Hierarchical Clustering for Volumetric Scalar Fields

By

CHRISTOPHER S. CO
B.S. (University of California, Davis) 2001

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Committee in charge

2002

Hierarchical Clustering for Volumetric Scalar Fields

Christopher S. Co
June 2002
Computer Science

Hierarchical Clustering for Volumetric Scalar Fields

Abstract

We present a flexible method by which large unstructured scalar fields can be represented in a simplified form. Using a parallelizable classification algorithm to build a cluster hierarchy, we generate a multiresolution representation of the original data. The method uses principal component analysis (PCA) for cluster classification and a fitting technique based on a set of radial basis functions. Once the cluster hierarchy has been generated, we utilize a variety of techniques for extracting different levels of resolution.

Professor Kenneth I. Joy
Dissertation Committee Chair

To my loving and ever supportive father.

Contents

List of Figures

List of Tables

Acknowledgments

No one person ever works alone. I owe a great deal to my advisor, Ken Joy, and to many others who helped produce the ideas and results discussed in this document. These collaborators include Bjoern Heckel, Hans Hagen of the University of Kaiserslautern, and Bernd Hamann.

Chapter 1

Introduction

Data simplification and multiresolution methods are necessary when exploring and visualizing large data sets. Researchers require high quality representations of their data without the excessive cost of representing the entire data set. Redundancies in data can be exploited to obtain a simplified representation which preserve important features.

We propose a framework for the construction of a hierarchical representation of scattered scalar field data. In a preprocessing step, we iteratively refine an initially coarse representation using clustering techniques to generate the hierarchy. At runtime, we extract data from this hierarchy to support interactive data exploration. Our technique is extremely flexible. Several methods for data interpolation may be used to obtain values for data points in the hierarchy, and multiple ways to extract data from the hierarchy may be defined. The resulting hierarchy depends on minimum cluster size, type of error measure used, and the partitioning method. The termination condition chosen determines the highest resolution represented by the data hierarchy.

The invariants in this method are the splitting criterion and structure of the data hierarchy. The cluster with the greatest error, is split in each refinement step. Binary partitioning is performed to create child clusters from the split cluster.

The preprocessing phase begins by computing an initial cluster. This cluster is used as the root node for a binary tree and added to the set of split candidates. Iteration then begins by obtaining the cluster with the greatest error from the candidate pool. A splitting plane is computed and used to partition the cluster into two sub-clusters. Approximate values and errors are computed for each sub-cluster. Value approximations are obtained using radial basis functions (RBFs). RBFs have been shown to be effective in computing interpolants for scattered data [1]. Child clusters are inserted into the set of split candidates and set to be the children of the cluster just split in the hierarchy. Cluster refinement continues until one of two termination conditions are satisfied. Refinement either terminates when a maximum number of iterations has been completed or when the errors of the clusters are below some user-defined threshold.

We will refer to the generated tree as a Cluster Binary Tree (CBT), cluster hierarchy, or data hierarchy.

The data extraction phase consists of a binary tree traversal over the CBT. We discuss two traversal methods: level-based and error-based. The level approach collects data in the hierarchy in a depth first fashion traversing the tree down to a maximum depth. The error approach gathers data in the cluster hierarchy based upon an error threshold. In this way, a multitude of resolutions are represented by one compact binary tree.

In Section 2, we review some previous work done in data simplification and cluster analysis. In Section 3, we discuss cluster partitioning methods, with emphasis on the use of PCA to orient the splitting plane. In Section 4, we describe methods for function value approximation at the cluster centers. In Section 5, we discuss extacting data from the hierarchy. In Section 6, we discuss some of our results and analyze the method.

Chapter 2

Related Work

An abundance of work has been done to create multiresolution representations for data with well defined structure. Many existing techniques apply only to structured rectilinear or curvilinear data [14, 6, 17, 12]. These techniques exploit special properties of the implicit or explicit connectivity in the data by superimposing hierarchical data structures, such as oct trees [14, 6], or a nesting of rectilinear grids in an adaptive mesh refinement (AMR) [17].

One drawback to these techniques is their limited application domain. *Scattered data* refers to data without implicit or explicit connectivity which typically cannot be handled by any of the mentioned techniques without imposing a grid structure, such as a Voronoi tessellation [13] or tetrahedrization [3, 4, 16, 2]. Once a grid structure has been imposed, simplification schemes over the simplex mesh can be performed to obtain multiple resolutions in a data hierarchy [3, 4, 16, 2]. Imposing such a structure may be disadvantageous, as additional storage is needed to represent it. Furthermore, creating an initial tessellation can be computationally expensive.

A wealth of scattered data methods exist in geometric modeling. For example, given a set of 3-D scan data, an implicit surface representation is constructed by fitting a function $F(x, y)$ to the given data [5, 7, 1, 11]. This reconstruction is useful since a mesh of any desired resolution can be generated by evaluating $F(x, y)$. The drawback of such a reconstruction technique is the computational cost to fit the surface. For example, complex optimization routines to determine ideal knot locations can take hours or even days, though these optimizations prove useful in modeling complex surfaces [5]. Such computational cost can be justified in the context of generating high quality surface models for industrial use. However, such highly accurate functions are seldom needed to visualize massive volume data.

Surface modelers strive to obtain 2-D parameterized surfaces. What we seek are 3-D parameterized volumes. Thus, 3-D data points in a 3-D scalar field become volumetric knots, whose equivalent in geometric modeling are 2-D surface knots. The increased dimensionality increases the computation load, further incentive to move to a less global optimization scheme.

In summary, keeping too explicit a representation can be impractical and memory intensive. At the same time, deriving too implicit a representation can be too computationally intensive. Results from each extreme were shown to provide excellent performance but at a high cost. It is our goal to find a good balance between explicit and implicit representation. Constructing a hierarchy of knots and applying localized data fitting steps can produce high quality results while at the same time not requiring extreme computational costs.

Obtaining the hierarchy can be accomplished using clustering techniques. Heckel et. al. showed that hierarchical cluster representation proved useful in classifying vector field data [9]. He generated vector field hierarchies by splitting vector data, separating regions in the domain of a vector field exhibiting different behaviour. Similar clustering techniques were applied to surface reconstruction by Heckel et. al [8]. Heckel et. al. used PCA to determine the degree of co-planarity of points lying on some unknown surface. This method creates near-planar tiles for approximating surfaces. These tiles, when stitched together, formed a closed surface. Using methods similar to these, we build a multiresolution representation for scattered scalar fields.

Chapter 3

Partitioning Clusters

Given a scalar field S defined over a set of points $p_i = (x_i, y_i, z_i),\ \ i = 1, 2, \ldots, n$, we let s_i be the scalar value associated with p_i, i.e.

$$S(\, x_i, y_i, z_i \,) = S(\, p_i \,) = s_i$$

We define a cluster C to be a subset of points in S.

We define the center of the cluster C, denoted by C_c as

$$C_c = \frac{1}{|C|} \sum_{j \in C} p_j$$

and the error C_e as

$$C_e \, = \, max \; | \, F_j \, - \, F_c \, |, \;\; j \in C$$

where each $F_j \, = \, S(\, p_j \,)$ and F_c is the value assigned to C_c. (Value approximation at the cluster center is discussed in Section 4.) In other words, the error C_e is the maximum deviation in scalar value between the approximated value and the values of the points in C. This error measure is simple to compute and suffices to identify clusters with high variation in scalar value.

Once a cluster has been chosen for splitting, [1] we partition the data points into two smaller clusters by defining a splitting plane that divides the cluster into two distinct groups. Center points, value approximations, and errors are computed for these two resulting sub-clusters. The sub-clusters are then reinserted into the candidate pool for further splitting.

[1]The cluster with the greatest error can be found efficiently by placing clusters into a priority queue, sorted based on cluster error.

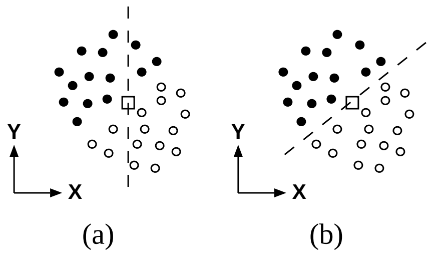


Figure 3.1: Comparison of splitting schemes. Black dots represent points of scalar value one, and white dots represent points of scalar value zero. (a) Non-optimal splitting using an axis-aligned scheme versus (b) near-optimal splitting

One method for defining the splitting plane is to use an axis-aligned scheme. In this method, the cluster center and one coordinate axis determine the splitting plane. This scheme, similar in nature to k-D tree style splitting [15], is efficient and allows us to determine the splitting plane simply, but it may not produce a near-optimal split. Figure 3.1 demonstrates this point. For this reason, we use PCA to determine a normal to orient the splitting plane. We will first discuss how 3-D PCA would be used on two dimensional scalar fields because it is easier to visualize. We then lift this scheme into $\mathbb{R}^4$ for application to 3-D scalar fields.

A detailed explanation of PCA is beyond the scope of this paper, and we refer the reader to [10] for details. Briefly, PCA computes a covariance or correlation matrix and performs an eigen-decomposition of this matrix. Eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \lambda_3$$

are obtained with corresponding eigenvectors

$$\widehat{e}_1,\ \widehat{e}_2,\ \text{and}\,\widehat{e}_3.$$

In three dimensions, $\langle \widehat{e}_1, \widehat{e}_2, \widehat{e}_3 \rangle$ define a local orthogonal coordinate system of an ellipsoid induced by the data points.

For bivariate scalar field data, we can perform PCA in 3-D to obtain a better orienting normal for a splitting line. We use the normal corresponding to the dominant axis of this ellipsoid, i.e., $\widehat{e}_1$. However, to partition the 2-D data points, we require a 2-D normal vector. We project the $\mathbb{R}^3$ eigenvector to $\mathbb{R}^2$, see Figure 3.2. In most cases, we obtain a suitable orienting normal by simply dropping the last component of the vector.

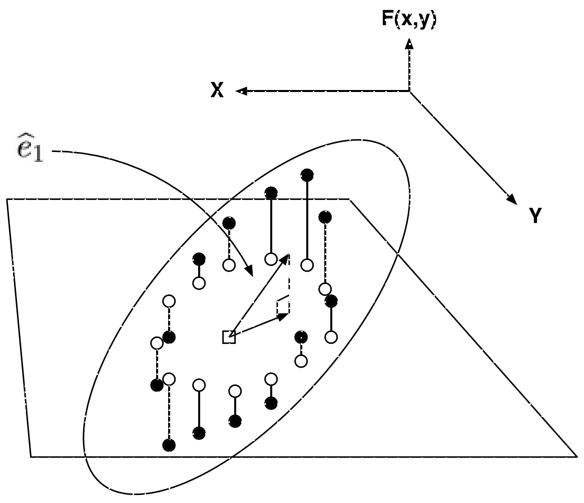


Figure 3.2: Example of 3-D PCA normal projected to $\mathbb{R}^2$ The white dots are data points in $\mathbb{R}^2$. The height of the black dots is indicative of the scalar value at the data point. The ellipsoid represents the local coordinate system computed by 3-D PCA. The dominant $\mathbb{R}^3$ eigenvector, $\widehat{e}_1$, is shown here, with its projection onto $\mathbb{R}^2$.

We must consider the case when $\widehat{e}_1$ returned by PCA is a multiple of the vector $\langle 0, 0, 1\rangle$. Such a normal projected to $\mathbb{R}^2$ corresponds to the null vector. In this case, we choose the second dominant eigenvector, $\widehat{e}_2$, which is guaranteed to be non-null when projected to $\mathbb{R}^2$, since it is orthogonal to $\widehat{e}_1$. In practice, this occurs infrequently. Figure 3.3 illustrates the progression of the cluster splitting procedure.

The technique just described generalizes to $\mathbb{R}^4$. PCA returns eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4$ with corresponding eigenvectors $\widehat{e}_1$, $\widehat{e}_2$, $\widehat{e}_3$, and $\widehat{e}_4$. Again, we use the projection of $\widehat{e}_1$ into $\mathbb{R}^3$ to define a splitting normal. When projection of $\widehat{e}_1$ maps to the null vector in $\mathbb{R}^3$, we choose $\widehat{e}_2$ as our orienting normal.

Cluster topology can have a great impact on the partitioning. When clusters become very thin in the split direction, it is possible that splitting the cluster into two sub-clusters produces a cluster with zero data points due to numerical error. When this occurs, the problem cluster is not re-inserted into the error queue. To avoid situations like this one, we attempt to make clusters as “round” as possible. Splitting along the dominant axis as defined by PCA accomplishes this objective.

Cluster size can also have a large effect on splitting. Eventually, clusters will contain

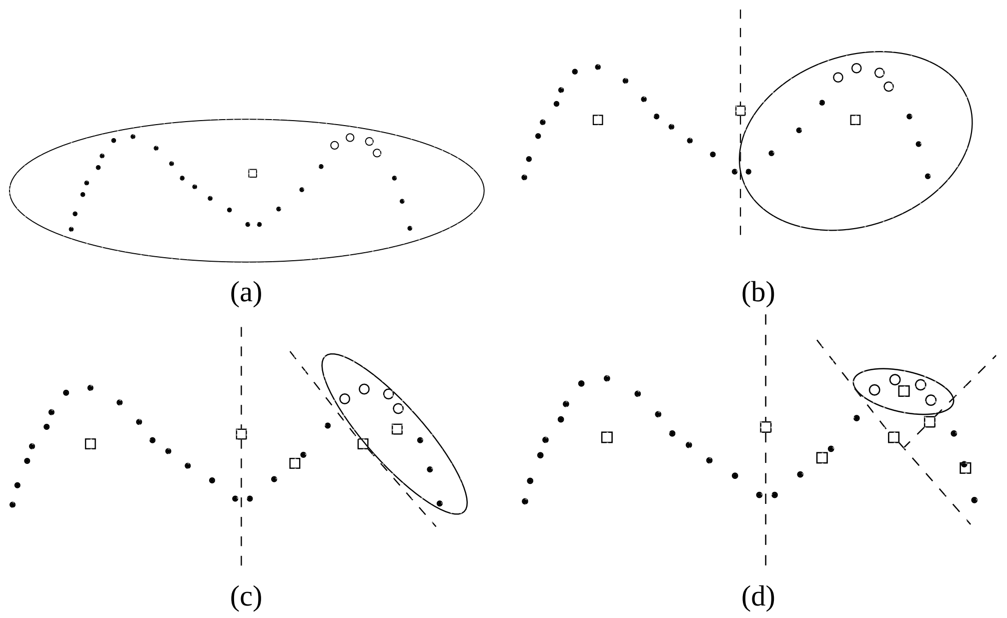


Figure 3.3: Example of clustering of 2-D scattered data. Black dots indicate a scalar value of one, and white dots indicate a scalar value of zero. Squares represent the cluster centers, which become new data points in the generated hierarchy. (a) CBT generation begins with one initial cluster ; (b) cluster is split into two sub-clusters ; only the right cluster is chosen for splitting ; (c) right cluster is split into two sub-clusters ; (d) final split ; all clusters have zero error.

only a few data points. A minimum cluster size can be defined to set a “pseudo-compression ratio” for the finest resolution in the data hierarchy. Setting a minimum cluster size can also reduce the occurrences of the zero-size cluster problem.

Chapter 4

Value Approximation

In the data hierarchy, each cluster is interpreted as a data point of a given resolution in the hierarchy, whose location is the cluster center, $C_c = (x_c, y_c, z_c)$, computed by the geometric mean of the points in the cluster.

To approximate a value at this location, we utilize a multiquadric method [5]. This type of approximation is commonly used in geometric modeling. This method effectively fits a function composed of a set of radial basis functions (RBFs) to the scattered data using least squares approximation to derive function constants and a good reparameterization of the data points.

Consider the cluster C with points $\{\, p_1, p_2, \ldots, p_n \,\}$, $p_i = (x_i, y_i, z_i)$, and corresponding scalar values $\{s_1, \ldots, s_n\}$.

We define a fitting function

$$F(x,y,z) = c + \sum_{j=1}^{N} a_j \, B_j(x,y,z),$$

where

$$\begin{aligned} B_j(x,y,z) &= [\,(x_j - x)^2 + (y_j - y)^2 \\ &\quad +(z_j - z)^2 + R^2\,]^{\pm 1/2} \end{aligned} \tag{4.1}$$

subject to conditions

$$F(p_i) = F_i = F(x_i, y_i, z_i) \; = \; s_i, \;\; i = \; 1, 2, \ldots, n\,. \tag{4.2}$$

We call c the *correction constant*, R the *multiquadric parameter*, and a_j the *blending coefficients*. The points p_i are often referred to as *knots*. Many times, equation (4.2) is too strict, and a least squares fitting is performed instead. Basis functions evaluated with $+1/2$ in the exponent are called *multiquadric* basis functions. When evaluated with $-1/2$, they are called *reciprocal multiquadric* basis functions.

It is important to make the distinction between n and N. The integer n denotes the number of scalar values in the cluster, while N denotes the number of data points used for the value approximation. Classically, N is set to be the same as n, which can be computationally expensive for large clusters. We use a local scheme by considering only the N nearest neighbors to C_c, i.e., $N = min(\,n, k\,)$, where k is some previously chosen integer, usually less than 30. These nearest neighbors can be computed in a number of ways. We make use of a k-D tree [15], where construction requires only $O(n\,log\,n)$ and searches require $O(log\,n)$.

The fitting scheme performs the minimization step

$$\underset{\{knots,R,a_j,c\}}{min} \sum_{i=1}^{N} \Big[\big(c + \sum_{j=1}^{N} a_j\, B_j(x_i, y_i, z_i) \big) - F_i \Big]^2,$$

which is separated into two minimization steps

$$\underset{\{knots,R\}}{min}\ \underset{\{a_j,c\}}{min} \sum_{i=1}^{N} \Big[\big(c + \sum_{j=1}^{N} a_j\, B_j(x_i, y_i, z_i) \big) - F_i \Big]^2,$$

where the inner minimization is performed by a least squares operation. The outer minimization is non-linear [5, 7].

After this optimization step, we obtain values for c, R, a_j, and basis function parameterization (x_j, y_j, z_j). We estimate the value at C_c as:

$$F(x_c, y_c, z_c) \,=\, c + \sum_{j=1}^{N} a_j\, B_j(x_c, y_c, z_c).$$

Our experiments have shown that this method works well, but at a high cost. The least squares fitting requires us to build an $n \times (n+1)$ linear system, i.e., an underdetermined system with an infinite number of solutions. A non-negativity constraint can be added, however, one must be careful to avoid near singular matrices. The major bottleneck is the non-linear optimization, which must optimize $3n + 1$ variables–the x-, y-, and z-components for each of n knot locations and R.

In geometric modeling, the correction constant is used smooth “bumpy” surfaces. Many scalar fields do not suffer as much from discontinuities. By removing the correction constant as a parameter we obtain a $n \times n$ linear system and no longer need to consider an underdetermined matrix problem.

We have found that increasing the local neighborhood for function value estimation precludes the necessity to optimize knot locations. For instance, we can obtain equivalent results using 25 nearest neighbors without knot optimization as we can using five neighbors and optimizing the knots. Removing the knot optimization step leaves only the multiquadric parameter R to optimize. Our experience shows that the R in this setting is frequently optimized to values so small that it is of neglible influence. Thus, fixing the multiquadric parameter to a sufficiently low value removes the need for any non-linear optimization. The blending coefficients in many cases compensates for any effect (or lack thereof) that the multiquadric parameter has on our local interpolating function. Thus, our final local interpolant is given by

$$F(x,y,z) \,=\, \sum_{j=1}^{N} a_j \, B_j(x,y,z),$$

where we determine a_j by solving

$$\underset{\{a_j\}}{min} \sum_{i=1}^{N} \Big[\big(\sum_{j=1}^{N} a_j \, B_j(x_i,y_i,z_i) \big) - F_i \Big]^2$$

and evaluate F at the cluster center C_c

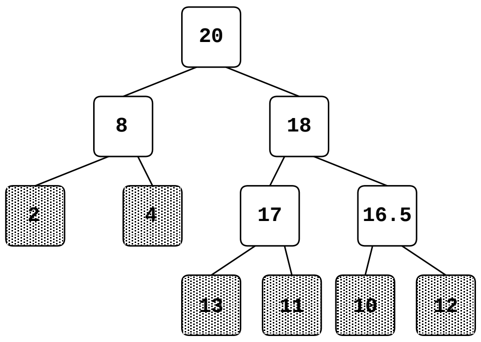
Chapter 5

Data Extraction and Traversal

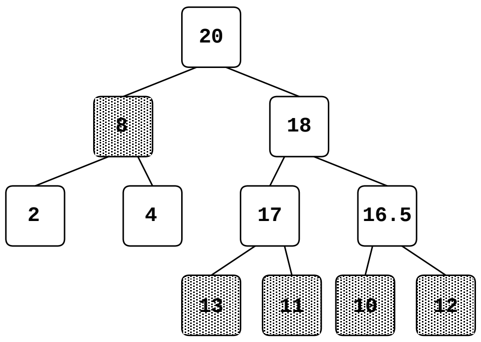
Extraction of data from the CBT is performed by traversing the tree in a depth first fashion, using either a *level-based* traversal that obtains the data points in the hierachy at a given level of the tree, or an *error-based* traversal which returns data points in the hierarchy that have an error below a given threshold. If a leaf node is encountered in the CBT during data extraction, we use the value of the leaf and continue traversal. The hierarchy can only guarantee data points with an error less than or equal to the maximum error at termination of the preprocessing.

Figure 5.1 illustrates the two traversal schemes over a small CBT. The error is shown inside each node. Note that the error-based approach offers a more compact representation, whereas the level-based approach provides better spatial distribution by covering the space spanned by the hierarchy with more data points. This phenomenon can be seen in the examples provided in Section 6.

The data extracted from the CBT defines a scattered scalar field in its own right. From this, a multitude of visualization techniques may be applied.



(a)



(b)

Figure 5.1: CBT hierarchy traversed in (a) a level-based manner with level request of 3 and (b) an error-based manner with error threshold of 10.00.

Chapter 6

Results

We have generated CBT hierarchies for three rectilinear data sets and used the same sampling resolutions to produce visualizations for comparison.

Table (6.1) summarizes the preprocessing results to generate the CBTs. For value approximation at the cluster centers and for sampling the data, we used 25 nearest neighbors and fixed the multiquadric parameter to 0.025. The minimum splittable cluster size was set to be two. All function values are between zero and 255.

One disadvantage of this technique is that maintaining the hierarchy for large data sets can be memory intensive. However, this method lends itself to data parallelism. Once a cluster is split, its sub-clusters can be further split on separate machines over a network. Since the hierarchy is based on a binary tree, merging the final results low cost. In the case of the argon bubble data set, we chose to divide the preprocessing work among three machines.

The argon bubble data set was too large to process on a single standard desktop workstation. We therefore processed it in parallel using three workstations. One parent machine performed

Data set	Number of splits	Skipped splits	CBT height	Error at termination
Teddy bear	200,000	10	22	8.15
Head	300,000	21	23	4.52
Argon bubble	600,001	7	25	0.92
Space shuttle	200,000	251	26	2.00

Table 6.1: CBT hierarchy statistics

an initial split and distributed the sub-clusters to two additional machines. The two additional machines each computed 300,000 splits on their respective halves and wrote their sub-trees to disk. The parent machine merged the two subtrees into the root tree, producing a unified cluster hierarchy.

As mentioned in Section 3, eventually clusters either converge to the granularity of the data set or suitable partitioning is not possible due to lack of precision. “Ignored splits” can result in the iterative refinement, and their numbers are listed in Table 6.1.

Lower resolutions introduce a certain amount of noise. Figure 6.1(a) shows a low resolution representation of the argon bubble using 11,246 data points, which is roughly 0.21% the size of the original data set. Fewer points exist outside the bubble itself, which causes “under-representation” in that region. Noise can be seen in the volume rendered result in Figure 6.1(d). We consider this effect a reasonable trade-off, since this region varies little in scalar value and therefore can be viewed as “less important.” This effect is not observed in higher resolutions, see Figure 6.1(b), which uses 215,633 data points (4.11% original data size). The volume rendered result shown in Figure 6.1(e) does not exhibit the noise seen in Figure 6.1(d). Figure 6.1(f) shows the volume rendered result of the finest resolution of the argon bubble cluster hierarchy. This resolution guarantees error values no greater than 0.92, using only 599,995 data points, which is about 11.44% the size of the original data. A summary for the images in Figure 6.1 is in Table 6.2.

Similar comments can be made for the results of the head data set. Figure 6.2(a) shows a low resolution representation of the head that produces a noisy volume rendered result as seen in Figure 6.2(d), where the entire back plate from the scan is not represented in detail. Salient features of the head are still discernable. Only about 5,000 points were used to construct the image, as opposed to roughly 2,000,000 points used to generate the volume rendered result of the original data shown in Figure 6.2(c). Deficiencies in low resolutions are eliminated in a higher resolution, as seen in Figures 6.2(b) and 6.2(e). It is difficult to see differences between the volume renderings shown in Figures 6.2(e) and 6.2(f). One can see that Figure 6.2(f) accurately captures the aliasing in the original scan data, as visible in the original data shown in 6.2(c). Table 6.3 provides a summary.

The images provided in Figure 6.3 of the teddy bear demonstrate some of the differences between error-based and level-based hierarchies. The argon bubble and head results were generated using an error-based approach, whereas the teddy bear results were generated using level-based extraction. We can see that the low resolution data set shown in Figure 6.3(a) exhibits good spatial

Data set	Figure	Error threshold	Size (points)	% Size
Argon bubble	(a), (d)	50.00	11,246	0.21 %
	(b), (e)	1.00	215,633	4.11 %
(original)	(c)	n/a	5,242,880	
	(f)	0.92	599,995	11.44 %

Table 6.2: Statistics for the argon bubble visualizations shown in Figure 6.1. The original data set consists of $320 \times 128 \times 128$ = 5,242,880 data points.

representation, resulting in an overall good volume rendered result shown in Figure 6.3(d). Details in the image appear "blobby," which is a common artifact of using multiquadric interpolating functions when using a low number of samples. Higher resolution representations, visualized in Figure 6.3(b), produce significantly better results, seen in Figure 6.3(e). The volume rendered result of the highest resolution in the cluster hierarchy, shown in Figure 6.3(f), is good in comparison to the volume visualization of the original data in Figure 6.3(c). Noise in the blue cloud surrounding the teddy bear are visible in Figure 6.3(f) and not in the original data, attributable to the fact that the maximum error at termination of the hierarchy generation was 8.15. The wood grain in the background is slightly "under-represented" in the highest resolution of the hierarchy. These problems can be resolved by allowing the hierarchy generation to continue for several more iterations until the error reaches a lower threshold. Table 6.4 provides a statistical comparison.

Figure 6.4 more noticeably illustrates the difference between traversing the hierarchy based on error and based on level. Level-based traversal produces a better spatial distribution between resolutions, whereas error-based extraction better resolves high error regions. This fact is exhibited by Figures 6.4(a) and 6.4(d). Table 6.5 summarizes some information about the levels in the hierarchy in Figure 6.4. Low resolutions in error traversal correspond to high error thresholds, whereas low resolutions in level traversal correspond to low level requests. Unlike the argon bubble, head, and teddy bear data sets, the space shuttle data set is the result of a fluid dynamics simulation, using multiple overlapping irregular grids.

It is difficult to analyze the running time of this algorithm. Running time depends strongly on the type of data being processed. Compared with data sets with relatively small variation in scalar value, data sets with large variation in scalar value do not converge as quickly to meet the error threshold. In the CBT generation phase, the primary bottleneck is value approximation at the

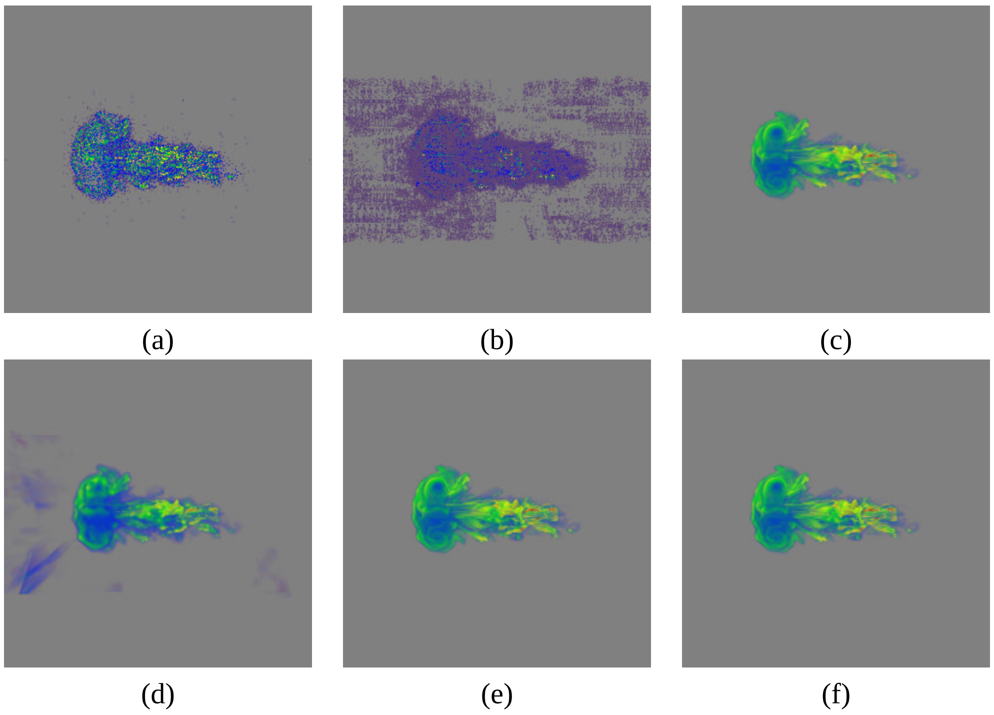


Figure 6.1: CBT and volume visualizations of a single time step of an argon bubble simulation. Figure (a) was extracted via an error-based traversal of the CBT with error threshold 50.00, while (b) used a threshold of 1.00. Figures (d) and (e) are their respective volume visualizations. Image (c) is a volume rendering of the original data. Image (f) is a volume rendering of the maximum resolution represented by the CBT. (This time-varying data set was provided by *The Center for Computational Sciences and Engineering at the Lawrence Berkeley National Laboratory.*)

Data set	Figure	Error threshold	Size (points)	% Size
Head	(a), (d)	60.00	17,533	0.84 %
	(b), (e)	8.00	222,088	10.59 %
(original)	(c)	n/a	2,097,162	
	(f)	4.52	299,980	14.30 %

Table 6.3: Statistics for the head visualizations shown in Figure 6.2 The original data set consists of $128 \times 128 \times 128$ = 2,097,980 data points.

Data set	Figure	Level	Size (points)	% Size
Teddy bear	(a),(d)	16	29,905	2.94 %
	(b),(e)	18	92,845	9.17 %
(original)	(c)	n/a	1,015,808	
	(f)	22	199,991	19.69 %

Table 6.4: Statistics for teddy bear visualizations shown in Figure 6.3. The original data set consists of $62 \times 128 \times 128$ = 1,015,808 data points.

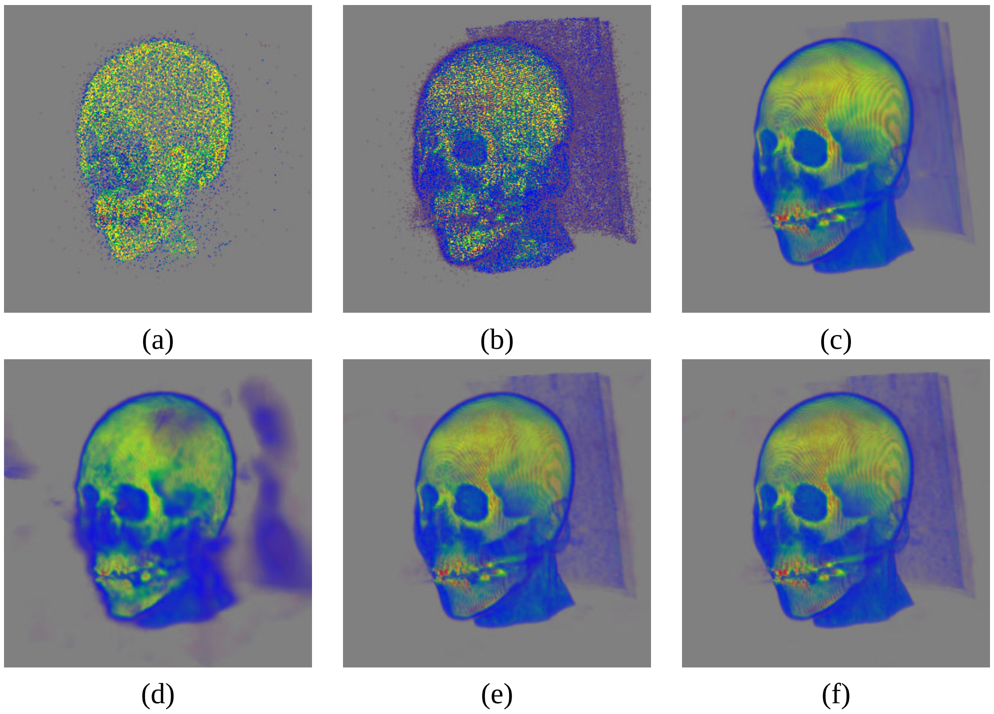


Figure 6.2: CBT and volume visualizations for a head data set. Figure (a) was extracted via error-based traversal of the CBT with error threshold 60.00, while (b) used a threshold of 8.00. Figures (d) and (e) are the respective volume visualizations. Image (c) is a volume rendering of the original data. Image (f) is a volume rendering of the maximum resolution represented by the CBT.

Data set	Figure	Type	Resolution	Size (points)	% Size
Space shuttle	(a)	error	30	4,939	0.59 %
	(b)	error	15	15,246	1.81 %
	(c)	error	8	41,369	4.90 %
	(d)	level	10	948	0.11 %
	(e)	level	12	3,529	0.41 %
	(f)	level	15	22,864	2.71 %

Table 6.5: Space shuttle CBT resolution sizes as shown in Figure 6.4. The original data set consists of 843,542 data points.

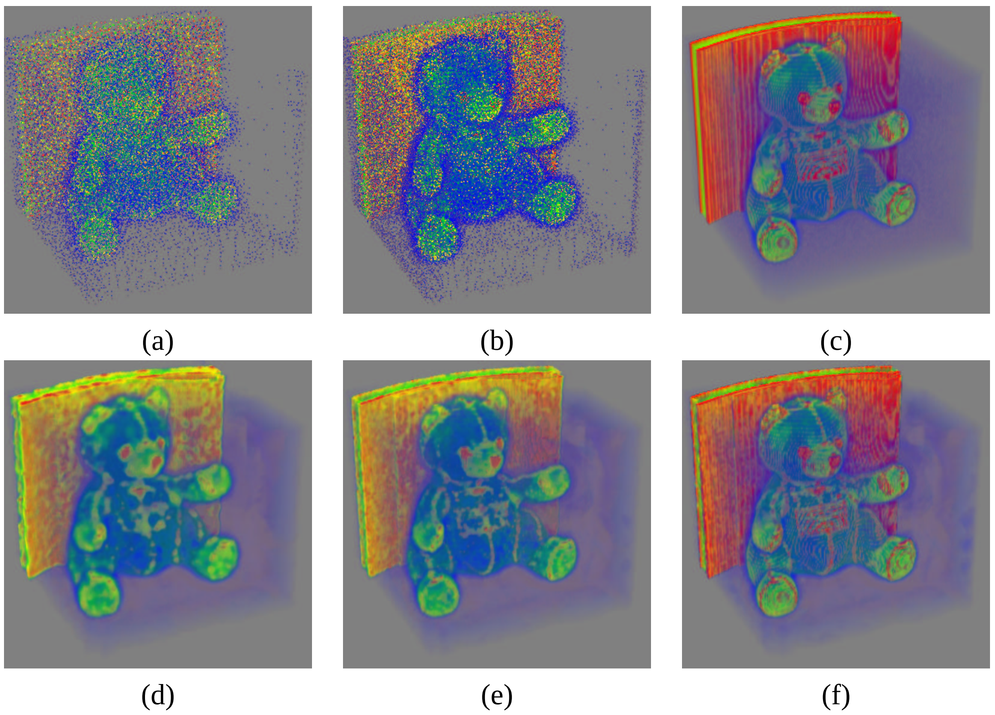
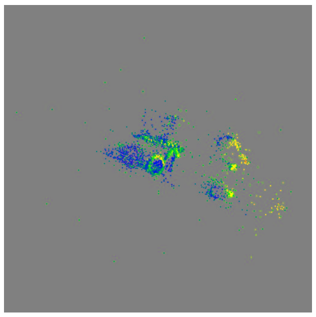
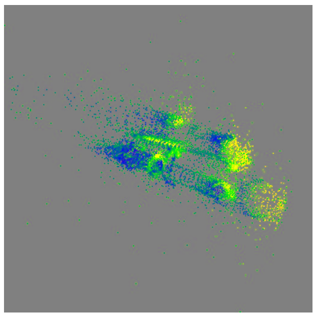
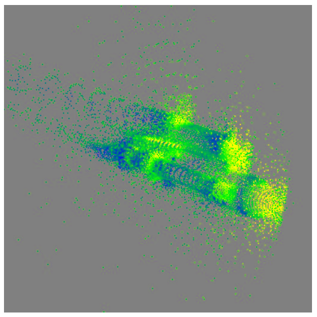
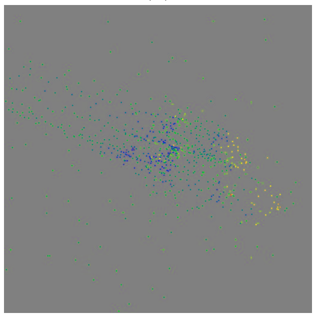


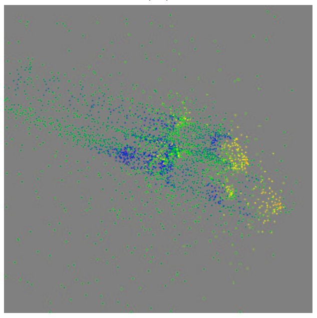
Figure 6.3: CBT and volume visualizations of an MRI scan of a teddy bear. Figure (a) was extracted via level-based traversal of the CBT with maximum depth 16, while (b) used a depth of 18. Figures (d) and (e) are their respective volume visualizations. Image (c) is a volume rendering of the original data. Image (f) is a volume rendering of the maximum resolution represented by the CBT.


(a)


(b)


(c)


(d)


(e)

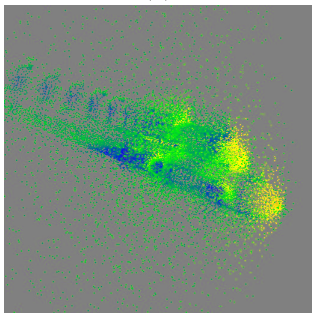

(f)

Figure 6.4: CBT visualizations for a space shuttle data set. Figures (a), (b), and (c) were extracted via error-based traversal of the hierarchy with error thresholds 30.00, 15.00, and 8.00, respectively. Figures (d), (e), and (f) were extracted via level-based traversal of the hierarchy with level requests 10, 12, and 15, respectively. Figures (a) and (d) correspond to low resolutions. Figures (b) and (e) correspond to medium resolutions. Figures (c) and (f) correspond to high resolutions. Information about each resolution is listed in Table 6.5.

cluster centers. When full linear and non-linear optimizations are implemented, then running time is expected to grow exponentially with respect to field size and number of iterations performed. The time required to maintain the error queue and to compute the k nearest neighbors to a given cluster center is also considerable. Let p be the number of iterations executed in the refinement process for n data points. We can expect to obtain the next split candidate from the error queue in constant time at a cost of approximately $O(\, log\, p\,)$ for inserting new clusters into the queue. Nearest neighbor computations can be efficiently computed in logarithmic time with respect to the size of the cluster.

The CBT's space requirement is linear with respect to the number of iterations, since each iteration of the refinement process can add at most two new members to the cluster hierarchy.

The time required to obtain a given resolution of the data from the cluster hierarchy is the time required to traverse the CBT, which is $O(\, 2p + 1\,)$, the time to perform a depth first traversal in the worst case. Extracting lower resolutions from the data hierarchy require less time due to the traversal termination conditions described in Section 5.

Chapter 7

Conclusions

We have presented a method for the creation of a multiresolution hierarchy through iterative refinement of a scalar field. The method uses clustering techniques to construct a data hierarchy This method is quite flexible. The choice of minimum cluster size can play a large role in the spatial distribution of cluster points as well as the size of the data hierarchy. The refinement termination condition can be based on a user-defined upper bound or on error threshold. Additional modes of data extraction could be defined by storing other parameters in the cluster hierarchy, such as gradient, function value range spanned by a cluster or spatial range covered by a cluster's bounding region. This flexibility allows one to customize the hierarchy to conform to application specific needs. To what degree these parameters can be used to enhance visualization is a topic for future work.

Many scalar field simplification methods for scattered data require an initial tessallation of the data points, which can be costly to compute and require significant amounts of memory to store connectivity information. The methods described in this paper do not require explicit connectivity information, though it could conceivably be useful for value approximation.

The multiresolution hierarchy can be constructed for large data sets as well, since cluster hierarchy generation can be distributed across several computers.

This method can be used to explore features in arbitrary data. Because the cluster with the most error is split at each iteration, dense clustering occurs in regions of high scalar value variation. Fewer clusters appear in lower variation regions. By virtue of this characteristic, this method has potential as a feature detection algorithm. Different features could be tracked by applying a transfer function filter to the scalar values prior to hierarchy generation. Different error metrics and extraction routines could further enhance detection of features.

Acknowledgements

This work was supported by the National Science Foundation under contract ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, and through the National Partnership for Advanced Computational Infrastructure (NPACI); the National Institute of Mental Health and the National Science Foundation under contract NIMH 2 P20 MH60975-06A2; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the Lawrence Berkeley National Laboratory. We also acknowledge the support of ALSTOM Schilling Robotics and SGI. We thank the members of the Visualization and Graphics Research Group at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis. We thank Dr. Zhaojun Bai for helping us in the context of linear systems and numerical algorithms.

Bibliography

[1] J.C. Carr, R.K. Beatson, J.B. Cherrie, T.J. Mitchell, W.R. Fright, B.C. McCallum, and T.R. Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings ACM SIGGRAPH 2001*, pages 67–76, Los Angeles, CA, August 2001. ACM SIGGRAPH.

[2] P. Cignoni, D. Constanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with error evaluation. In *Proceedings of the IEEE Visualization 2000 Conference*, pages 85–92. IEEE, 2000.

[3] P. Cignoni, L.D. Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplical complexes. In *Proceedings of the 1994 Symposium on Volume Visualization*, 1994.

[4] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution representation and visualization of volume data. 1997.

[5] R. Franke and H. Hagen. Least squares surface approximation using multiquadrics and parametric domain distortion. *CAGD 16*, pages 177–196, 1999.

[6] L.A. Freitag and R.M. Loy. Adaptive, multiresolution visualization of large data sets using a distributed memory octree. In *Proceedings of SC99: High Performance Networking and Computing*, 1999.

[7] H. Hagen, R. Franke, and G. Nielson. Repeated knots in least squares multiquadric functions. *Computing Suppl. 10*, pages 177–187, 1995.

[8] B. Heckel, A.E. Uva, B. Hamann, and K.I. Joy. Surface reconstruction using adaptive clustering methods. In G. Brunnett, H. Bieri, and G. Farin, editors, *Geometric Modeling, Computing Supplement 14*, pages 199–218. Springer-Verlag, 1999.

[9] B. Heckel, G. Weber, B. Hamann, and K.I. Joy. Construction of vector field hierarchies. In D. Ebert, M. Gross, and B. Hamann, editors, *Proceedings IEEE Visualization '99*, San Francisco, California, October 1999. IEEE.

[10] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, New York, NY, 1986.

[11] S. Lee, G. Wohlberg, and S.Y. Shin. Scattered data interpolation with multilevel b-splines. In *IEEE Transactions on Visualization and Computer Graphics*, volume 3. IEEE, 1997.

[12] J. Meredith and K.-L Ma. Multiresolution view-dependent splat based volume rendering of large irregular data. In *Proceedings of IEEE Symposium on Parallel and Large Data Visualization and Graphics*. IEEE, 2001.

[13] D.M. Mount. Voronoi diagrams on the surface of a polyhedron. Technical report, University of Maryland, 1985.

[14] D.V. Pinskiy, E.S. Brugger, H.R. Childs, and B. Hamann. An octree-based multiresolution approach supporting interactive rendering of very large volume data sets. In H.R. Arabnia, R.F. Erbacher, X. He, C. Knight, B. Kovalerchuk, M.M.-O. Lee, Y. Mun, M. Sarfraz, J. Schwing, and M.H.N. Tabrizi, editors, *Proceedings of The 2001 International Conference on Imaging Science, Systems, and Technology (CISST 2001)*, volume 1, pages 16–22, Athens, Georgia, 2001.

[15] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[16] I.J. Trotts, B. Hamann, and K.I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 3(5):224–237, July-August 1999.

[17] G.H. Weber, O. Kreylos, T.J. Ligocki, J.M. Shalf, H. Hagen, B. Hamann, K.I. Joy, and K.-L Ma. High-quality volume rendering of adaptive mesh refinement data. In T. Ertl, B. Girod, G. Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, Los Alamitos, California, 2001. IEEE, IEEE Computer Society Press.