Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering

Alexander Hinneburg
hinneburg@informatik.uni-halle.de

Daniel A. Keim
keim@informatik.uni-halle.de

Institute of Computer Science, University of Halle
Kurt-Mothes-Str.1, 06120 Halle (Saale), Germany

Abstract

Many applications require the clustering of large amounts of high-dimensional data. Most clustering algorithms, however, do not work effectively and efficiently in high-dimensional space, which is due to the so-called "curse of dimensionality". In addition, the high-dimensional data often contains a significant amount of noise which causes additional effectiveness problems. In this paper, we review and compare the existing algorithms for clustering high-dimensional data and show the impact of the curse of dimensionality on their effectiveness and efficiency. The comparison reveals that condensation-based approaches (such as BIRCH or STING) are the most promising candidates for achieving the necessary efficiency, but it also shows that basically all condensation-based approaches have severe weaknesses with respect to their effectiveness in high-dimensional space. To overcome these problems, we develop a new clustering technique called *OptiGrid* which is based on constructing an optimal grid-partitioning of the data. The optimal grid-partitioning is determined by calculating the best partitioning hyperplanes for each dimension (if such a partitioning exists) using certain projections of the data. The advantages of our new approach are (1) it has a firm mathematical basis (2) it is by far more effective than existing clustering algorithms for high-dimensional data (3) it is very efficient even for large data sets of high dimensionality. To demonstrate the effectiveness and efficiency of our new approach, we perform a series of experiments on a number of different data sets including real data sets from CAD and molecular biology. A comparison with one of the best known algorithms (BIRCH) shows the superiority of our new approach.

Proceedings of the 25th VLDB Conference, Edinburgh, Scotland, 1999.

1 Introduction

Because of the fast technological progress, the amount of data which is stored in databases increases very fast. This is true for traditional relational databases but also for databases of complex 2D and 3D multimedia data such as image, CAD, geographic, and molecular biology data. It is obvious that relational databases can be seen as high-dimensional databases (the attributes correspond to the dimensions of the data set), but it is also true for multimedia data which - for an efficient retrieval - is usually transformed into high-dimensional feature vectors such as color histograms [SH94], shape descriptors [Jag91, MG95], Fourier vectors [WW80], and text descriptors [Kuk92]. In many of the mentioned applications, the databases are very large and consist of millions of data objects with several tens to a few hundreds of dimensions.

Automated clustering in high-dimensional databases is an important problem and there are a number of different clustering algorithms which are applicable to high-dimensional data. The most prominent representatives are partitioning algorithms such as CLARANS [NH94], hierarchical clustering algorithms, and locality-based clustering algorithms such as (G)DBSCAN [EKSX96, EKSX97] and DBCLASD [XEKS98]. The basic idea of *partitioning algorithms* is to construct a partition of the database into k clusters which are represented by the gravity of the cluster (k-means) or by one representative object of the cluster (k-medoid). Each object is assigned to the closest cluster. A well-known partitioning algorithm is CLARANS which uses a randomised and bounded search strategy to improve the performance. *Hierarchical clustering algorithms* decompose the database into several levels of partitionings which are usually represented by a dendrogram - a tree which splits the database recursively into smaller subsets. The dendrogram can be created top-down (divisive) or bottom-up (agglomerative). Although hierarchical clustering algorithms can be very effective in knowledge discovery, the costs of creating the dendrograms is prohibitively expensive for large data sets since the algorithms are usually at least quadratic in the number of data objects. More efficient are *locality-based clustering algorithms* since they usually group neighboring data elements into clusters based on local conditions and therefore allow the clustering to be performed in one scan of the database. DBSCAN, for example, uses a density-based notion of clusters and allows the discovery of arbitrarily shaped clusters. The basic idea is that for each point of a cluster the density of data points in the neighborhood has to exceed some threshold. DBCLASD also works locality-based but in contrast to DBSCAN assumes that the points inside of the clusters are randomly distributed, allowing DBCLASD to work without any input parameters.

A problem is that most approaches are not designed for a clustering of high-dimensional data and therefore, the performance of existing algorithms degenerates rapidly with increasing dimension. To improve the efficiency, optimised clustering techniques have been proposed. Examples include Grid-based clustering [Sch96], BIRCH [ZRL96] which is based on the Cluster-Feature-tree, STING which uses a quadtree-like structure containing additional statistical information [WYM97], and DENCLUE which uses a regular grid to improve the efficiency [HK98]. Unfortunately, the curse of dimensionality also has a severe impact on the effectiveness of the resulting clustering. So far, this effect has not been examined thoroughly for high-dimensional data but a detailed comparison shows severe problems in effectiveness (cf. section 2), especially in the presence of noise. In our comparison, we analyse the impact of the dimensionality on the effectiveness and efficiency of a number of well-known and competitive clustering algorithms. We show that they either suffer from a severe breakdown in efficiency which is at least true for all index-based methods or have severe effectiveness problems which is basically true for all other methods. The experiments show that even for simple data sets (e.g. a data set with two clusters given as normal distributions and a little bit of noise) basically none of the fast algorithms guarantees to find the correct clustering.

From our analysis, it gets clear that only condensation-based approaches (such as BIRCH or DENCLUE) can provide the necessary efficiency for clustering large data sets. To better understand the severe effectiveness problems of the existing approaches, we examine the impact of high dimensionality on condensation-based approaches, especially grid-based approaches (cf. section 3). The discussion in section 3 reveals the source of the problems, namely the inadequate partitioning of the data in the clustering process. In our new approach, we therefore try to find a better partitioning of the data. The basic idea of our new approach presented in section 4 is to use contracting projections of the data to determine the optimal cutting (hyper-)planes for partitioning the data. If no good partitioning plane exist in some dimensions, we do not partition the data set in those dimensions. Our strategy of using a data-dependent partitioning of the data avoids the effectiveness problems of the existing approaches and guarantees that all clusters are found by the algorithm (even for high noise levels), while still retaining the efficiency of a grid-based approach. By using the highly-populated grid cells based on the optimal partitioning of the data, we are able to efficiently determine the clusters. A detailed evaluation 5 shows the advantages of our approach. We show theoretically that our approach guarantees to find all center-defined clusters (which roughly spoken correspond to clusters generated by a normal distribution). We confirm the effectiveness lemma by an extensive experimental evaluation on a wide range of synthetic and real data sets, showing the superior effectiveness of our new approach. In addition to the effectiveness, we also examine the efficiency, showing that our approach is competitive with the fastest existing algorithms (BIRCH) and (in some cases) even outperforms BIRCH by up to a factor of about 2.

2 Clustering of High-Dimensional Data

In this section, we discuss and compare the most efficient and effective available clustering algorithms and examine their potential for clustering large high-dimensional data sets. We show the impact of the curseof dimensionality and reveal severe efficiency and effectiveness problems of the existing approaches.

2.1 Related Approaches

The most efficient clustering algorithms for low-dimensional data are based on some type of hierarchical data structure. The data structures are either based on a hierarchical partitioning of the data or a hierarchical partitioning of the space.

All techniques which are based on partitioning the data such as R-trees do not work efficiently due to the performance degeneration of R-tree-based index structures in high-dimensional space. This is true for algorithms such as DBSCAN [EKSX96] which has an almost quadratic time complexity for high-dimensional data if the R*-tree-based implementation is used. Even if a special indexing techniques for high-dimensional data is used, all approaches which determine the clustering based on near(est) neighbor information do not work *effectively* since the near(est) neighbors do not contain sufficient information about the density of the data in high-dimensional space (cf. section 3.2), which means that algorithms such as the k-means or DBSCAN algorithm do not work effectively on high-dimensional data.

A more efficient approach is BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) which uses a data partitioning according to the expected cluster structure of the data [ZRL96]. BIRCH uses a hierarchical data structure called CF-tree (Cluster Feature Tree) which is a balanced tree for storing the clustering features. BIRCH tries to build the best possible clustering using the given limited (memory) resources. The idea of BIRCH is store similar data items in the node of the CF-tree and if the algorithm runs short of main memory, similar data items in the nodes of the CF-tree are condensed. BIRCH uses several heuristics to find the clusters and to distinguish the clusters from noise. Due to the specific notion of similarity used to determine the data items to be condensed BIRCH is only able to find spherical clusters. Still, BIRCH is one of the most efficient algorithms and needs only one scan of the the database (time complexity $O(n)$), which is also true for high-dimensional data.

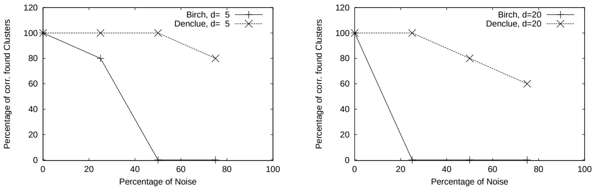
On low-dimensional data, space partitioning methods which in general use a grid-based approach such as STING (STatistical INformation Grid) [WYM97] and WaveCluster [SCZ98] are of similar efficiency (the time complexity is $O(n)$), but better effectiveness than BIRCH (especially for noisy data and arbitrary-shape clusters) [SCZ98]. The basic idea of STING is to divide the data space into rectangular cells and store statistical parameters (such as mean, variance, etc.) of the objects in the cells. This information can then be used to efficiently determine the clusters. WaveCluster[SCZ98] is a wavelet-based approach which also uses a regular grid for an efficient clustering. The basic idea is map the data onto a multi-dimensional grid, apply a wavelet transformation to the grid cells, and then determine dense regions in the transformed domain by searching for connected components. The advantage of using the wavelet transformation is that it automatically provides a multiresolutions representation of the data grid which allows an efficient determination of the clusters. Both, STING and WaveCluster have only been designed for low-dimensional data. There is no straight-forward extention to the high-dimensional case. In WaveCluster, for example, the number of grid cells grows exponentially in the number of dimensions (d) and determining the connected components becomes prohibitively expensive due to the large number of neighboring cells. An approach which works better for the high-dimensional case is the DENCLUE approach [HK98]. The basic idea is to model the overall point density analytically as the sum of influence functions of the data points. Clusters can then be identified by determining density-attractors and clusters of arbitrary shape can be easily described by a simple equation based on the overall density function. DENCLUE has been shown to be a generalization of a number of other clustering algorithm such as k-means and DBSCAN [HK99], and it also generalizes the STING and WaveCluster approaches. To work efficiently on high-dimensional data, DENCLUE uses a grid-based approach but only stores the grid-cells which contain data points. For an efficient clustering, DENCLUE connects all neighboring populated grid cells of a highly-populated grid cell.

2.2 Comparing the Effectiveness

Unfortunately, for high-dimensional data none of the approaches discussed so far is fully effective. In the following preliminary experimental evaluation, we briefly show that none of the existing approaches is able to find all clusters on high-dimensional data. In the experimental comparison, we restrict ourselves to the most efficient and effective algorithms which all use some kind of aggregated (or condensed) information being stored in either a cluster-tree (in case of BIRCH) or a grid (in case of STING, WaveCluster, and DENCLUE). Since all of them have in common that they condense the available information in one way or the other, in the following we call them condensation-based approaches.

For the comparison with respect to high-dimensional data, it is sufficient to focus on BIRCH and DENCLUE since DENCLUE generalizes STING and WaveCluster and is directly applicable to high-dimensional data. Both DENCLUE and BIRCH have a similar time complexity, in this preliminary comparison we focus on their effectiveness (for a detailed comparison, the reader is referred to section 5). To show the effectiveness problems of the existing approaches on high-dimensional data, we use synthetic data sets consisting of a number of clusters defined by a normal distribution with the centers being randomly placed in the data space.

In the first experiment (cf. Figure 1), we analyse the percentage of clusters depending on the percentage of noise in the data set. Since at this point, we are mainly interested in getting more insight into the problems of clustering high-dimensional data, a sufficient measure of the effectiveness is the percentage of correctly found clusters. Note that BIRCH is very sensitive to noise for higher dimensions for the realistic situation that the data is read in a random order. If however the data points belonging to clusters are known a-priori (which is not a realistic assumption), the cluster points can be read first and in this case, BIRCH provides a much better effectiveness since the CF-tree does not degenerate as much. If the clusters are inserted first, BIRCH provides about the same effectiveness as grid-based approaches such as DENCLUE. The curves show how critical the dependency on the insertion order becomes for higher dimensional data. Grid-based approaches are in general not sensitive to the insertion order.



(a) Dimension 5 (b) Dimension 20

Figure 1:
Comparison of DENCLUE and Birch on noisy data

The second experiment shows the average percentage of clusters found for 30 data sets with different positions of the clusters. Figure 2 clearly shows that for high dimensional data, even grid-based approaches such as DENCLUE are not able to detect a significant percentage of the clusters, which means that even DENCLUE does not work fully effective. In the next section, we try to provide a deeper understanding of these effects and discuss the reasons for the effectiveness problems.

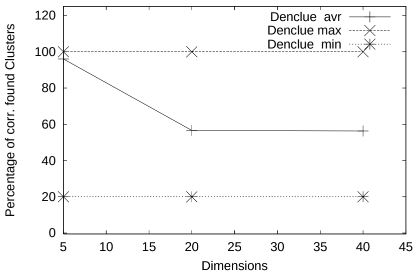


Figure 2: Effects of grid based Clustering (DENCLUE)

2.3 Discovering Noise in High-Dimensional Data

Noise is one of the fundamental problems in clustering large data sets. In high-dimensional data sets, the problem becomes even more severe. In this subsection, we therefore discuss the general problem of noise in high-dimensional data sets and show that it is impossible to determine clusters in data sets with noise correctly in linear time (in the high-dimensional case).

Lemma 1 (Complexity of Clustering)
The worst case time complexity for a correct clustering of high-dimensional data with noise is superlinear.

Idea of the Proof:
Without loss of generality, we assume that we have $O(n)$ noise in the database (e.g., 10% noise). In the worst case, we read all the noise in the beginning which means that we can not obtain any clustering in reading that data. However, it is also impossible to tell that the data we have already read does not belong to a cluster. Therefore, in reading the remaining points we have to search for similar points among those noise points. The search for similar points among the noise points can not be done in constant time ($O(1)$) since we have $O(n)$ noise points and in high-dimensional space, an $O(1)$ access to similar data in the worst case (based on techniques such as hashing or histograms) is not possible even for random noise (cf. section 3 for more details on this fact). The overall time complexity is therefore superlinear. $\Box$

The lemma implies that the clustering of data in high-dimensional data sets with noise is an inherently non-linear problem. Therefore, any algorithm with linear time complexity such as BIRCH can not cluster noisy data correctly. Before we introduce our algorithm, in the following we provide more insight into the effects and problems of high-dimensional clustering.

3 Grid-based Clustering of High-dimensional Spaces

In the previous section, we have shown that grid-based approaches perform well with respect to efficiency and effectiveness. In this section, we discuss the impact of the "curse of dimensionality" on grid-based clustering in more detail. After some basic considerations defining clustering and a very general notion of multidimensional grids, we discuss the effects of different high-dimensional data distributions and the resulting problems in clustering the data.

3.1 Basic Considerations

We start with a well known and widely accepted definition of clustering (cf. [HK98]). For the definition, we need a density function which is determined based on kernel density estimation.

Definition 2 (Density Function)
Let D be a set of n d-dimensional points and h be the smoothness level. Then, the **density function** $\hat{f^D}$ *based on the kernel density estimator K is defined as:*

$$\hat{f^D}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Kernel density estimation provides a powerful framework for finding clusters in large data sets. In the statistics literature, various kernels K have been proposed. Examples are square wave functions or Gaussian functions. A detailed introduction into kernel density estimation is beyond the scope of this paper and can be found in [Sil86],[Sco92]. According to [HK98], clusters can now be defined as the maxima of the density function, which are above a certain noise level ξ.

Definition 3 (Center-Defined Cluster)
A center-defined cluster for a maximum x^* of the density function $\hat{f^D}$ is the subset $C \subseteq D$, with $x \in C$ being density-attracted by x^* and $\hat{f^D}(x^*) \geq \xi$. Points

$x \in D$ are called outliers if they are density-attracted by a local maximum x_o^* with $\hat{f}^D(x_o^*) < \xi$.

According to this definition, each local maximum of the density function which is above the noise level ξ becomes a cluster of its own and consists of all points which are density-attracted by the maximum. The notion of *density-attraction* is defined by the gradient of the density function. The definition can be extended to clusters which are defined by multiple maxima and can approximate arbitrarily-shaped clusters.

Definition 4 (Multicenter-Defined Cluster)
A multicenter-defined cluster for a set of maxima X is the subset $C \subseteq D$, where

1. *$\forall x \in C\ \exists x^* \in X :\ f_B^D(x^*) \geq \xi$, x is density-attracted to x^* and*
2. *$\forall x_1^*, x_2^* \in X : \exists$ a path $P \subset F^d$ from x_1^* to x_2^* above noise level ξ.*

As already shown in the previous section, grid-based approaches provide an efficient way to determine the clusters. In grid-based approaches, all data points which fall into the same grid cell are aggregated and treated as one object. In the low-dimensional case, the grid can be easily stored as an array which allows a very efficient access time of $O(1)$. In case of a high-dimensional grid, the number of grid cells grows exponentially in the number of dimensions d which makes it impossible to store the grid as a multi-dimensional array. Since the number of data points does not grow exponentially, most the grid cells are empty and do not need to be stored explicitly, but it is sufficient to store the populated cells. The number of populated cells is bounded by the number of non-identical data points.

Since we are interested in arbitrary (non-equidistant, irregular) grids, we need the notion of a cutting plane which is a (d-1)-dimensional hyperplane cutting the data space into the grid cells.

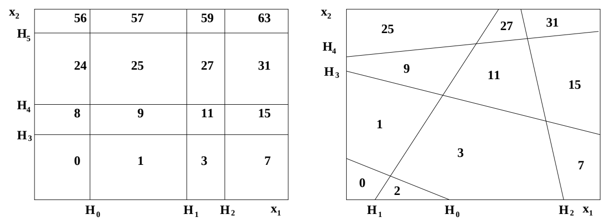
Definition 5 (Cutting Plane)
*A **cutting plane** is a $(d-1)$-dimensional hyperplane consisting of all points y which fulfil the equation $\sum_{i=1}^{d} w_i y_i = 1$. The cutting plane partitions $\mathbb{R}^d$ into two half spaces. The decision function $H(x)$ determines the half space, where a point $x \in \mathbb{R}$ is located:*

$$H(x) = \begin{cases} 1 & , \sum\limits_{i=1}^{d} w_i x_i \geq 1 \\ 0 & , else \end{cases}$$

Now, we are able to define a general notion of arbitrary (non-equidistant, irregular) grids. Since the grid can not be stored explicitly in high-dimensional space, we need a coding function c which assigns a label to all points belonging to the same grid cell. The data space S as well as the grid cells are defined as right semi-opened intervals.

Definition 6 (Multidimensional Grid)
*A **multidimensional grid** G for the data space S is defined by a set $H = \{H_1, \ldots, H_k\}$ of $(d-1)$-dimensional cutting planes. The coding function $c^G : S \longrightarrow \mathbb{N}$ is defined as follows:*

$$x \in S,\ c(x) = \sum_{i=1}^{k} 2^i \cdot H_i(x).$$



(a) regular (b) irregular
Figure 3: Examples of 2-dimensional Grids

Figure 3 shows two examples of two-dimensional grids. The grid in figure 3a shows a non-equidistant regular (axes-parallel cutting planes) grid and 3b shows a non-equidistant irregular (arbitrary cutting planes) grid. The grid cells are labeled with value determined by the coding function according to definition 6. In general, the complexity of the coding function is $O(k \cdot d)$. In case of a grid based on axes parallel hyperplanes, the complexity of the coding function becomes $O(k)$.

3.2 Effects of Different Data Distributions

In this section, we discuss the properties of different data distributions for an increasing number of dimensions. Let us first consider uniformly distributed data. It is well-known, that uniform distributions are very unlikely in high-dimensional space. From a statistical point of view, it is even impossible to determine a uniform distribution in high-dimensional space a-posteriori. The reason is that there is no possibility to have enough data points to verify the data distribution by a statistical test with sufficient significance. Assume we want to characterize the distribution of a 50-dimensional data space by an grid-based histogram and we split each dimension only once at the center. The resulting space is cut into $2^{50} \sim 10^{14}$ cells. If we generate one billion data points by a uniform random data generator, we get about 10^{12} cells filled with one data point which is about one percent of the cells. Since the grid is based on a very coarse partitioning (one cutting plane per dimension), it is impossible to determine a data distribution based on one percent of the cells. The available information could justify a number of different distributions including a uniform distribution. Statistically, the number of data points is not high enough to determine the distribution of the data. The problem is that the number of data points can not grow exponentially with the dimension, and therefore, in high-dimensional space it is generally impossible to determine the distribution of the data with sufficient statistical significance. (The only thing which can be verified easily is that the projections onto the dimensions follow a uniform distribution.) As a result of the sparsely filled space, it is very unlikely that data points are nearer to each other than the average distance between data points, and as a consequence, the difference between the distance to the nearest and the farthest neighbor of a data point goes to zero in high-dimensional space (see [BGRS99] for a recent theoretical proof of this fact).

Now let us look at normally distributed data. A normal distribution is characterized by the center point (expected value) and the standard deviation (σ). The distance of the data points to the expected point follows a Gaussian curve but the direction from the expected point is randomly chosen without any preference. An important observation is that the number of possible directions from a point grows exponentially in the number of dimensions. As a result, the distance among the normally distributed data points increases with the number of dimensions although the distance to the center point still follows the same distribution. If we consider the density function of the data set, we find that it has a maximum at the center point although there may be no data points very close to the center point. This results from the fact that it is likely that the data points slightly vary in the value for one dimension but still the single point densities add up to the maximal density at the center point. The effect that in high dimensional spaces the point density can be high in empty areas is called the *empty space phenomenon* [Sco92].

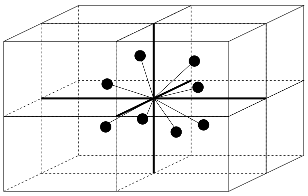


Figure 4:
Example Scenario for a Normal Distribution, $d = 3$

To illustrate this effect, let us consider normally distributed data points in $[0, 1]^d$ with $(0.5, \ldots, 0.5)$ as center point and a grid based on splitting at 0.5 in each dimension. The number of directions from the center point now directly corresponds to the number of grid cells which is exponential in d (2^d). As a consequence, most data points will fall into separate grid cells (Figure 4 shows an example scenario for $d = 3$). In high dimensions, it is unlikely that there are any points in the center and that populated cell are adjacent to each other on a high-dimensional hyperplane which is again an explanation of the high inter-point distances.

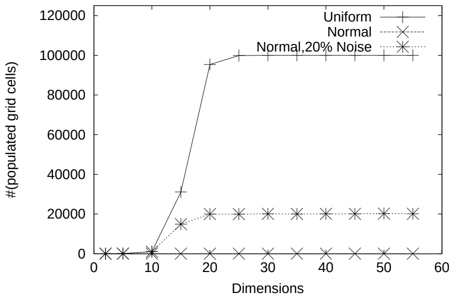


Figure 5:
#(Populated Grid Cells) for Different Data Distributions

To show the effects of high-dimensional spaces on grid-based clustering approaches, we performed some interesting experiments based on using a simple grid and counting the number of populated grid cells containing different numbers of points. The resulting figures show the effects of different data distributions and allow interesting comparisons of the distributions. Figure 5 shows the total number of populated cells (containing at least one data point) depending on the dimensionality. In the experiments, we used three data sets consisting of 100000 data points generated by a uniform distribution (since it is impossible to generate uniform distributions in high-dimensional space we use a uniformly generated distribution of which the projections are at least uniformly distributed), a normal distribution containing 5 clusters with $\sigma = 0.1$ and centers uniformly distributed in $[0, 1)^d$ and a combination of both (20% of the data is uniformly distributed). Based on the considerations discussed above, it is clear that for the uniformly distributed data as many cells as possible are populated which is the number of available cells (for $d \leq 20$) and the number of data points (for $d \geq 25$). For normally distributed data, the number of populated data points is always lower but still increases for higher dimensions due to the many directions the points may vary from the center point (which can not be seen in the figure). The third data set is a combination of the other two distributions and converges against the percentage of uniformly distributed data points (20000 data points in the example).

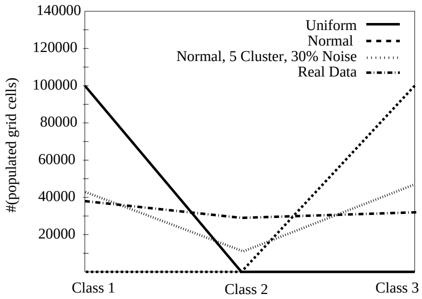


Figure 6:
#(Populated Grid Cells) for Molecular Biology Data

It is now interesting to use the same approach for a better understanding of real data sets. It can be used to analyse and classify real data sets by assessing how clustered the data is and how much noise it contains. Figure 6 shows the results for a real data set from a molecular biology application. The data consists of about 100000 19-dimensional feature vectors describing peptides in a dihedral angle space. In figure 6, we show the number of data points which fall into three different classes of grid cells. Class one accounts for all data points which fall into grid cells containing a small number of data points (less than 2^4), class two those which contain a medium number of data points (between 2^4 and 2^8 data points), and class three those which contain a large number of data points (more than 2^8 data points). We show the resulting curves for uniformly distributed data, normally distributed data, a combination of both, and the peptide data. Figure 6 clearly shows that the peptide data corresponds neither to uniformly nor to normally distributed data, but can be approximated most closely by a combination of both (with about 40% of noise). Figure 6 clearly shows that the peptide data sets contains a significant portion of noise (corresponding to a uniform distribution) but also contains clear clusters (corresponding to the normal distribution). Since real data sets may contain additional structure such as dependencies between dimensions or extended clusters of arbitrary shape, the classification based on counting the grid cells of the different population levels can not be used to detect all properties of the data and completely classify real data sets. However, as shown in Figure 6 the approach is powerful and allows to provide interesting insights into the properties of the data (e.g., percentage of noise and clusters). In examining other real data sets, we found that real high-dimensional data is usually highly clustered but mostly also contains a significant amount of noise.

3.3 Problems of Grid-based Clustering

A general problem of clustering in high-dimensional spaces arises from the fact that the cluster centers can not be as easily identified (by a large number of almost identical data points) as in lower dimensional cases. In grid-based approaches it is possible that clusters are split by some of the (d-1) dimensional cutting planes and the data points of the cluster are spread over many grid cells. Let us use a simple example to exemplify this situation. For simplification, we use a grid where each dimension is split only once. In general, such a grid is defined by d $(d-1)$-dimensional hyperplanes which cut the space into 2^d cells. All cutting planes are parallel to $(d-1)$ coordinate axes. By cutting the space into cells, the naturally neighborhood between the data points gets lost. A worst case scenario could be the following case. Assume the data points are in $[0,1]^d$ and each dimension is split at 0.5. The data points lie on a hyper sphere with small radius $\epsilon > 0$ round the split point $(0.5, 0.5, \ldots, 0.5)$. For $d > 20$, most of the points would be in separate grid cells despite the fact that they form a cluster. Note that there are 2^d cells adjacent to the split point. Figure 4 tries to show this situation of a worst case scenario for a three-dimensional data set. In high-dimensional data, this situation is likely to occur and this is the reason for the effectiveness problems of DENCLUE (cf. Figure 2).

An approach to handle the effect is to connect adjacent grid cells and treat the connected cells as one object. A naive approach is to test all possible neighboring cells of a populated cell whether they are also populated. This approach however is prohibitively expensive in high-dimensional spaces because of the exponential number of adjacent neighbor grid cells. If a grid with only one split per dimension is used, all grid cells are adjacent to each other and the number of connections becomes quadratic in the number of grid cells, even if only the $O(n)$ populated grid cells are considered. In case of finer grids, the number of neighboring cells which have to be connected decreases but at the same time, the probability that cutting planes hit cluster centers increases. As a consequence, any approach which considers the connections for handling the effect of splitted clusters will not work efficiently on large databases, and therefore another solution guaranteeing the effectiveness while preserving the efficiency is necessary for an effective clustering of high-dimensional data.

4 Efficient and Effective Clustering in High-dimensional Spaces

In this section, we propose a new approach to high-dimensional clustering which is efficient as well as effective and combines the advantages of previous approaches (e.g, BIRCH, WaveCluster, STING, and DENCLUE) with a guaranteed high effectiveness even for large amounts of noise. In the last section, we discussed some disadvantages of grid-based clustering, which are mainly caused by cutting planes which partition clusters into a number of grid cells. Our new algorithm avoids this problem by determining cutting planes which do not partition clusters.

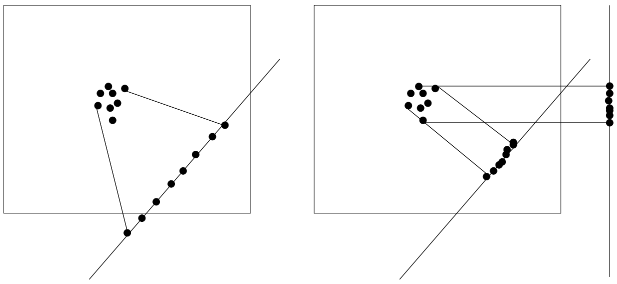
There are two desirable properties for good cutting planes: First, cutting planes should partition the data set in a region of low density (the density should be at least low relative to the surrounding region) and second, a cutting plane should discriminate clusters as much as possible. The first constraint guarantees that a cutting plane does not split a cluster, and the second constraint makes sure that the cutting plane contributed to finding the clusters. Without the second constraint, cutting planes are best placed at the borders of the data space because of the minimal density there, but it is obvious that in that way clusters can not be detected. Our algorithm incorporates both constraints. Before we introduce the algorithm, in the following we first provide some mathematical background on finding regions of low density in the data space which is the basis for our optimal grid partitioning and our algorithm *OptiGrid*.

4.1 Optimal Grid-Partioning

Finding the minima of the density function $\hat{f}^D$ is a difficult problem. For determining the optimal cutting plane, it is sufficient to have information on the density on the cutting plane which has to be relatively low. To efficiently determines such cutting planes, we use contracting projections of the data space.

Definition 7 (Contracting Projection)
A contracting projection for a given d-dimensional data space S and an appropriate metric $\|\cdot\|$ is a linear transformation P defined on all points $x \in S$

$$P(x) = Ax \quad with \quad \|A\| = \max_{y \in S}\left(\frac{\|Ay\|}{\|y\|}\right) \leq 1 \,.$$



(a) general (b) contracting
Figure 7: General and Contracting Projections

Now we can proof a main lemma for the correctness of our algorithm, which states that the density at a point x' in a contracting projection of the data is an upper bound for the density on the plane, which is orthogonal to the projection plane.

Lemma 8 (Upper Bound Property)
Let $P(x) = Ax$ be a contracting projection, $P(D)$ the projection of the data set D, and $\hat{f}^{P(D)}(x')$ the density for a point $x' \in P(S)$. Then,

$$\forall x \in S \text{ with } P(x) = x' : \; \hat{f}^{P(D)}(x') \geq \hat{f}^D(x) \,.$$

Proof: First, we show that the distance between points becomes smaller by the contracting projection P. According to the definition of contracting projections, for all $x, y \in S$:

$$\|P(x) - P(y)\| = \|A(x-y)\| \leq \|A\| \cdot \|x-y\| \leq \|x-y\|$$

The density function which we assume to be kernel based depends monotonically on the distance of the data points. Since the distance between the data points becomes smaller, the density in the data space S grows. □

The assumption that the density is kernel based is not a real restriction. There are a number of proofs in the statistical literature that non-kernel based density estimation methods converge against a kernel based method [Sco92]. Note that Lemma 8 is a generalization of the Monotonicity Lemma in [AGG+98]. Based on the preceeding lemma, we now define an optimal grid-partitioning.

Definition 9 (Optimal Grid-Partioning)
For a given set of projections $P = \{P_0, \ldots, P_k\}$ and a given density estimation model $\hat{f}(x)$, the **optimal l grid partitioning** *is defined by the l best separating cutting planes. The projections of the cutting planes have to separate significant clusters in at least one of the projections of P.*

The term *best separating* heavily depends on the considered application. In section 5 we discuss several alternatives and provide an efficient method for normally distributed data with noise.

4.2 The OptiGrid Algorithm

With this definition of optimal grid-partitioning, we are now able to describe our general algorithm for high-dimensional data sets. The algorithm works recursively. In each step, it partitions the actual data set into a number of subsets if possible. The subsets which contain at least one cluster are treated recursively. The partitioning is done using a multidimensional grid defined by at most q cutting planes. Each cutting plane is orthogonal to at least one projection. The point density at a cutting planes is bound by the density of the orthogonal projection of the cutting plane in the projected space. The q cutting planes are chosen to have a minimal point density. The recursion stops for a subset if no good cutting plane can be found any more. In our implementation this means that the density of all possible cutting planes for the given projections is above a given threshold.

OptiGrid(data set D, q, min_cut_score)

1. Determine a set of contracting projections $P = \{P_0, \ldots, P_k\}$
2. Calculate all projections of the data set $D \rightarrow P_0(D), \ldots, P_k(D)$
3. Initialize a list of cutting planes $BEST\_CUT \leftarrow \emptyset$, $CUT \leftarrow \emptyset$
4. FOR i=0 TO k DO
   (a) $CUT \leftarrow$ Determine best_local_cuts($P_i(D)$)
   (b) $CUT\_SCORE \leftarrow$ Score best_local_cuts($P_i(D)$)
   (c) Insert all cutting planes with a score $\geq$ min_cut_score into $BEST\_CUT$
5. IF $BEST\_CUT = \emptyset$ THEN RETURN D as a cluster

6. Determine the q cutting planes with highest score from $BEST\_CUT$ and delete the rest
7. Construct a Multidimensional Grid G defined by the cutting planes in $BEST\_CUT$ and insert all data points $x \in D$ into G
8. Determine clusters, i.e. determine the highly populated grid cells in G and add them to the set of cluster C
9. Refine(C)
10. FOREACH Cluster $C_i \in C$ DO OptiGrid(C_i, q, min_cut_score)

In step 4a of the algorithm, we need a function which produces the best local cutting planes for a projection. In our implementation, we determine the best local cutting planes for a projection by searching for the leftmost and rightmost density maximum which are above a certain noise level and for the $q-1$ maxima in between. Then, it determines the points with a minimal density in between the maxima. The position of the minima determines the corresponding cutting plane and the density at that point gives the quality (score) of the cutting plane. The estimation of the noise level can be done by visualizing the density distribution of the projection. Note that the determination of the q best local cutting planes depends on the application. Note that our function for determining the best local cutting planes adheres to the two constrains for cutting planes but additional criteria may be useful.

Our algorithm as described so far is mainly designed to detect center-defined clusters (cf. step 8 of the algorithm). However, it can be easily extended to also detect multicenter-defined clusters according to definition 4. The algorithm just has to evaluate the density between the center-defined clusters determined by OptiGrid and link the clusters if the density is high enough.

The algorithm is based on a set of projections. Any contracting projection may be used. General projections allow for example the detection of linear dependencies between dimensions. In cases of projections $P : \mathbb{R}^d \to \mathbb{R}^{d'}$, $d' \leq 3$ an extension of the definitions of multidimensional grid and cutting planes is necessary to allow a partitioning of the space by polyhedrons. With such an extension, even quadratic and cubic dependencies which are special kinds of arbitrary shaped clusters may be detected. The projections can be determined using algorithms for principal component analysis, techniques such as FASTMAP [FL95] or projection pursuit [Hub85]. It is also possible to incorporate prior knowledge in that way.

4.3 Complexity

In this section, we provide a detailed complexity analysis and hints for an optimized implementation of the OptiGrid algorithm. For the analysis, we assume that the data set D contain N $d-$dimensional data points. Frist, we analyse the complexity to the main steps. The first step takes only constant time, because we use a fixed set of projections. The number of projections k can be assumed to be in $O(d)$. In the general case, the calculation of all projections of the data set D may take $O(N \cdot d \cdot k)$, but the case of axes parallel projections $P : \mathbb{R}^d \to \mathbb{R}$ it is $O(N \cdot k)$.

The determination of the best local cutting planes for a projection can be done based on 1-dimensional histograms. The procedure takes time $O(N)$. The whole loop of step 4 runs in $O(N \cdot k)$.

The implemention of the multidimensional grid depends on the number of cutting planes q and the available resources of memory and time. If the possible number of grid cell n^G, i.e $n^G = 2^q$, does not exceed the size of the available memory, the grid can be implemented as an array. The insertion time for all data points is then $O(N \cdot q \cdot d)$. If the number of potential grid cells becomes to large, only the populated grid cells can be stored. Note that this is only the case, if the algorithm has found a high number of meaningful cutting planes and therefore, this case only occurs if the data sets consists of a high number of clusters. In this case, a tree or hash based data structure is required for storing the grid cells. The insertion time then becomes to $O(N \cdot q \cdot d \cdot I)$ where I is the time to insert a data item into the data structure which is bound by $minq, logN$. In case of axes parallel cutting planes, the insertion time is $O(N \cdot q \cdot I)$. The complexity of step 7 dominates to complexity of the recursion.

Note that the number of recusions depends on the data set and can be bounded by the number of clusters $\#C$. Since q is a constant of our algorithm and since the number of clusters is a constant for a given data set, the total complexity is between $O(N \cdot d)$ and $O(d \cdot N \cdot logN)$. In our experimental evaluation (cf. 5), it is shown that the total complexity is slightly superlinear as implied by our complexity lemma 1 in section 2.3.

5 Evaluation

In this section, first we provide a theoretical evaluation of the effectiveness of our approach and then, we demonstrate the efficiency and effectiveness of OptiGrid using a variety of experiments based on synthetic as well as real data sets.

5.1 Theoretical Evaluation of the Effectiveness

A main point in the OptiGrid approach is the choice of the projections. The effectiveness of our approach depends strongly on the set P of projections. In our implementation, we use all projections d $P : \mathbb{R}^d \to \mathbb{R}$ to the d coordinate axes. The resulting cutting planes are obviously axes parallel.

Let us now examine the discriminative power of axes parallel cutting planes. Assume for that purpose two clusters with same number of points. The data points of both clusters follow an independent normal distribution with standard deviation $\sigma = 1$. The centers of both clusters have a minimal distance of 2σ. The worst case for the discrimination by axes parallel cutting planes is that the cluster centers are on a diagonal of the data space and have minimal distance. Figure 8 shows an example for the 2-dimensional case and the L_1-metric. Under these conditions we can prove that the error of partitioning the two clusters with axes parallel cutting plane is limited by a small constant in high-dimensional spaces.

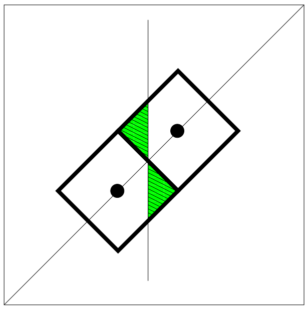


Figure 8:
Worst case for partitioning Center-defined Clusters

Lemma 10 (Error-Bound)
Using the L_1-metric, two d-dimensional center defined clusters approximated by two hyper spheres with radius 1 can be partitioned by an axes parallel cutting plane with at most $(1/2)^{d+1}$ percent wrongly classified points.

Proof: According to the density of the projected data, the optimal cutting plane is in middle between the two clusters but axes parallel. Figure 8 shows the worst case scenario for which we estimate the error. The two shaded regions in the picture mark the wrongly classified point sets. The two regions form a d-dimensional hypercube with an edge length $e = \frac{\sqrt{d}}{2}$. The maximal error can be estimated by the fraction of the volume of the clusters and the volume of the wrongly classified point sets.

$$error_{max} = \frac{V_{bad}}{V_{cluster}} = \frac{(1/2 \cdot \sqrt{d})^d}{2(\sqrt{d})^d} = (1/2)^{d+1} \ .$$

The error bound for metrics L_p, $p \geq 2$ is lower than the bound for the L_1 metric. □

Note that in higher dimensions the error bound converges against 0. An important aspect however is that both clusters are assume to consist of the same number of data points. The split algorithm tends to preserve the more dense areas. In case of clusters with very different numbers of data points the algorithm splits the smaller clusters more than the larger ones. In extreme cases, the smaller clusters may then be rejected as outliers. In such cases, after determining the large clusters a reclustering of the data set without the large clusters should be done. Because of the missing influence of the large clusters the smaller ones now becomes more visible and will be correctly found.

An example for the first steps of partitioning a two-dimensional data set (data set DS3 from BIRCH) is provided in Figure 9. It is interesting to note that regions containing large clusters are separated first. Although not designed for low-dimensional data, OptiGrid still provides correct results on such data sets.

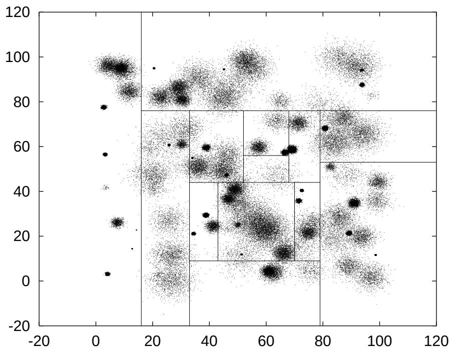
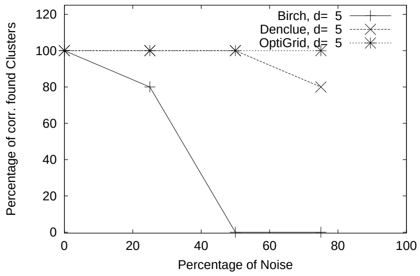


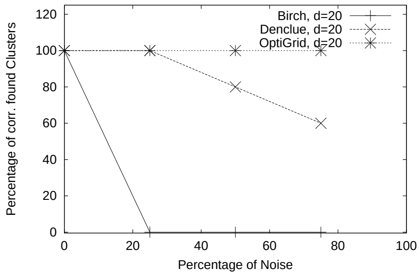
Figure 9:
The first determined cut planes of OptiGrid (q=1)

5.2 Experimental Evaluation of Effectiveness and Efficiency

In this section, we provide a detailed experimental evaluation of the effectiveness and efficiency of OptiGrid. We show that BIRCH is more effective and even slightly more efficient than the best previous approaches, namely BIRCH and DENCLUE.



(a) Dimension 5



(b) Dimension 20

Figure 10: Dependency on the Noise Level

The first two experiments showing the effectiveness of OptiGrid use the same data sets as used for showing the problems of previous BIRCH and DENCLUE in subsection 2.2. As indicated by the theoretical considerations, OptiGrid provides a better effectiveness than BIRCH and DENCLUE. Figure 10a shows the percentage of correctly found clusters for $d = 5$ and 10b shows the same curves for $d = 20$. OptiGrid correctly determines all clusters in the data sets, even for very large noise levels of up to 75%. Similar curves result for $d > 20$.

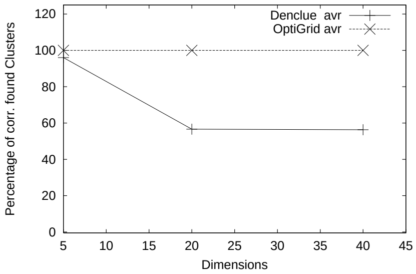


Figure 11: Dependency on the Dimensionality

The second experiments show the average percentage of correctly found clusters for data sets with different cluster centers (cf. figure 2). Again, OptiGrid is one hundred percent effective since it does not partitions clusters due to the optimal grid-partitioning algorithm.

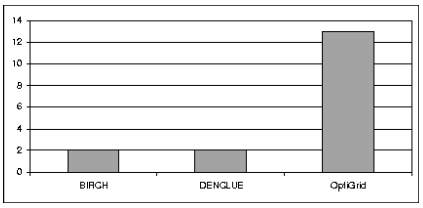


Figure 12:
Effectiveness on Real Molecular Biology Data

For the comparison with real data, we used two data sets which have been previously used in evaluating high-dimensional clustering algorithms [HK98]. The first data set results from an application in the area of molecular biology. The data used for our tests comes from a complex simulation of a very small but flexible peptide. (The time for performing the simulation took several weeks of CPU-time.) The data generated by the simulation describes the conformation of the peptide as a 19-dimensional point in the dihedral angle space [DGRJS97]. The simulation was done for a period of 50 nanoseconds with two snapshots taken every picosecond, resulting in about 100000 data points. The purpose of the simulation was to determine the behavior of the molecule in the conformation space. Due to the large amount of high-dimensional data, it is difficult to find, for example, the preferred conformations of the molecule. This, however, is important for applications in the pharmaceutical industry since small and flexible peptides are the basis for many medicaments. The flexibility of the peptides, however, is also responsible for the fact that the peptide has many intermediate conformations which causes the data to contain large amounts of noise (about 40 percent). In identifying clusters in this data set, OptiGrid turned out to be much more effective than DENCLUE and BIRCH (cf. figure 12). Both BIRCH and DENCLUE are able to find 2 clusters while OptiGrid determines 13 clusters (with a comparable efficiency). Describing the consequences of the identified structure in the data is beyond the scope of this paper, but has an important applications in the pharmaceutical industry.

The second real data set is polygonal CAD data which is transformed into 11-dimensional feature vectors using a Fourier transformation. The number of data points used in the experiments is 36000. In this experiment, OptiGrid and DENCLUE found the same number of clusters but BIRCH did not provide any clustering results. In this case, however, OptiGrid was about 10 times faster than DENCLUE.

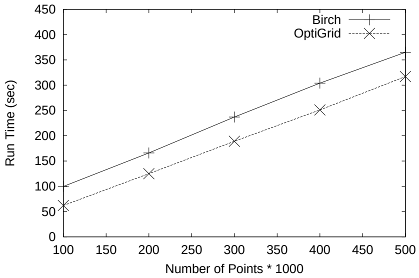


Figure 13: Performance Comparison

Although the previous experiments have shown that the three approaches are incomparable with respect to their effectiveness, in the following we still compare their efficiency. In the comparison, we use synthetic data sets with $d = 40$, 25% noise, and an increasing number of data points ($N = 100000$ to $N = 500000$). Figure 13 shows the resulting performance values for OptiGrid and BIRCH. It is clear that both algorithms are have about the same dependency on the number of data points N(i.e. slightly superlinear in N) but OptiGrid is about 50 seconds faster than BIRCH. This results in a factor of about 2 for $N = 100000$ and a factor of about 1.4 for $N = 500000$). Note that the times measured for BIRCH do not contain the time needed for the clustering phase (phase 4 in BIRCH). The real performance advantage of OptiGrid is therefore even higher.

6 Conclusions

In this paper, we propose a new approach to clustering in large high-dimensional databases with noise. We discuss the weaknesses of previous approaches and provide some interesting insights into the problems of clustering high-dimensional data. We formally introduce multidimensional grids and show that the complexity of a correct clustering of high-dimensional data sets with noise is superlinear in the number data points. We then provide a new algorithm which is fully effective, i.e. is able to correctly determine all clusters, but at the same time is very efficient. The new approach is based on an optimal grid-partitioning which uses a number of projections to determine the optimal cutting planes. In the experiments, we show that our approach avoids all the effectiveness problems of previous approaches and in some cases, it is even more efficient compared to the most efficient known approach (BIRCH). Due to the promising results of the experimental evaluation, we believe that our method will have a noticeable impact on the way clustering of high-dimensional data will be done in the future. Our plans for future work include an application and fine tuning of our method for specific applications and an extension to other projections.

References

[AGG+98] R. Aggrawal, J. Gehrke, D. Gunopulos, P. Raghavan: *'Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications'*, Proc. ACM SIGMOD Int. Conf. on Managment of Data, pp. 94-105, 1998.

[BGRS99] Beyer K., Goldstein J., Ramakrishnan R., Shaft U.: *'When is nearest neighbor meaningful? '*, Proc. 7nd Int. Conf. on Database Theory (ICDT), 1999.

[DJSGM97] X. Daura, B. Jaun, D. Seebach, W.F. van Gunsteren A.E. Mark: *'Reversible peptide folding in solution by molecular dynamics simulation'*, submitted to Science (1997)

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.: *'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'*, Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1996.

[EKSX97] Ester M., Kriegel H.-P., Sander J., Xu X.: *'Density-Connected Sets and their Application for Trend Detection in Spatial Databases'*, Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1997.

[EKX95] Ester M., Kriegel H.-P., Xu X.: *'Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification'*, Proc. 4th Int. Symp. on Large Spatial Databases, 1995, in: Lecture Notes in Computer Science, Vol. 951, Springer, 1995, pp. 67-82.

[FL95] Faloutsos C., Lin K.D.: *'FastMap: A ast Algorithm for Indexing,Data-Mining and Visualisation of Traditional and Multimedia Datasets'*, Proc. ACM SIGMOD Int. Conf. on Managment of Data, 1995.

[FH75] Fukunaga K.,Hostler L.D.: *'The estimation of the gradient of a density function, with application in pattern recognation'*, IEEE Trans. Info. Thy. 1975, IT-21, 32-40

[HK98] Hinneburg A., Keim D.A.: *'An Efficient Approach to Clustering in Large Multimedia Databases with Noise'*, Proc. 4rd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1998.

[HK99] Hinneburg A., Keim D.A.: *'A General Approach to Clustering in Large Multimedia Databases with Noise'*, submitted to the Knowledge Discovery and Data Mining Journal.

[Hub85] Huber P.J.: *'Projection Pursuit'*, The Annals of Statistics, Vol. 13, No.2, 1985, pp. 435-474.

[Jag91] Jagadish H. V.: *'A Retrieval Technique for Similar Shapes'*, Proc. ACM SIGMOD Int.Conf. on Management of Data, 1991, pp. 208-217.

[Kuk92] Kukich K.: *'Techniques for Automatically Correcting Word in Text'*, ACM Computing Surveys, Vol. 24, No. 4, 1992, pp. 377-440.

[MG95] Methrotra R., Gray J. E.: *'Feature-Index-Based Similar Shape retrieval'*, Proc. of the 3rd Working Conf. on Visual Database Systems, 1995.

[MN89] Mehlhorn K., Naher S.: *'LEDA, a libary of efficient data types and algorithms'*, University of Saarland, FB Informatik, TR A 04/89.

[NH94] Ng R. T., Han J.: *'Efficient and Effective Clustering Methods for Spatial Data Mining'*, Proc. 20th Int. Conf. on Very Large Data Bases, Morgan Kaufmann, 1994, pp. 144-155.

[Sch96] Schikuta E.: *'Grid clustering: An efficient hierarchical clustering method for very large data sets'*, Proc. 13th Conf. on Pattern Recognition, Vol. 2, IEEE Computer Society Press, pp. 101-105.

[Sco92] Scott D.W.: *'Multivariate Density Estimation'*, Wiley & Sons 1992.

[Sil86] Silverman B.W.: *'Density Estimation'*, Chapman & Hall 1986.

[SCZ98] Sheikholeslami G., Chatterjee S., Zhang A.: *'WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases'*, Proc. 24th VLDB Conference, 1998.

[SH94] Shawney H., Hafner J.: *'Efficient Color Histogram Indexing'*, Proc. Int. Conf.on Image Processing, 1994, pp. 66-70.

[WW80] Wallace T., Wintz P.: *'An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors'*, Computer Graphics and Image Processing, Vol. 13, 1980.

[WYM97] Wang W., Yang J., Muntz R.: *'STING: A Statistical Information Grid Approach to Spatial Data Mining'*, Proc. 23rd Int. Conf. on Very Large Data Bases, Morgan Kaufmann, 1997.

[XEKS98] Xu X., Ester M., Kriegel H.-P., Sander J.: *'A Nonparametric Clustering Algorithm for Knowlege Discovery in Large Spatial Databases'*, will appear in Proc. IEEE Int. Conf. on Data Engineering, 1998, IEEE Computer Society Press.

[ZRL96] Zhang T., Ramakrishnan R., Linvy M.: *'BIRCH: An Efficient Data Clustering Method for very Large Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM Press, 1996, pp. 103-114.