Paper ID: 394

FindOut: Finding Outliers in Very Large Datasets *

Dantong Yu, Gholam Sheikholeslami and Aidong Zhang
Department of Computer Science and Engineering
State University of New York at Buffalo
Buffalo, NY 14260, USA

Abstract

Signal processing techniques have been introduced to transform images for image enhancement, filtering and restoration, analysis, and reconstruction. In this paper, we apply signal processing techniques to solve important KDD problems. In particular, we introduce a novel deviation (or outlier) detection approach, termed *FindOut*, based on wavelet transform. The main idea in FindOut is to remove the clusters from the original data and then identify the outliers. Although previous research showed that such techniques may not be effective because of the nature of the clustering, FindOut can successfully identify various percentages of outliers from large datasets. Thus, we can combine both clustering and outliers detection in a unified approach. Experimental results on very large datasets are presented which show the efficiency and effectiveness of the proposed approach.

1 Introduction

Data mining is a process of extracting valid, previously unknown, and ultimately comprehensible information from large databases and using it to make crucial decisions. Modern companies are awash in data on customers, clients, suppliers and industry trends. But data is of little use without intelligence. Here Intelligence refers to combing through the data to notice patterns, devise rules, and make predictions about future. Data mining technology permits organizations to make the most effective use of the vast amounts of data that they have gathered. For example, in the banking industry, data mining can be used in modeling and predicting credit fraud, evaluating risk, performing trend analysis, analyzing profitability, and helping with direct marketing campaigns.

One of the very interesting problems arising recently in the data mining research community is the deviation (or outliers) detection problem. Here outliers refer to those exceptions

*This research is partially supported by an NSF CAREER grant IIS-9733730.

which deviate from the values anticipated based on a given (usually statistical) model or from some previously known expectation and norm [17]. The identification of outliers can lead to the discovery of truly unexpected knowledge in areas such as electronic commerce exceptions, bankruptcy, credit card fraud, and even the analysis of performance statistics of stock exchange [13]. Such knowledge can lead to detecting abnormal events happened in the past or predicting potential trends in the future. Such potential trends may become new directions for future invest, marketing, and other purposes.

1.1 Related Work

Many clustering algorithms have been recently proposed. They include partitioning algorithms such as CLARANS [15], improved k-means algorithms [4, 6], hierarchical algorithms such as BIRCH [24], density-based methods such as DBSCAN [5] and CURE [7], or grid-based algorithms including STING [21], DBCLASD [22], and WaveCluster [18]. Some other recent clustering algorithms are CLIQUE [1], DENCLUE [8]. The goal of these methods is to detect the clusters in the data space. They try to remove or at least tolerate the outliers to better detect the clusters.

Most of the previous research in outlier detection has been in the field of statistics [9, 11, 3]. These methods usually make assumptions about data distribution, statistical distribution parameters, type or number of outliers. However, these parameters generally cannot be easily determined which can make these method difficult to apply. Ruts and Rousseeuw proposed a depth-based method to detect the outliers [16]. The objects are organized in layers in the data space according to some definition for depth. Outliers are expected to be in the deep layers. Such methods do not have the distribution fitting problem, however, in practice for multidimensional spaces, they may not be applicable. Sarawagi *et al.* proposed a discovery-driven method that search for exception is guided by pre-computed indicators of exceptions at various levels of detail in a data cube [17]. They consider a value in a cell of a data cube to be an exception, if it is significantly different from anticipated value based on a statistical model. Although this method can handle hierarchies and ordered dimensions, but model selection still is an issue for it.

Arning *et al.* introduced a method for outlier detection which relies on the observation that after seeing a series of similar data, an element disturbing the series is considered an outlier [2]. Their method requires a function that can yield the degree to which a data element causes the dissimilarity of the data set to increase. It looks for the subset of data that lead to the greatest reduction in Kolmogorov complexity for the amount of data discarded [2].

Knorr and Ng presented algorithms to detect Distance-Based outliers [12, 13]. They consider an object O in a dataset T a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O. Their index-based algorithm executes a range search with radius D for each object. If number of objects in its D-neighborhood exceeds a threshold, the search stops and that object is declared as a non-outlier, otherwise it is an outlier. In their cell-based approach, they quantize the feature space and assign the objects to the cells. For each cell, they consider two sets of neighboring cells L_1 and L_2. By checking L_1 and L_2 neighbors they rule out testing unnecessary cells and speed up outlier detection. Their experiments show that their cell-based algorithm is the most efficient when the number of dimensions is less than or equal to 4. However for higher number of dimensions, its efficiency decreases [13].

We propose the cell-based method FindOut for outlier detection. But we look at the problem from a signal processing point of view and apply signal processing techniques to address this problem.

1.2 Our Contributions

Signal processing techniques have been introduced to transform images for image enhancement, filtering and restoration, analysis, and reconstruction. In this paper, we apply signal processing techniques to solve important KDD problems. In particular, we introduce a novel deviation (or outlier) detection approach, termed *FindOut*, based on wavelet transform. FindOut is a by-product of WaveCluster [18]. The main idea in FindOut is to remove the clusters from the original data and thus identify the outliers. Previous research has shown that such techniques may not be effective because of the nature of the clustering [17]. Utilizing WaveCluster, FindOut can successfully identify various percentages of outliers from large datasets. As a result, we can combine both clustering and outliers detection in a unified approach. Experimental results on very large datasets will demonstrate the effectiveness and efficiency of the proposed approach.

The rest of the paper is organized as follows. Section 2 presents the outliers detection approach. Section 3 presents the experimental results. Section 4 offers conclusion.

2 Finding Outliers

2.1 Brief description about Liner Space System

From the original dataset, information for detecting outliers is usually hidden and scattered. We introduce *linear space system* to expose and gather such information. Given an input dataset, we summarize it as a distribution function and input to the linear space system. We then get a new function which is called *transformed distribution function.* If the transformed distribution function can make the outliers more salient than other data points, we then can easily detect outliers within the dataset.

According to [10], many signal processing operations can be modeled as a linear space system. The two-dimensional case was introduced in [10]. For a d-dimensional space, a linear space (LS_g) system can be completely described by its *impulse response* function $g(x_1, x_2, \ldots, x_d)$. The function $g(x_1, x_2, \ldots, x_d)$ is also called *kernel* function. When the input to the system is an impulse $\delta(x_1, x_2, \ldots, x_d)$, the output will be exactly the kernel function $g(x_1, x_2, \ldots, x_d)$. Thus by inputting an impulse, we can obtain kernel function and understand the functionality of the system. The system works as follows:

$$\begin{array}{c} \text{Input} \\ f(x_1, x_2, \ldots, x_d) \end{array} \longrightarrow \boxed{\begin{array}{c} \text{Linear space system} \\ g(x_1, x_2, \ldots, x_d) \end{array}} \longrightarrow \begin{array}{c} \text{Output} \\ \hat{f}_g(x_1, x_2, \ldots, x_d) \end{array}$$

where $f(x_1, x_2, \ldots, x_d)$ and $\hat{f}_g(x_1, x_2, \ldots, x_d)$ are the input and output signals. For such a system, the output $\hat{f}_g(x_1, x_2, \ldots, x_d)$ is defined as the *convolution* of $f(x_1, x_2, \ldots, x_d)$ with the impulse response $g(x_1, x_2, \ldots, x_d)$ and is defined as follows:

$$\hat{f}_g(x_1, x_2, \ldots, x_d) = \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_{d} f(x'_1, x'_2, \ldots, x'_d) \cdot g(x_1 - x'_1, \ldots, x_d - x'_d) dx'_1, \ldots, dx'_d.$$

For discrete functions and a given linear system, this becomes:

$$\begin{aligned} \hat{f}_g(i_1, i_2, \ldots, i_d) &= f(i_1, i_2, \ldots, i_d) \star g(i_1, i_2, \ldots, i_d) \\ &= \sum_{k_1=1}^{m_1} \sum_{k_2=1}^{m_2} \cdots \sum_{k_d=1}^{m_d} f(k_1, k_2, \ldots, k_d) \cdot g(i_1 - k_1, i_2 - k_2, \ldots, i_d - k_d), \end{aligned}$$

where $\star$ is the convolution operation. When the kernel is a wavelet function w, we call the linear system as *wavelet transform system.* We denote the output as $\hat{\rho_w}$ when the input is a function ρ.

2.2 Problem Formalization

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* to generate the *quantized space.* 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 by a single pass through the dataset the containment relations are discovered 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 $\mathbf{c_i} \cdot \mathbf{param}$, which is called as *density*. Thus a *density function* $\rho(\mathbf{c_i})$ is specified in the quantized space. A cell in the quantized space with 0 count value is called an *empty cell*. A cell in the quantized space with nonzero count value is called a *nonempty cell*. In our approach we apply wavelet transform on $\mathbf{c_i} \cdot count$ values. The *transformed space* is the result after wavelet transformation on the count values of the cells in the quantized space. The procedure can be easily illustrated by linear space system. A wavelet transform is described by wavelet function w, thus a linear space system LS_w is defined based on w. A density function ρ is defined on the quantized space which maps a cell $\mathbf{c_i}$ to $\mathbf{c_i} \cdot \mathbf{param}$. The result of LS_w is a new density function $\hat{\rho}_w$ on the transformed space which maps $\mathbf{c_i}$ to transformed value of $\mathbf{c_i} \cdot \mathbf{param}$, as follows:

$$\begin{array}{c} \text{Input} \\ \rho(\mathbf{c_i}) \end{array} \longrightarrow \boxed{\begin{array}{c} \text{Linear space system} \\ w \end{array}} \longrightarrow \begin{array}{c} \text{Output} \\ \hat{\rho}_w(\mathbf{c_i}) \end{array}$$

We introduce the following definitions to be used in the rest of the paper.

Definition 1 *(Significant cell) Given a linear space system LS_w and density function ρ, a cell $\mathbf{c}$ is a* significant *cell if $\hat{\rho}_w(\mathbf{c}) \geq \tau$, where $\tau = p \times \mathcal{V}$, p is input parameter and $\mathcal{V}$ is a statistical value of $\hat{\rho}_w$.*

The statistical value $\mathcal{V}$ can be the average value, maximum value or summation of $\hat{\rho}_w$.

Definition 2 *(ϵ-neighbor) A cell $\mathbf{c_1}$ is an ϵ-neighbor of cell $\mathbf{c_2}$ if both are significant cells in transformed space and $D(\mathbf{c_1}, \mathbf{c_2}) \leq \epsilon$, where D is an appropriate distance metric and $\epsilon > 0$.*

Definition 3 *(Cluster) A cluster $\mathcal{C}$ is a set of significant cells $\{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_m}\}$ which are ϵ-connected in the transformed space.*

Informally, we say that an object is outlier if it does not belong to any clusters. We give a precise definition of the outliers as follows:

Definition 4 *(Outlier) Given a linear space system LS_w and density function ρ, an object in a cell* $\mathbf{c}$ *is outlier if $\hat{\rho}_w(\mathbf{c}) < \tau$, where τ is given by* **Definition** 1.

Our definition can be related to the outliers as defined by Knorr and Ng [13]. For a dataset, it defines a density function ρ. We define a kernel function g as follows:

$$g(\mathbf{c_i}) = \begin{cases} 1 & \text{if } D(\mathbf{c_i}, \vec{\mathbf{0}}) \leq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{\mathbf{0}}$ is the origin in the data space and D is a distance function. A linear system LS_g is defined based on g. An object $\mathbf{o_i}$ in LS_g is outlier if $\hat{\rho}_g(\mathbf{o_i}) < p \times N$, where N is the size of data. Thus the outlier under this linear system is similar to the one defined in [13].

Given a set of N data points $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_N}\}$ in the d-dimensional feature space, our goal is to find the outliers as defined above. We propose FindOut to detect outliers in very large datasets with low number of dimensions.

2.3 Multidimensional Data versus Multidimensional Signals

We first discuss the relationship between multidimensional data and multidimensional signals and show how to use wavelet transform to detect the inherent relationships in the data. We propose to look at the multidimensional data space from a signal processing perspective. The collection of data in the multidimensional data space composes a d-dimensional signal. The high frequency parts of the signal correspond to the regions of the data space where there is a rapid change in the distribution of data, 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 data space where the data are concentrated. For example, in a two-dimensional data space, each row or column can be considered as a one-dimensional signal, so the whole data space will be a two-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 data is high, we can apply signal processing techniques to find the high frequency and low frequency parts of d-dimensional signal representing the data, resulting in detecting the clusters. The key idea is to apply signal processing methods to transform the space and find the dense regions in the transformed space. The objects in the remaining non-dense regions will be the outliers.

2.4 Wavelet Transform

Wavelet transform is a signal processing technique that decomposes a signal into different frequency subbands (for example, high frequency subband and low frequency subband). A one-dimensional signal s can be filtered by convolving the filter coefficients c_k with the signal values:

$$\hat{s}_i = \sum_{k=0}^{M-1} c_k s_{i+k-\frac{M}{2}}, \tag{1}$$

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 interesting filters. For example, Figure 1 shows the Cohen-Daubechies-Feauveau(2,2) biorthogonal wavelet [20].

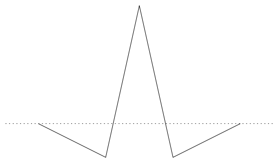


Figure 1: Cohen-Daubechies-Feauveau (2,2) biorthogonal wavelet.

We now briefly review wavelet-based multi-resolution decomposition. More details can be found in Mallat's paper [14]. 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 S_0 by convolving it with the low pass filter $\tilde{H}$ and down sampling the signal by two [14]. 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. Figure 2 illustrates the method.

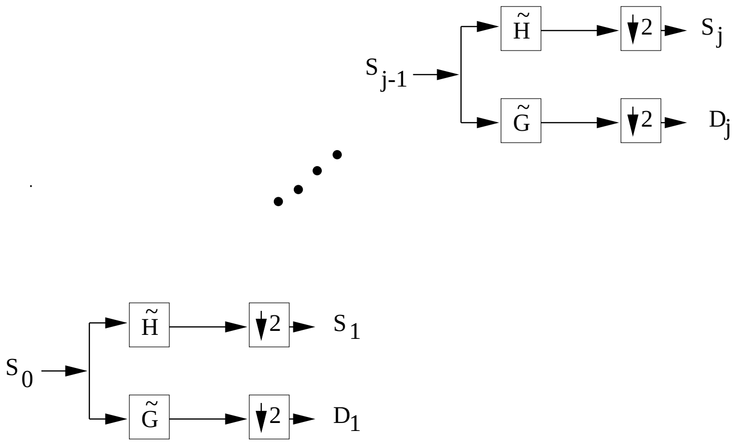


Figure 2: Block diagram of multi-resolution wavelet transform.

D_j denotes the difference between S_j and S_{j-1} 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 d-dimensional data space in which one-dimensional transform can be applied multiple times.

2.5 Outlier Detection

We proposed WaveCluster which partitions the feature space into cells to generate a quantized space [18]. It then applies wavelet transform on the quantized space. After finding the clusters in the transformed space, it assigns labels to the cells according to the cluster that they belong to. Finally it assigns labels to the objects in the cells based on the cluster label of each cell. WaveCluster can detect arbitrary shape clusters such as convex, concave, or nested clusters [18]. Using multiresolution property of wavelets, WaveCluster detects the clusters at different degrees of detail and does not require the exact number of clusters as input. WaveCluster takes advantage of the filters in wavelet transform and removes the noise from the feature space without requiring extra processing time. It is a very fast and efficient method for very large databases with low number of dimensions. Using parallel processing it can be further speeded up.

Wavelet transform convolves its input with a low pass filter. A low pass filter has the property of removing the outliers (noise) from the input. It considers the influence of the neighboring objects in computing its output. For example Figure 3-a presents a one-dimensional dataset (signal) containing some noise. Figure 3-b shows the signal after convolving with a low pass filter. It shows how the low pass filter removes the outliers and smoothes the input data.

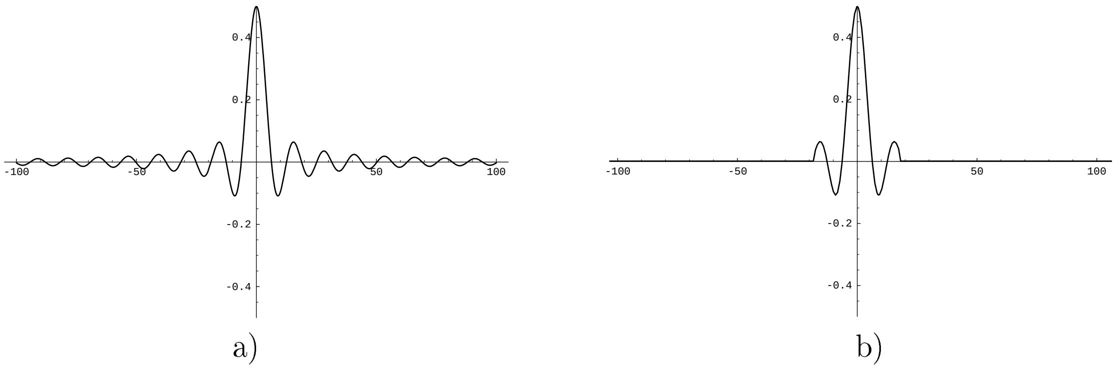


Figure 3: a) Original signal; b) Signal after convolving with low pass filter.

WaveCluster takes advantage of this property of low pass filters and removes the outliers that may cause problem in detecting the clusters. However, in FindOut our goal is to *find* the outliers and not to remove them. The main idea in FindOut is to remove the clusters from the original data and thus identify the outliers. We now present the definition of $\ominus$ to subtract quantized spaces.

Definition 5 *(Scale-compatibility): Given two quantized spaces $S_1 = A_1 \times A_2 \ldots \times A_d$ and $S_2 = B_1 \times B_2 \ldots \times B_d$, let m_i be the number of cells along dimension A_i, and n_i be the number of cells along B_i. S_1 and S_2 are* scale-compatible *if*

$$\forall i \ \ 1 \leq i \leq d \ \ \exists s_i \in Z^+ \mid m_i = s_i n_i, \quad or \quad n_i = s_i m_i.$$

Definition 6 *(Subtraction of quantized spaces): Given two scale-compatible spaces S_1 and S_2, $R = S_1 \ominus S_2$ is the result of their subtraction. If $S_1(a_1, a_2, \ldots, a_d)$ is nonempty and $S_2(b_1, b_2, \ldots, b_d)$ is empty then $R(c_1, c_2, \ldots, c_d)$ will be nonempty, otherwise $R(c_1, c_2, \ldots, c_d)$ will be empty, where $c_i = s_i a_i$, or $c_i = s_i b_i$,*

We present FindOut as a tool that can be added to WaveCluster. So we consider it in the context of WaveCluster and make of use WaveCluster's intermediate data structures. The main steps of FindOut's algorithm are as follows:

Algorithm FindOut
Input: Multidimensional data objects' feature vectors
Output: Outliers

1. Apply WaveCluster.
   1.1. Quantize feature space. /* S_{orig} */
   1.2. Apply wavelet transform. /* $S_{l,t}$ */
   1.3. Find the connected components (clusters).
   1.4. Map data to the clusters. /* $S_{l,c}$ */
2. $S_{l,out} \leftarrow S_{orig} \ominus S_{l,c} \ominus S_{l,D}$.

Based on the definition of cells in Section 2.2, objects will be assigned to the cells in step 1.1 of algorithm. We denote this quantized space as S_{orig}. Applying wavelet transform (step 1.2) on S_{orig} results in a set of subbands $S_{l,t}$ for each level l of the transform. For each level l, $S_{l,t}$ consists of one approximation space (signal) $S_{l,A}$ and a set of detail spaces (signals) $S_{l,D}$ (as defined in section 2.4). The clusters are detected in the transformed spaces $S_{l,A}$ in step 1.3. In step 1.4, the clusters will be mapped to the original data and will be represented in the spaces $S_{l,c}$. In step 2 of FindOut, the subtraction results of spaces will be stored in $S_{l,out}$ which only contain the outliers at different levels l. Since in wavelet transform we always downsample the signals by two, all the above spaces are scale-compatible.

Figure 4-a shows an example of a two-dimensional feature space S_{orig}. The gray scale value of each pixel in image reflects the number of objects in the corresponding cell in the original feature space. Darker pixels have more objects. Figure 4-b shows the two clusters detected by WaveCluster in $S_{1,c}$. If we remove the pixels from the original space which correspond to the cluster cells, we will get the space $S_{orig} \ominus S_{1,c}$ shown in Figure 4-c. For better visualization, all nonempty cells in Figure 4-c (and 4-d) are shown in black instead of gray scale. Figure 4-c shows that in addition to the outliers, the points in the boundaries of clusters are also included in the result. The goal of WaveCluster is to detect the core of the main clusters in the dataset. So it applies those wavelet transforms that can clear the area around the clusters to make them more distinct. However such points should not be considered as outliers. To detect the clusters we use the information in the low frequency subbands (approximation signals). The other detail frequency subbands $S_{l,D}$ have information about the boundaries of the clusters. We can apply this information and remove the cluster boundary points from the set of outliers. To do so, we subtract $S_{l,D}$ from the spaces in the previous step and get the results shown in Figure 4-d.

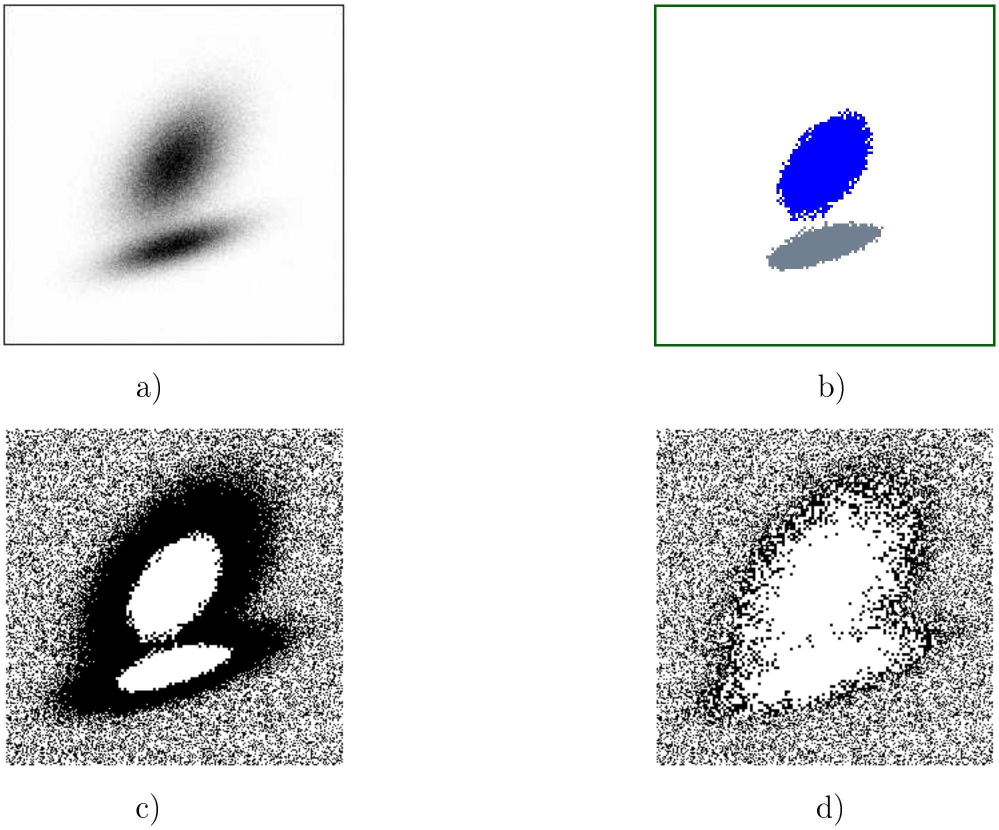


Figure 4: a) Original data; b) Clusters detected by WaveCluster; c) Original data $\ominus$ clusters; d) Outliers detected by FindOut.

As we mentioned earlier, we present FindOut as a tool in WaveCluster. If we are only interested in the outliers and not the clusters, some steps of WaveCluster can be skipped with little effect on the effectiveness of FindOut. This will make FindOut more efficient. For example, we do not have to find the connected components and instead of $S_{l,c}$ (result of step 1.4), we can subtract $S_{l,A}$ from original space S_{orig}. Based on our experiments, usually the results from the first level of wavelet transform contain all the outliers, thus we can avoid the computation required by the next levels of the wavelet transform. We are working on these issues as part of our on going research.

2.6 Time Complexity

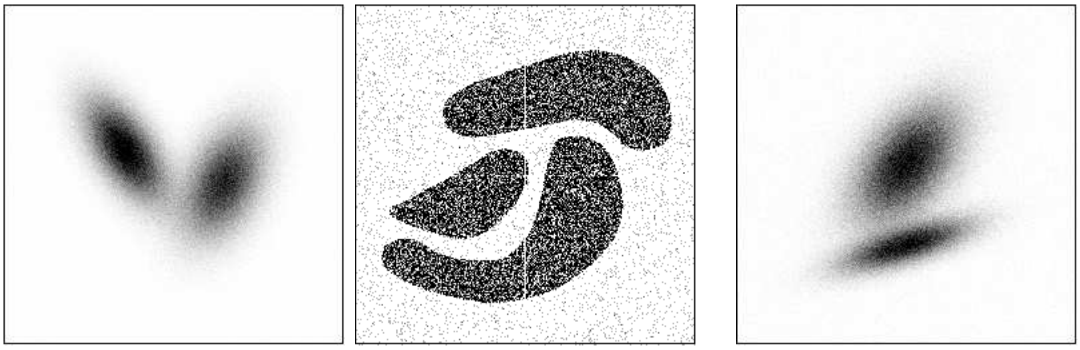
The time complexity of the the steps 1.1 and 1.4 of FindOut's algorithm is $O(N)$, because they scan all the database objects. Assuming m cells in each dimension of feature space, there would be $K = m^d$ cells. Complexity of applying wavelet transform would be $O(K)$. To find the connected components, the required time is $O(K)$. Subtracting the spaces in step 2 has also a complexity of $O(K)$. Thus the time complexity of processing data (without considering I/O) which is performed in steps 1.2, 1.3 and 2 would in fact be $O(K)$. Since this algorithm is applied on very large databases with low number of dimensions, we can assume that $N \geq K$. As an example, for a database with 1,000,000 objects when the number of dimensions d is less than or equal to 6, 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 $O(N)$.

3 Experiments

In this section, we evaluate the performance of FindOut and demonstrate its effectiveness and efficiency on different types of distributions of data. Tests were done on synthetic datasets generated by us and on a real-world dataset. We present the experiments on 2-dimensional datasets so that the results can be easily visualized. FindOut can be applied to 3,4 or 5 low dimensions. As for higher dimensions, [23] provided a hash-based algorithm to address the high-dimensionality issue. All of the tests were done on a Sun UltraSPARC 168 MHz machine having 1024 MB of main memory.

3.1 Synthetic Datasets

Figure 5 shows the synthetic datasets used in the experiments. DS1 contains 735,000 data points with 5% outliers. DS2-10 contains 65,193 data points with 10% outliers. DS2-5, DS2-15, DS2-20 and DS2-25 refer to the same as DS2 but with 5%, 15%, 20% and 25% outliers, respectively. DS3-10 contains 550,000 data points with 10% outliers. DS3-5, DS3-15, DS3-20, DS3-25 and DS3-50 are different versions of DS3-10 with different percentage of outliers.



a) DS1 b) DS2-10 c) DS3-10

Figure 5: The datasets used in the experiments.

Find outliers in large datasets. All the datasets used in the experiments contain more than 50,000 data points. For a fixed quantization, larger datasets don't affect CPU time of the outlier detection. FindOut can successfully handle arbitrarily large number of data points. Figure 6 shows the outliers detected for DS1.

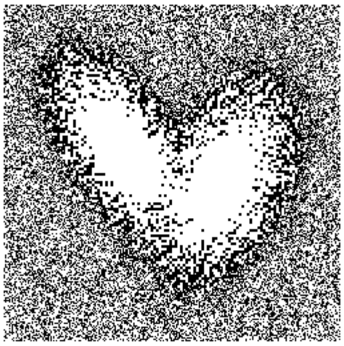


Figure 6: The outliers of DS1

Different percentage outliers in large datasets. FindOut is very effective in handling outliers. Figure 7 shows outliers that have been detected among five datasets with different percentages of randomly generated outliers. Our approach can work well on different scales of outliers.

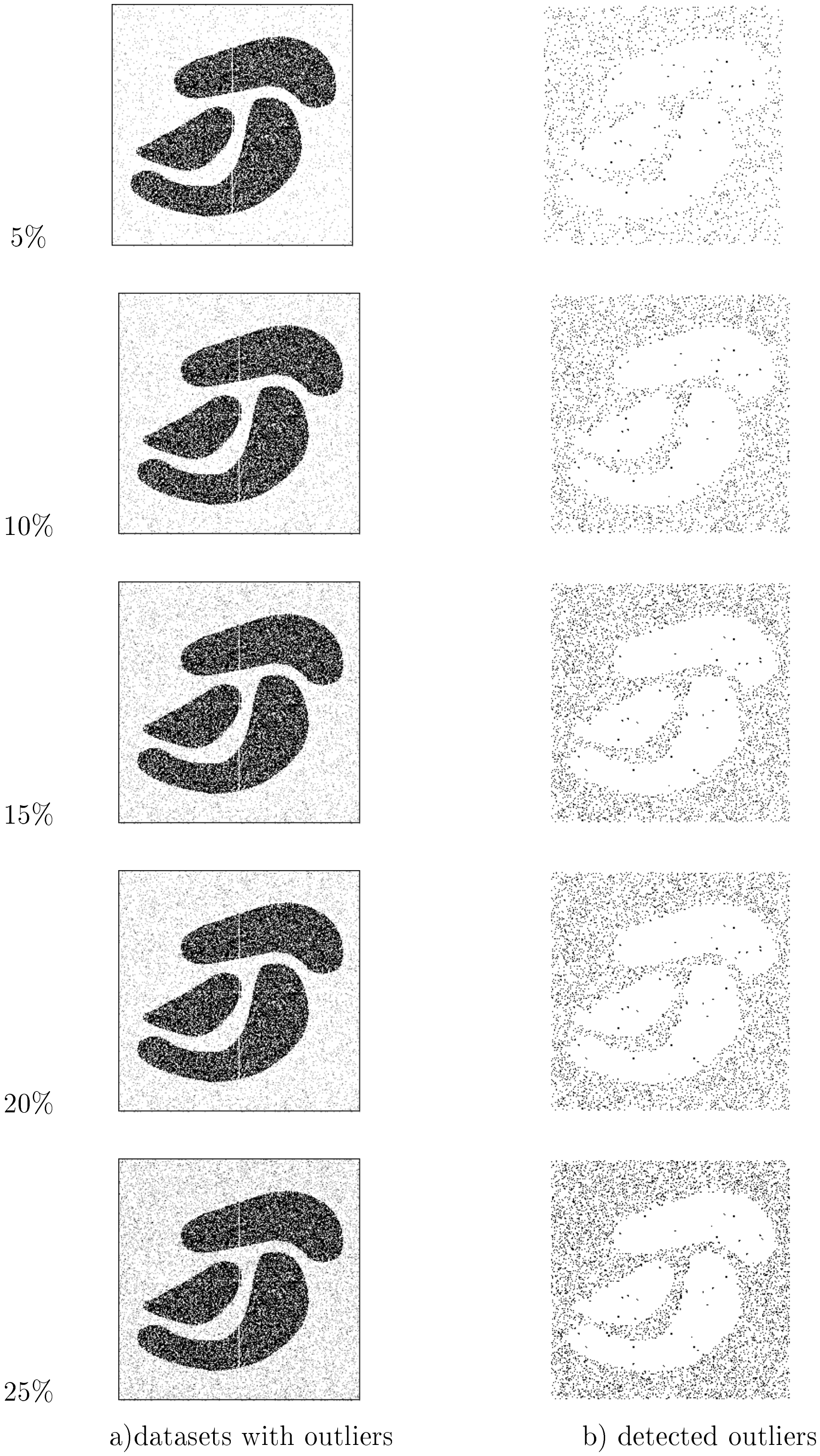
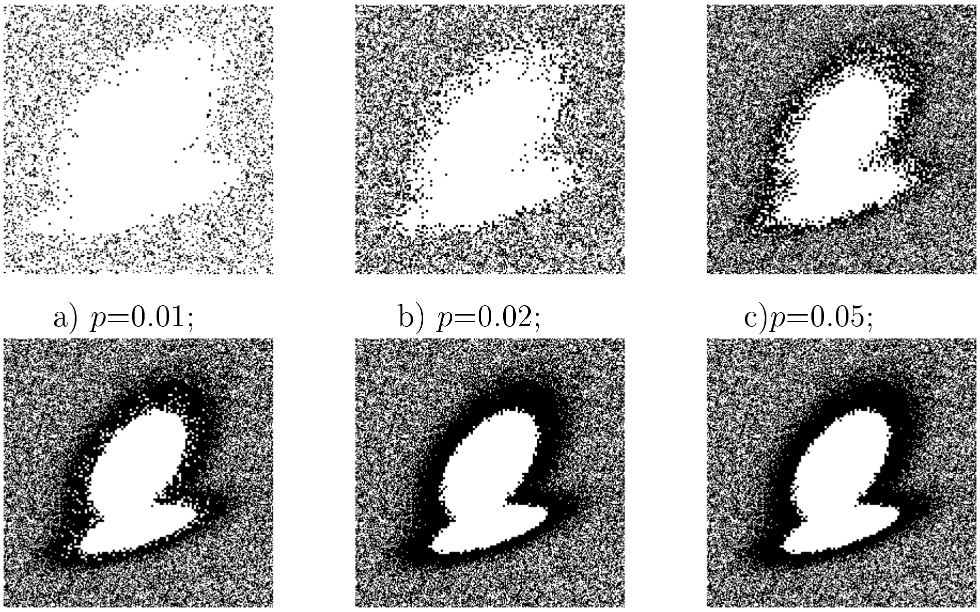


Figure 7: FindOut on datasets with different levels of outliers:

The important role of thresholding in outliers detection. The selection of threshold p (defined in Section 2.2) directly affects the results of the outlier detection. Higher thresholds may result in larger number of outliers. We used a simple uniform threshold value in our tests. But for better performance in real-world datasets, the knowledge such as intensity characteristics of datasets and percentage of outliers can be used in thresholding. Figure 8 shows different threshold values applied for detection outliers in dataset DS3-10. We can see that the real dense and sparse areas do not change too much when threshold changes. But the *border* area between them is highly affected by the chosen threshold.



a) p=0.01; b) p=0.02; c)p=0.05;

d) p=0.10; e) p=0.25; f) p=0.50.

Figure 8: The outliers of DS3-10 with different threshold values p.

Timing on different quantization and wavelet transform. Quantization m affects running time. When m is low (very coarse quantization), the total number of cells is small, thus it takes less time. Running time is also affected by wavelet functions with different width. The width of wavelet function determines the range of neighborhood. When a large neighborhood which affects a data point is considered, the longer it takes on detecting outliers. Figure 9 shows CPU-time when wavelets Haar, Cohen-Daubechies-Feauveau(2,2), Cohen-Daubechies-Feauveau(4,2) [20], and Daubechies [19] are used on dataset DS3-10 under different quantizations.

The code and the data reported in [13] was not available to do experimental comparisons. Their experiments show that their cell-based algorithm is the most efficient when the number of dimensions is less than or equal to 4. However for higher number of dimensions, its

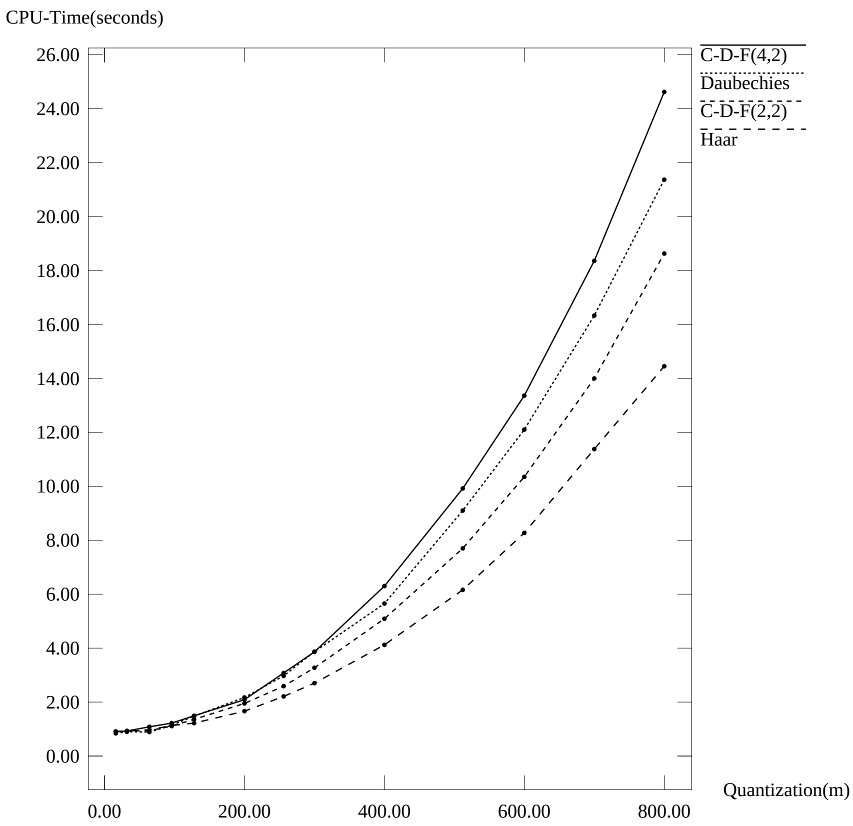


Figure 9: Run time under different wavelets and quantizations.

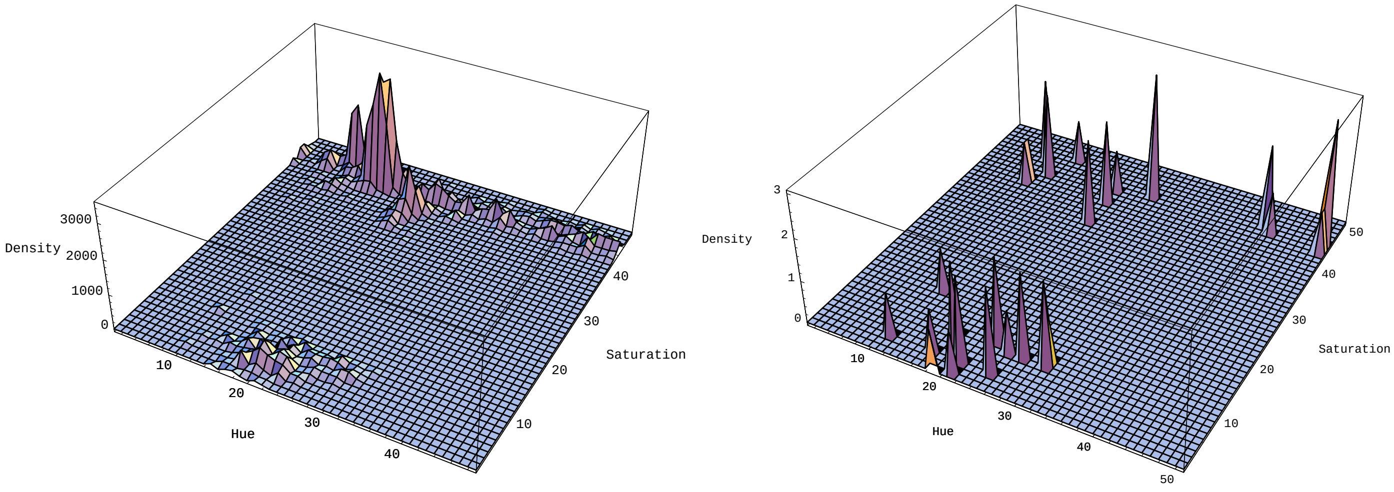
efficiency decreases [13]. Using hash-based strategy, FindOut can work with high dimensions with arbitrary quantization (See [23]). Table 1 shows the timing requirements of FindOut with different number of dimensions and quantizations. The size of the testing dataset is $100,000$.

Number of Dimensions	m				
	32	64	96	128	256
5	118	132	216	256	340
10	1,042	1647	2,166	2,680	3,280
15	2,374	3958	7,332	7,156	8,080
20	3,742	6153	10,983	10,660	13,400

Table 1: Execution time (in seconds) for different number of dimensions and quantizations m.

3.2 A Real-World Dataset

Dataset sat-hue is created based on the image feature vectors from University of California at Irvine Machine Learning Repository. The size of the dataset is 392,402 and each element describes hue and saturation values of segmentations of seven types of images: brickface, sky, foliage, cement, window, path and grass. To scale up the test, we randomly generated 10 to 1000 vectors around each basic feature vector within the hypersphere centered at the basic feature vector with a small radius r. Figure 10 shows the histogram of dataset sat-hue and detected outliers. All the detected points have exceptional hue and saturation values.



a)Dataset sat-hue; b)Detected Outliers;

Figure 10: Dataset sat-hue

4 Conclusion

This paper has addressed an interesting question of how to apply signal processing techniques to solve important KDD problems and the effectiveness of such approaches. We presented a novel deviation (or outlier) detection approach, termed FindOut, based on wavelet transform. By combining outliers detection with clustering, FindOut can successfully identify various percentages of outliers from large datasets. Thus, our approach is highly cost-effective. Experimental results on very large data sets have demonstrated the efficiency and effectiveness of the proposed approach. This approach can have many potential uses. Based on the found outliers, - we can build a model that describes what kind of data have abnormal distance to *majority*, thus become outliers. This model can be used to predict future datasets.

References

[1] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 94–105, Seattle, WA, 1998.

[2] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 164–169, 1996.

[3] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley, 3rd edition, 1994.

[4] P. S. Bradley, Usama Fayyad, and Cory Reina. Scaling clustering algorithms to large databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 9–15, New York, August 1998.

[5] 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.

[6] Usama Fayyad, Cory Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 194–198, New York, August 1998.

[7] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. Cure: An efficient clustering algorithm for large databases. In *Proceedings of the ACM SIGMOD conference on Management of Data*, pages 73–84, Seattle, WA, 1998.

[8] Alexander Hinneburg and Daniel A.Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 58–65, New York, August 1998.

[9] D. Hoaglin, F. Mosteller, and J. Tukey. *Understanding Robust and Exploratory Data Analysis.* John Wiley, New York, 1983.

[10] Ramesh Jain, Rangachar Kasturi, and Brian G.Schunck. *Machine Vision.* MIT PRESS, 1995.

[11] R. Johnson. *Applied Multivariate Statistical Analysis.* Prentice Hall, 1992.

[12] E. Knorr and R. Ng. A unified notion of outliers: Properties and computation. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 219–222, 1997.

[13] E. Knorr and R. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th VLDB conference*, pages 392–403, New York, August 1998.

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

[15] 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, Santiago, Chile, 1994.

[16] Ida Ruts and Peter J. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23:153–168, 1996.

[17] S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. In *Proceedings of the Sixth International Conference on Extending Database Technology (EDBT)*, Spain, March 1998.

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

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

[20] 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.

[21] 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.

[22] X. Xu, M.Ester, H. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 324–331, Orlando, Fl, February 1998.

[23] D. Yu, S. Chatterjee, G. Sheikholeslami, and A. Zhang. 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, November 1998.

[24] 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.