visualizing our clusters as inherently 3-manifold structures, we must replace eigenvalues of value zero by an artificial, small number. This number has to be chosen large enough, such that sampling the density function for visualization purposes does not miss a cluster. The number, therefore, is dependent upon the sampling rate. With replacing eigenvalues of zero we can always define a cluster-specific coordinate system whose eigendirections are all different from the null vector. This allows definition of our density functions for all possible types of clusters and to visualize all types of clusters as truly volumetric entities.

6. VISUALIZATION

Currently, we support four different display modes in three-dimensional object space. If the object space has more than three dimensions, a three-dimensional hyperspace has to be selected for the rendering process. One of the visualization modes directly uses the cluster hierarchy. The centers of clusters at a certain level in the hierarchy are displayed. By changing this level, one can observe how clusters are nested in the hierarchy. By clicking on the centers, different functions can be activated, for example, to show a cluster's Gabriel neighbors, its associated data points, or its local coordinate system. Figure 4 shows an example for this display mode - three clusters (large, green spheres) generated from 16 randomly distributed data points (small, red spheres). The size of the spheres representing clusters indicates the number of associated data objects. Gray lines describe the local coordinate systems and blue lines are connecting Gabriel neighbors. This visualization mode proved to be useful to get a first impression about the data set and find appropriate parameters for rendering the density functions.

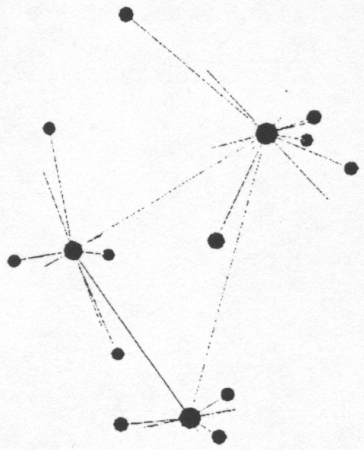

Figure 4: Viewing the output of the classification process.

We use three standard visualization techniques to render density functions: slicing, contouring, and ray casting. Since all density functions are known explicitly once cluster centers, eigendirections, and eigenvalues have been determined, we can evaluate them at arbitrary locations, which is needed for slicing. Alternatively, we can evaluate them on a uniform-rectilinear grid, which is needed for contouring and ray casting. For these visualization techniques, the gradient vector is determined at a given point and used for Phong shading.

Slicing is performed in the following way: Once the user has defined an arbitrary cutting plane, we discretize this plane using a uniform-rectilinear grid and evaluate the underlying density function at all grid points. The individual quadrilaterals defined by the grid in the cutting plane are then Gouraud-shaded. It is possible to move a cutting plane in real time through the density function.

Contouring is achieved by means of the Marching Cubes algorithm: The density function is evaluated at the grid points of a uniform-rectilinear mesh that discretizes the region of interest in three dimensions. The grid implies a piecewise trilinear approximation of the underlying continuous density function. We construct a triangulation of a contour by determining the individual contour pieces for each grid cell.

Our system supports two standard ray casting techniques: Levoy's and Sabella's algorithms. Again, we evaluate the underlying density function on a uniform-rectilinear grid and assume a piecewise trilinear variation for the approximation of the density field. Given a user-specified eye point and line of sight, we shoot a two-dimensional array of rays into the set of data cubes and integrate the density function along each one of those viewing rays. The result is a two-dimensional image showing the various levels in the cluster hierarchy. At this point, it becomes evident why we chosen our cluster-specific density functions in the way that we did: Clusters consisting of a small number of points have associated density functions that are non-zero over only small regions, yet their associated maximal density values (at their centers) are very high. This produces translucent images where the small clusters stand out as high-intensity points shining through the next levels in the cluster hierarchy.

7. EXAMPLES

We have applied our hierarchical classification and visualization approach to various test data. We provide two examples: (1) four overlapping spheres places on the corners of a square; and (2) data points along a helix.

For the sphere data set, four thousand data points were generated which are uniformly distributed in four spheres. These spheres are located on the corners of a square and overlap by 10% of their radius. A cluster hierarchy consisting of seven clusters and four leaf clusters is determined. For the visualization, the root cluster and the four leaf clusters are selected. In figure 5, ray casting is applied to the density function of the cluster hierarchy. A low weight is used for the root cluster and high weights are associated with the leaf clusters. The resulting picture shows a hull around the data set and four overlapping spheres inside it. In figure 6, the density function is replaced by the corresponding boundary density function. Comparing figure 5 and figure 6 shows that when the boundary function is used, more levels in the hierarchy can be displayed at once.

Pictures generated from the boundary density function appear sharper. The density function stresses the 'influence' of a cluster center at a certain point. The farther away a point is from a cluster center, the smaller is the influence of that center and the smaller is its color contribution.

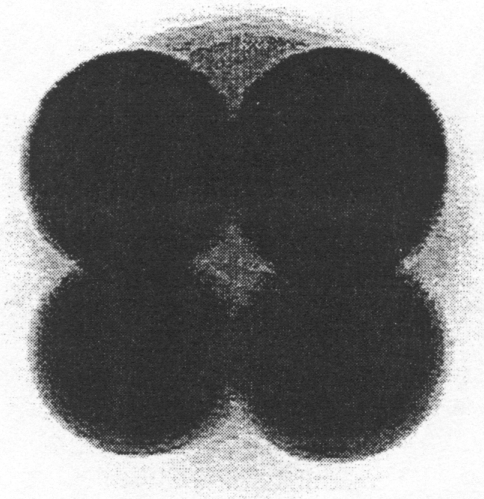

Figure 5: Ray casting applied to the density function of the sphere data set.

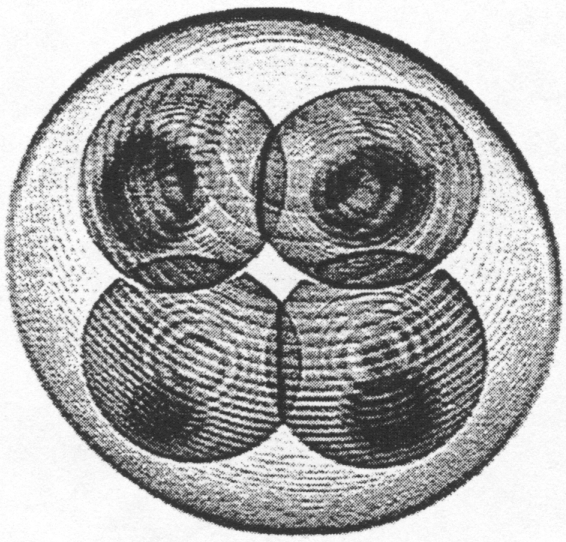

Figure 6: Ray casting applied to the boundary density function of the sphere data set.

In figure 7, slicing is applied to the sphere data set. Three orthogonal and transparent cutting planes are used. Contouring is applied in figure 8 to render the density function. The pictures for the sphere data set indicate that the clustering algorithm correctly identifies the overlapping spherical clusters.

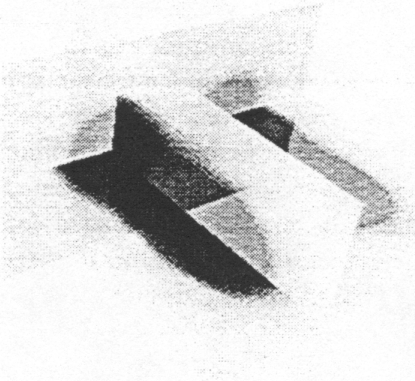

Figure 7: Ray casting applied to the density function of the sphere data set.

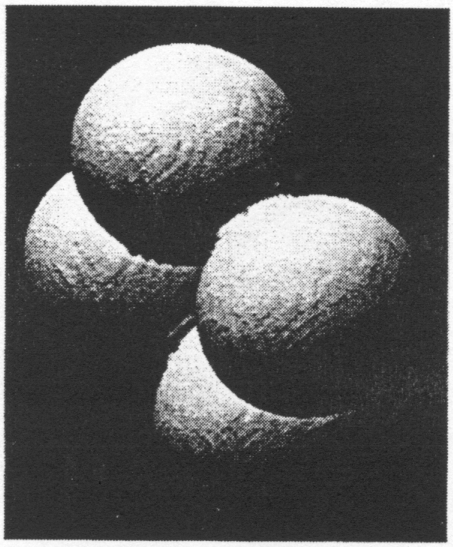

Figure 8: Contouring applied to the density function of the sphere data set.

The helix data set is obtained by randomly choosing points along the curve [sin(t), cos(t), t], t varying between 0 and 6π, and randomly perturbing the points away from the exact helix. The data set used to generate the pictures below consists of 5000 samples. A classification with 30 leaf clusters is determined; these clusters are selected for the visualization process. Figure 9 shows ray casting applied to the density functions of the leaf clusters. High transparency of the clusters creates an X-ray-like image. In figure 10, the boundary density function is rendered with low transparency value. The pictures show ellipsoids with different orientation and shape representing the underlying helix structure. In figure 11, the shape of the helix is reconstructed by applying contouring on the density function.

Figure 9: Ray casting applied to the density function of the helix data set.

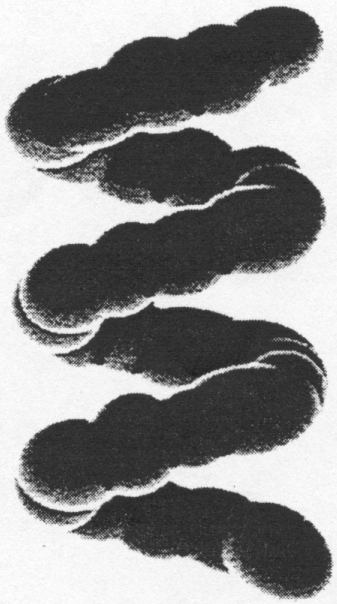

Figure 10: Ray casting applied to the boundary density function of the helix.

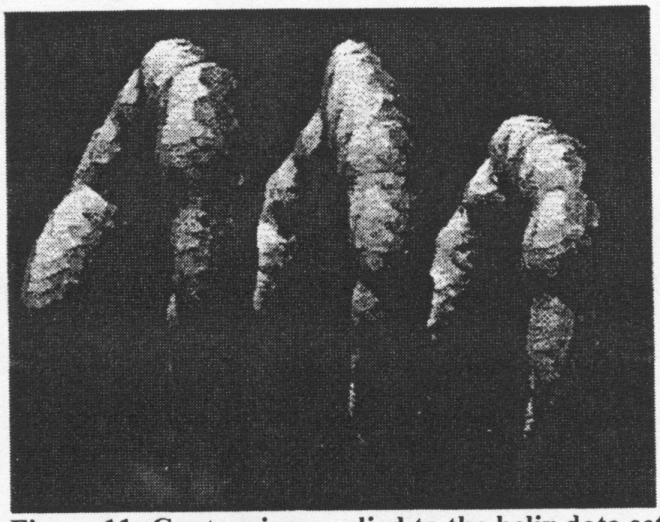

Figure 11: Contouring applied to the helix data set.

8. CONCLUSION AND FUTURE WORK

We presented a new approach to visualize cluster hierarchies. It allows a user to analyze and interactively explore large data sets. The fundamental idea is to represent a cluster hierarchy as a density function, which is rendered using standard visualization techniques. Our approach enables examination of data sets on different level of detail and local refinement of the level of abstraction. The classification provides a reduction of the original data set while minimizing the deviation error. This is important for allowing interactive data exploration in real-time.

The classification process and the density function concept are applicable to data sets of an arbitrary number of dimensions, however visualization is currently limited to three-dimensional object (sub-)space. In order to display complete data sets of higher dimensionality, transformations like (1) nonlinear projection based on Kohonen's topology preserving maps [5][9] or multidimensional scaling [6] could be applied. Alternatively, we are currently working on new visualizations techniques for multi-variant data sets that can be used for exploring multiresolution hierarchies. We are also working on the parallelization

of the classification and visualization processes to utilize multiprocessor workstations and network of workstations. This will enable interactive exploration of a larger selection of clusters in real-time.

Other extensions of our work will include research on adaptively changing the level of detail. For example, if the viewer is very far away from a helix-shaped object as described in chapter 7, only the outer hull will be displayed. Moving closer to the helix will incrementally reveal the underlying shape of the object.

Currently, the primitives used for rendering are hyperellipsoids. The shape of these elements results from the metric used to measure dissimilarities between objects. Other primitives, for example polyhedrons, could be derived from the cluster hierarchy by applying feature detection techniques.

9. ACKNOWLEDGEMENTS

This work was supported by the IBM Almaden Research Center, the National Science Foundation under contracts ASC-9210439 (Research Initiation Award) and ASC 9624034 (CAREER Award) to Mississippi State University and the University of California, Davis. In addition, this project was funded by the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP, and Lawrence Livermore National Laboratory under contract W-7405-ENG-48 (B335358) to the University of California, Davis. We would like to thank the members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis.

10. REFERENCES

1. Christopher Ahlberg, Erik Wistrand, "IVEE: An Information Visualization and Exploration Environment", Proceedings of IEEE Information Visualization 1995, Computer Soc. Press, Los Alamitos, 1995.
2. Gregory M. Nielson, Thomas A. Foley, Bernd Hamann, David Lane, "Visualization and Modeling Scattered Multivariante Data", IEEE Computer Graphics and Applications,11(3), 1991.
3. Allan D. Gordan, "Hierarchical Classification", in: P. Arabie, L. J. Hubert, G. De Soete, eds., *Clustering and Classification*, World Scientific Publ., River Edge, 1996.
4. Phipps Arabie, Lawrence J. Hubbert, "An Overview of Combinatorial Data Analysis", in: P. Arabie, L. J. Hubert, G. De Soete, eds., *Clustering and Classification*, World Scientific Publ., River Edge, 1996.
5. Martin A. Kraaijveld, Jianchang Mao, Anil K. Jain, "A Nonlinear Projection Method Based on Kohonen's Topology Preserving Maps", in: IEEE Transactions on Neural Networks, 6(3), May 1995.
6. A. Mead, "Review of the development of multi-dimensional scaling methods", The Statistician, 33:27-35, 1992.
7. Richard Franke, Gregory M. Nielsen, "Scattered Data Interpolation and Applications: A Tutorial and Survey", in: Hagen, H. and Roller D., eds., *Geometric Modeling*, Springer-Verlag, New York, 1991.
8. Pak Chung Wong, R. Daniel Bergeron, "Multivariate Visualization using Metric Scaling", Proceedings of IEEE Visualization 1997, IEEE Computer Society Press, Los Alamitos, pp. 111-118.
9. Teuvo Kohonen, "Self-Organizing Maps", Series in Information Sciences, Vol. 1, Springer-Verlag, Heidelberg, 1995.
10. Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, "The KDD Process for Extracting Useful Knowledge from Volumes of Data", in: *Communications of the ACM*, 39(11), pp. 27-34, November 1996.
11. P. Arabie, L. J. Hubert, G. De Soete, "Clustering and Classification", World Scientific Publ., River Edge, 1996.
12. M. de Berg, M. van Kreveld, O. Overmars, O. Schwarzkopf, "Computational Geometry - Algorithms and Applications", Springer-Verlag, Berlin, 1997.
13. A. Okabe, B. Boots, K. Sugihara, "Spatial Tessellations", John Wiley & Sons, Chichester, 1992.
14. Marc Levoy, "Display of surfaces from volume data", IEEE Computer Graphics & Applications, (10):3, pp. 29-37, 1988.
15. G. G. Robertson, J. D. Mackinlay, S. K. Card, "Cone Trees: Animated 3d visualization of hierarchical information", Proceedings of ACM CHI '91 Conference on Human Factors in Computing Systems, pp. 189-194, 1991.
16. R. Hendley, "Case Study - Narcissus: Visualization Information", Proceedings of IEEE Information Visualization '95, IEEE Computer Society Press, Los Alamitos, pp. 90-96,1995.

11. BIOGRAPHIES

Bjoern Heckel is a Ph.D. student at the University of California, Davis. His primary research interests are data analysis and visualization. He is the author/co-author of 14 research papers. Heckel worked as a programmer and technical consultant for a variety of companies including IBM Germany, and Aon Corporation. From May 1995 to May 1996 he worked at the NSF Engineering Research Center for Computational Field Simulation and was one of the charter members of the Hector project. Since June 1997 he has been employed by IBM's Almaden Research Center in San Jose, California, where he is doing research on the representation of large sets of multimedia objects found in digital libraries.

In 1996, Heckel received an M.S. in computer science from the Friedrich-Alexander-University, Erlangen/Nuremberg, Germany, with a minor in computer-integrated manufacturing.

Heckel is a member of the Association for Computing Machinery (ACM).

Bernd Hamann is an Associate Professor of Computer Science and Co-Director of the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis, and an Adjunct Professor of Computer Science at Mississippi State University. From 1991 to 1995 he was a faculty member in the Department of Computer Science at Mississippi State University and a research faculty member at the NSF Engineering Research Center for Computational Field Simulation. His primary interests are visualization, computer graphics, and computer-aided geometric design (CAGD). He is the author or co-author of 50 research papers and has presented his research at leading conferences in the U.S. and in Europe.

Hamann received a B.S. in computer science, a B.S. in mathematics, and an M.S. in computer science from the Technical University of Braunschweig, Germany. He received his Ph.D. in computer science from Arizona State University in 1991.

Hamann was awarded a 1992 Research Initiation Award by Mississippi State University, a 1992 Research Initiation Award by the National Science Foundation, and a 1996 CAREER Award by the National Science Foundation. In 1995, he received a Hearin-Hess Distinguished Professorship in Engineering by the College of Engineering, Mississippi State University.

Hamann is a member of the Association for Computing Machinery (ACM), the Institute of Electrical and Electronics Engineers (IEEE), and the Society for Industrial and Applied Mathematics (SIAM).