

USING PROJECTIONS TO VISUALLY CLUSTER HIGH-DIMENSIONAL DATA

The High-Dimensional Eye system proves that a tight integration of advanced clustering algorithms and state-of-the-art visualization techniques can help us better understand and effectively guide the clustering process, and thus significantly improve the clustering results.

The difficult task of sifting through the data generated by the extensive use of computers today—and locating concise and interpretable information within that data—is called knowledge discovery in databases (KDD). *Data mining* refers to one specific step in the KDD process—namely, to the application of algorithms that can extract hidden patterns from data (see the “KDD Process” sidebar for more information).

The data mining technique we focus on in this article is called *clustering*—partitioning a set of data vectors into clusters and noise such that data vectors within the clusters are similar to each other and that the data items in different clusters or noise partitions are not. Recent research has proposed many algorithms for clustering.¹ In all these methods, the user must choose the algorithm and adjust it to application-domain-specific knowledge via parameters. Because these modifications strongly affect the algorithm’s final

result, the process must be repeated multiple times to get useful results with different algorithms and parameters. Visualization techniques could support this step, deepen our understanding of the process, and increase our confidence in the results. Visual data exploration can deal easily with highly inhomogeneous and noisy data, for example, and its intuitive nature requires no understanding of complex algorithms.

The approach we present here combines an advanced clustering algorithm with new visualization methods for a more effective interactive clustering. We start with an efficient clustering algorithm based on a generalized multidimensional grid.² It uses axes-parallel projections as well as complex hyperpolygonal objects as separators in the multidimensional space. Choosing the axes-parallel projections and specifying the separators to be used in building the multidimensional grid, however, are two difficult problems that cannot be fully automated. Our visual clustering system, the High-Dimensional Eye, guides the user through the process of clustering and therefore helps improve the clustering results significantly.

Clustering

The statistics, machine learning, and KDD literature propose many specific clustering algo-

gorithms. *Partitioning algorithms* partition the database into k clusters, which are represented by the gravity of the cluster (k -means) or by one representative data point (k -medoid).³ *Hierarchical clustering algorithms* decompose the database into several levels of partitionings, which are usually represented by a dendrogram—a tree that splits the database recursively into smaller subsets. *Locality-based clustering algorithms* usually group neighboring data elements into clusters based on local conditions.^{4,5}

Most approaches are not designed for clustering of high-dimensional data, so the performance (meaning the efficiency and effectiveness) of existing algorithms degenerates rapidly with increasing dimension. To improve performance, optimized clustering techniques have emerged.⁶⁻⁹ Unfortunately, the curse of dimensionality severely affects the resulting clustering's effectiveness (or quality), especially in the presence of noise. In previous work, we showed that existing clustering methods suffer from either a severe breakdown in efficiency (true for all index-based methods) or have severe effectiveness problems (basically true for all other methods).¹⁰

In image similarity, a simple feature vector is color distribution. In three-dimensional similarity search, a feature vector is the shape of the object described, for example, by its Fourier transformation. The selection of important properties depends strongly on the application. The properties must represent the characteristics of one particular object as well as the diversity of the objects. We can define the clustering problem as one of partitioning a set of data vectors into clusters and noise such that data vectors within the clusters are similar to each other and that the data items in different clusters or noise partitions are not similar. Determining the similarity between two data items, however, is difficult and depends on the task and application. This is even more important because the clustering results strongly depend on the notion of similarity used; exactly how to measure similarity effectively in high-dimensional feature spaces remains an open research question.

Another problem of all clustering algorithms is that one measure of similarity might not be sufficient for the whole data set but could work effectively for a subset of the data—for example, a subset of the data points or of the dimensions.

Projected Clustering

The idea of *projected clustering* has attracted a lot of attention during the past few years. (Infor-

The KDD Process

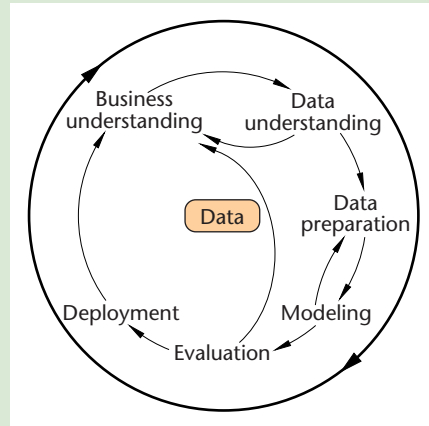
The steps of the knowledge discovery process involve understanding the application domain and the specification of the overall process's goal, acquiring and preprocessing data, selecting and applying data mining algorithms, interpreting the findings, and analyzing success.

The KDD process is a circular flow. After the last step, new questions arise that lead to a new iteration of the process. (A detailed description of the KDD process appears elsewhere.¹) Figure A shows a common model of a data mining project that reflects these steps called the CRISP (Cross-Industry Standard Process for Data Mining), a nonproprietary, documented, and freely available data mining model.

Reference

1. U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to Knowledge Discovery in Databases," *AI Magazine*, vol. 17, no. 3, 1996, pp. 37-54.

Figure A. The current process model for data mining provides an overview of a data mining project's life cycle. It contains the corresponding phases of a project, their respective tasks, and the relationships between these tasks.



mally, projected clustering means that only some promising attributes are used to compute the similarity and thus determine the clusters. In cases with three-dimensional vectors, for example, it might be possible to neglect one dimension, so the remaining two-dimensional subspace is a projection of the original three-dimensional one. HD-Eye uses projected clustering.) The projected clustering problem consists of two main tasks—namely, finding useful projections of the high-dimensional data and determining clusters in the projections. Both tasks depend on each other because a projection is only useful if it provides a good clustering, and a useful projection is needed to determine a projected cluster. Therefore, an algorithm for the projected clustering problem must decide whether a clustering is good for a given projection, and it must search the space of projections based on that criterion.

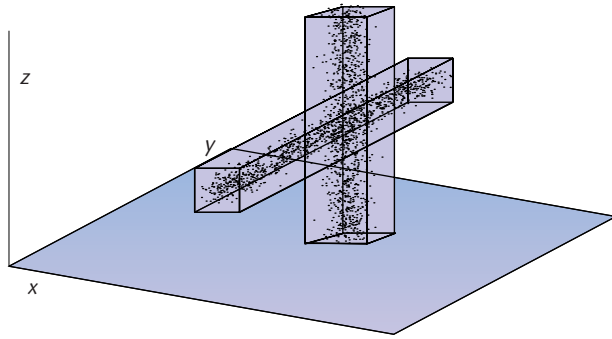


Figure 1. A three-dimensional data set with two projected clusters defined in two dimensions. The clusters are bounded in two dimensions but have a large extent in the third dimension.

One of the first algorithms to deal with projected clustering—Clique¹¹—mines the projection space from the bottom up by searching quantitative frequent item sets (histogram bins). Proclus¹² and Orclus¹³ are k -means-like algorithms that need the number of clusters and the associated projections' average dimensionality as parameters. The most recent method, Doc,¹⁴ defines a projected cluster as a hyperbox—the maximum bounding rectangle of all data points belonging to the cluster—with a boundary size of w (one of the algorithm's parameters) in the bounded dimensions and an unbounded size in the others. Doc uses sampling to center the boxes around some randomly chosen data points. The result is a set of hyperboxes that are bound in some dimensions and contain the projected clusters.

Visualization and Projected Clustering

Visualization technology can help users apply clustering algorithms. Several visualization techniques are useful in data mining, including geometric projection techniques such as projection matrices and parallel coordinates, icon-based techniques, hierarchical techniques, graph-based techniques, pixel-oriented techniques, and combinations thereof. Examples of well-known visual data exploration systems include XmDv,¹⁵ XGobi,¹⁶ and Parallel Visual Explorer.¹⁷ Most visualization techniques are used in conjunction with interaction and distortion techniques.¹⁸ Existing techniques, however, do not allow effective support of the projection-finding process, which is needed for efficient clustering in high-dimensional space.

In previous work, we investigated the link between clustering in projections and visualization

and developed a first prototype of HD-Eye.¹⁹ In the prototype, we combined automated techniques for finding clusters in one-dimensional projections with appropriate visualizations that let the user correct the automatically found results. In the new version, described in this article, we add functionality to better handle two-dimensional projections and provide additional features for combining automatic and interactive clustering. In addition, we tightly integrate HD-Eye with a database system.

The basic idea of projected clustering is to define the similarity among a cluster's data points as distance in some projection of the high-dimensional space. Figure 1 shows an example data set containing two projected clusters that are defined in dimensions $x \times y$ and $x \times z$, respectively. Projected clustering of high-dimensional data is difficult because we have to find the data points belonging to a cluster as well as a useful projection for defining the clustering.

To explain our approach of projected clustering, we must first define the data space, data set, and density functions.

Definition 1: data space and data set. The d -dimensional data space $F = F_1 \times \dots \times F_d$ is defined by d bounded intervals $F_i \subset \mathbb{R}$, $1 \leq i \leq d$. The data set $D = \{x_1, \dots, x_n\} \subset F \subset \mathbb{R}^d$ consists of n d -dimensional data points $x_i \in F$, $1 \leq i \leq n$.

The *probability density function* is a fundamental concept in statistics. In the multivariate case, we have random variables from which we can observe data points in F . The density function f gives a natural description of the distribution in F and allows probabilities to be found from the relation

$$P(x \in \text{Reg}) = \int_{x \in \text{Reg}} f(x) dx \quad (1)$$

for all regions $\text{Reg} \subset F$.

We can use the density function as a basis on which to build clusters from observed data points. *Kernel density estimation* provides a powerful and effective method for estimating the unknown density function f in a nonparametric way from a sample of data points.

Definition 2: kernel density estimation. Let $D \subset F \subset \mathbb{R}^d$ be a data set, h be the smoothness level, and $|\cdot|$ an appropriate metric with $x, y \in F$, $\text{dist}(x, y) = |x - y|$. Then, the kernel density function \hat{f}^D based on the kernel density estimator K is defined as

$$\hat{f}^D(x) = \frac{1}{nb} \sum_{i=1}^n K\left(\frac{x-x_i}{b}\right), 0 \leq \hat{f}^D. \quad (2)$$

The statistics literature has proposed various kernels K , including square-wave and Gaussian functions. Figure 2 shows an example for the density function of a two-dimensional data set using a square-wave and Gaussian kernel. A detailed introduction to kernel density estimation appears elsewhere.^{6,20,21}

We can define clusters as sets of data points that are density-attracted to a local maximum of the density function.²² The definition of projected clustering is similar but considers a projection of the data space instead of the full-dimensional space. Each cluster can be defined in different projections, so determining all the projected density functions to find the clusters is computationally infeasible.

Practical approaches based on an iterative partitioning of the data try to partition the data space without partitioning any of the clusters. The most efficient way to partition a data set is to use low-dimensional projections to split the data. If we use just one-dimensional projections, the resulting clusters are hyperboxes. If we use two-dimensional projections, we can separate arbitrarily shaped regions of the data. In both cases, the idea is to split the data set only in the considered projection to help us find projected clusters. A difficulty is that the set of dimensions needed to separate a cluster will likely differ for each cluster. Furthermore, two clusters could overlap in some dimensions.

How can we characterize the projected clusters we find with HD-Eye? A hyperbox can describe the cluster, but a better description is the sequence of split operations needed to separate them. Starting with the whole data set, in each iteration, the user selects one or more one- or two-dimensional projections that indicate distinguishable regions. These regions are separated automatically or by hand and each is processed recursively. The definition of a projected cluster reinforces that not all dimensions are required to define the clusters and lets the clusters have arbitrary shapes.

HD-Eye

HD-Eye combines visualization techniques with an advanced algorithm to perform projected clustering. Here, we'll describe the concepts of density-based single-linkage separators and the separator tree. The concept of separators is needed to find well distinguishable regions, and the con-

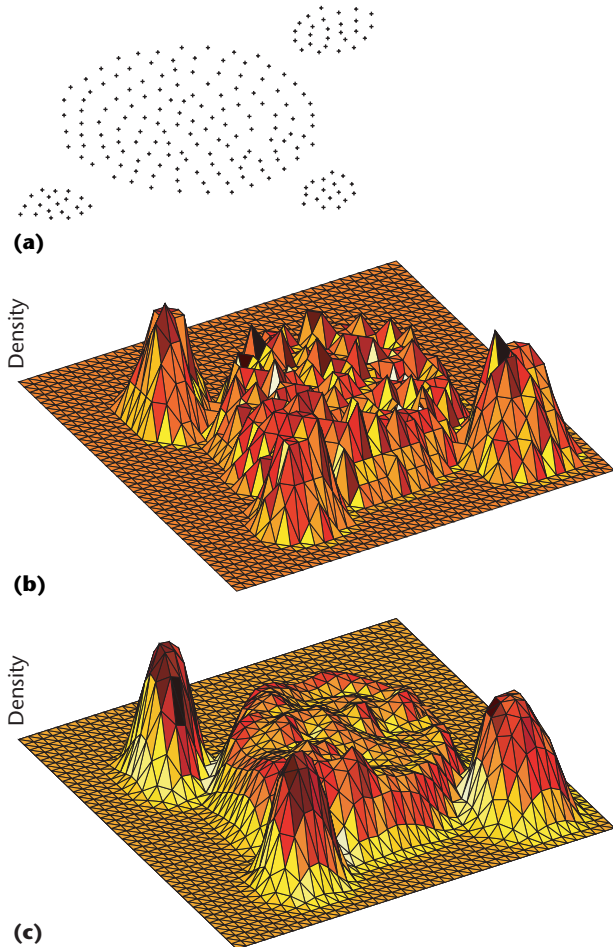


Figure 2. Density functions of (a) a two-dimensional data set using (b) a square-wave kernel and (c) a Gaussian kernel. The subfigures (b) and (c) show the density function corresponding to different kernel function K —namely, the square-wave kernel and the Gaussian kernel. In general, the choice of a Gaussian kernel results in a smoother density function.

cept of the separator tree helps keep track of the iterative and recursive nature of our approach. We'll also present an efficient algorithm for determining such separators, before finally introducing some visualization techniques to support the process of finding the projected clusters.

The two main tasks of finding the appropriate projections for partitioning the data and finding and specifying good separators based on these projections need visual support. Both tasks require the human user's intuition and creativity and cannot be done automatically.

Separators and Separation Tree

The HD-Eye framework uses a concept for clus-

tering that is similar to the one decision trees use for classification. A decision tree comprises several nodes, each containing a decision rule that splits the data to achieve purer label sets in the child nodes. In our case, we do not use class labels to find a split—rather, we use a density function. The equivalent to the decision rule is the *separator*, which partitions and labels the data space's points according to a clustering scheme.

Definition 3: separator. A separator is a point-labeling function $S: F \rightarrow \{0, \dots, Split(S) - 1\}$ that uses a density estimator to assign each point in the continuous data space F a label (or integer) to distinguish different clusters or groups of clusters. The number of separated regions is $Split(S) > 0$.

Given a set of points $A \subset R^d$, we denote the subset of points $\{x \in A: S(x) = i\}$ as $S^i(A)$; there might exist empty regions $0 \leq i < Split(S)$, $S^i(A) = \emptyset$. To automatically separate regions in a one- or two-dimensional projection, we use a separator as introduced in definition 3. The density estimator used in the separator definition is defined in the one- or two-dimensional subspace of the particular projection.

Besides simple partitioning hyperplanes, we mainly use two-dimensional projections for a specific type of separators: the *density-based single-linkage separator* defines clusters as regions of maximal size in the data space, while the density all over the cluster is larger than a minimum threshold.²³ According to this scheme, a cluster is arbitrarily shaped, so it cannot be represented by a single centroid and the nearest-neighbor rule (like a simple k -means clustering). Clusters found with the density-based single-linkage separator are useful for approximating complex correlations.

Based on ideas from other research, our informal definition for a density-based single-linkage clustering makes no assumption about the density estimation function used.^{9,22}

Definition 4: density-based single-linkage clustering. Given a set of objects described by the set of feature vectors $D = \{x_1, \dots, x_n\}$, the partition $C = \{C_1, \dots, C_l\}$ of D is a density-based single-linkage clustering for a given density function $\hat{f}(\cdot)$ and a noise level ξ , if and only if the following properties are satisfied for all C_i , $i = 1, \dots, l$: the cluster C_i is nonempty; the density on a path that connects two points in C_i and is completely contained in C_i does not fall below the threshold ξ , and the cluster C_i has maximal size. Points $x \in D$ with $\hat{f}(x) < \xi$ are called *outlier*.

The intuition behind this definition is that a cluster is a connected region R in the continuous data space F , the density at all points of R is above the noise threshold ξ , and the clusters are isolated by low-density valleys. For our separator approach, we need an algorithm that can find the valley with the lowest density separating at least two clusters or groups of clusters.

Because we want to find separators in many (two-dimensional) projections of the data space, we need a quality measure that compares different separators. The quality measure for the density-based single-linkage scheme does not have a straightforward definition. Again, though, low-density valleys separate clusters.

We define the separation quality q_{sep} depending on the maximum density at the borders of the cluster regions in the continuous data space F . The point at a cluster's border with the maximum density is the point on which the cluster is hard to distinguish from neighboring regions. The set $Border(S) \subseteq F$ of cluster border points is determined by a separator S . Intuitively, it is defined as the set of points $x \in F$ with the property that each arbitrary small neighborhood around x contains two points with different cluster labels. This definition is valid because a separator labels the data points as well as all the continuous data space's points. The separation quality is defined as

$$q_{sep}(S) = 1 - MAX\{\hat{f}^D(x) | x \in Border(S)\}. \quad (3)$$

Figure 3 shows an example data set's separator quality. The color shows the density on the border between the two clusters; the separator quality corresponds to the inverse of the border's maximum density.

Because a single separator does not necessarily separate all of a data set's clusters at the same time, we need a more global structure to integrate multiple separators defined in different projections. One way to accomplish this is with separator trees. Similar to a decision tree, which is an assemblage of classification rules forming a classifier, a separator tree is a collection of separators forming a cluster model for the given data with regard to the user's intention.

Definition 5: separator tree. A separator tree T is a tree that corresponds to a recursive partitioning of a data set D . A node v of T corresponds to a cluster region $R_v \subset F$. Except the leaves, each node v has an assigned separator S_v that splits the

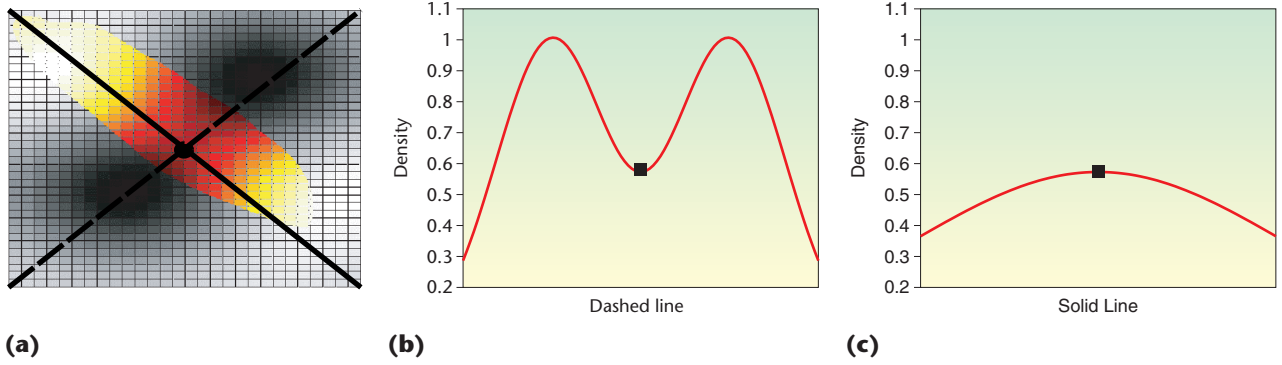


Figure 3. The quality of the density-based single-linkage separator is determined as the maximum density at the border between the clusters (solid line): (a) the overall picture with two clusters and the color-coded density and the density at (b) the dashed line and (c) the solid line. The separation quality in the example is 0.57.

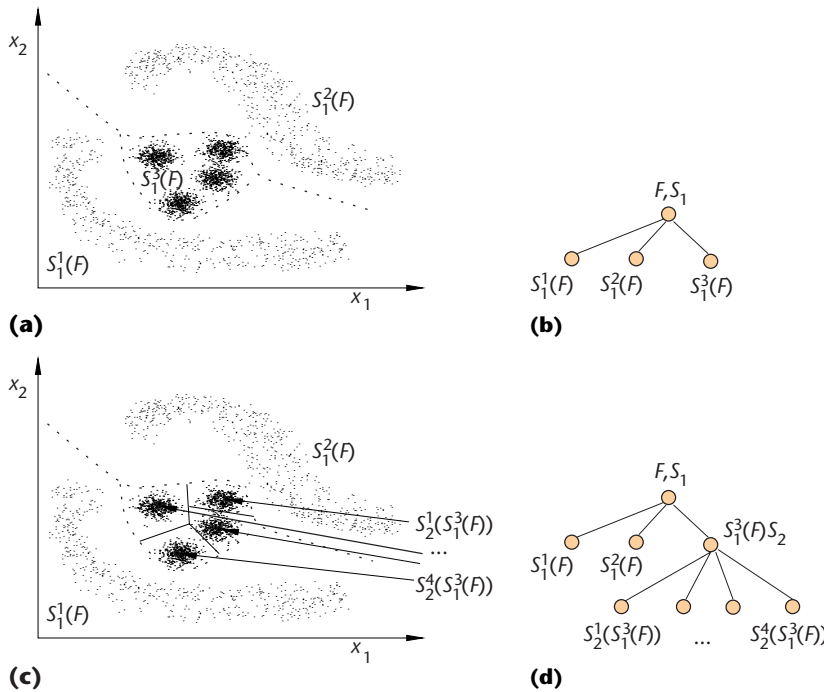


Figure 4. Growing the separator tree using a separator. (a) The separator S_1 partitions the data set F into three subsets—namely, $S_1^1(F)$, $S_1^2(F)$, and $S_1^3(F)$ —and its (b) corresponding separator tree. The dotted line represents the border between those three regions. (c) The separator S_2 splits $S_1^3(F)$ into four regions and its (d) corresponding separator tree.

corresponding region R_v . The i th son of v corresponds to the region $S_v^i(R_v)$. The root node's region is the whole data space F .

The split operation grows the tree: it takes a leaf node l of a separator tree T , assigns a separator S to l , and thus defines the new son nodes of l (see Figure 4). The region R of l is decomposed into subregions, each containing different clusters or groups of clusters. Because the separators are not defined in the same dimensions, we can integrate complex separators from different projections into a single clustering model.

Algorithm

We found an efficient algorithm that finds approximated density-based single-linkage separators in two-dimensional projections. By definition, the original single-linkage problem has quadratic runtime complexity in the number of data points—each data point must be compared with every other to see if they can be linked (or whether the density between them is high enough). For large databases, this is prohibitively expensive.

To make the algorithm scalable, we first reduce the set of data points $D = \{x_1, \dots, x_n\}$ to a set of centroids $P = \{p_1, \dots, p_k\}$. We can do this with a variant of the k -means algorithm, which

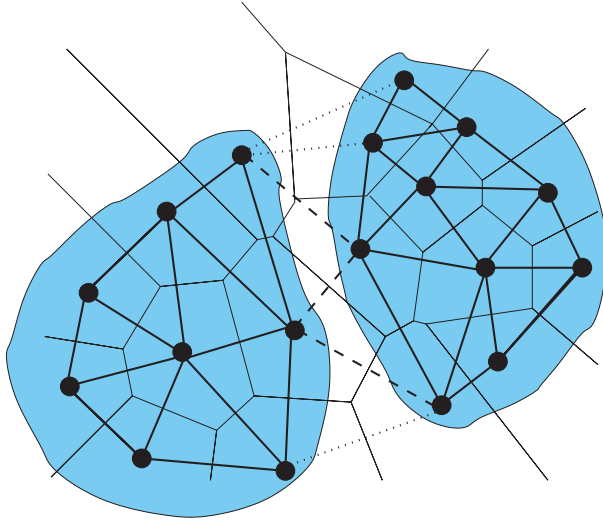


Figure 5. A Voronoi diagram (thin solid lines) with an underlying data distribution (shaded areas) and the Delaunay graph (thick solid and dashed lines). The thick dotted edges are the remaining edges of the Delaunay graph, which the separator algorithm does not delete. The dashed edges have a low density and are deleted by the density-based single-linkage separator.

uses a histogram as an approximation of the data set. The reduction decreases the number of objects from n to k with $k \ll n$.

After reducing the data set, the second task is to determine the connected high-density regions based on the data set's centroid representation. For two-dimensional projections, we use the weighted Delaunay graph,²³ which is called a cluster graph:

$$G = (V, E, w) \text{ with } V = \{p_i : 1 \leq i \leq k\}, \quad (4)$$

where E is defined according to the Delaunay triangulation, and

$$\forall e = (p, p') \in E, \quad w(e) = \min \left\{ \hat{f}(p + t \cdot (p' - p)) : t \in [0, 1] \right\}. \quad (5)$$

Because the density function is not available in analytical form, the weight determination implementation works with a discretization of the line between the centroids and estimates the density on $r \geq 1$, $r \in \mathbb{N}$ points at the line between p and p' .

Centroids belonging to the same high-density region are at least transitively linked in the cluster graph with edges of large weights. To separate such regions, the algorithm deletes the edges in ascending order according to their weights until the graph splits into two connected compo-

nents. The threshold for the noise level ξ is set to the weight of the last deleted edge. We assume that all nodes with a lower density than ξ approximate noise points and collect them in a special prototype subset P_0 . The following algorithm describes the method in pseudo code. (Figure 5 shows an example of the algorithm's result.²⁴)

db_slink_separator ($G(V = P, E, w), \hat{f}$)

Require: $G(V = P, E, w)$ initial cluster graph

Ensure: P_1, P_2, \dots contains the centroids in connected high-density regions isolated by density valleys with maximal density ξ (separation quality), P_0 contains prototypes approximating noise

- 1: $\xi \leftarrow 0, P_0 \leftarrow \emptyset$
- 2: **while** G is connected **do**
- 3: determine e with $w(e) = \min\{w(e'), e' \in G.E\}$
- 4: $\xi \leftarrow w(e)$
- 5: $G.delete_edge(e)$
- 6: $P_0 \leftarrow P_0 \cup \{p \in G.V, \hat{f}(p) < \xi\}$
- 7: delete all nodes p from G with $\hat{f}(p) < \xi$
- 8: **end while**
- 9: $P_1, P_2, \dots \leftarrow \text{Determine_Connected_Components}(G)$
- 10: **return** $(P_1, P_2, \dots, P_0, \xi)$

The separator algorithm's output is a partition of P into two subsets of centroids (which approximates connected clusters or cluster groups) and possibly a subset of centroids (which approximates outliers P_0). We can label a data point x (or assign it to a cluster) by looking for the nearest centroid $p_{I(x)} \in P$ and determining the index i of the subset of centroids with $p_{I(x)} \in P_i$. (Note that $I(x)$ determines the index of the nearest centroid to x .) Therefore, the Voronoi edges that correspond to the deleted edges in the induced Delaunay graph approximate the separating density valley.

We can use a recursive variant of the separator algorithm to approximate the single-linkage hierarchy. In this case, the subsets of centroids as well as the cluster graph's corresponding subgraphs are inputs of recursive calls of the separator algorithm. The following algorithm approximates the hierarchy; the intermediate cluster descriptions C can be stored in a tree to form the hierarchy:

db_slink_hierarchy ($G(V = P, E, w), \hat{f}$)

Require: $G(V = P, E, w)$ cluster graph

- 1: **if** $\#(P) = 1$ **then**
- 2: **return**
- 3: **end if**
- 4: $(P_1, P_2, \dots, P_0, \xi) \leftarrow \text{db_slink_separator}(G(V = P, E, w), \hat{f})$ {We assume G

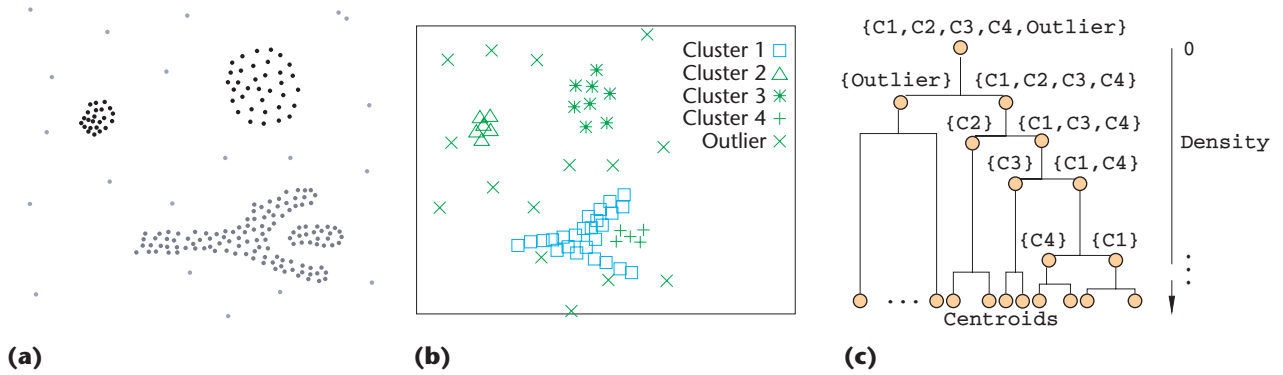


Figure 6. The hierarchical splitting of (a) a data set with multiple arbitrarily shaped clusters: (b) the result of the recursive splittings using the density-based single-linkage separator and (c) the upper part of the single-linkage hierarchy.

- contains only nodes with $\hat{f}(p) \geq \xi$ and edges with $w(e) \geq \xi$ after the call.)
- 5: $C \leftarrow \{P_1, P_2, \dots\}$ {Note that P_0 does not belong to the actual clustering C .}
 - 6: **for all** $P_i \in C$ **do**
 - 7: $V_{sub} \leftarrow P_i, E_{sub} \leftarrow \{e(p, p') : p, p' \in P \text{ and } e \in G, E\}$
 - 8: **db_slink_hierarchy**($G_{sub}(V_{sub}, E_{sub}, w), \hat{f}$)
 - 9: **end for**

Figure 6b shows the approximated clustering determined by `db_slink_hierarchy`; Figure 6c shows the upper part of the corresponding hierarchy. Using the density-based single-linkage separator recursively until all clusters are separated can interactively integrate the relevant parts of the hierarchy. The separation quality of the clusters on each level is reflected by the noise level ξ (or splitting density), which increases for each recursive splitting. Higher splitting density indicates low separation quality. We can use the separator tree to exclude branches of the hierarchy and encourage further separation. In this way, we can handle arbitrarily shaped clusters of different densities.

Visualizations

So how does HD-Eye visually support the process of finding projected clusters? Figure 7 shows an overview screenshot of the HD-Eye system. The middle part of the figure shows an overview of the one-dimensional projection's density distribution. Each arc shows a one-dimensional histogram in which the density is mapped to a color (dark means high density). The right part (with the green squares) shows two-dimensional density distributions. The algorithm determines a density-based single-linkage separator for the selected projection and shows it as a graph in the detailed plot.

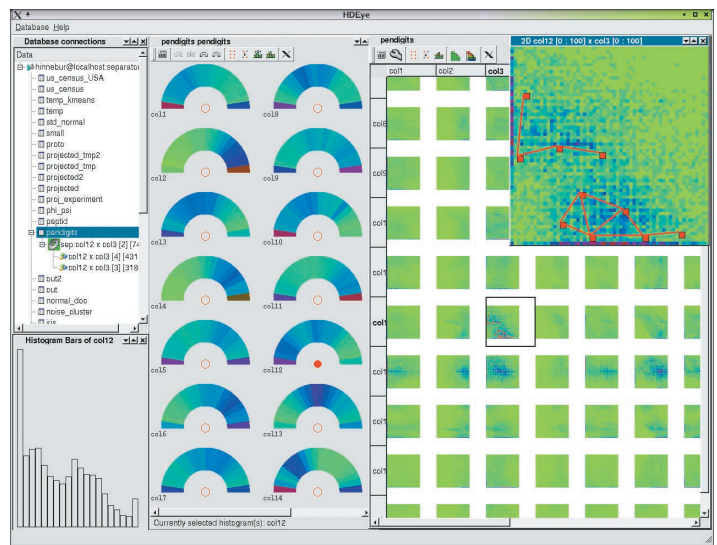


Figure 7. An initial snapshot of HD-Eye with different visualization techniques. For the selected dimension (the arc with the filled red dot), the detailed distribution is shown as bar chart in the lower left corner. Because the distribution shows multiple peaks, the user might be interested in that dimension. The right part (with the green squares) shows two-dimensional density distributions. The selected one is the projection 12×3 , which is also shown in detail in the right upper corner.

To find splits in the one-dimensional projections, the user can determine meaningful values for the noise level ξ . In Figure 8a, ξ is set to zero, and in Figure 8b, it's set to 6 percent to help the automated split algorithm find good splits.¹⁹ The part of the histogram below the noise level appears in yellow. The rule of thumb is to select the noise level as low as possible (to keep many points in clusters), but high enough to get a good visualization where regions can be visually separated.

One important aspect, particularly for two-

Figure 8. One-dimensional projections with different noise levels: (a) ξ = zero and (b) ξ = six percent. This increase removes regions with low density, helping the user find good separators more easily.

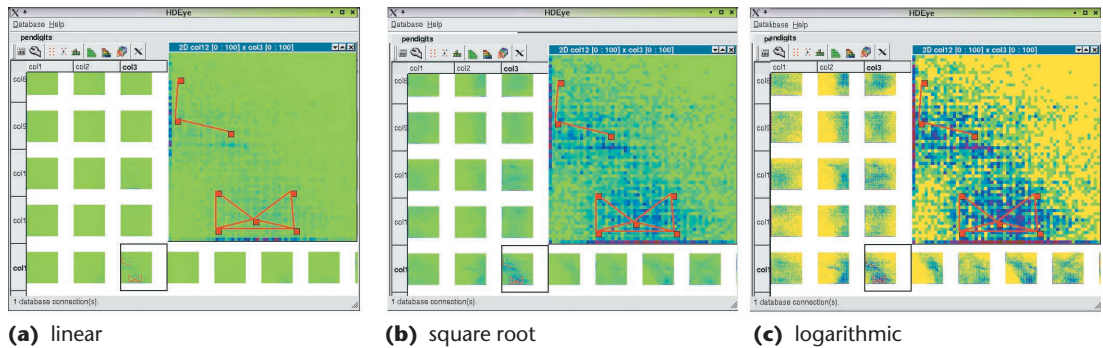
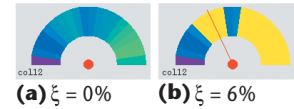


Figure 9. The pendigits data set with different color mappings: (a) linear, (b) square root, and (c) logarithmic. The color mapping choice strongly effects the visual perception; unfortunately, the color cannot be chosen in advance. The user could select different mappings to find the one that gives the best contrast.

dimensional projections, is color mapping. A linear mapping of density to color yields poor results (as Figure 9a shows). To improve the contrast, we need a nonlinear mapping. Figures 9b and 9c show the result of a square-root and logarithmic color mapping, respectively, which means that the square root and the logarithm of the density are mapped to color. The clusters are much easier to perceive in the nonlinear mappings than in the linear mapping.

Putting HD-Eye to the Test

To evaluate HD-Eye, we used the *ecoli* and *pendigits* data sets from the University of California at Irvine’s Machine Learning Repository (see www.ics.uci.edu/~mllearn/MLRepository.html). To show how to apply HD-Eye, we use the *ecoli* data set, which records information about *E. coli* proteins. The data set is challenging because of the overlap between clusters and varying cluster sizes: five clusters have 143, 77, 52, 35, and 20 data points, respectively, and the remaining three clusters consist of only nine points. The data set is labeled, meaning we can verify our visual clustering’s validity.

At the beginning, the user sees the circular histograms of all dimensions and a matrix of all two-dimensional density plots for the whole data set (see Figure 10). Dense regions are colored blue; sparse regions appear in green.

Looking at the circular histograms, the user might notice that dimension *alm2* has a bimodal distribution, which window 1 in Figure 11 shows. To achieve a good partitioning, the user can split the data between the two bumps by placing a separator (the red line in window 1) between them. The system automatically splits the data at the separator and computes the histograms for the two partitions, which windows 2 and 3 show, respectively. Window 4 shows the new separation tree. You can see that the distributions of dimensions *acc* and *alm1* in windows 2 and 3 (corresponding to the two partitions of the data set) differ from each other and from the whole data set’s

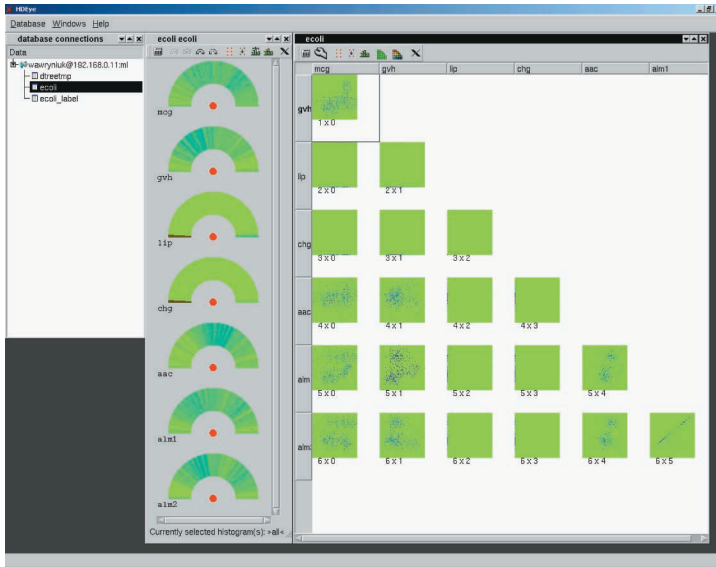


Figure 10. An initial snapshot of HD-Eye. One window contains circular histograms, and the others contain all two-dimensional density plots. The visual information corresponds to the whole data set.

distributions. This shows that the split operation was successful, because the user can see that the two subsets in windows 2 and 3 differ significantly. Using the known labeling of the data set, we verified the split operation's importance and determined that one partition contains two clusters and the other contains three.

As we mentioned earlier, one-dimensional projections are not sufficient for clustering data. To show this, let's demonstrate the use of two-dimensional projections. Looking at the matrix in Figure 12, the user sees that one density plot (marked with a 1) reveals two clusters and can split the data. To determine the separator, we use the density-based single-linkage separator. Figure 13 shows the result of the split; windows 3 and 4 show the corresponding matrix of density plots for each partition. You can see that the two partitions the split generated uncover a completely different distribution in some of the two-dimensional projections, which is best seen in the projection $alm2 \times mcg$. Window 1 shows the density plot of the projection $alm2 \times mcg$ for one partition, and window 2 shows it for the other. One partition has a higher density in the upper region, whereas the other partition has a higher density in the bottom one. Using the available labeling, we verified this observation and confirmed that the partitions represent different clusters of the data set.

The general strategy of working with HD-Eye is as follows. First, the user scrutinizes the histograms and two-dimensional density plots to find interesting properties such as bimodal distributions or noise regions. After identifying them, he or she separates the data either with a one-dimensional or two-dimensional separator and repeats the process with the obtained partitions recursively as long as he or she detects interesting distributions.

When applying this strategy, we find all five major clusters in the *ecoli* data set with a high-quality clustering (classification rate of 84 percent). The best results achieved with automated classification algorithms have a classification rate of 81 percent. This shows that interactive clustering using visual and automated algorithms can provide better results than automated algorithms can achieve alone. In addition to the high quality, HD-Eye gives the user a visual impression of the data and more confidence in the results.

We used the *ecoli* data set because of its simplicity to show how HD-Eye works. Let's look at how the system handles lots of dimensions and data points—namely, the *pendigits* data set, which contains 7,494 tuples and 16 dimensions that describe handwritten digits. The *pendigit* data set

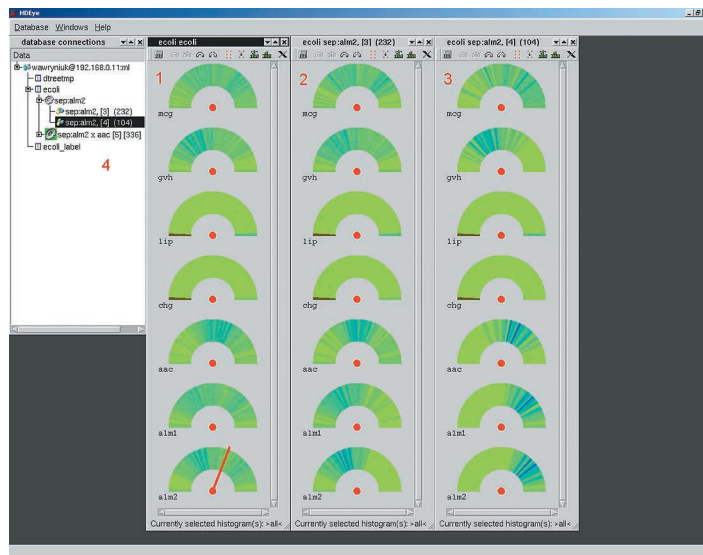


Figure 11. The effect of a one-dimensional split operation. The applied separator appears as a red line in window 1. The operation splits the whole data set into two partitions; Windows 2 and 3 show the corresponding histograms of the two partitions.

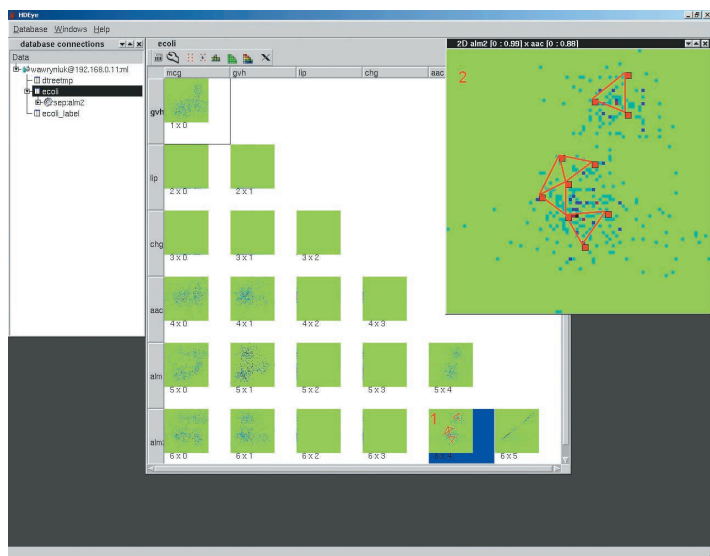


Figure 12. How a two-dimensional split operation works. The density plot marked with 1 shows two clusters, and the window marked with 2 is an enlargement. The enlargement shows the induced Delaunay graph, which assigns the data points to the two clusters.

was created by collecting samples of handwritten digits from different writers. To represent digits as constant length feature vectors, a resampling algorithm applying a linear interpolation between pairs of points is used. Applying our strategy, we discovered 19 clusters in the data. This is surprising, because we expected only 10 clusters, corresponding to the 10 digits in the data (see Table 1

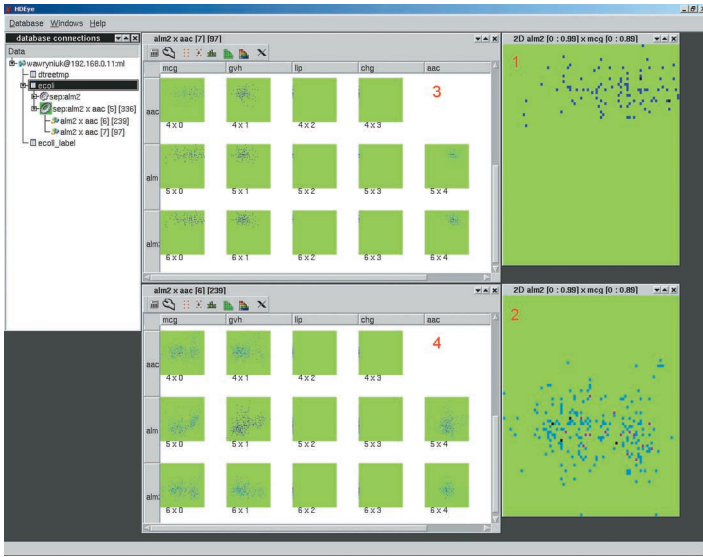


Figure 13. The effect of a two-dimensional split operation. Windows 3 and 4 show density plots for the partitions resulting from a split operation. Windows 1 and 2 emphasize that the two subsets have significantly different characteristics.

for the clustering’s confusion matrix). Identifying the data points corresponding to digits 2, 3, 4, 6, and 7 is easy in the sense that one node in the separator tree lets us cut away almost all the data points labeled with the given digit and only the points with the given label. The data points cor-

responding to the remaining digits require more cuts. Different nodes in the separation tree let us cut away a certain small amount of data points labeled with a given digit.

How can we rate the clustering’s quality? For each cluster, we determine the digit with the highest support (the bold numbers in Table 1) and state that the particular cluster forms a sub-cluster of all data points labeled with that digit. Adding all the bold numbers in Table 1, we get the number of data points assigned to the correct clusters. In this experiment, we assigned 80 percent of the data points to the correct cluster, which is a good result for the given data set.


The HD-Eye system combines an advanced algorithm with visualization techniques that can support the clustering process via a visual representation of the important information. The visualization technique uses a pixel-oriented view that lets the user place cluster separators directly or automatically in visualizations. Experimental evaluation shows that the combination of automatic and visual techniques significantly improves the effectiveness of the data mining process and provides a better understanding of the results. We used our algorithm on labeled data. Each data item belongs

Table 1. The confusion matrix for our clustering of the pendigits data set. Columns correspond to real clusters (digits 0 to 9) and rows correspond to the found clusters (cluster 1 to 19).

	0	1	2	3	4	5	6	7	8	9	10
1	13				698		3			3	717
2					3	3				180	186
3	518					8			48		574
4	3	22	8		37	12	683	1	2	17	785
5			1	1	1	3				110	116
6		3		1	2	89				4	99
7	29					3	7		179		218
8		38		102		113		17	4	169	443
9		10	3	5	29			48	1	65	161
10		365		1				2	13	63	444
11		32		603		15	11	85	60	80	886
12		16				17			2	12	47
13		181	743	3		7	8	18		1	961
14	1	112		3	5			35			156
15			13			1	7	567	103	8	699
16						62			4		66
17			12			385	1	4	33		435
18	3							1	263		267
19	213				5	2			7	7	234
Σ	780	779	780	719	780	720	720	778	719	719	7494

to a different class, which the label annotates. Our algorithm classified 84 percent with the right label in contrast to older results with 81 percent.

The combination of automated and visual techniques, which HD-Eye demonstrates, will have a noticeable effect on clustering high-dimensional data in the context of data mining. Our plans for future work include applying and fine-tuning our method for specific applications as well as an extension to include other visual representations.

The HD-Eye software is available from <http://hdeye.sourceforge.net>. A further extension is the possibility of merging different results from multiple users analyzing the same or slightly different data. This can be a way to combine the view of two experts working on the same data. 

References

- D.A. Keim and A. Hinneburg, "Clustering Techniques for Large Data Sets: From the Past to the Future," *Proc. Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 1999, pp. 141–181.
- A. Hinneburg and D.A. Keim, "Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering," *Proc. 25th Int'l Conf. Very Large Databases*, Morgan Kaufmann, 1999, pp. 506–517.
- R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. 20th Int'l Conf. Very Large Databases*, Morgan Kaufmann, 1994, pp. 144–155.
- M. Ester et al., "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. 2nd Int'l Conf. Knowledge Discovery and Data Mining*, AAAI Press, 1996, pp. 226–231.
- X. Xu et al., "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases," *Proc. 14th Int'l Conf. Data Eng.*, IEEE CS Press, 1998, pp. 324–331.
- E. Schikuta, "Grid-Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets," *Proc. 13th Int'l Conf. Pattern Recognition*, IEEE CS Press, 1996, pp. 101–105.
- T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1996, pp. 103–114.
- W. Wang, J. Yang, and R.R. Muntz, "Sting: A Statistical Information Grid Approach to Spatial Data Mining," *Proc. 23rd Int'l Conf. Very Large Databases*, Morgan Kaufmann, 1997, pp. 186–195.
- A. Hinneburg and D.A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," *Proc. 4th Int'l Conf. Knowledge Discovery and Data Mining*, AAAI Press, 1998, pp. 58–65.
- A. Hinneburg and D.A. Keim, "Clustering Methods for Large Databases: From the Past to the Future," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1999, p. 509.
- R. Aggarwal et al., "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1998, pp. 94–105.
- C.C. Aggarwal et al., "Fast Algorithms for Projected Clustering," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 1999, pp. 61–72.
- C.C. Aggarwal and P.S. Yu, "Finding Generalized Projected Clusters in High Dimensional Spaces," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 2000, pp. 70–81.
- C.M. Procopiuc et al., "A Monte Carlo Algorithm for Fast Projective Clustering," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 2002, pp. 418–427.
- M.O. Ward, "XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data," *Proc. Visualization '94*, IEEE CS Press, 1994, pp. 326–336.
- A. Buja, D.F. Swayne, and D. Cook, "Interactive High-Dimensional Data Visualization," *J. Computational and Graphical Statistics*, vol. 5, no. 1, 1996, pp. 78–99.
- A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry," *Proc. Visualization '90*, IEEE CS Press, 1990, pp. 361–370.
- D. Keim and M. Ward, "Visual Data Mining Techniques," to be published in *Intelligent Data Analysis: An Introduction*, M. Bertold and D. Hand, eds., Springer, 2003.
- A. Hinneburg, M. Wawryniuk, and D. A. Keim, "HD-Eye: Visual Mining of High-Dimensional Data," *IEEE Computer Graphics & Applications*, vol. 19, no. 5, 1999, pp. 22–31.
- B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman & Hall, 1986.
- M.P. Wand and M.C. Jones, *Kernel Smoothing*, Chapman & Hall, 1995.
- A. Hinneburg and D. Keim, "A General Approach to Clustering in Large Databases with Noise," to be published in *J. Knowledge and Information Systems*, Springer, 2002.
- F. Aurenhammer, "Voronoi Diagrams: A Survey of a Fundamental Geometric Data Structure," *ACM Computing Surveys*, vol. 23, no. 3, 1991, pp. 345–405.
- R. Klein, *Concrete and Abstract Voronoi Diagrams*, Springer, 1989.

Alexander Hinneburg is working in the area of mining large databases, information visualization, and bioinformatics. He received his diploma and PhD in computer science from the University of Halle. Contact him at the Inst. of Computer Science, Univ. of Halle, Von-Seckendorff-Platz 1, 06120 Halle (Saale) Germany; hinneburg@informatik.uni-halle.de.

Daniel A. Keim is a full professor and head of the database and visualization group in the Department of Computer and Information Science at the University of Constance, Germany. His research interests include data mining and information visualization, as well as similarity search and indexing in multimedia databases. He is a member of the IEEE CS, ACM, and GI (the German Society for Informatics). Contact him at the Dept. of Computer and Information Science, Univ. of Constance, Universitätsstr. 10-D 78, 78457 Konstanz, Germany; keim@informatik.uni-konstanz.de.

Markus Wawryniuk is a PhD student in the Department of Computer and Information of the University of Constance, Germany. His research interest include data mining and information visualization. He is a member of the IEEE CS and the ACM. Contact him at the Dept. of Computer and Information Science, Univ. of Constance, Universitätsstr. 10-D 78, 78457 Konstanz, Germany; wawryniu@informatik.uni-konstanz.de.