The VLDB Journal (2000) 8: 289–304

The VLDB Journal

WaveCluster: a wavelet-based clustering approach for spatial data in very large databases

Gholamhosein Sheikholeslami, Surojit Chatterjee, Aidong Zhang

Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA

Edited by S. Christodoulakis. Received June 9, 1998 / Accepted July 8, 1999

Abstract. Many applications require the management of spatial data in a multidimensional feature space. Clustering large spatial databases is an important problem, which tries to find the densely populated regions in the feature space to be used in data mining, knowledge discovery, or efficient information retrieval. A good clustering approach should be efficient and detect clusters of arbitrary shape. It must be insensitive to the noise (outliers) and the order of input data. We propose *WaveCluster*, a novel clustering approach based on wavelet transforms, which satisfies all the above requirements. Using the multiresolution property of wavelet transforms, we can effectively identify arbitrarily shaped clusters at different degrees of detail. We also demonstrate that *WaveCluster* is highly efficient in terms of time complexity. Experimental results on very large datasets are presented, which show the efficiency and effectiveness of the proposed approach compared to the other recent clustering methods.

1 Introduction

Following the current research achievements [JM95, COM95], the visual content in an image may be represented by a set of features such as texture, color and shape. In a database, the features of an image can be represented by a set of numerical numbers, termed a feature vector. Various dimensions of feature vectors are used for content-based retrieval. We use the term *spatial data* to refer to those data which are 2D and 3D points, polygons, and points in some d-dimensional feature space [EKSX98]. In this paper we explore a *data-clustering* method in the spatial data-mining problem. Spatial data mining is the discovery of interesting characteristics and patterns that may exist in large spatial databases. Usually the spatial relationships are implicit in nature. Because of the huge amounts of spatial data that may be obtained from satellite images, medical equipments, geographic information systems (GIS), image database exploration, etc., it is expensive and unrealistic for the users to examine spatial data in detail. Spatial data mining aims to automate the process of understanding spatial data by representing the data in a concise manner and reorganizing spatial databases to accommodate data semantics. It can be used in many applications such as seismology (grouping earthquakes clustered along seismic faults), minefield detection (grouping mines in a minefield), and astronomy (grouping stars in galaxies) [AF97, BR95].

The aim of *data-clustering* methods is to group the objects in spatial databases into meaningful subclasses. Due to the huge amount of spatial data, an important challenge for clustering algorithms is to achieve good time efficiency. Also, due to the diverse nature and characteristics of the sources of the spatial data, the clusters may be of arbitrary shapes. They may be nested within one another, may have holes inside, or may possess concave shapes. A good clustering algorithm should be able to identify clusters irrespective of their shapes or relative positions. Another important issue is the handling of noise. Noise objects (outliers) refer to the objects which are not contained in any cluster and should be discarded during the mining process. The results of a good clustering approach should not become affected by the different ordering of input data and should produce the same clusters. In other words, it should be order-insensitive with respect to input data.

The complexity and enormous amount of spatial data may hinder the user from obtaining any knowledge about the number of clusters present. Thus, clustering algorithms should not assume to have the input of the number of clusters present in the spatial domain. To provide the user maximum effectiveness, clustering algorithms should classify spatial data at different levels of detail. For example, in an image database, the user may pose queries like whether a particular image is of type agricultural or residential. Suppose the system identifies that the image is of agricultural category. The user may be just satisfied with this broad classification. Again, the user may enquire about the actual type of the crop that the image shows. This requires clustering at hierarchical levels of coarseness, which we call the *multiresolution* property.

In this paper, we propose a spatial data-mining method, termed *WaveCluster*. We consider the spatial data as multidimensional signals and apply signal-processing techniques

– wavelet transforms to convert the spatial data into the frequency domain [SCZ98]. In wavelet transform, convolution with an appropriate kernel function results in a transformed space where the natural clusters in the data become more distinguishable. We then identify the clusters by finding the dense regions in the transformed domain. WaveCluster conforms with all the requirements of good clustering algorithms as discussed above. It can handle any large spatial datasets efficiently. It discovers clusters of any arbitrary shape and successfully handles noise, and it is totally insensitive to the ordering of the input data. Also, because of the signal-processing techniques applied, the *multiresolution* property is attributed naturally to WaveCluster. To our knowledge, no method currently exists which exploits these properties of wavelet transform in the clustering problem in spatial data mining. It should be noted that use of WaveCluster is not limited only to the spatial data, and it is applicable to any collection of attributes with ordered numerical values.

The rest of the paper is organized as follows. Section 2 formalizes the problem and defines its scope. We discuss the related work in Sect. 3. In Sect. 4, we present the motivation behind using signal-processing techniques for clustering large spatial databases. This is followed by a brief introduction on wavelet transform. Section 5 discusses our clustering method WaveCluster and analyzes its complexity. In Sect. 6, we present the experimental evaluation of the effectiveness and efficiency of WaveCluster using very large datasets. Finally in Sect. 7, concluding remarks are offered.

2 Problem formalization

Following the definition of Agrawal et al. [AGGR98], let $A = A_1, A_2, \ldots, A_d$ be a set of bounded, totally ordered domains and $S = A_1 \times A_2 \times \ldots \times A_d$ be a d-dimensional numerical space or feature space. $A_1, \ldots, A_d$ are referred as dimensions of S. The input dataset is a set of d-dimensional points $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_N}\}$, where $\mathbf{o_i} = \langle o_{i1}, o_{i2}, \ldots, o_{id} \rangle$, $1 \leq i \leq N$. The j-th component of $\mathbf{o_i}$ is drawn from domain A_j.

We first partition the original feature space into nonoverlapping hyper-rectangles, which we call *cells*. The cells are obtained by segmenting every dimension A_i into m_i number of intervals. Each cell $\mathbf{c_i}$ is the intersection of one interval from each dimension. It has the form $\langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$, where $c_{ij} = [l_{ij}, h_{ij})$ is the right open interval in the partitioning of A_j. Each cell $\mathbf{c_i}$ has a list of statistical parameters $\mathbf{c_i} \cdot \mathbf{param}$ associated with it.

We say that a point $\mathbf{o_k} = \langle o_{k1}, \ldots, o_{kd} \rangle$ is contained in a cell $\mathbf{c_i}$ if $l_{ij} \leq o_{ki} < h_{ij}$ for $1 \leq j \leq d$. The list $\mathbf{c_i} \cdot \mathbf{param}$ keeps track of the statistical properties such as aggregation, mean, variance, and the probability distribution of the data points contained in the cell $\mathbf{c_i}$. In general, in grid-based approaches the containment relations are discovered by a single pass through the dataset and appropriate statistical parameters are computed. Each cell has information about the density of the data contained in the cell. Thus, the collection of $\mathbf{c_i} \cdot \mathbf{param}$ summarizes the dataset.

We choose the number of points contained in each cell as the only statistic to be used. That is, we use $\mathbf{c_i} \cdot count$ to be the $\mathbf{c_i} \cdot \mathbf{param}$. In our approach, we apply wavelet transform on $\mathbf{c_i} \cdot count$ values. We define the *transformed space* as the set of cells after wavelet transformation on the count values of the cells in the quantized space.

In our approach, we take an all point stand on defining clusters, i.e., we consider that all the points within a cluster are representatives of the cluster. We introduce the following definitions to be used for the rest of the paper.

Definition 1. *(Empty cell) A cell in the quantized space with 0 count value is called an empty cell.*

Definition 2. *(Nonempty cell) A cell in the quantized space with nonzero count value is called a nonempty cell.*

Definition 3. *(Significant cell) A cell is a significant cell if its count field*[1] *in the transformed space is above a certain threshold τ.*

Definition 4. *(ϵ-neighbor) A cell $\mathbf{c_1}$ is an ϵ-neighbor of cell $\mathbf{c_2}$ if either both are significant cells (in transformed space) or nonempty cells (in quantized space) and $D(\mathbf{c_1}, \mathbf{c_2}) \leq \epsilon$, where D is an appropriate distance metric and $\epsilon > 0$.*

We can extend the definition of ϵ-neighborhood to k-ϵ-neighborhood.

Definition 5. *(k-ϵ-neighbor) A cell $\mathbf{c_1}$ is a k-ϵ-neighbor of a cell $\mathbf{c_2}$ if both are significant cells (in transformed space) or both are nonempty cells (in quantized space) and if $\mathbf{c_1}$ is one of the k prespecified ϵ-neighbors of $\mathbf{c_2}$.*

Definition 6. *(k-connected) Two cells $\mathbf{c_1}$ and $\mathbf{c_2}$ are k-connected if there is a sequence of cells $\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_j}$ such that $\mathbf{p_1} = \mathbf{c_1}$ and $\mathbf{p_j} = \mathbf{c_2}$ and $\mathbf{p_{i+1}}$ is a k-ϵ-neighbor of $\mathbf{p_i}$, $1 \leq i \leq j$.*

Definition 7. *(Cluster) A cluster $\mathscr{C}$ is a set of significant cells $\{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_m}\}$ which are k-connected in the transformed space.*

Given a set of N database points $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_N}\}$ in the d-dimensional feature space S, our goal is to find the clusters as defined above. After clustering, each cell in the feature space S will have a label indicating the cluster that it belongs to. In this paper, we propose WaveCluster to cluster very large databases with low number of dimensions, that is, we assume that N is very large and d is low.

3 Related work

We can categorize the clustering algorithms into four main groups: partitioning algorithms, hierarchical algorithms, density-based algorithms and grid-based algorithms.

3.1 Partitioning algorithms

Partitioning algorithms construct a partition of a database of N objects into a set of K clusters. Usually, they start

[1] In case we keep more elaborate information about each cell, then we can specify a range of values for each of the parameters of each cell for determining whether it is a significant cell.

with an initial partition and then use an iterative control strategy to optimize an objective function. There are mainly two approaches: i) k-means algorithm, where each cluster is represented by the center of gravity of the cluster; ii) k-medoid algorithm, where each cluster is represented by one of the objects of the cluster located near the center.

PAM [KR90] uses a k-medoid method to identify the clusters. PAM selects K objects arbitrarily as medoids and swaps with other objects until all K objects qualify as medoids. PAM compares an object with an entire dataset to find a medoid; thus, it has a slow processing time, $O(K(N-K))^2$. CLARA (Clustering LARge Applications) [KR90] draws a sample of the dataset, applies PAM on the sample, and finds the medoids of the sample.

Ng and Han introduced CLARANS (Clustering Large Applications based on RANdomized Search), which is an improved k-medoid method [NH94]. This is the first method that introduces clustering techniques into spatial data-mining problems and overcomes most of the disadvantages of traditional clustering methods on large datasets. Although CLARANS is faster than PAM, but still slow and, as mentioned in [WYM97], its computational complexity is Ω (KN^2). Moreover, because of its randomized approach, for large values of N, quality of results cannot be guaranteed.

In general, k-medoid methods do not present enough spatial information when the cluster structures are complex.

3.2 Hierarchical algorithms

Hierarchical algorithms create a hierarchical decomposition of the the database. The hierarchical decomposition can be represented as a *dendrogram* [Gor81]. The algorithm iteratively splits the database into smaller subsets until some termination condition is satisfied. Hierarchical algorithms do not need K as an input parameter, which is an obvious advantage over partitioning algorithms. The disadvantage is that the termination condition has to be specified.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [ZRL96] uses a hierarchical data structure called CF-tree for incrementally and dynamically clustering the incoming data points. CF-tree is a height-balanced tree which stores the clustering features. BIRCH tries to produce the best clusters with the available resources. They consider that the amount of available memory is limited (typically much smaller than the dataset size) and want to minimize the time required for I/O. In BIRCH, a single scan of the dataset yields a good clustering, and one or more additional passes can (optionally) be used to improve the quality further. So, the computational complexity of BIRCH is $O(N)$. BIRCH is also the first clustering algorithm to handle noise [ZRL96]. Since each node in a CF-tree can only hold a limited number of entries due to its size, it does not always correspond to a natural cluster [ZRL96]. Moreover, for different orders of the same input data, it may generate different clusters. In other words, it is order-sensitive. In addition, as our experimental results showed, if the clusters are "spherical" or convex in shape, BIRCH performs well; however, for other shapes it does not do as well. This is because it uses the notion of radius or diameter to control the boundary of a cluster.

3.3 Density-based algorithms

Pauwels et al. proposed an unsupervized clustering algorithm to locate clusters by constructing a density function that reflects the spatial distribution of the data points [PFG97]. They modified the nonparametric density estimation problem in two ways. Firstly, they use cross-validation to select the appropriate width of the convolution kernel. Secondly, they use difference-of-gaussians *(DOGs)* that allows for better clustering and frees the need to choose an arbitrary cut-off threshold. Their method can find arbitrary shape clusters and does not make any assumptions about the underlying data distribution. They have successfully applied the algorithm to color segmentation problems. This method is computationally very expensive [PFG97]. So it can make the method impractical for very large databases.

Ester et al. presented a clustering algorithm DBSCAN relying on a density-based notion of clusters. It is designed to discover clusters of arbitrary shapes [EKSX96]. The key idea in DBSCAN is that, for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points, i.e., the density in the neighborhood has to exceed some threshold. DBSCAN can separate the noise (outliers) and discover clusters of arbitrary shape. It uses R*-tree to achieve better performance. But the average runtime complexity of DBSCAN is $O(NlogN)$.

3.4 Grid-based algorithms

Recently a number of algorithms have been presented which quantize the space into a finite number of cells and then do all operations on the quantized space. The main characteristic of these approaches is their fast processing time, which is typically independent of the number of data objects. They depend only on the number of cells in each dimension in the quantized space.

Wang et al. propose a STatistical INformation Grid-based method (STING) for spatial data mining [WYM97]. They divide the spatial area into rectangular cells using a hierarchical structure. They store the statistical parameters (such as mean, variance, minimum, maximum, and type of distribution) of each numerical feature of the objects within cells. STING goes through the dataset once to compute the statistical parameters of the cells; hence, the time complexity of generating clusters is $O(N)$. The other previously mentioned clustering approaches do not explain if (or how) the clustering information is used to search for queries, or how a new object is assigned to the clusters. In STING, the hierarchical representation of grid cells is used to process such cases. After generating the hierarchical structure, the response time for a query would be $O(K)$, where K is the number of grid cells at the lowest level [WYM97]. Usually $K << N$, which makes this method fast. However, in their hierarchy, they do not consider the spatial relationship between the children and their neighboring cells to construct the parent cell. This might be the reason for the *isothetic* shape of resulting clusters, that is, all the cluster boundaries are either horizontal or vertical, and no diagonal boundary is detected. It lowers the quality and accuracy of clusters, despite the fast processing time of this approach.

Xu et al. proposed DBCLASD (Distribution Based Clustering of LArge Spatial Databases) [XMKS98]. DBCLASD assumes that the points inside a cluster are uniformly distributed. For each point in the cluster, the nearest point which is not inside the cluster is detected. Then it defines a *nearest neighbor distance set* as the set of all distances between each point in the cluster and its nearest point outside the cluster. Then it defines a *cluster* to be a nearest neighbor distance set that has the expected distribution with a required confidence level. DBCLASD incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distribution. DBCLASD is able to find arbitrarily shaped clusters. Furthermore, DBCLASD does not require input parameters to do the clustering. The experimental results presented by Xu et al. shows that it is slower than DBSCAN which has a complexity of $O(NlogN)$. Also, it assumes that points inside a cluster are uniformly distributed, which may not be the case in many applications.

We propose WaveCluster, which is a grid-based approach. The proposed approach is very efficient, especially for very large databases. The computational complexity of detecting clusters in our method is $O(N)$. The results are not affected by noise and the method is not sensitive to the order of input objects to be processed. WaveCluster is well capable of finding arbitrary-shape clusters with complex structures such as concave or nested clusters at different scales, and does not assume any specific shape for the clusters. A priori knowledge about the exact number of clusters is not required in WaveCluster. However, an estimation of expected number of clusters helps in choosing the appropriate resolution of clusters.

4 Relating spatial data to multidimensional signals

In this section, we discuss the relationship between spatial data and multidimensional signals, and show how to use wavelet transforms to illustrate the inherent relationships in spatial data.

4.1 Spatial data versus multidimensional signals

The primary motivation for applying signal-processing primitives to spatial databases comes from the observation that the multidimensional spatial data objects can be represented in a d-dimensional *feature space*. The numerical attributes of a spatial object can be represented by a *feature vector*, where each element of the vector corresponds to one numerical attribute, or *feature*. These feature vectors of the spatial data can be represented in the spatial area, which is termed *feature space*, where each dimension of the feature space corresponds to one of the features (numerical attributes). For an object with d numerical attributes, the feature vector will be one point in the d-dimensional feature space. The feature space is usually not uniformly occupied by the feature vectors. Clustering the data identifies the sparse and the dense places, and hence discovers the overall distribution of patterns of the feature vectors.

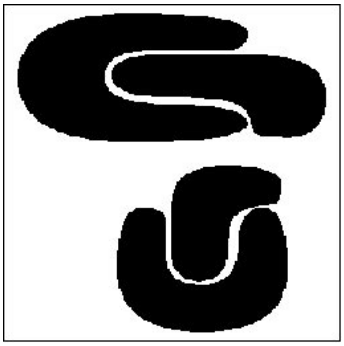


Fig. 1. A sample 2D feature space

4.2 Wavelet-based clustering

We propose to look at the feature space from a signal-processing perspective. The collection of objects in the feature space composes a d-dimensional signal. The high-frequency parts of the signal correspond to the regions of the feature space where there is a rapid change in the distribution of objects, that is, the boundaries of clusters. The low-frequency parts of the d-dimensional signal which have high amplitude correspond to the areas of the feature space where the objects are concentrated, in other words, the clusters themselves. For example, Fig. 1 shows a 2D feature space, where the 2D data points have formed four clusters. Note that Fig. 1 and also the figures in Sect. 5 are the *visualizations* of the 2D feature spaces, and each point in the images represents the feature values of one object in the spatial datasets. Each row or column can be considered as a 1D signal, so the whole feature space will be a 2D signal. Boundaries and edges of the clusters constitute the high-frequency parts of this 2D signal, whereas the clusters themselves, correspond to the parts of the signal which have low frequency with high amplitude. When the number of objects is high, we can apply signal-processing techniques to find the high-frequency and low-frequency parts of the d-dimensional signal representing the feature space, resulting in detecting the clusters.

Wavelet transform is a signal-processing technique that decomposes a signal into different frequency subbands (for example, high-frequency subband and low-frequency subband). The wavelet model can be generalized to d-dimensional signals in which a 1D transform can be applied multiple times. Methods have been used to compress data [HJS94], or to extract features from signals (images) using wavelet transform [SC94, JFS95, SZ97, SZB97]. For each object, the extracted features form a feature vector that can be represented by a point in the d-dimensional feature space. A spatial database will be the collection of such points. Wavelet transform has been applied on the objects to generate the feature vectors (feature space). The key idea in our proposed approach is to apply wavelet transform on the *feature space* (instead of the objects themselves) to find the dense regions in the feature space, which are the clusters. The next subsection discusses the strategy and motivation of using wavelet transform on d-dimensional feature spaces.

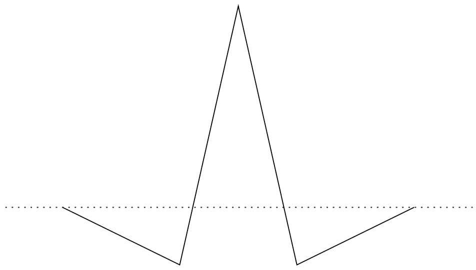


Fig. 2. Cohen-Daubechies-Feauveau (2,2) biorthogonal wavelet

4.3 Applying wavelet transform

Wavelet transform is a type of signal representation that can give the frequency content of the signal at a particular instant of time by filtering. A 1D signal s can be filtered by convolving the filter coefficients c_k with the signal values:

$$\hat{s}_i = \sum_{k=1}^{M} c_k s_{i+k-\frac{M}{2}},$$

where M is the number of coefficients in the filter and $\hat{s}$ is the result of convolution [HJS94]. Wavelet transform provides us with a set of attractive filters. For example, Fig. 2 shows the Cohen-Daubechies-Feauveau (2,2) biorthogonal wavelet.

The motivation for using wavelet transform and thereby finding connected components in the transformed space is drawn from the following observations.

- **Unsupervised clustering.** The *hat-shape* filters emphasize regions where points cluster, but simultaneously tend to suppress weaker information in their boundary. Intuitively, dense regions in the original feature space act as *attractors* for the nearby points and at the same time as *inhibitors* for the points that are not close enough. This means clusters in the data and clear regions around them automatically stand out, so that they become more distinct [PFG97]. It makes finding the connected components in the transformed space easier than in the original feature space, because the dense regions in the feature space will be more salient. Figure 3a shows an example of a feature space before and after transform. In this case, we have used Cohen-Daubechies-Feauveau (2,2) biorthogonal transform. Two cluster centers were first placed in a 2D feature space and then 500,000 points were generated around them following bivariate normal distribution. Then 25,000 uniformly distributed random noise points were added to the data to check the effect of applying wavelet transform on them. As the figure shows, the clusters in the transformed space are more salient and thus easier to be found.
- **Effective removal of noise objects.** Noise objects are the objects that do not belong to any of the clusters, and usually their presence causes problems for the current clustering methods. Applying wavelet transform removes the noise in the original feature space, resulting in more accurate clusters. As we will show, we take advantage of low-pass filters used in the wavelet transform to automatically remove the noise. Figure 3 shows that majority of the noise objects in the original space are removed after the transformation.
- **Multiresolution.** The multiresolution property of wavelet transform can help in detecting the clusters at different levels of detail. As will be shown later, wavelet transform provides multiple levels of decompositions, which results in clusters at different scales from fine to coarse. The appropriate scale for choosing clusters can be decided based on the user's needs.
- **Cost efficiency.** Since applying wavelet transform is very fast, it makes our approach cost-effective. As will be shown in the experiments, clustering very large datasets takes only a few seconds. Using parallel processing, we can obtain even faster responses.

Applying wavelet transform on a signal decomposes it into different frequency subbands [Mal89a]. We now briefly review wavelet-based multiresolution decomposition. More details can be found in Mallat's paper [Mal89b]. To have multiresolution representation of signals, we can use discrete wavelet transform. We can compute a coarser approximation of the 1D input signal S_0 by convolving it with the low-pass filter $\tilde{H}$ and downsampling the signal by 2 [Mal89b]. By downsampling, we mean skipping every other signal sample (For example, one row in a 2D feature space). All the discrete approximations S_j, $1 < j < J$ (J is the maximum possible scale), can thus be computed from S_0 by repeating this process. Resolution becomes coarser with increasing j. For example, the third approximation of S_0 (that is S_3) is coarser than the second approximation S_2. Figure 4 illustrates the method.

We can extract the difference of information between the approximation of signal at scale $j-1$ and j. D_j denotes this difference of information and is called *detail signal* at the scale j. We can compute the detail signal D_j by convolving S_{j-1} with the high-pass filter $\tilde{G}$ and returning every other sample of output. The wavelet representation of a discrete signal S_0 can therefore be computed by successively decomposing S_j into S_{j+1} and D_{j+1} for $0 \leq j < J$. This representation provides information about signal approximation and detail signals at different scales.

We can easily generalize the wavelet model to 2D feature space, in which we can apply two separate 1D transforms [HJS94]. We can represent a 2D feature space as an image where each pixel of image corresponds to one cell in the feature space. The 2D feature space (image) is first convolved along the horizontal (x) dimension, resulting in a low-pass image L and a high-pass image H. We then downsample each of the convolved images in the x dimension by 2. Both L and H are then convolved along the vertical (y) dimension, resulting in four subimages: LL, LH, HL, and HH. Once again, we downsample the subimages by 2, this time along the y dimension. The 2D convolution decomposes an image into an *average signal* (LL) and three *detail signals* which are directionally sensitive: LH emphasizes the horizontal image features, HL the vertical features, and HH the diagonal features.

Figure 5 shows the wavelet representation of the image in Fig. 1 at three scales. At each level, subband LL (wavelet approximation of original image) is shown in the upper left quadrant. Subband LH (horizontal edges) is shown in the

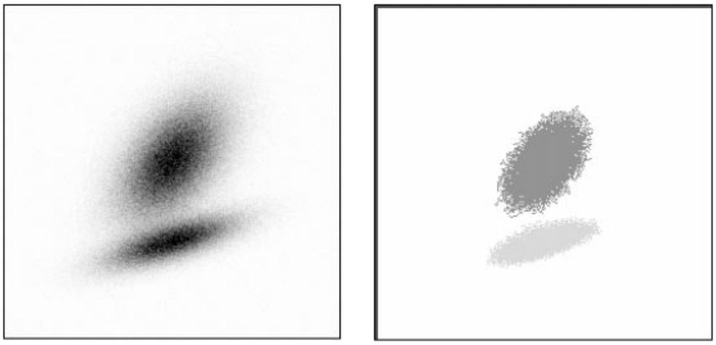


Fig. 3a. Original feature space. **b** Transformed space

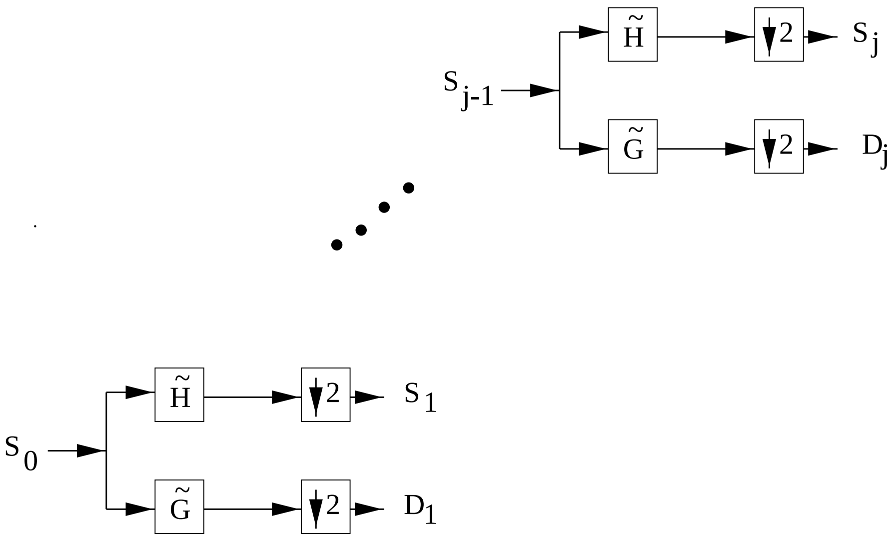


Fig. 4. Block diagram of multiresolution wavelet transform

upper right quadrant, subband HL (vertical edges) is displayed in the lower left quadrant, and subband HH (corners) is in the lower right quadrant.

The above wavelet model can similarly be generalized for d-dimensional feature space, where 1D wavelet transform will be applied d times. As mentioned earlier, we apply wavelet transform on the feature vectors of objects. At different scales, it decomposes the original feature space into an approximation, or *average subband (feature space)*, which has information about content of clusters, and *detail subbands (feature spaces)*, which have information about boundaries of clusters. The next section describes how we use this information to detect the clusters.

5 WaveCluster

In this section, we introduce our proposed algorithm and discuss its properties. The time complexity analysis of the algorithm is then presented.

5.1 Algorithm

Given a set of spatial objects $\mathbf{o_i}, 1 \leq i \leq N$, the goal of the algorithm is to detect clusters and assign labels to the objects based on the cluster that they belong to. The main idea in WaveCluster is to transform the original feature space by applying wavelet transform and then find the dense regions in the new space. It yields sets of clusters at different resolutions and scales, which can be chosen based on the user's needs. The main steps of WaveCluster are shown in Algorithm 1.

Algorithm 1

Input: Multidimensional data objects' feature vectors

Output: clustered objects

1. Quantize feature space, then assign objects to the cells.
2. Apply wavelet transform on the quantized feature space.
3. Find the connected components (clusters) in the subbands of transformed feature space, at different levels.
4. Assign labels to the cells.
5. Make the lookup table.
6. Map the objects to the clusters.

5.1.1 Quantization

The first step of the WaveCluster algorithm is to quantize the feature space, where each dimension A_i in the d-dimensional

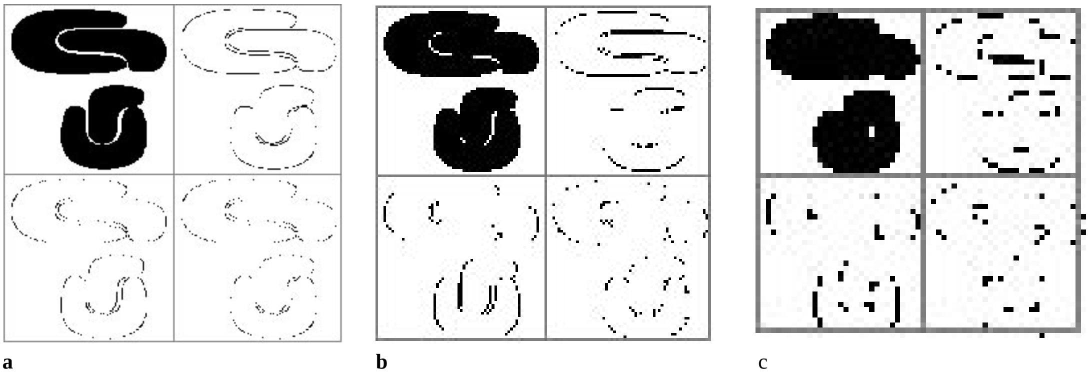


Fig. 5. Multiresolution wavelet representation of the feature space in Fig. 1 at **a** scale 1; **b** scale 2; **c** scale 3

feature space will be divided into m_i intervals. If we assume that m_i is equal to m for all the dimensions, there would be m^d cells in the feature space. Then, the corresponding cell for the objects will be determined based on their feature values. A cell $\mathbf{c_i} = \langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$ contains an object $\mathbf{o_k} = \langle o_{k1}, \ldots, o_{kd} \rangle$, if

$$l_{ij} \leq o_{ki} < h_{ij}, \qquad 1 \leq j \leq d.$$

We may recall that $c_{ij} = [l_{ij}, h_{ij})$ is the right open interval in the partitioning of A_j. For each cell, we count the number of objects contained in it to represent the aggregation of the objects. The number (or size) of these cells and the aggregation information in each cell are important issues that affect the performance of clustering. We discuss these quantization issues in the next section. Because of the multiresolution property of wavelet transform, we consider different cell sizes at different scales of transform.

5.1.2 Transforming and clustering

In the second step, discrete wavelet transform will be applied on the quantized feature space. Applying wavelet transform on the cells in $\{\mathbf{c_j} : 1 \leq j \leq \mathscr{J}\}$ results in a new feature space, and hence new cells $\{\mathbf{t_k} : 1 \leq k \leq \mathscr{K}\}$. Given the set of cells $\{\mathbf{t_k} : 1 \leq k \leq \mathscr{K}\}$, WaveCluster detects the connected components in the transformed feature space. Each connected component is a set of cells in $\{\mathbf{t_k} : 1 \leq k \leq \mathscr{K}\}$ and is considered as a cluster. Corresponding to each resolution r of wavelet transform, there would be a set of clusters $\mathscr{C}_r$, where, usually at the coarser resolutions, the number of clusters is less. In the experiments, we applied each of the three-level wavelet transforms Daubechies, Cohen-Daubechies-Feauveau ((4,2) and (2,2)) [Vai93, SN96, URB97]. Average subbands (feature spaces) give approximations of the original feature space at different scales, which help in finding clusters at different levels of details. For example, as shown in Fig. 5, for a 2D feature space, the subbands LL show the clusters at different scales.

We use the algorithm in [Hor88] to find the connected components in the 2D feature space (image). The same concept can be generalized for higher dimensions. In our implementation, we have $k = 8$ and $\epsilon = \sqrt{2}$ for k-ϵ-neighborhood as defined in Sect. 2. That is, a significant cell a in the transformed feature space is k-ϵ-neighbor of another cell b if a lies within one of the eight grid cells surrounding cell b. The connected component analysis consists of scanning through the image once to find all the connected components, and then equivalence analysis to relabel the components. This takes care of components with holes and concave shapes. There are many well-known algorithms for finding connected components in images and we used the one mentioned in [Hor88] for our purpose. Figure 9 in Sect. 5 shows the clusters that WaveCluster found at each scale in different colors.

5.1.3 Label cells and make lookup table

Each cluster w, $w \in \mathscr{C}_r$, will have a cluster number w_n. In the fourth step of the algorithm, WaveCluster labels the cells in each cluster in the transformed feature space with its cluster number. That is,

$$\forall w \ \ \forall \mathbf{t_k},\ \mathbf{t_k} \in w \Longrightarrow l_{t_k} = w_n, \quad w \in \mathscr{C}_r,$$

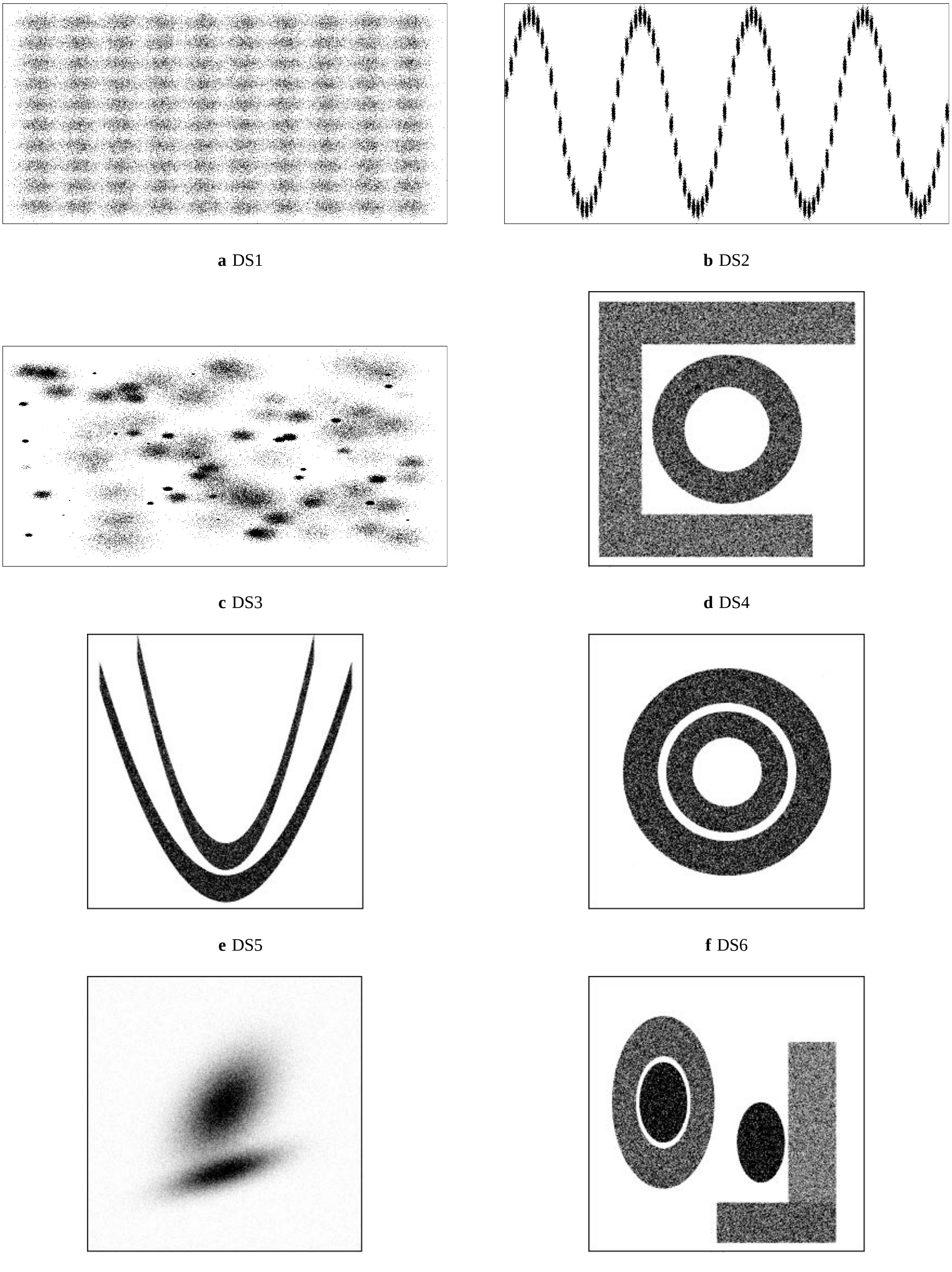
where l_{t_k} is the label of the cell $\mathbf{t_k}$. The clusters that are found are in the transformed feature space and are based on wavelet coefficients. Thus, they cannot be directly used to define the clusters in the original feature space. WaveCluster makes a lookup table LT to map the cells in the transformed feature space to the cells in the original feature space. Each entry in the table specifies the relationship between one cell in the transformed feature space and the corresponding cell(s) of the original feature space. So the label of each cell in the original feature space can be easily determined. Finally, WaveCluster assigns the label of each cell in the feature space to all the objects whose feature vector is in that cell, and thus the clusters are determined. That is,

$$\forall w \ \ \forall \mathbf{c_j}, \ \ \forall \mathbf{o_i} \in \mathbf{c_j}, \qquad l_{o_i} = w_n, \quad w \in \mathscr{C}_r, \ \ 1 \leq i \leq N,$$

where l_{o_i} is the cluster label of object $\mathbf{o_i}$.

5.2 Properties of WaveCluster

When the objects are assigned to the cells of the quantized feature space at step 1 of the algorithm, the final content of the cells is independent of the order in which the objects are presented. The following steps of the algorithm will be performed on these cells. Hence, the algorithm will have the same results for any different order of input data, so it is order insensitive with respect to input objects. As will



a DS1 **b** DS2

c DS3 **d** DS4

e DS5 **f** DS6

g DS7 **h** DS8

Fig. 6a–h. Visualization of some of the datasets used in the experiments.

be formally and experimentally shown later, the required time for WaveCluster to detect the clusters is linear in terms of number of input data, and it cannot go below that, because all the data should be at least read. After reading the data, the processing time will be only a function of number of cells in the feature space. Thus, it makes WaveCluster very efficient, especially for very large numbers of objects. WaveCluster will be especially very efficient for the cases where the number of cells m and the number of feature space dimensions d are low. Minefield detection and some seismology applications are examples where we have low-dimensional (two dimensions) feature spaces.

All the grid-based approaches for clustering spatial data suffer from the *modifiable areal cell problem (MAUP)* first addressed by Openshaw in 1977, and Openshaw and Taylor in 1981 [Ope77, OT81]. The problem occurs in terms of scaling and aggregation. The problem of scaling is in selecting the appropriate size and number of cells to represent the spatial data. There are infinitely large numbers of ways in which the cells may be organized and their size be specified. Aggregation is the problem of summarizing the data *contained* in each cell. In our case, we use a simple accumulative approach where the number of the data points contained in a cell summarizes all information about the cell. But there might be other measures which characterize the data more appropriately. In his paper, Openshaw [Ope77] defines this problem mathematically and discusses some heuristics to solve the problem.

All the present grid-based algorithms suffer from these problems. In general, when the quantization value m is too low (very coarse quantization), more objects will be assigned to the same cell, and there is higher probability for the objects from different clusters to belong to the same cell. We call this case *under-quantization* problem. This results in merging of the two clusters and mislabeling their objects; thus, the quality of clustering decreases. In contrast, if the quantization value m is too high (very fine quantization), each object will be in a separate cell, which might be far from the other cells. We call this *over-quantization* problem. Over-quantization can result in many unnecessary small clusters (that might be later removed as noise) and does not find the real clusters; thus, it will also decrease the quality of clustering. Aggregation also plays a role in clustering and it depends on the kind of algorithm used for clustering. In STING, each cell maintains a list of statistical attributes, like number of objects in the cell, mean of values, standard deviation of values, min, max, type of distribution of the values in the cell [WYM97]. In CLIQUE proposed by Agrawal et al., each cell is classified as dense or not based on the count value in each cell [AGGR98]. But none of the methods discusses the problems regarding aggregation.

We argue that, in this context, scaling is an inherent problem in what a human user can call a cluster, in other words, the definition of cluster. As Openshaw and Taylor stated, it seems very unlikely that there will ever be either a purely statistical or mathematical solution for MAUP [OT81]. To have an optimal quantization, application domain information should be incorporated. Openshaw provided a geographical solution to scale and aggregation problems in region-building, partitioning, and spatial modeling [Ope77]. However, as he mentions, although his approach seems to work, and perhaps provides the only real solution to a complicated problem, it has its own weaknesses [OT81]. Quantization is a problem that *all* grid-based algorithms suffer from. However, while other existing grid-based clustering methods ignore this problem, WaveCluster has the advantage of producing clusters at multiple scales at the same time. This means that the results of WaveCluster implicitly reflect multiple quantizations of the feature space, resulting in multiple sets of clusters that can be selected based on the user's requirements.

We may use a heuristic-based approach to experimentally find a good quantization. We can start with very small size grid cells (over-quantized feature space) and try to find the clusters. Most likely, no clusters will be found at this step. We can then increase the size of cells and find the possible clusters. If no acceptable clusters are found, we repeat the process after enlarging the size of cells. This process can be continued until we obtain some acceptable clusters. At this phase, WaveCluster, using the multiresolution property of wavelet transform, can provide multiple sets of clusters at different scales. This approach to finding an appropriate quantization will increase the overall time to cluster the database. However, given the appropriate quantization, the required time complexity of WaveCluster will still be $O(N)$. Finding the suitable quantization is a common problem for all grid-based methods and this cost should be considered for all of them.

WaveCluster finds the connected components in the *average* subband (LL) of the wavelet transformed feature space, as the output clusters. As mentioned in Sect. 4.3, average subband is constructed by convolving the low-pass filter along each dimension and downsampling by two. So a wavelet-transformed cell will be affected by the content of cells in the neighborhood covered by the filter. It means that the spatial relationships between neighboring cells will be preserved. The algorithm to find the connected components labels each cell of feature space with respect to the cluster that it belongs to. The label of each cell is determined based on the labels of its neighboring cells [Hor88]. It does not make any assumptions about the shape of connected components and can find convex, concave, or nested connected components. Hence, WaveCluster can detect arbitrary shapes of clusters.

WaveCluster applies wavelet transform on the feature space to generate multiple decomposition levels. Each time we consider a new decomposition level, we ignore some details in the average subband and effectively increase the size of a cell's neighborhood whose spatial relationship is considered. This results in sets of clusters with different degrees of details after each decomposition level of wavelet transform. In other words, we will have multiresolution clusters at different scales, from fine to coarse. For example, in Sect. 6, Fig. 9 shows an example where a three-level wavelet transform is applied and the output clusters after each transform are presented. At scale 1, we have the four fine clusters, and at the next scale, two of those clusters are merged. At scale 3, we have only two coarse clusters representing original feature space. In our approach, a user does not have to know the exact number of clusters. However, a good estimation of the number of clusters helps in choosing the appropriate scale and the corresponding clusters. One of the effects of applying a low-pass filter on the feature space is the removal of noise. WaveCluster takes advantage of this property, and removes the noise from the feature space automatically. Figure 3a shows an example where about 25,000 noise objects are scattered in the feature space. After applying wavelet transform, the noise objects are removed, and thus WaveCluster can detect the clusters correctly.

5.3 Time complexity

Let N be the number of objects in the database, where N is a very large number. Assume the feature vectors of objects are d-dimensional, resulting in a d-dimensional feature space. As we mentioned in Sect. 2, the current version of WaveCluster is designed for the cases where N is very large and d is low. The time complexity of the first step of WaveCluster algorithm is $O(N)$, because it scans all the database objects and assigns them to the corresponding cells. Assuming m cells in each dimension of feature space, there would be $K = m^d$ cells. Complexity of applying wavelet transform on the quantized feature space (step 2) will be $O(ldK) = O(dK)$, where l is a small constant representing the length of the filter used in the wavelet transform. Since we assume that the value of d is low, we can consider it as a constant, thus $O(dK) = O(K)$. If we apply wavelet transform for T levels of decomposition, since for each level, we downsample the space by 2, for $d \geq 2$, the required time would be

$$O\left(K + \frac{K}{2^d} + \frac{K}{\left(2^d\right)^2} + \ldots + \frac{K}{\left(2^d\right)^T}\right)$$
$$= O\left(K \sum_{i=0}^{T} \frac{1}{\left(2^d\right)^i}\right) = O\left(K \sum_{i=0}^{T} \left(2^{-d}\right)^i\right)$$
$$= O\left(K \frac{1 - \left(2^{-d}\right)^{T+1}}{1 - 2^{-d}}\right) \leq O\left(\frac{4}{3}K\right) .$$

That means the cost to apply wavelet transform for multiple levels would be at most $O(\frac{4}{3}K)$. It shows that we can have multiresolution presentation of the clusters very cost-effectively. To find the connected components in the transformed feature space, the required time will be $O(cK) = O(K)$, where c is a small constant. Making the lookup table requires $O(K)$ time. After reading data objects, the processing of data is performed in steps 2 to 5 of the algorithm. Thus, the time complexity of processing data (without considering I/O) would, in fact, be $O(K)$, which is independent of the number of data objects (N). The time complexity of the last step of WaveCluster algorithm is $O(N)$. Since this algorithm is applied on very large databases with a low number of dimensions, we can assume that $N \geq K$. As an example, for a database with 1,000,000 objects where the number of dimensions d is less than or equal to six, and the number of intervals m is 10, this condition holds. Thus, based on this assumption, the overall time complexity of the algorithm will be $O(N)$. It should be noted that, because of the way that we find the connected components (and hence the clusters), the number of clusters does not affect the time complexity of WaveCluster. In other words, WaveCluster's time complexity is independent of the number of clusters.

During applying wavelet transform on each dimension of the feature space, the required operations for each feature space cell can be carried out independent of the other cells. Thus, using parallel processing can speed up transforming the feature space. The connected component analysis can also be speeded up using parallel processing [NS80, SV82]. Parallel processing algorithms will be especially useful when the number of cells m or the number of dimensions d is high. For a large number of dimensions, we may have $N < K = m^d$. For such cases, we can also perform principle component analysis [Sch92] to find the most important features and to reduce the number of dimensions to a value f such that $N > m^f$. We have provided another solution using a hash-based data structure for the cases when number of dimensions is high, which is presented in [YCSZ98].

6 Performance evaluation

In this section, we evaluate the performance of WaveCluster and demonstrate it's effectiveness on different types of distributions of data. Tests were done on synthetic datasets generated by us and also on datasets used to evaluate BIRCH [ZRL96]. We mainly compare our clustering results with BIRCH.

Synthetic dataset generation

For the experiments, we used the datasets generated by both our own synthetic generator and the ones used by [ZRL96]. In the dataset generation method described in [ZRL96], cluster centers are first placed at certain locations in the space. The data points of each cluster are generated according to a 2D normal distribution whose mean is the center and whose variance is specified. Datasets DS1, DS2 and DS3 are the same as used by [ZRL96]. They are shown in Fig. 6a–c respectively. Each dataset consists of $100,000$ points. The points in DS3 are randomly distributed, while those of DS1 and DS2 are distributed in a grid and sine curve pattern, respectively.

We designed our own synthetic dataset generator for performing further experiments. The data generator allows control over the structure, number of clusters, probability distribution, and size of the datasets. It also allows us to add different proportion of noise to the generated datasets. We generated 14 new datasets to perform experiments.

We generated DS4 by spreading points in 2D space following uniform random distribution in the shapes of rectangles and annular region. DS4 contains $228,828$ data objects spread in two clusters as shown in Fig. 6d. For generating dataset DS5, we spread points around two parabolas following uniform random distribution. Dataset DS5 has $250,000$ data objects, containing two concave clusters in the shape of parabolas.

Dataset DS6 was generated by spreading $275,429$ random data objects following uniform distribution in two concentric annulus regions. We randomly generate two floating-point numbers in the feature space, one for each dimension. We then check whether the data object defined by these two features falls in the annular region defined by the inner radius, center and the width. The parameters used for generating this dataset are shown in Table 1. The parameter r is the radius of the void circle inside the annulus, w is the width of the annulus, and x and y define the location of the center of the annulus.

We used a technique similar to one described in [ZRL96] to generate the dataset DS7. Two cluster centers are first placed on the 2D plane and then $500,000$ data objects are spread following 2D normal distribution around these

Table 1. Parameters for generating DS6

Parameters	r	w	x	y
Inner Circle	20.0	15.0	60.0	60.0
Outer Circle	40.0	20.0	60.0	60.0

Table 2. Parameters for generating DS7

Parameters	μ_x	μ_y	σ_x	σ_y	ρ
Cluster1	125.0	55.0	60.0	13.0	0.7
Cluster2	125.0	120.0	50.0	30.0	0.5

points. After that, $75,000$ (15%) random noise objects were added to the dataset, making the total number of data objects $575,000$. For the 2D normal distribution, we used the polar method proposed by Box et al. as described in [Knu98]. The dataset is shown in Fig. 6g. The parameters used for this are shown in Table 2, where μ_x and μ_y specify the mean in each dimension, i.e., the location of the cluster center, σ_x and σ_y specify the variance in each dimension and ρ specifies the correlation coefficient between the variables in each dimension.

Generation of dataset DS8 follows a combination of strategies used for generating DS6 and DS4. We create two concentric annular region, one filled circle and an "L-shaped" cluster. There is a total $252,869$ data objects in DS8.

We also had several other datasets to study certain characteristics of WaveCluster. One group of datasets was used to verify the sensitivity of processing time of WaveCluster with increasing number of clusters. To make a fair comparison, we made the total number of data objects the same, but varied the number of clusters in these datasets. Each dataset has $1,000,000$ data objects and $20,000$ noise objects. The number of clusters in these datasets range from 2 to 100. The clusters are either rectangles (following a uniform random distribution) or ellipsoids (following 2D normal random distribution, as described before). The results for these experiments are reported in Table 3. The generation of rectangular clusters closely follows the method described in [ZM97]. We also generated several noisy versions of the DS7 dataset to verify the noise removal property of WaveCluster. We added different proportions (5%, 10%, 15%, 20%, 25%) of noise to the original DS7 dataset to create these datasets. The number of objects in them are $525,000$, $550,000$, $575,000$, $600,000$, and $625,000$, respectively. The visualizations of these datasets and WaveCluster's results on them are presented in Fig. 10.

Clustering very large datasets

All the datasets used in the experiments contain typically more than 100,000 data points. DS1,DS2 and DS3 each has 100,000 data points. WaveCluster can successfully handle an arbitrarily large number of data points. Figure 7 shows WaveCluster's performance on DS1. Here, a map-coloring algorithm has been used to color the clusters. Neighboring clusters have different colors. But nonneighboring clusters might be allocated the same color. In Fig. 3, we showed the clustering results for a dataset with more than 500,000 objects.

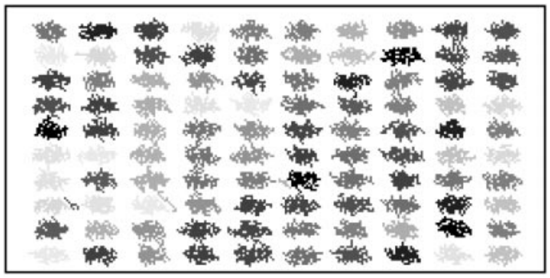


Fig. 7. WaveCluster on DS1

Clustering arbitrary shapes

As we mentioned earlier, spatial data-mining methods should be capable of handling any arbitrarily shaped clusters. Figure 6d presents the DS4 dataset. There are two arbitrarily shaped clusters in the original data. Figure 8a shows clustering of DS4 using WaveCluster and BIRCH. This result emphasizes effectiveness of the methods which do not assume the shape of the clusters a priori.

Clustering at different resolutions

WaveCluster has the remarkable property that it can be used to cluster at different granularities according to the user's requirement. Figure 9 displays the results of WaveCluster on DS8 (Fig. 6h). At scale 1, we have the four fine clusters, and at the next scale, two of those clusters are merged. At scale 3, we have only two coarse clusters representing original feature space. This illustrates how WaveCluster finds clusters at different degrees of detail. This property of WaveCluster provides the user with the flexibility to modify queries based on initial results.

Handling noise objects

WaveCluster is very effective in handling noise. The dataset presented in Fig. 3 has 500,000 objects in two clusters plus 25,000 (5%) noise objects. We generated new datasets from it, where 50,000, 75,000, 100,000, and 125,000 (10%, 15%, 20%, and 25%) uniformly distributed noise objects were added to datasets. The datasets and the corresponding clusters detected by WaveCluster are shown in Fig. 10. WaveCluster successfully removes all the random noise and produces the two intended clusters in all cases. Also, because the time complexity of the processing phase of WaveCluster is $O(K)$ (where K is the number of grid cells), the time taken to find the clusters in the noisy version of the data is the same as in the one without noise.

Clustering nested and concave patterns

WaveCluster can successfully cluster any complex pattern consisting of nested or concave clusters. From Fig. 6f (DS6)

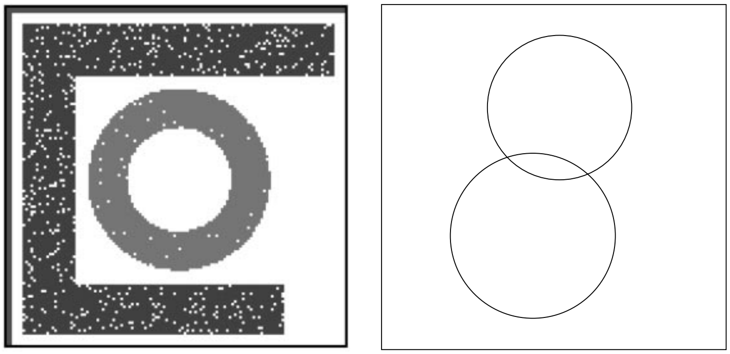


Fig. 8a,b. Clustering results on DS4: **a** WaveCluster, **b** BIRCH

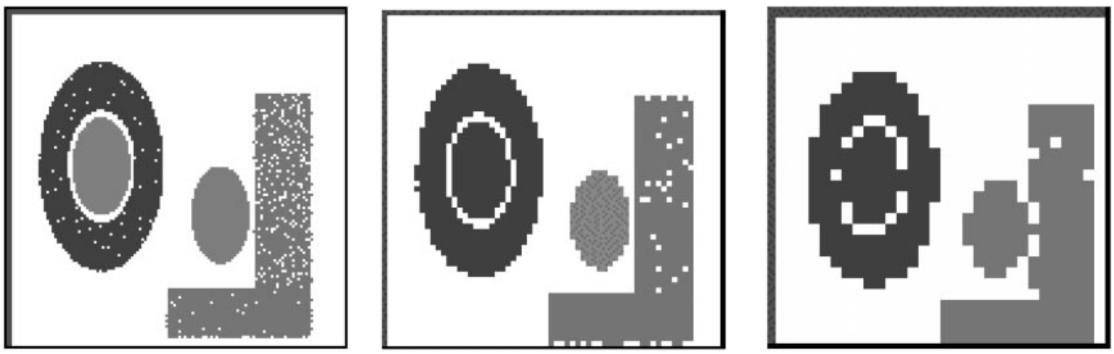


Fig. 9a–c. WaveCluster output clusters of DS8 at **a** scale 1; **b** scale 2; **c** scale 3

and Fig. 11a we see that WaveCluster's result is very accurate on nested clusters. Figure 11b shows BIRCH's result on the same dataset.

Figure 6g (DS5) shows an example of a concave shape data distribution. Figure 12a and b compare the clustering produced by WaveCluster and BIRCH. From these results, it is evident that WaveCluster is very powerful in handling any type of sophisticated patterns.

Comparing different number of clusters

We generated nine datasets, each having 1,000,000 data objects, and added 20,000 noise objects to them. These datasets have the same number of data objects (1,020,000), but have a different number of clusters ranging from 2 to 100 clusters. Table 3 summarizes the required quantization and processing time for these datasets. We applied Cohen-Daubechies-Feauveau (2,2) wavelet transform and used 256×256 quantization in these experiments. As this table shows, the number of clusters has no effect on the timing requirements of WaveCluster. It verifies our discussion in Sect. 5.3 that WaveCluster's time complexity is independent of the number of clusters.

Comparison of timing requirements

We now compare the timing requirements of WaveCluster, BIRCH, and CLARANS as shown in Tables 4 and 5. We ran BIRCH on all the datasets. CLARANS requires the information about all the database objects to be loaded into memory, and its run time is very large when there is a large number of objects. Thus, we were unable to run it. Based on the comparison of BIRCH and CLARANS presented in [ZRL96], we estimated the performance of CLARANS. Running code for DBSCAN and STING was not available; thus, we were not able to do experiments with it. We observe that on an average CLARANS is 22 times slower than BIRCH. We show the time requirements for quantization and processing separately for WaveCluster. All the experiments were carried out on a SUN SPARC workstation using 168 MHz UltraSparc CPU with SunOS operating system and 1024 MB memory. We applied Cohen-Daubechies-Feauveau (2,2) wavelet transform in the experiments reported in Table 5.

Table 3. Required time (in seconds) for different number of clusters with same number of points

Number of clusters	2	4	5	10	20	25	40	50	100
Quantization time	49.1	49.1	54.7	50.9	52.0	52.0	51.7	52.0	54.1
Processing time	2.1	2.1	2.1	2.2	2.15	2.3	2.1	2.1	2.2

Table 5 shows the average quantization time required in WaveCluster. It also presents the processing time when different number of grid cells are used in quantization. The values of m_1 and m_2 specify the number of cells in horizontal and vertical dimensions, respectively. The total required time to cluster using WaveCluster is the summation of pro-

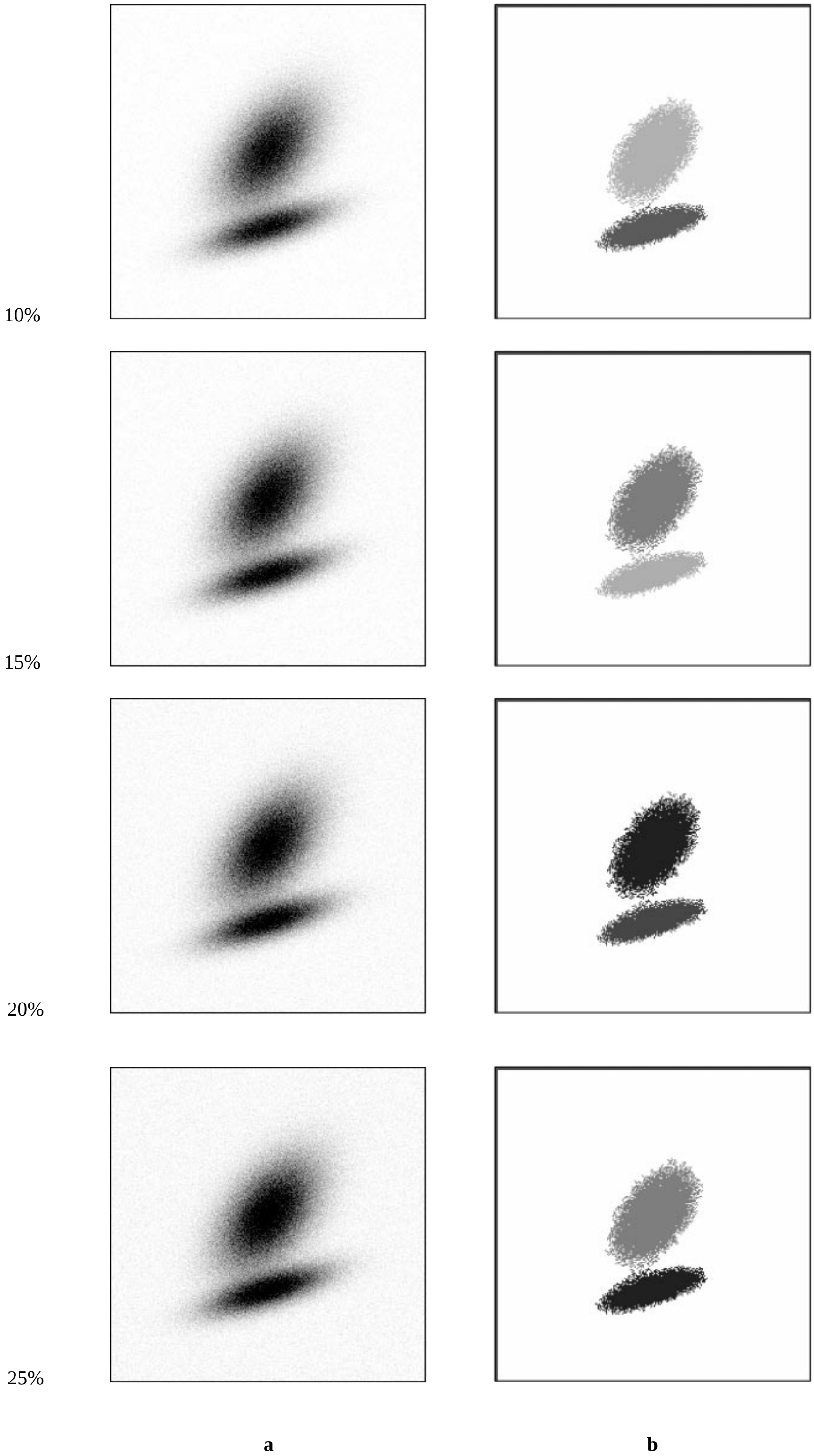
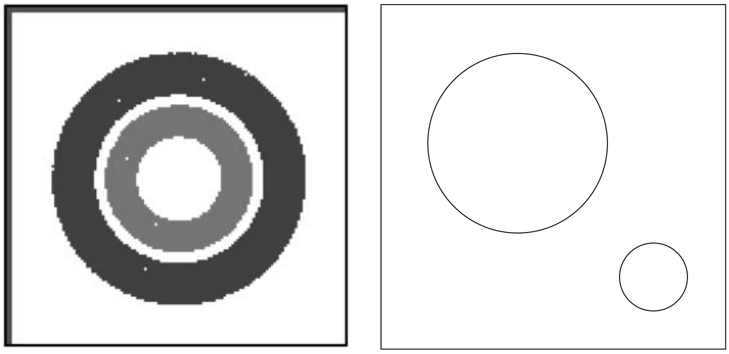


Fig. 10a,b. WaveCluster on datasets with different levels of noise: **a** noisy datasets; **b** clusters

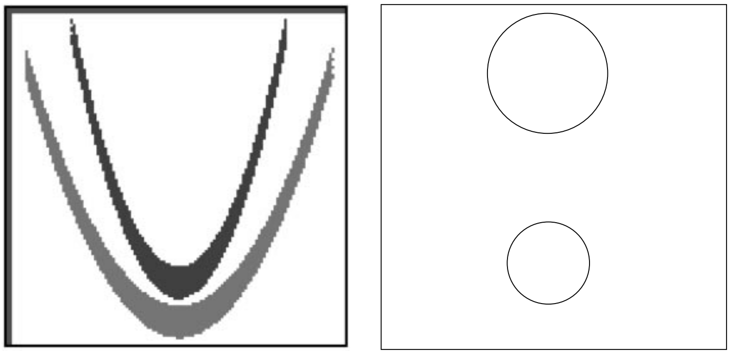
cessing and quantization time. We observe that WaveCluster outperforms BIRCH and CLARANS by a large margin when we use the finest quantization (512×1024), which takes the longest among the quantizations in our experiments. Even if we add up the processing time for all different quantizations, the total time would still be less than that of the other clustering methods.

The processing time of WaveCluster is almost independent of the distribution of the spatial objects, and most importantly it is even independent of number of objects present



a **b**

Fig. 11a. WaveCluster on DS6; **b** BIRCH on DS6



a **b**

Fig. 12a. WaveCluster on DS5; **b** BIRCH on DS5

Table 4. Required time (in seconds) for different datasets using CLARANS and BIRCH

Dataset	DS6	DS5	DS4	DS1	DS2	DS3
Number of data	275,429	250,000	228,828	100,000	100,000	100,000
CLARANS	2378.2	2376.0	2085.6	1232.0	1093.0	1089.4
BIRCH	108.1	108.0	94.8	56.0	49.7	49.5

in the space. As Table 5 shows, the time taken by WaveCluster is heavily dominated by the time to read the input data from disk. A faster method to do I/O will make the algorithm much faster. The experimental results demonstrate WaveCluster to be a stable and efficient clustering method.

As Table 5 shows, the processing time (without considering I/O) is not a function of the number of data objects. For datasets of different sizes, WaveCluster requires almost similar processing time (given the same quantization). As mentioned in Sect. 5.3, the time complexity of processing data is linear in terms of number of the feature space cells ($O(K)$). The timing results shown in Table 5 verify this property of WaveCluster. When we have less cells (coarser quantization), the required time is less. Quantization time includes the time to read the input data and assign them to the cells, and hence is a function of number of input data. That is, as shown in Table 5, the required quantization time for larger datasets is larger than that of smaller datasets.

Clustering at different quantizations

In Table 5 we showed how quantization affects the processing time, and thus the overall efficiency of WaveCluster. We now present our experimental results regarding the effect of quantization on the quality of clustering. We performed experiments on the dataset DS1 that has 100 clusters. Table 6 shows the number of clusters found by WaveCluster where different quantizations were used. In Sect. 5.2, we discussed the problems of scaling, aggregation, under-quantization, and over-quantization. When we used the fine quantization, 2048×4096, almost all the 100 were eliminated as noise (over-quantization). On the other hand, when the objects were quantized coarsely (under-quantization), (for example 32×64 or 64×128), most of the clusters were merged to each other, yielding low-quality results. When we used 256×512 quantization, almost all the 100 clusters were correctly detected and we obtained the best results. Table 6 shows that, for 1024×2048 quantization, WaveCluster also detects about 100 clusters. However, due to over-quantization and because of low density of objects at the border of clusters, most such border objects were not assigned to the clusters. Thus, for this case, the results were not satisfactory.

7 Conclusion

In this paper, we presented the clustering approach termed *WaveCluster*. This grid-based approach applies wavelet transform on the quantized feature space and then detects the

Table 5. Required time (in seconds) for different datasets using WaveCluster.

Dataset			DS6	DS5	DS4	DS1	DS2	DS3
Number of data			275,429	250,000	228,828	100,000	100,000	100,000
Processing time	m_1	m_2						
	512	1024	5.9	5.7	6.3	5.6	5.8	6.0
	512	512	3.5	3.5	3.4	3.8	3.4	3.3
	256	512	2.2	2.1	2.0	2.3	2.1	2.0
	256	256	1.4	1.5	1.5	1.5	1.4	1.4
	128	256	1.2	1.1	1.1	1.2	1.2	1.1
	128	128	0.9	1.0	1.0	1.0	0.9	1.0
	64	128	1.0	0.9	0.8	1.1	1.0	0.9
	64	64	0.9	0.9	0.9	0.9	0.9	0.8
Quantization time			11.7	11.0	9.7	5.4	5.6	5.5

Table 6. Number of clusters found for DS1 using different quantizations.

m_1	2048	1024	512	256	128	64	32
m_2	4096	2048	1024	512	256	128	64
Number of Clusters	1	110	203	105	48	13	3

dense regions in the transformed space. Applying wavelet transform makes the clusters more distinct and salient in the transformed space, and thus eases their detection. Using the multiresolution property of wavelet transform, WaveCluster can detect the clusters at different scales and levels of detail, which can be very useful in the user's applications. Moreover, applying wavelet transform removes the noise from the original feature space, and thus enables WaveCluster to handle them properly and find more accurate clusters. WaveCluster does not make any assumption about the shape of clusters and can successfully detect arbitrary-shape clusters such as concave or nested clusters. It is a very efficient method with a time complexity of $O(N)$, where N is the number of objects in the database, which makes it especially attractive for very large databases. WaveCluster is insensitive to the order of input data to be processed. Current clustering techniques do not address these issues sufficiently, although considerable work has been done in addressing each issue separately. Our experimental results demonstrated that WaveCluster can outperform other recent clustering approaches. WaveCluster is the first attempt to apply the properties of wavelet transform in the clustering problem in spatial data mining.

Acknowledgements. We would like to thank Drs. Raymond Ng, Tian Zhang, and Joerg Sander for providing information and the source code for CLARANS, BIRCH, and DBSCAN respectively.

References

[AF97] Allard D, Fraley C (1997) Non-parametric maximum likelihood estimation of features in spatial process using voronoi tesselation. *J Am Stat Assoc* 92 (440): 1485–1493

[AGGR98] Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high-dimensional data for data mining applications. In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, Seattle, Wash., pp 94–105

[BR95] Byers S, Raftery AE (1995) Nearest neighbor clutter removal for estimating features in spatial point processes. Technical Report 295, Department of Statistics, University of Washington

[COM95] Gudivada VN, Raghavan VV (1995) Special Issue on Content-Based Image Retrieval Systems. *IEEE Comput* 28(9)

[EKSX96] Ester M, Kriegel H, Sander J, Xu X (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proceedings of 2nd International Conference on KDD*

[EKSX98] Ester M, Kriegel H, Sander J, Xu X (1998) Clustering for mining in large spatial databases. *KI-Journal*, Special Issue on Data Mining

[Gor81] Gordon AD (1981) *Classification Methods for the Exploratory Analysis of Multivariate Data*. Chapman and Hall, London

[HJS94] Hilton ML, Jawerth BD, Sengupta A (1994) Compressing Still and Moving Images with Wavelets. *Multimedia Syst* 2(5): 218–227

[Hor88] Horn BKP (1988) *Robot Vision*, fourth edition. MIT Press, Cambridge, Mass.

[JFS95] Jacobs CE, Finkelstein A, Salesin DH (1995) Fast multiresolution image querying. In: *SIGGRAPH 95*, Los Angeles, Calif.

[JM95] Jain R, Murthy SNJ (1995) Similarity Measures for Image Databases. *Proc. SPIE (Storage and Retrieval of Image and Video Databases III)*: 58–67

[Knu98] Knuth DE (1998) *The Art of Computer Programming*, third edition. Addison-Wessley, Reading, Mass.

[KR90] Kaufman L, Rousseeuw PJ (1990) *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, Chichester

[Mal89a] Mallat S (1989) Multiresolution approximation and wavelet orthonormal bases of $L^2(R)$. *Trans Am Math Soc* 315: 69–87

[Mal89b] Mallat S (1989) A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans Pattern Anal Mach Intell* 11: 674–693

[NH94] Ng RT, Han J (1994) Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, pp 144–155

[NS80] Nassimi D, Sahni S (1980) Finding connected components and connected ones on a mesh-connected parallel computer. *SIAM J Comput* 9: 744–757

[Ope77] Openshaw S (1977) A geographical solution to scale and aggregation problems in region-building, partitioning and spatial modelling. *Trans Inst Brit Geogr* 2: 459–472

[OT81] Openshaw S, Taylor P (1981) *Quantitative Geography: A British View* (Chapter: The Modifiable Areal Unit Problem), pages 60–69. Routledge, London

[PFG97] Pauwels EJ, Fiddelaers P, Van Gool L (1997) DOG-based unsupervized clustering for CBIR. In *Proceedings of the 2nd International Conference on Visual Information Systems*, San Diego, Calif., pp 13–20

[SC94] Smith JR, Chang S (1994) Transform Features For Texture Classification and Discrimination in Large Image Databases. In *Proceedings of the IEEE International Conference on Image Processing*, pp 407–411

[Sch92] Schalkoff R (1992) *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, New York

[SCZ98] Sheikholeslami G, Chatterjee S, Zhang A (1998) *WaveCluster*: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In: *Proceedings of the 24th VLDB conference*, New York, pp 428–439,

[SN96] Strang G, Nguyen T (1996) *Wavelets and Filter Banks*. Wellesley-Cambridge Press, Wellesley, MA

[SV82] Shiloach Y, Vishkin U (1982) An $O(logn)$ parallel connectivity algorithm. *J Algorithms* 3: 57–67

[SZ97] Sheikholeslami G, Zhang A (1997) An Approach to Clustering Large Visual Databases Using Wavelet Transform. In: *Proceedings of the SPIE Conference on Visual Data Exploration and Analysis IV*, San Jose, Calif., pp 322–333

[SZB97] Sheikholeslami G, Zhang A, Bian L (1997) Geographical Data Classification and Retrieval. In: *Proceedings of the 5th ACM International Workshop on Geographic Information Systems*, Las Vegas, Nev., pp 58–61

[URB97] Uytterhoeven G, Roose D, Bultheel A (1997) Wavelet transforms using lifting scheme. Technical Report ITA-Wavelets Report WP 1.1. Katholieke Universiteit Leuven, Department of Computer Science, Belgium

[Vai93] Vaidyanathan PP (1993) *Multirate Systems And Filter Banks*. Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, N.J.

[WYM97] Wang W, Yang J, Muntz R (1997) STING: A Statistical Information Grid Approach to Spatial Data Mining. In: *Proceedings of the 23rd VLDB Conference*, Athens, Greece, pp 186–195

[XMKS98] Xu X, Ester M, Kriegel H, Sander J (1998) A distribution-based clustering algorithm for mining in large spatial databases. In: *Proceedings of the 14th International Conference on Data Engineering*, Orlanda, Fla., pp 324–331

[YCSZ98] Yu D, Chatterjee S, Sheikholeslami G, Zhang A (1998) Efficiently detecting arbitrary-shaped clusters in very large datasets with high dimensions. Technical Report 98-8, State University of New York at Buffalo, Department of Computer Science and Engineering

[ZM97] Zait M, Messatfa H (1997) A comparative study of clustering methods. *Future Generation Comput Syst* 13: 149–159

[ZRL96] Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, Montreal, Canada, pp 103–114