WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases [*]

Gholamhosein Sheikholeslami
Computer Science Dept.
SUNY at Buffalo
Buffalo, NY 14260, USA
gsesfah@cs.buffalo.edu

Surojit Chatterjee
Computer Science Dept.
SUNY at Buffalo
Buffalo, NY 14260, USA
surojit@cs.buffalo.edu

Aidong Zhang
Computer Science Dept.
SUNY at Buffalo
Buffalo, NY 14260, USA
azhang@cs.buffalo.edu

Abstract

Many applications require the management of spatial data. 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 outliers (noise) and the order of input data. We propose *WaveCluster*, a novel clustering approach based on wavelet transforms, which satisfies all the above requirements. Using multi-resolution property of wavelet transforms, we can effectively identify arbitrary shape clusters at different degrees of accuracy. We also demonstrate that WaveCluster is highly efficient in terms of time complexity. Experimental results on very large data sets are presented which show the efficiency and effectiveness of the proposed approach compared to the other recent clustering methods.

[*]This research is supported by Xerox Corporation.

Proceedings of the 24th VLDB Conference
New York, USA, 1998

1 Introduction

In this paper we explore a *data clustering* method in the multidimensional 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 of the spatial objects, 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 or outliers. Outliers refer to spatial 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 get 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 objects at different levels of accuracy. 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 and 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 *multi-resolution* property.

In this paper, we propose a spatial data mining method, termed *WaveCluster*. We consider the multidimensional spatial data as a multidimensional signal and we apply signal processing techniques - wavelet transforms to convert the spatial data into the frequency domain. 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 outliers, and it is totally insensitive to the ordering of the input data. Also, because of the signal processing techniques applied, the *multi-resolution* 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. We first discuss the related work in Section 2. In Section 3, 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 4 discusses our clustering method WaveCluster and analyzes its complexity. In section 5, we present the experimental evaluation of the effectiveness and efficiency of WaveCluster using very large data sets. Finally in Section 6, concluding remarks are offered.

2 Related Work

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

2.1 Partitioning Algorithms

Partitioning algorithms construct a partition of a database of N objects into a set of K clusters. Usually they start 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 swap with other objects until all K objects qualify as medoids. PAM compares an object with entire data set 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 data set, applies PAM on the sample, and finds the medoids of the sample.

Ng and Han introduced CLARANS (Clustering Large Applications based on RANdomaized 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 data sets. Although CLARANS is faster than PAM, but it is still slow and as mentioned in [WYM97], its computational complexity is $\Omega(KN^2)$. 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.

2.2 Hierarchical Algorithms

Hierarchical algorithms create a hierarchical decomposition of the the database. The hierarchical decomposition can be represented as a *dendrogram*. 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 data set 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 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. Also as our experimental results show, if the clusters are not "spherical" in shape, BIRCH does not perform well. This is because it uses the notion of radius or diameter to control the boundary of a cluster.

2.3 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 non-parametric density estimation problem in two ways. Firstly, they use cross-validation to select the appropriate width of convolution kernel. Secondly, they use Difference-of-Gaussians *(DOG's)* 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 run time complexity of DBSCAN is $O(NlogN)$.

2.4 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 proposed 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 data set 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.

We propose WaveCluster, which is a grid-based approach. The proposed approach is very efficient, specially for very large databases. The computational complexity of generating clusters in our method is $O(N)$. The results are not affected by outliers and the method is not sensitive to the order of the number 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.

3 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.

3.1 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 an n-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 n numerical attributes, the feature vector will be one point in the n-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.

3.2 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 an n-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 n-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, Figure 1 shows a 2-dimensional feature space, where the two dimensional data points have formed four clusters. Note that Figure 1 and also the figures in Section 5 are the *visualizations* of the 2-dimensional 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 one-dimensional signal, so the whole feature space will be a 2-dimensional signal. Boundaries and edges of the clusters constitute the high frequency parts of this 2-dimensional 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 n-dimensional signal representing the feature space, resulting in detecting the clusters.

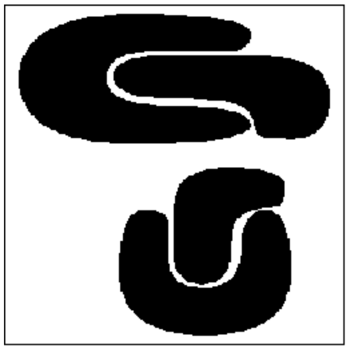


Figure 1: A sample 2-dimensional feature space.

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 n-dimensional signals in which one-dimensional 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, SZ97, SZB97]. 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 n-dimensional feature spaces.

3.3 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 one-dimensional 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. Wavelet transform provides us with a set of attractive filters. For example, Figure 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.

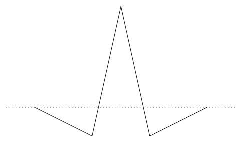


Figure 2: Cohen-Daubechies-Feauveau (2,2) wavelet.

- **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 automatically stand out and clear regions around them, so that they become more distinct [PFG97]. It makes finding the connected components in the transformed space easier than that of the original feature space, because the dense regions in the feature space will be more salient. Figure 3 shows an example of a feature space before and after applying Cohen-Daubechies-Feauveau(2,2) biorthogonal transform. As the figure shows, the clusters in the transformed space are more salient and thus easier to find.

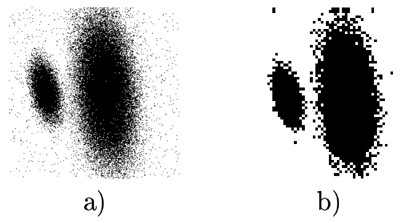


Figure 3: Feature space: a)original; b) transformed.

- **Effective Removal of Outliers**: As we will show, we take advantage of low-pass filters used in the wavelet transform to automatically remove the outliers. Figure 3 shows that the outliers in the original space are removed after the transformation.

- **Multi-resolution**: Multi-resolution property of wavelet transform can help detecting the clusters at different levels of accuracy. As it will be shown later, we can apply wavelet transform multiple times which results in clusters at different scales from fine to coarse.

- **Cost Efficiency**: Since applying wavelet transform is very fast, it makes our approach cost-effective. As it will be shown in the experiments, clustering very large datasets takes only a few seconds. Using parallel processing we can get even faster responses.

Applying wavelet transform on a signal decomposes it into different frequency sub-bands [Mal89a]. We now briefly review wavelet-based multi-resolution decomposition. More details can be found in Mallat's paper [Mal89b]. To have multi-resolution representation of signals we can use discrete wavelet transform. We can compute a coarser approximation of the one-dimensional input signal A_0 by convolving it with the filter $\tilde{H}$ and down sampling the signal by two [Mal89b]. By down sampling, we mean skipping every other signal sample (for example one row in a 2-dimensional feature space). $\tilde{H}$ acts as a low pass filter. All the discrete approximations A_j, $1 < j < J$, (J is the maximum possible scale), can thus be computed from A_0 by repeating this process. Scales become coarser with increasing j. For example, the third approximation of A_0 (that is A_3) is coarser than the second approximation A_2. Figure 4 illustrates the method.

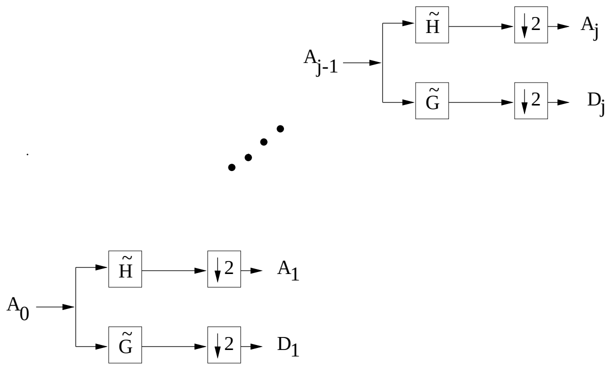


Figure 4: Block diagram of multi-resolution wavelet transform.

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 A_{j-1} with the high pass filter $\tilde{G}$ and returning every other sample of output. The wavelet representation of a discrete signal A_0 can therefore be computed by successively decomposing A_j into A_{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 2-dimensional feature space, in which we can apply two separate one-dimensional transforms [HJS94]. We can represent a 2-dimensional feature space as an image where each pixel of image corresponds to one unit cell in the feature space. The two-dimensional 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.

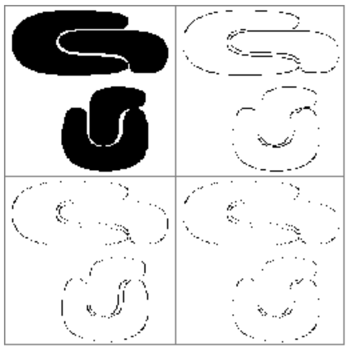
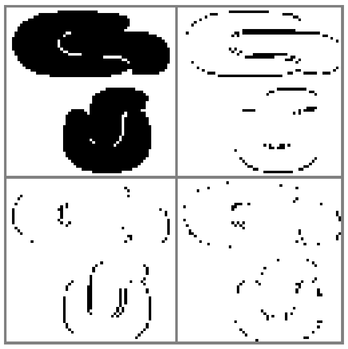
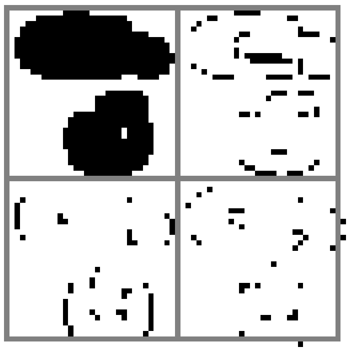

a)

b)

c)

Figure 5: Multi-resolution wavelet representation of the feature space in Figure 1 at a) scale 1; b) scale 2; c) scale 3.

Figure 5 shows the wavelet representation of the image in Figure 1 at three scales from fine to coarse. At each level, sub-band LL (wavelet approximation of original image) is shown in the upper left quadrant. Sub-band LH (horizontal edges) is shown in the upper right quadrant, sub-band HL (vertical edges) is displayed in the lower left quadrant, and sub-band HH (corners) is in the lower right quadrant.

The above wavelet model can similarly be generalized for n-dimensional feature space, where one-dimensional wavelet transform will be applied n 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. Next section describes how we use this information to detect the clusters.

4 WaveCluster

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

4.1 Algorithm

Given a set of spatial objects $o_i, 1 \leq i \leq N$, 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 users 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 units.
2. Apply wavelet transform on the feature space.
3. Find the connected components (clusters) in the
   subbands of transformed feature space,
   at different levels.
4. Assign label to the units.
5. Make the lookup table.
6. Map the objects to the clusters.
```

4.1.1 Quantization

The first step of WaveCluster algorithm is to quantize the feature space, where each dimension i in the d-dimensional 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 units in the feature space. Then the objects will be assigned to these units based on their feature values. Let $F_k = (f_1, f_2, \ldots, f_d)$ be the feature vector of the object o_k in the original feature space. Let $M_j = (v_1, v_2, \ldots, v_d)$ denote a unit in the original feature space where v_i, $1 \leq v_i \leq m_i$, $1 \leq i \leq d$, is the location of the unit on the axis $\mathcal{X}_i$ of the feature space. Let s_i be the size of each unit in the axis $\mathcal{X}_i$. An object o_k with the feature vector $F_k = (f_1, f_2, \ldots, f_d)$ will be assigned to the unit $M_j = (v_1, v_2, \ldots, v_d)$ if

$$\forall i \quad (v_i - 1)s_i \leq f_i < v_i s_i, \quad 1 \leq i \leq d$$

The number (or size) of these units is an important issue that affects the performance of clustering. Because of multi-resolution property of wavelet transform, we consider different unit sizes at different scales of transform.

4.1.2 Transform

In the second step, discrete wavelet transform will be applied on the quantized feature space. Applying wavelet transform on the units M_j results in a new feature space and hence new units T_k. Given the set of units T_k, WaveCluster detects the connected components in the transformed feature space. Each connected component is a set of units T_k and is considered as a cluster. Corresponding to each resolution r of wavelet transform, there would be a set of clusters C_r, where usually at the coarser resolutions, number of clusters is less. In the experiments, we applied wavelet transform three times and tried Haar, Daubechies, Cohen-Daubechies-Feauveau ((4,2) and (2,2)) transforms [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 Figure 5, for a 2-dimensional feature space, the subbands LL show the clusters at different scales. We use the algorithm in [Hor88] to find the connected components in the 2-dimensional feature space (image). The same concept can be generalized for higher dimensions. Figure 12 in Section 5, shows the clusters that WaveCluster found at each scale in different colors.

4.1.3 Label and Make Look Up Table

Each cluster c, $c \in C_r$, will have a cluster number c_n. In the fourth step of algorithm, WaveCluster labels the units in the feature space that are included in a cluster, with its cluster number. That is,

$$\forall c \ \ \forall T_k, \ T_k \in c \Longrightarrow l_{T_k} = c_n, \quad c \in C_r,$$

where l_{T_k} is the label of the unit 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 units in the transformed feature space to the units in the original feature space. Each entry in the table specifies the relationship between one unit in the transformed feature space and the corresponding unit(s) of the original feature space. So the label of each unit in the original feature space can be easily determined. Finally, WaveCluster assigns the label of each unit in the feature space to all the objects whose feature vector is in that unit, and thus the clusters are determined. Formally,

$$\forall c \ \ \forall M_j, \ \ \forall o_i \in M_j, \quad l_{o_i} = c_n, \ \ c \in C_r, \ \ 1 \le i \le N,$$

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

4.2 Properties of WaveCluster

When the objects are assigned to the units of the quantized feature space at step 1 of the algorithm, the final content of the units is independent of the order in which the objects are presented. The next steps of the algorithm will be performed on these units. 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 it will 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, processing time will be just a function of number of units in the feature space. Thus, it makes WaveCluster very efficient, specially for very large number of objects. WaveCluster will be specially very efficient for the cases where the number of units m and the number of feature space dimensions d are low. Minefield detection and some seismology applications are examples where we have 2-dimensional feature spaces.

WaveCluster finds the connected components in the *average* subband (LL) of the wavelet transformed feature space, as the output clusters. As mentioned in Section 3.3, average subband is constructed by convolving the low pass filter along each dimension and down sampling by two. So a wavelet transformed unit will be affected by the content of units in the neighborhood covered by the filter. It means that the spatial relationships between neighboring units will be preserved. The algorithm to find the connected components, labels each unit of feature space with respect to the cluster that it belongs to. The label of each unit is determined based on the labels of its neighboring units [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 shape clusters.

WaveCluster applies wavelet transform multiple times. Each time we apply another transform, we ignore some details in the average subband, and effectively increase the size of a unit's neighborhood whose spatial relationship is considered. This results in sets of clusters with different degrees of details after each application of wavelet transform. In other words, we will have multi-resolution clusters at different scales, from fine to coarse. For example, in Section 5, Figure 12 shows an example where wavelet transform is applied three times and the output clusters after each transform are presented. At scale 1 we have 31 fine clusters, and at the next scale some of those clusters are merged. At scale 3, we have only two coarse clusters representing original feature space. In our approach, user does not have to know the exact number of clusters, however a good estimation of number of clusters helps in choosing the appropriate scale and the corresponding clusters. One of the effects of applying low pass filter on the feature space is the removal of noise. WaveCluster takes advantage of this property, and removes the noise and outliers from the feature space automatically. Figure 6-d shows an example where about 1000 outliers are scattered in the feature space, but after applying wavelet transform, they are removed and thus WaveCluster can detect the clusters correctly.

4.3 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. 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 units. Assuming m units in each dimension of feature space, there would be $K = m^d$ units. Complexity of applying wavelet transform on the feature space (step 2) will be $O(lK) = O(K)$, where l is a small constant representing the length of filter used in the wavelet transform. To find the connected components in the 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 number of data objects (N). The time complexity of the last step of WaveCluster algorithm is $O(N)$. Since we assume this algorithm is applied on very large databases, that is $N >> K$, so $O(N) > O(K)$, thus the overall time complexity of the algorithm will be $O(N)$. During applying wavelet transform on each dimension of the feature space, the required operations for each feature space unit can be carried out independent of the other units. Thus, using parallel processing we 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 specially useful when the number of units m or the number of dimensions d is high. For 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 $N > m^f$. So the required time for WaveCluster algorithm will be linear in terms of N (number of objects in the database).

5 Performance Evaluation

In this section, we evaluate the performance of WaveCluster and demonstrate its 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. We first introduce the test datasets.

Synthetic Datasets

Datasets DS1,DS2 and DS3 are the same as used by [ZRL96]. They are shown in Figure 6 a, b, and c. Each dataset consists of 100,000 points. The points in DS3 are randomly distributed while DS1 and DS2 are distributed in a grid and sine curve pattern respectively. The other datasets shown in Figure 6 were generated using our own dataset generator. Data set DS4 is the *noisy* version of DS5 that is generaterd by scattering 1000 random noise points on the original dataset.

Clustering Very Large Datasets

All the datasets used in the experiments contain typically more than 10,000 data points. DS1, DS2 and DS3 each has 100,000 data points. WaveCluster can successfully handle 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 non-neighboring clusters might be allocated the same color. [1]

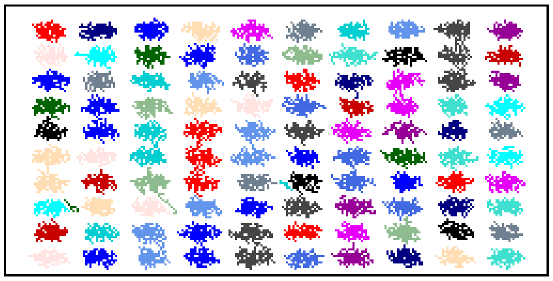
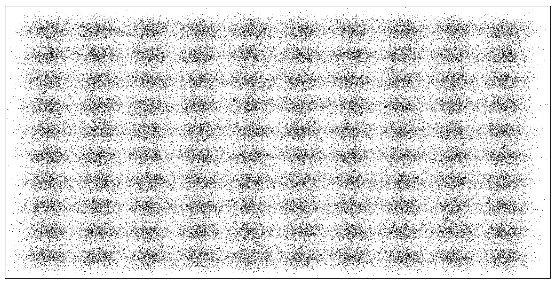


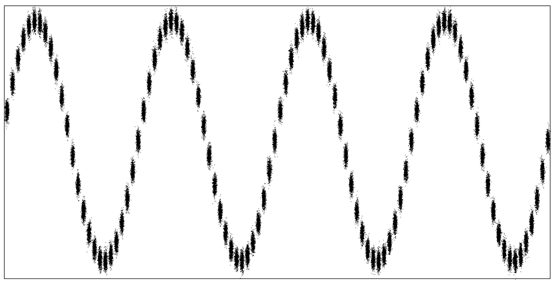
Figure 7: WaveCluster on DS1

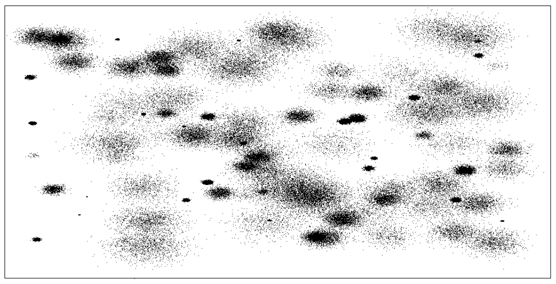
Clustering Arbitrary Shapes

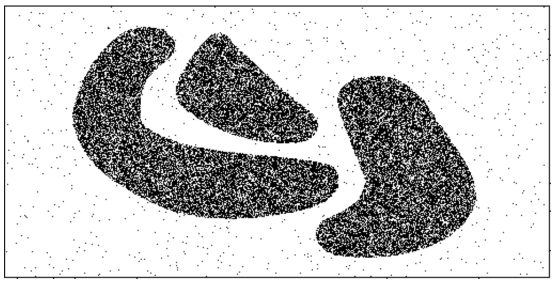
As we mentioned earlier, spatial data mining methods should be capable of handling any arbitrary shaped clusters. There are 3 arbitrary shaped clusters in dataset DS5. Figure 8-a shows WaveCluster clustering of DS5. Figure 8-b shows BIRCH clustering for the same data set. This result emphasizes deficiency of the methods which assume the shape of the clusters a priori.

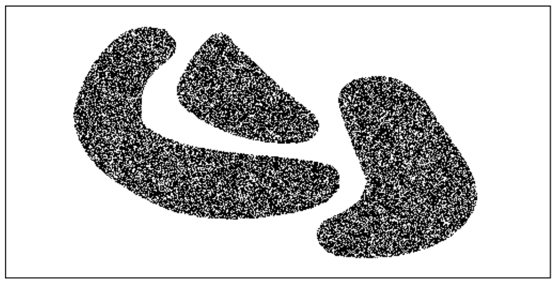
[1]The paper with the colored clusters is available in *http://www.cs.buffalo.edu/pub/WWW/faculty/azhang/*

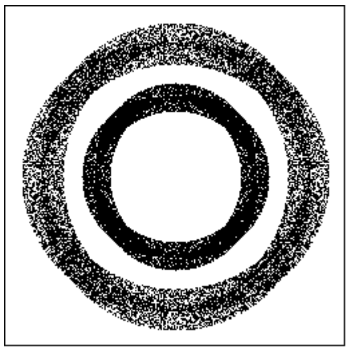

a) DS1

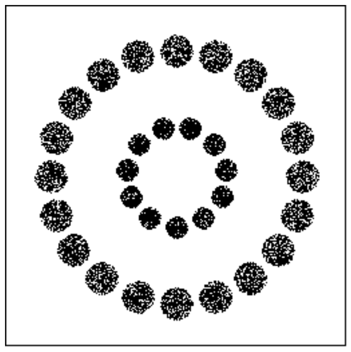

b) DS2


c) DS3


d) DS4


e) DS5


f) DS6


g) DS7

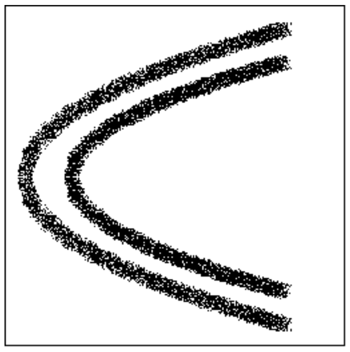
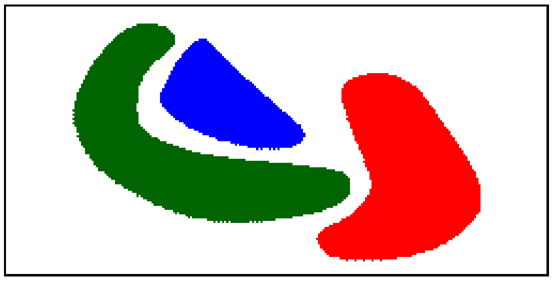
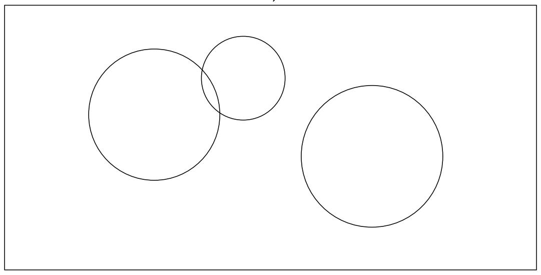

h) DS8

Figure 6: The datasets used in the experiments.



a)



b)

Figure 8: a) WaveCluster on DS5; b) BIRCH on DS5

Handling Outliers

WaveCluster is very effective in handling outliers. Figure 9 shows clustering achieved by WaveCluster on DS4 dataset (noisy version of DS5 dataset). It successfully removes all random noise and produces three intended clusters. Also, because of $O(K)$ (where K is the number of grid points), the time complexity of the processing phase of WaveCluster, that is, the time taken to find the clusters in the noisy version of the data is the same as that on without noise.

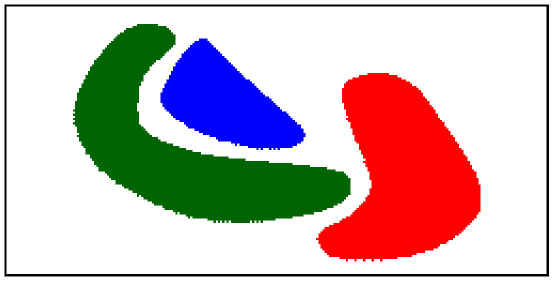
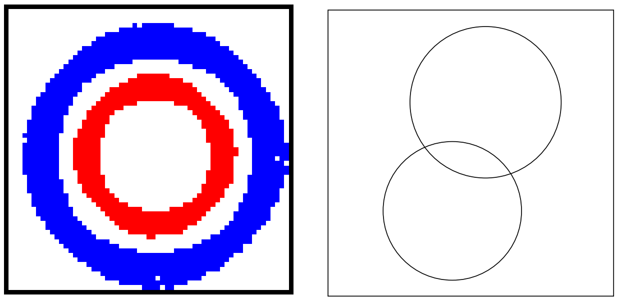


Figure 9: WaveCluster on DS4.

Clustering Nested and Concave Patterns

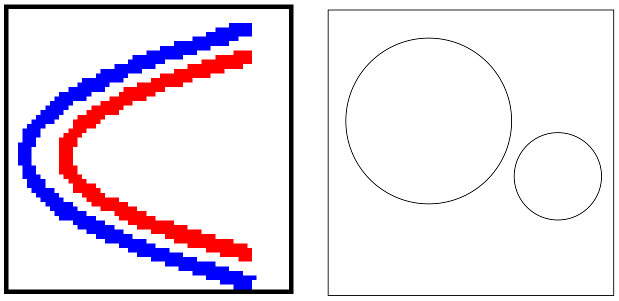
WaveCluster can successfully clusterify any complex pattern consisting of nested or concave clusters. From Figure 6-f and Figure 10-a we see that WaveCluster's result is very accurate on nested clusters. Figure 10-b shows BIRCH's result on the same dataset.

Figure 6-h shows DS8 as an example of a concave shape data distribution. Figures 11-a and 11-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.



a) b)

Figure 10: a) WaveCluster on DS6; b) BIRCH on DS6



a) b)

Figure 11: a) WaveCluster on DS8; b) BIRCH on DS8

Clustering at Different Resolutions

WaveCluster has the remarkable property that it can be used to cluster at different granularities according to user's requirement. Figure 12 displays the results of wavelet transform on DS7. This illustrates how WaveCluster finds clusters at different degrees of accuracy. This property of WaveCluster provides the user with the flexibility to modify queries based on initial results. BIRCH does very well on dataset DS7, but it generates only one set of clusters.

Comparison of Timing Requirements

We now compare the timing requirements of WaveCluster, BIRCH, and CLARANS as shown in Table 1. 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 are large number of objects. Thus, we were unable to run it. Based on the comparison of BIRCH and CLARANS presented in [ZRL96], we estimate the performance of CLARANS. We observe that on an average CLARANS is 22 times slower than

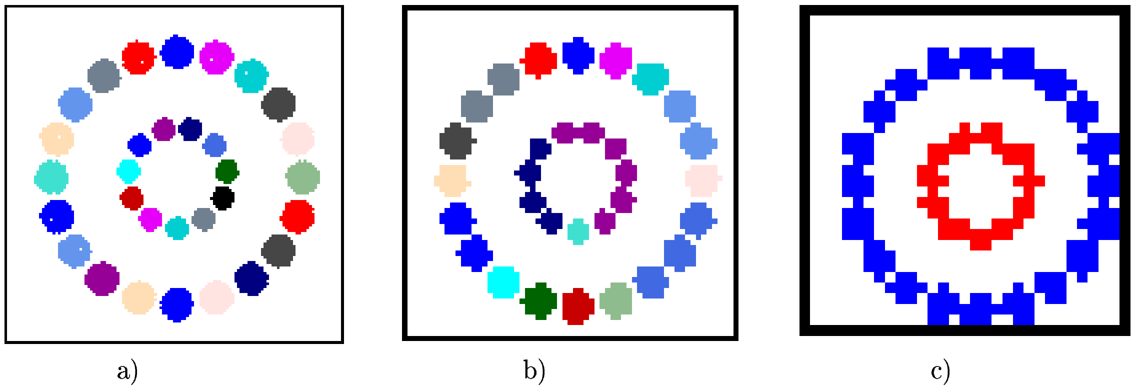


Figure 12: WaveCluster output clusters of DS7 at a) scale 1; b) scale 2; c) scale 3.

Number of data	DS1 100,000	DS2 100,000	DS3 100,000	DS4 60,266	DS5 59,266	DS6 40,000	DS7 18,509	DS8 16,928
CLARANS	1232.0	1093.0	1089.4	258.3	255.2	369.6	92.4	308.0
BIRCH	56.0	49.7	49.5	11.7	11.6	16.8	4.2	10.9
WaveCluster (I/O)	3.40	3.40	3.40	1.29	1.20	0.80	0.36	0.32
WaveCluster (processing)	1.46	1.36	1.41	0.33	0.33	0.18	0.17	0.17
WaveCluster (Total)	4.86	4.66	4.81	1.62	1.53	0.98	0.53	0.49

Table 1: Required time (in seconds) for the clustering approaches.

BIRCH. Because of the sensitiveness to the ordering of the input data, BIRCH produces different timing results for different ordering. For example, BIRCH takes 14.1 seconds to run on original DS8 dataset, but when we sorted the data based on the first dimension it took 10.9 seconds. We use the lower time for comparisons. We show the time requirements for I/O 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 observe that 1) WaveCluster outperforms BIRCH and CLARANS by a large margin. On an average it is 8 to 10 times faster than BIRCH which in turn is 20 to 30 times faster than CLARANS; 2) 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 in the space. As Table 1 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 a whole lot faster. The experimental results demonstrates WaveCluster to be a stable and efficient clustering method.

6 Conclusion

In this paper, we presented the clustering approach WaveCluster. It applies wavelet transform on the spatial data feature space which helps in detecting arbitrary shape clusters at different scales. It is a very efficient method with time complexity of $O(N)$, where N is the number of objects in the database, which makes it specially attractive for very large databases. WaveCluster is insensitive to the order of input data to be processed. Moreover, it is not affected by the outliers and can handle them properly. Our experimental results demonstrated that WaveCluster can outperform the other recent clustering approaches. WaveCluster is the first attempt to apply the properties of wavelet transform in the clustering problem in spatial data mining.

Acknowledgments

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] D. Allard and C. Fraley. Non parametric maximum likelihood estimation of features in spatial process using voronoi tesselation. *Journal of the American Statistical Association*, December 1997.

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

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

[HJS94] Michael L. Hilton, Bjorn D. Jawerth, and Ayan Sengupta. Compressing Still and Moving Images with Wavelets. *Multimedia Systems*, 2(5):218–227, December 1994.

[Hor88] Berthold Klaus Paul Horn. *Robot Vision*. The MIT Press, forth edition, 1988.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[Mal89a] S. Mallat. Multiresolution approximation and wavelet orthonormal bases of $L^2(R)$. *Transactions of American Mathematical Society*, 315:69–87, September 1989.

[Mal89b] S. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trnasactions on Pattern Analysis and Machine Intelligence*, 11:674–693, July 1989.

[NH94] R. T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proceedings of the 20th VLDB Conference*, pages 144–155, 1994.

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

[PFG97] E.J. Pauwels, P. Fiddelaers, and L. Van Gool. DOG-based unsupervized clustering for CBIR. In *Proceedings of the 2nd International Conference on Visual Information Systems*, pages 13–20, 1997.

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

[Sch92] Robert Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, Inc., 1992.

[SN96] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks*. Wellesley-Cambridge Press, 1996.

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

[SZ97] G. Sheikholeslami and A. Zhang. 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, February 1997.

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

[URB97] Greet Uytterhoeven, Dirk Roose, and Adhemar Bultheel. Wavelet transforms using lifting scheme. Technical Report ITA-Wavelets Report WP 1.1, Katholieke Universiteit Leuven, Department of Computer Science, Belgium, April 1997.

[Vai93] P. P. Vaidyanathan. *Multirate Systems And Filter Banks*. Prentice Hall Signal Processing Series. Prentice Hall, Englewood Cliffs, NJ,, 1993.

[WYM97] Wei Wang, Jiong Yang, and Richard Muntz. STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd VLDB Conference*, pages 186–195, Athens, Greece, 1997.

[ZRL96] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, 1996.