ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, No.8, August 2008, pp.1965−1979
DOI: 10.3724/SP.J.1001.2008.01965
© 2008 by *Journal of Software*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

数据聚类中基于浓度噪音消除的可视化参数选择方法*

钱 宇+

(Department of Pathology, University of Texas Southwestern Medical Center, Dallas, 75390, USA)

A Visual Approach to Parameter Selection of Density-Based Noise Removal for Effective Data Clustering

QIAN Yu+

(Department of Pathology, University of Texas Southwestern Medical Center, Dallas, 75390, USA)

+ Corresponding author: E-mail: yu.qian@utsouthwestern.edu

Qian Y. A visual approach to parameter selection of Density-based noise removal for effective data clustering. *Journal of Software*, 2008,19(8):1965−1979. http://www.jos.org.cn/1000-9825/19/1965.htm

Abstract: Traditional visual data mining relies on visualization techniques to disclose implicit information and relationship among data through utilizing human capability of pattern recognition. As an important step in data clustering, noise removal is a challenging topic as domain-specific noise is not well defined and cannot be removed by generic process of data cleaning. This paper addresses two conjugated and reciprocal issues in the use of visualization in noise removal: choosing appropriate visualization techniques based on data removing methods, and designing processing algorithms that suit visualization. The goal is a synthesis of visualization techniques and data mining methods to enhance the overall performance while reducing the subjective factor in visual mining procedure. A visual data cleaning approach called *CLEAN* is proposed to assist spatial data clustering in four important aspects: removal of domain-specific noise, visualization of data quality, selection of algorithm parameters, and measurement of noise removing methods on parameter sensitiveness. Experiments show that the visualization models in *CLEAN* do assist effective discovery of natural spatial clusters in a noisy environment.

Key words: information visualization; data mining; clustering; noise removal

摘 要: 可视化技术的发展极大地提高了传统数据挖掘技术的效率.通过结合人类识别模式的能力,计算机程序能够更有效的发现隐藏在数据中的规律和信息.作为聚类分析的重要步骤,噪音消除一直都是困绕数据挖掘研究者的问题,尤其对于不同领域的应用,由于噪音的模型和定义不同,单一的数据处理方法无法有效而准确地去除域相关的噪音.本文针对这一问题,提出了一个新型的可视化噪音处理方法 CLEAN.CLEAN 的独特之处在于它设计的噪音处理技术和提出的可视化方法有机地结合在一起.噪音处理算法为可视化模型生成所需数据,同时针对噪音处理算法选择可视化方法,从而达到提高整个数据处理系统性能的目的.这样不仅降低了噪音去除过程中主观因素的影响,还可以帮助数据挖掘程序去除领域相关的噪音.同时源数据的质量,算法参数的选择和不同噪音去除算法的精确性都可以在所使用的可视化模型中反映出来.实验表明 CLEAN 能够有效地帮助空间数据聚类算法在噪音环境下发现数据的自然聚类.

关键词: 信息可视化;数据挖掘;聚类;噪音消除

* Received 2008-01-16; Accepted 2008-04-18

中图法分类号: TP181 文献标识码: A

1 Introduction

The importance of handling a rapid growth of data has been recognized since the wide application of database technology and large-scale data collection mechanisms. Computer-aided data processing techniques have proved their utilities in managing large amount of data from various perspectives. Among these techniques, data mining or knowledge discovery in database (KDD), which refers to the non-trivial process of discovering interesting, implicit, and unknown knowledge from large databases[1], has been widely applied to many communities. While the purpose of data mining is still primarily concerned with extracting knowledge from very large databases[2], its techniques evolves to be more user-oriented. Users are allowed to get insights from data during the mining process. The extracted patterns, models, or relationships among the data are becoming easily interpreted and understood. Many of the benefits come from effective data visualization, an important component of visual data mining (VDM)[3], which emphasizes the human-computer interaction in mining process to deliver intuitive mining results.

The goal of visualization is to provide qualitative insight into data, processes, and concepts through using visual pattern recognition ability humans possess[4]. Visualization could bridge the two most powerful information-processing systems: human and computer. While humans are capable of providing overviews of patterns and detecting outliers, they are limited in the ability of handling scale and are easily overwhelmed by the volumes of data. Data mining could complement human abilities through processing large amount of data automatically. Combining the two approaches for knowledge discovery is clearly promising[5]. Visualization plays a key role in the combination as transforming and presenting problems visually could provide new insight and pave the way to their solutions[6].

As a primary data mining technique, clustering is the process of grouping a set of objects into classes or clusters so that objects within a cluster have high similarity in comparison to one another, but are dissimilar to objects in other clusters[7]. Since the definition of similarity varies in different domains, clustering usually relies heavily on the application and the database at hand, which leads to a trade-off between using automated algorithms that rely on statistical properties or implicit heuristics for the clustering process and allowing the user to influence the results through his/her domain knowledge[8]. This paper attempts to explore the complementary roles of visualization techniques and mining methods so that a balance can be struck in-between. On one hand, through visualization, users become active participants in the clustering process by providing domain-specific information and tuning algorithm parameters. On the other hand, the characteristics and properties of data mining methods and features of data are visualized as feedbacks to the user. While traditional visual data clustering focuses on inventing visual representations of source data that allow users to group data directly through visual inspection, our approach still relies on mining techniques to find clusters, in which visualization is not directly applied to source data but used to guide the mining techniques to produce final clusters. This leads to a fundamental difference between our approach and previous work because the data to visualize in our approach reflects characteristics of the mining algorithm or the impact of the algorithm on the source data, e.g., parameters, internal data flow etc., while traditional approaches visualize source data or its transformed forms directly and consider less the connections between mining algorithms and the data to be processed. The role of visualization plays in our approach is bridging the user not to source data but to the mining techniques that process the source data, which not only reduces the subjective factor involved in manual operations performed on source data but also connects the human capability of pattern recognition and the mining power of computers.

The means of visual support for a data clustering method may vary for different user requirements. It, however, must serve in one of the three stages of the overall clustering process: preprocessing, grouping, and post-processing. Since domain-specific requirements and knowledge are usually involved in the preprocessing stage in guiding the grouping process, we are particularly interested in supporting user decisions in this stage. We propose two visualization models for noise removal, a crucial preprocessing step for many clustering algorithms. The first model visualizes raw data in a 3D colored space so that users can see the impact of algorithm parameters directly, while the second model visualizes results of different parameter selections abstractly and therefore can serve high-dimensional data sets, if necessary.

There have been many non-visual clustering methods for discovering patterns of different shapes, densities, and sizes in spatial databases[9−11]. There are also some visual approaches for high dimensional data clustering[12,13]. None of them, however, addresses the domain-specific noise removal problem: noise cannot be defined or removed without the support from domain users because noise in one case may not be noise in another. This paper aims at improving the situation. The visualization models presented in this paper are integrated into a visual data cleaning approach called CLEAN (CLeaning by Eliminating Ambiguous Noise) to address the issue of domain-specific noise removal. CLEAN provides users with a layered and colored 3D visualization so that noise can be defined and identified with domain knowledge. The key difference between the visualization models in CLEAN and a direct plotting of 2D or 3D data is in our data preparation method before producing the visualization. Although direct plotting allows users to judge which data is noise through visual inspection, users cannot "tell" the computer their judgments unless they specify noise points one by one, which is unfeasible in large databases. In contrast, the data preparation method used in this paper is designed to ease the burden of visualization through separating data points into small groups. Each data point has obtained a group-ID before visualization so that users can specify noise points using their group-IDs. For 2D data points, their group-IDs are used as their third coordinates to deliver a 3D visualization. For 3D data points, different groups will be represented with different colors in the direct plotting. Users can judge which groups are noise easily through visual inspection and specify the corresponding group-IDs or colors to the algorithm to allow customizable noise removal. The assignment of group IDs is not a random procedure and relies on the K-core algorithm[14], which partitions a given graph into many groups called "cores". The cores preserve the density information of the given graph and therefore can be regarded as an initial separation of noise from true data. The K-core method is well-studied in literature. A core with a small K is usually more possible to be noise than a core with a big K. Another advantage of using "cores" as groups is that the "cores" are inherent to the given graph as they are defined as subgraphs with maximum node degree smaller than K. That is, the granularities of these groups (cores) are predefined by the graph connections, i.e., the data itself, and have nothing to do with how we visualize them. Based on the definition, the number of cores is always equal to or smaller than the largest node degree in the graph. Thus the number of groups (cores) is much smaller than the number of data points. Therefore our approach is much more efficient than removing noise points one by one. The way of generating core-IDs will be introduced in Section 2 and its utility will be explained in Section 3.

For high-dimensional data, we propose another visualization model to assist noise removal. The proposed model, called *Waterfall*, typically represents the resulting profiles of different parameters in noise removal in a waterfall-like pattern. The waterfall model does not visualize higher dimensional data directly. It instead visualizes the relation between the output of the data preparation method and the two algorithm parameters in a 3D coordinate system. Since the data preparation method separates the given data and removes noise, visualizing its output provides an approximate overview on the data distribution. Further, the relationship between the two method parameters can be reflected through a 2D projection of the produced 3D graph on the surface decided by the two

axes corresponding to the two method parameters. If different combinations of the two parameters produce similar noise removing results, the noise removing method might not be sensitive to parameter selection; otherwise, the noise removing method is parameter sensitive. Thus we can evaluate the parameter sensitivity of a noise removing method. Also parameter tuning becomes easy. The appropriate pair of the two parameters can be selected through checking the visualization of the algorithm output and the relationship between the two parameters. Compared with a direct plotting of raw data, the waterfall model has several features: firstly, it is independent from dimensionality of raw data. Secondly, it allows a quick and approximate overview on data distribution. Thirdly, it discovers the relationship between algorithm parameters. Lastly, it can measure the parameter sensitivity of a noise removing method.

According to Han, *et al*.[7], clustering algorithms can be classified into hierarchical method, partitioning method, density-based method, and grid-based method. Visualization models introduced in this paper are designed to suit representative density-based clustering algorithms. This is based on our observation that most representative density-based approaches use a similar pair of parameters on noise removal. Since the axes of the visualization models are method parameters, a reassignment of the axes can provide visual support for different noise removing methods, as long as these methods use parameters in a similar way. While noise removal can help clustering, some clustering methods can be used to output data items that do not belong to any cluster as noise, such as Birch[15], and scale-space filter method[16]. A main difference between our approach and clustering filter methods is that they do not employ visualization to encode domain knowledge and therefore is incapable of facilitating customized noise removal. The purposes and applications of these methods are also different.

The CLEAN approach has been used in the data clustering framework FACADE[17] to provide effective visual support for noise removal and parameter tuning. Some of the visualization models in CLEAN have also been applied to assist post-classification of remote sensing images[18], with results significantly outperforming those of conventional methods.

The rest of the paper is organized as follows. Section 2 describes the data preparation method we use to preprocess the given data so that visualization can be easily applied. Section 3 presents the visualization model for ambiguous noise removal. The waterfall model is introduced in Section 4. Section 5 illustrates how CLEAN and the visualization models are used in effective spatial data clustering. Section 6 concludes the paper.

2 Data Preparation

As mentioned in Section 1, our approach explores the interacting roles of data mining methods and visualization techniques. This section will introduce our noise removing method, which prepares the data for visualization through a two-step approach: first modeling the given data set with a *k*-mutual neighborhood graph; then applying a fast graph partitioning method, the *k*-core algorithm[14], to decompose the *k*-mutual neighborhood graph into small groups of data points. Then a 3D layered visualization of the small groups will be provided to the user for customizable noise removal. Section 2.1 will introduce the graph construction process. The decomposition of the graph will be described in Section 2.2.

2.1 *K*-mutual neighborhood graph

There are two commonly used graph structures for data clustering: *k*-nearest neighbor graph and *k*-mutual neighbor graph. As illustrated in Fig.1, each vertex of a *k*-nearest neighborhood graph represents a data item. For each pair of data items, if either of them is among the *k*-most similar data items of the other, there exists an edge between the two corresponding vertices. Compared with a *k*-nearest neighborhood graph, for the same data set and *k*, a *k*-mutual neighborhood graph contains fewer edges: for each pair of data items, only if both of them are among

Generally, the advantages of representing data using a *k*-nearest/mutual graph include: first, data points that are far apart are completely disconnected from the graph. Also the constructed graph is able to represent the natural density dynamically. In a dense region, the neighborhood radius of a data point is determined narrowly, and in a sparse region, the neighborhood radius becomes wide. Thirdly, the number of graph edges is linear to the number of vertices. The first two advantages ensure that the graph structure can be used to distinguish noise and true data. In particular, our experiments will show that a *k*-mutual graph suits our noise removing approach.

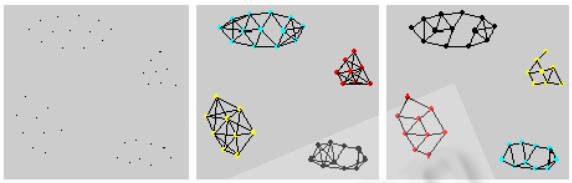


Fig.1 (a) a 2D data set; (b) its 4-nearest neighborhood graph; (c) its 4-mutual neighborhood graph

2.2 The *k*-core algorithm

The main task of the data preparation is to divide the data into different groups. In our study, the task is accomplished through partitioning the constructed *k*-mutual graph into small sets of vertices called *cores*. The notion of a core was introduced by Seidman[14]. Let $G = (V, E)$ be a graph, where V is the set of vertices and E is the set of edges. A sub-graph $H_k = (W, E \mid W)$ induced by the set W is a *k-core* or a *core of order k* iff $\forall\ v$ in W: degree $(v) \geq k$ and H_k is the maximum sub-graph with this property. The core of maximum order is also called the *main core*. The cores have the following properties:

- They are nested: $\forall\ i<j \rightarrow H_j \subseteq H_i$.
- There exists an efficient algorithm to determine the core hierarchy.
- A core is not necessarily a connected sub-graph.

The algorithm for determining the core hierarchy is simple: from a given graph $G=(V, E)$, recursively delete all vertices of degrees less than k and lines incident with them, the remaining graph is the *k*-core. Known as the *k*-core algorithm, it costs only $O(m)$ time, where m is the number of edges for the given graph[14]. The purpose of the *k*-core algorithm is to determine a *core-ID* for each vertex. Since cores are nested, a vertex may belong to multiple cores. The core-ID of a vertex is defined as the biggest order of these cores. After applying core-decomposition, the vertices are divided into different groups according to their core-IDs. The *k*-core algorithm has previously been used to produce layouts for very big graphs[19] in which its efficiency and effectiveness are verified. The combination of *k*-mutual graph and *k*-core algorithm has been shown very effective for spatial noise removal[18]. To avoid confusion, we will hereafter use k_c as the k used in *k*-core algorithm and k_m as the k used in *k*-mutual graph. For a *k*-mutual graph with n vertices and m edges, we have $m \leq k_m\ n/2$ if n is the number of vertices**, so applying *k*-core algorithm to a *k*-mutual graph requires only linear time.

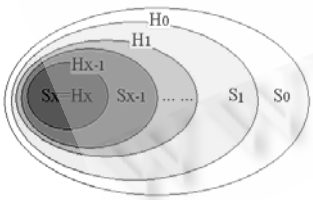


Fig.2 Example sketch map of core hierarchy

A sketch map of the core hierarchy is shown in Fig.2. Given a dataset D, suppose that the corresponding *k*-mutual graph is $G=(V, E)$, $|V|=n$ and $|E|=m$. Apply the *k*-core algorithm to G, and obtain the set of cores, denoted

** Given a data set, the corresponding *k*-nearest neighborhood graph $G=(V, E)$, and the *k*-mutual neighborhood graph $G'=(V', E')$, we have $|V|=|V'|$, $|E'| \leq |E| \leq k|V|$, and $|E|+|E'|=k|V|$, so $|E'| \leq k|V'|/2$.

by H_0, H_1, ..., H_{x-1}, H_x, where H_i represents the core of order i. The higher the order is, the darker the area is. To understand the concept of core-ID, let $S_x = H_x$, $S_{x-1}=H_{x-1}-H_x$,…, $S_1 = H_1 - H_2$, $S_0 = H_0 - H_1$, and $S_x = H_x$ is the main core, a core-ID of a vertex is i if and only if the vertex belongs to S_i.

So far there have been two parameters that would affect the result of noise removal: k_m and k_c. While choosing k_c through visualization will be discussed in Section 3, Section 4 will provide a visual support for the selections of both of them. Applying the core decomposition on the k-mutual neighborhood graph constructed on the given data set will be referred to as *core-based noise removal* in this paper. Then we visualize the core hierarchy to allow an overview on the data distribution and customizable noise removal. Section 3 will also describe this idea.

3 Separating Noise from True Data

This section presents the visualization model that assists noise removal through the application of the k-core algorithm. The model visualizes the output of the k-core algorithm and plots the data points in their original scale and dimension so that users can recognize and specify domain-specific noise through the system.

3.1 Motivation

Spatial databases usually store and manage geometric, geographic, or spatial data that describes natural or man-made space. They may contain various types of noise in different sizes, shapes, and densities. Defining what noise is becomes a domain-related problem so that no general method can be applied. Figure 3 shows a benchmark data set used by CHAMELEON[11], where the thick horizontal line crossing "GEORGE" is domain-specific: it should be regarded as noise in area of letter recognition; but it could also be true data representing a style or decoration, depending on different applications. No general definition of noise based on density or size can be applied in such a case. Besides, since the thick line not only overlaps the true data but also has a similar density, it poses a challenging problem for existing noise removing methods. Therefore, user participation becomes inevitable.

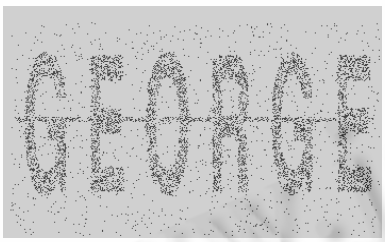


Fig.3 Benchmark data set DS1 that contains domain-specific noise

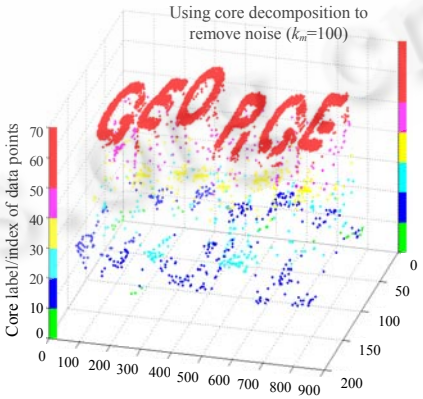


Fig.4 Visualization of the core hierarchy of DS1

3.2 The 3D layered visualization model

Given that we have two or three (for 3D data) data dimensions and four graphical dimensions: x-axis, y-axis, z-axis, and color, the coordinate assignment is as follows. For 2D data sets, the two data dimensions are mapped to x- and y-axis, respectively, and the core-ID is mapped to z-axis. Besides, the core-ID is mapped to color in a multiple to one manner: 10 continuous core-IDs correspond to one kind of color. For a 3D data set, the three data dimensions are mapped to x-, y-, and z-axis, respectively, and the core-ID is mapped to a color in the same manner

Figure 4 illustrates the 3D layered visualization of the core hierarchy of the 2D data set shown in Fig.3. Points of different cores are represented in different layers and with different colors. While different cores mean different densities or sizes, the data set is "segmented" into many small groups. Users can judge which groups are noise easily by visual inspection. As shown in Fig.4, the thick line crossing "GEORGE" has been separated into a layer far below the true data and the 3D visualization allows the user to select an appropriate threshold to remove noise easily and intuitively. There are two operating modes available to the user through the visualization. The basic mode accepts a user input defining the boundary layer, i.e., k_c, and then the layers above k_c will be regarded as true data and kept while other layers will be removed as noise. In the advanced mode, the system can accept a set of user inputs to specify which layers are true data and which are noise. Thus noise removal becomes a customizable process. For example, in Fig.4, the user can keep the thick line by setting its layer as true, or the user can even remove the right foot of the letter "R" if he/she thinks the correct word should be "GEOPGE".

For a 3D data set, the system assigns the three data dimensions of the given data set to x-, y-, and z-axis, respectively, and uses color to represent the fourth dimension of core-ID in distinguishing different cores. Users can specify noise groups by assigning different colors. As shown in Fig.5, the synthetic 3D data set consists of 10000 3D vectors which form a sphere and three lines penetrating through the sphere. The three data dimensions correspond to the three system coordinates respectively. The k-mutual neighborhood graph constructed on the data set consists of two different cores: the first one includes the vectors that form the three blue lines and the second core includes those that form the sphere. Users can distinguish them by their colors. Either the line or the sphere could be domain-specific noise for different applications. Users can choose to remove the lines or the sphere based on their domain requirements.

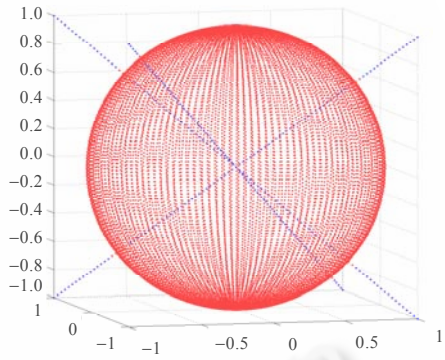


Fig.5 3D example visualized in cores

3.3 Rotation and slicing

Both the basic and advanced modes in the layered display allow users to use the rotation and slicing operations of the 3D visualization to remove noise interactively. Rotation helps users to see the exact layered structure of the data. Users can obtain the core-ID of the data group in which they are interested by changing the angle of view to bring the group to the front and clicking the mouse on it. Then a simple query can be used to extract the group according to its core-ID. To illustrate the process, suppose the correct word is "GEOPGE" for a domain user A, who first discovers the core-ID of the right foot of "R" is 52 through rotation. Then he/she can use the following query to get the desirable data: ((*coreID*>52) and (*coreID*<70)) or ((*coreID*>45) and (*coreID*<52)). The query result includes two slices: the slice between 53 and 69 and the one between 46 and 51. The system will return the query result in a 2D surface, as depicted in Fig.6.

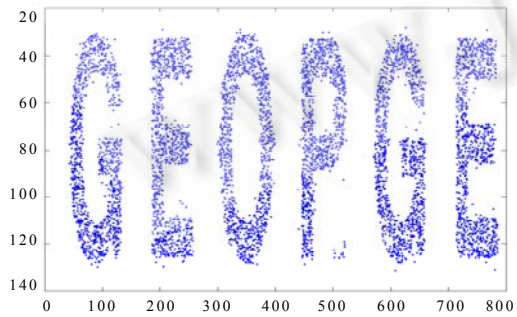


Fig.6 Data slice of DS1 for one user

Suppose there is another user B who thinks "GEORGE" is what he/she needs. He/she only needs to change the

above query to include the slice with core ID=52, i.e., the right foot of "R", to get the desirable result, which is shown in Fig.7.

3.4 Visual comparison of different noise removing methods

Another desirable feature of the layered model is its applicability to other data clustering methods, so that it can be used to compare the results of our core-based noise removing technique and any other technique in an integrated view. This feature works only for 2D data sets. The idea is to apply the *k*-core algorithm before applying the other noise removing method to be compared. The results of the two techniques are thus merged into a single view. For core-based noise removal, true data and noise are separated into different levels. For the other method to be compared, different colors are assigned to true data and noise. Since both the colors and the levels of the data points are shown in a single view, it becomes immediately clear which method is more effective than the other. The display, therefore, provides intuitive comparison and quality measurement between different noise removing methods. The comparison between different results in an integrated view is more accurate and convenient than the visual comparison on two separate views.

Fig.7 Different data slice of DS1

Figure 8 visualizes the noise removing result of DBSCAN[20], a well-known density-based clustering algorithm. The data is still presented in multiple layers according to the core-IDs of data points. Color is used to represent the result of DBSCAN: The red points mean true data and the black points are noise. Such an integration view shows the difference clearly between the core-based noise removal and the method used for testing and comparison.

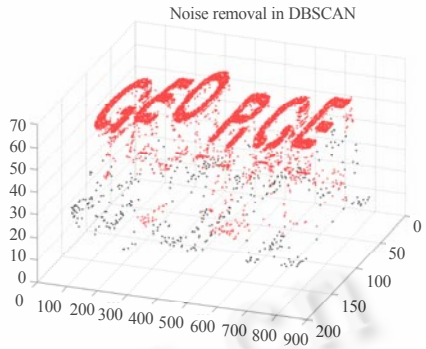


Fig.8 Visual comparison of results of DBSCAN and core-based removal

Although the layered model can help removing domain-specific noise, it cannot visualize higher dimensional (over 3) data. Also it can intuitively guide the user on selecting k_c but not k_m. The following section introduces another visualization technique to assist users in such parameter tuning.

4 Parameter Tuning

This section will propose a waterfall visualization model to allow an intuitive parameter tuning for representative clustering methods. The basic idea of the waterfall model is to visualize the relationship between the algorithm output and the parameters so that the user can select right parameters intuitively. This model not only avoids direct visualization of higher dimensional data but also provides an overview of the data characteristics useful for noise removal.

4.1 Motivation

Parameter tuning has always been a challenging issue in data clustering. Many current data clustering algorithms hardwire some algorithm parameters to the values that are identified through ad hoc and try-and-error experiments. For example, DBSCAN[20] uses two parameters: *Eps* and *MinPts* for its noise removal. The neighborhood within a radius *Eps* of a given object is called the *Eps*-neighborhood of the object and an object with at least *MinPts* of objects within its *Eps*-neighborhood is called a core object. Then noise is defined as non-core

objects which do not lie within the *Eps*-neighborhood of any core object. It is easy to understand that the selection of *Eps* and *MinPts* would be difficult for unknown data sets while the selection would affect the result of noise removal significantly. Other clustering methods that use similar parameters include OPTICS[9], DENCLUE[21], and SNN[10]. All of these algorithms require two parameters to complete their noise removing processes and the two parameters appear to be related to each other on the output of the algorithm. If we can work out a visualization way to assist the parameter tuning for one of these methods, it can also be applied to other methods.

With the core-based noise removing technique, two parameters affecting the result are k_m and k_c, as described in Section 2.2. To study the relationship between the output of the *k*-core algorithm and the two parameters, we started our experiments also by try-and-error on many benchmarks. Our experiments reveal an important phenomenon: for a wide range of k_m, there exists a corresponding k_c so that the pair (k_m,k_c) removes noise effectively. This phenomenon, however, does not help solve the parameter selection problem completely because: first, this property may not be general enough for all kinds of data sets. Second, even if k_m can be chosen freely, it is difficult to select the correspondingly k_c correctly when the data is high-dimensional, which the current 3D layered visualization does not support.

We therefore propose the waterfall model to visualize the relationship between the parameters and the algorithm output. The two parameters are mapped to two axes of a 3D coordinate system. The ratio of the size of the data after noise removal to the original data size represents the third axis. An important property of this model is its independence from the dimensionality of the given data because it visualizes the noise removing method instead of data.

4.2 The waterfall visualization model

The coordinate system of the waterfall model has four graphical dimensions: *x*-axis, *y*-axis, *z*-axis, and color, corresponding to k_m, k_c, the ratio of the data sizes before and after noise removal (denoted by *r*), and the five colors simulating a waterfall. The colors, from light green to dark blue, correspond to the amount of true data. The more true data identified, the bluer the graph appears. This allows a quick capture of the data quality, i.e. if the waterfall appears to be blue, it means the data set contains little noise.

Figure 9 shows the resulting visualization for the benchmark data set shown in Fig.3. To save computation time, each time we increase the value of k_m by ten or k_c by five and find how many data points are removed. The increment of k_m and k_c determines the size of the mesh grid, as shown in Fig.9. The relationship between the algorithm parameters and the output can be described as follows. When fixing k_m, the size of the remaining data decreases as k_c increases. The dropping of the size of the remaining data becomes dramatic when the true data has been removed, which produces a waterfall style of visualization. The sudden dropping can be justified as follows. Because true data usually are of large size and have similar properties on distance or shape, the parameter change will make either few of them or most of them removed, which will not be a gradual process. When k_m increases, the connections among both noise and true data will increase, thus a corresponding bigger k_c is required for the removal of the same size of noise.

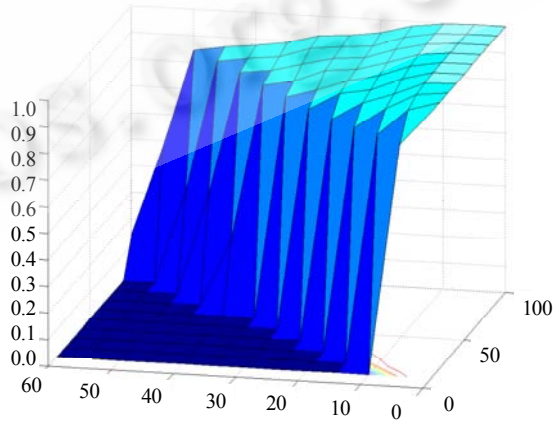


Fig.9 Waterfall model for DS1 with the core-based removal

The waterfall visualization, on one hand, supports our hypothesis that for a wide range of k_m there is a k_c to remove noise effectively, on the other hand, detects the noise ratio in this data set is about 15% because the waterfall drops down dramatically at z=0.85. This is very useful for data mining methods as prior knowledge about the given unknown data. The visualization can also measure the quality of a noise removing method. If a noise removing algorithm is not sensitive to its parameters, for a wide range of one of the parameter, there should be a value for the other parameter so that the pair can remove noise correctly. Shown in the visualization, the waterfall should drop at similar place if the algorithm is insensitive to parameter selection because the noise ratio is a fixed value for a given data set. For example, in Fig.9 all lines drop at about 0.85.

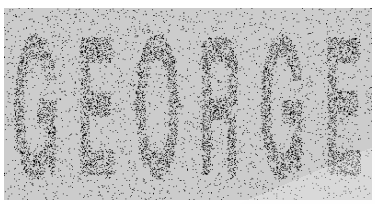


Fig.10 DS2: Another benchmark

Generally, the shape of the waterfall is affected by both the method and the data set. Figure 10 shows DS2, another benchmark data set. Its waterfall model, illustrated in Fig.11, shows a different waterfall outline from that of DS1. The comparison between Fig.9 and Fig.11 indicates that DS2 has a little higher noise ratio than DS1.

All the above features of waterfall can be seen clearly through two 2D projections on the back surface and on the bottom surface, which will be presented in the following section.

4.3 2D projections

We first project the 3D waterfall of DS1 to the surface decided by z-axis (r) and y-axis (k_c), i.e., the back surface, as illustrated in Fig.12. Similarly, DS2 also has the projection, as shown in Fig.13. From the comparison between Figs.12 and 13, we can discover information about the DS1 and DS2. Firstly, the graph of DS1 has a bigger blue area than that of DS2, so DS1 contains less noise. Secondly, the waterfall of DS1 drops at about z=0.85 while that of DS2 drops at about z=0.77, which represents the approximate noise ratio of them. Thirdly, all lines of the waterfall of DS1 drop at similar place while those of DS2 vary a little, which means the parameter selection for DS2 is harder than for DS1. Lastly, the waterfall of DS2 drops twice, implying that there exists a large amount of noise in DS2 that differs greatly from the true data on distribution, which makes the noise removal a sudden drop instead of a gradual process as k_c increases. This matches the generation process of DS2 exactly because DS2 is a combination of the true data of DS1 and 3500 random noise points which has a much smaller density compared with the true data. Thus the first drop represents the removal of these noise points totally and the second drop removes the true data. In contrast, DS1 contains the thick line which reduces the density difference between noise and the true data, so the removing process is a gradual one.

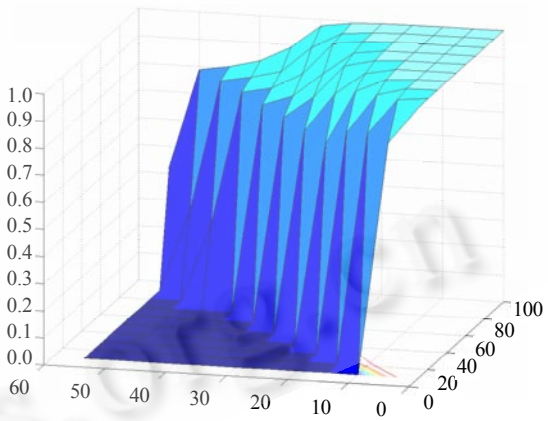


Fig.11 Waterfall model for DS2

Now let us study the relationship between the two algorithm parameters through another 2D projection on the surface decided by x-axis (k_m) and y-axis (k_c), i.e., the bottom surface. The 2D projection is obtained through cutting the waterfall horizontally at the dropping position. For example, for DS1, we cut the waterfall with the horizontal surface z=0.85 and get the slice, as shown in Figure 14, in which we can find that the relationship between k_m and k_c is very close to the function k_m =2k_c. This relationship can be used to select the parameter when the other parameter

can be fixed. All pairs (k_m, k_c) satisfying k_m =2k_c would remove the 15% noise effectively. By checking the slice of the waterfall model, users are freed from the complicated parameter tuning job.

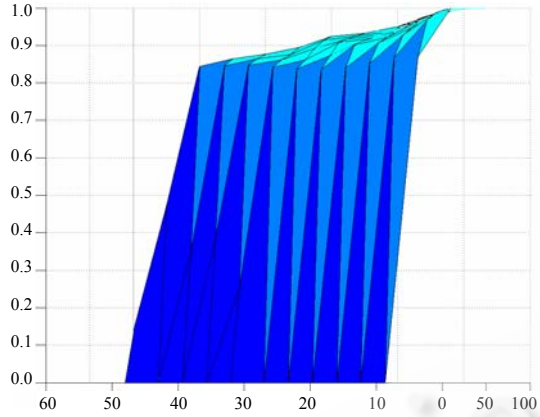


Fig.12 2D projection of DS1 on the back surface

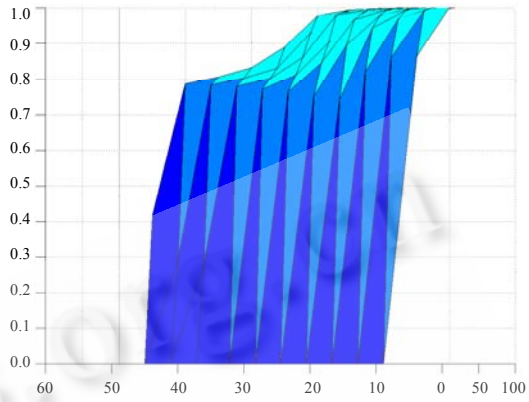


Fig.13 2D projection of DS2 on the back surface

4.4 Visual comparison of different clustering methods

Using similar coordinate assignments, the waterfall visualization model can be applied to most density-based noise removing methods. We produced one for DBSCAN through the following mappings: *Eps* and *MinPts* are mapped to *x*-axis and *y*-axis, respectively, and z-axis still represents the ratio of the size of the remaining data to the original data. The 3D waterfall visualization for DS2 with DBSCAN is shown in Fig.15 while the 2D projection on the back side is in Fig.16. A comparison between the visualizations of the two different methods leads to the measurement of the methods on the parameter sensitivity. The inconsistent dropping position shown in Figs.15 and 16 indicates that DBSCAN is more parameter-sensitive than the core-base noise removing approach.

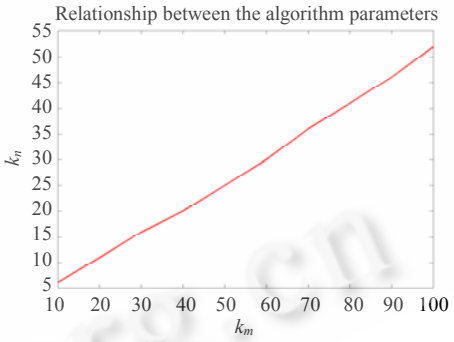


Fig.14 Relationship between k_m and k_c for DS1: A projection on the bottom surface

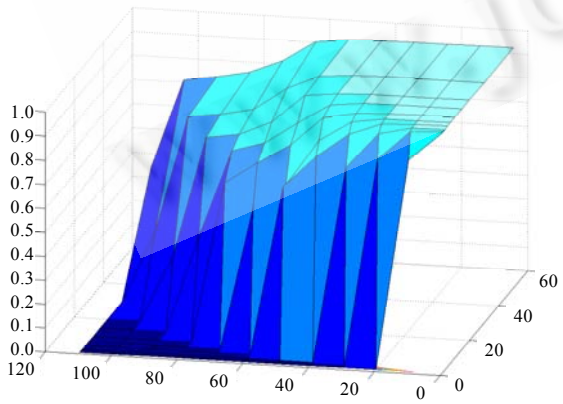


Fig.15 3D Waterfall model of DS2 with DBSCAN

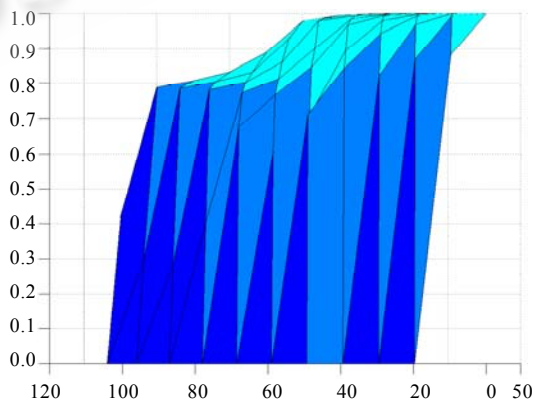


Fig.16 2D projection in waterfall of DS2 with DBSCAN

In summary, the waterfall visualization has four advantages:

- It supports high-dimensional data because it does not show the data directly but the method parameters and results.
- It shows the approximate noise ratio for a given data set, which makes later knowledge discovery and information understanding effective.
- It measures the quality of a noise removing method on parameter sensitiveness.
- It produces a relationship graph between the two algorithm parameters to assist parameter tuning. Users can choose the appropriate pair of parameters based on the visualization and the approximate noise ratio.

5 Visualized Data Clustering

This section demonstrates how visualization models are used in producing user-oriented clustering results with CLEAN. Each clustering step will be visualized to allow users to evaluate the whole process. Section 5.1 will introduce the preprocessing stage where CLEAN is used. The group merging is described in Section 5.2.

5.1 Visualized noise removal and data compression

The preprocessing includes two operations: data cleaning, and compression. Data cleaning, i.e., noise removal, is used to guarantee a correct clustering, while data compression aims at accelerating the later steps. The benchmark data set[11] used in this section is shown in Fig.17.

Fig.17 DS3: Benchmark data set to be clustered

Fig.18 20-Mutual neighborhood graph from DS3

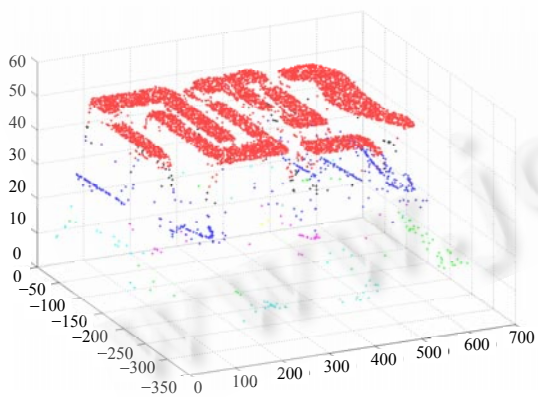


Fig.19 Core hierarchy of DS3

The noise removing process consists of three steps:

- Constructing a k-mutual neighborhood graph on the data set, as illustrated in Fig.18.
- Applying the k-core algorithm to decompose the graph into smaller groups.
- Choosing an appropriate k_c to remove noise with the help of 3D visualization of core hierarchy, shown in Fig.19.

The layered visualization model is applicable only if the data set is 2D or 3D. For higher dimensional data, we apply the waterfall model to discover the relationship between the parameters so that the appropriate pair of parameter can be chosen. The resulting DS3 after noise removal is shown in Fig.20. A data compression method called GraphZip[22] is applied on the clean data to produce a compact representation of the data so that the later clustering process can be accelerated, as shown in Fig.21.

5.2 Animated group merging and result visualization

After preprocessing, each data point of the compressed data set represents a small group of the original data

points. The final clusters can be produced through merging small groups in a pair by pair way until the total group number decreases to a desirable value. Such a process is called hierarchical merging. Figure 22 shows the different groups represented with different colors. When two groups are merged, their colors will become the same. We implement the merging process through animation so that users can find which pair of groups is being merged and decide to terminate the merging process.

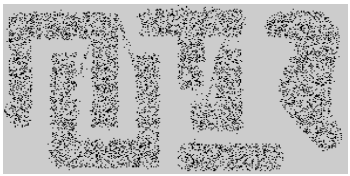


Fig.20 DS3 data after noise removal

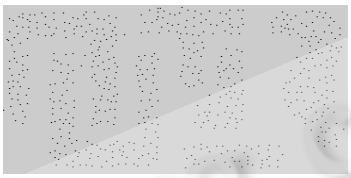


Fig.21 Compressed result of DS3

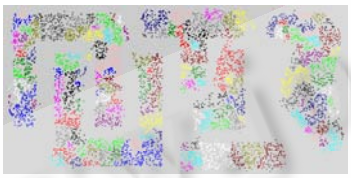


Fig.22 Subgroups before merging

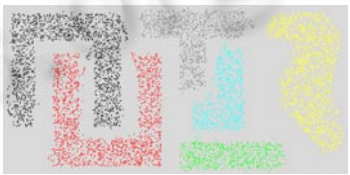


Fig.23 Final clustering result of DS3

During the hierarchical merging process, a criterion is needed to judge which pairs of groups should be merged, as described in following: we continuously choose the most appropriate pair, from the initial groups, to merge until user termination or reaching one cluster. At each hierarchical level a value M is computed for each pair of groups, denoted by $M(i,j)$ for groups i and j. The hierarchical process merges the pair of groups that maximizes M at each hierarchical level until all the groups have been merged, and there is a combination criterion to specify how to compute M. The value M is computed based on the k-mutual graph constructed on the original data set. Each data point of the initial groups has its corresponding vertex in the k-mutual graph. Suppose the k–mutual graph is $G=(V, E)$ and $S_1, S_2, \ldots, S_t$ are sets of vertices corresponding to t initial groups, we denote the number of edges between S_i and S_j as $E(i,j)$, and the size of S_i is defined as the number of vertices in S_i, denoted by $|S_i|$. The combination criterion is defined as follows:

$$M(i,j)=E(i,j)^2/\mathrm{MIN}(|S_i|,|S_j|) \quad (1)$$

Formula (1) favors the number of connections between two groups over their sizes. The more connections, the more likely the two groups will be merged. On the other hand, if the connection is the same for two pairs of groups, formula (1) will merge the pair containing the smallest group first. Formula (1) favors adding points to a big group as long as the number of the points being added is small enough. Thus formula (1) can add small groups of points to the clusters continuously. According to the definition of formula (1), small groups will be merged into big groups in an order according to the number of connections. Using formula (1) will not merge two natural clusters until all of their sub-clusters have been merged.

The final clustering result of DS3 is shown in Figure 23 after merging the number of groups into 6. The 6 natural clusters contained in DS3 have been discovered successfully. Apart from DS3, we have also applied the visualized clustering approach to two other benchmark data sets used in CHAMELEON[11] that contain noisy spatial data. Figure 24 shows the data sets in (a) and (c) and their corresponding clustering results in (b) and (d), respectively.

5.3 Result comparison

The final clustering results of the three data sets DS3, DS4, and DS5 are illustrated in Fig.25. Each produced cluster has its own gray level. Figure 25 (a) shows the clustering results of CHAMELEON while Fig.25 (b) lists the results of CLEAN. Since CHAMELEON experimentally outperformed previous systems on cluster quality, this section compares the experimental results of CLEAN with CHAMELEON only. Although the clusters in Fig.25 (b) appear very similar to those in Fig.25 (a), CLEAN runs faster than CHAMELEON, requires less user-supplied parameters, and most importantly, is able to remove noise.

Then we compare different noise removing methods. The methods to compare include the core-based noise removal in CLEAN and the degree-based noise removal proposed in SNN[10]. The comparison result is shown in Fig.26, which shows that the ambiguous noise cannot be completely removed by SNN. Besides, with the support of the visualization models, the parameter selection of CLEAN is much easier than that of SNN.

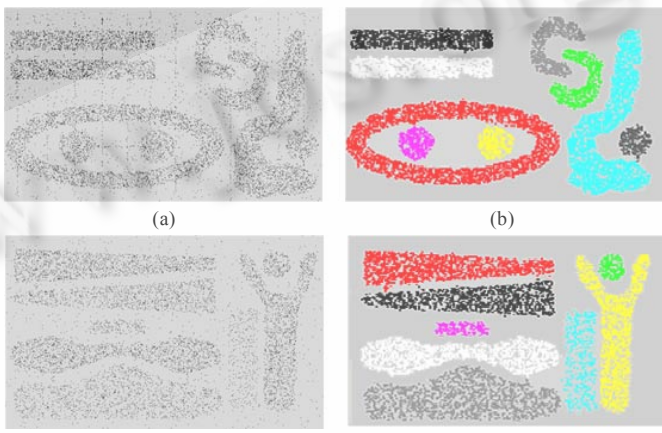


Fig.24 (a) DS4; (b) clustering result of DS4 with 9 clusters discovered; (c) DS5; (d) clustering result of DS5 with 8 clusters discovered

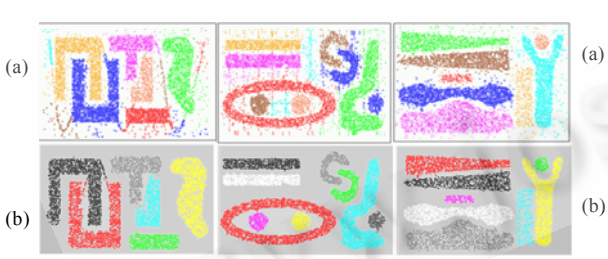


Fig.25 Clustering results of (a) CHAMELEON; (b) CLEAN

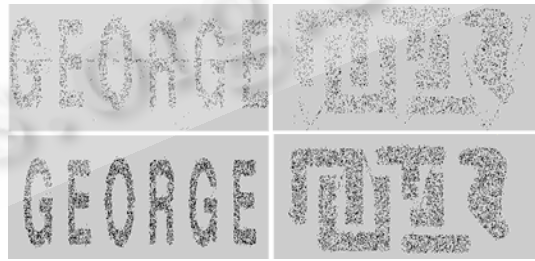


Fig.26 Noise removal with (a) SNN; (b) CLEAN

6 Conclusion

For existing data clustering algorithms, noise removal is a necessary but challenging process. This paper presents a visualization-based approach to parameter tuning and domain-specific noise removal problem. We devised two visualization models, which allow customizable noise removal and can assist discovery of relationship between the algorithm output and its parameters. In particular, the waterfall model can be used to measure the data quality by outputting the noise ratio of the data set. Also, different noise removing methods can be measured and compared at the same time by applying the waterfall model on them. Based on these visualization models and

core-based processing techniques, this paper described a visualization-based data clustering procedure. The experiments demonstrate that the visualization techniques and core-based methods are complementary to each other and the natural clusters can be identified effectively. Future work will be focused on utilizing visualization techniques to address challenging mining problems including high-dimensional data clustering. At the same time properties of the waterfall model will be further studied to evaluate whether it can be applied to more data analysis methods.

References:

[1] Fayyad UM, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery: An overview. In: Fayyad UM, *et al*., eds. Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press. 1996. 1−36.

[2] Fayyad UM, Uthurusamy R. Evolving data mining into solutions for insights. Communications of the ACM, 2002,45(8):28−31.

[3] Kopanakis I, Theodoulidis B. Visual data mining modeling techniques for the visualization of mining outcomes. Journal of Visual Languages and Computing, 2003,(14):543−589.

[4] Ward MO, Zheng J. Visualization of spatio-temporal data quality. In: Proc. of the GIS/LIS. 1993. 727−737.

[5] Fayyad UM, Grinstein G. Information Visualization in Data Mining and Knowledge Discovery. Morgan Kaufmann Publishers, 2001. 182−190.

[6] Inselberg A. Data mining and visualization of high dimensional data. In: Proc. of the Workshop on Visual Data Mining. 2001. 65−81.

[7] Han J, Kamber M, Tung AKH. Spatial clustering methods in data mining: A survey. In: Miller H, Han J, eds. Geographic Data Mining and Knowledge Discovery. Taylor and Francis. 2001. 1−29.

[8] Hinneburg A. Density-Based clustering in large databases using projections and visualizations [Ph.D. Thesis]. Konstanz: University of Konstanz, 2002.

[9] Ankerst M, *et al*. OPTICS: Ordering points to identify the clustering structure. In: Proc. of the 1999 ACM SIGMOD Conf. on Management of Data. ACM Press, 1999. 49−60.

[10] Ertoz L, Steinbach M, Kumar V. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In: Proc. of the 3rd SIAM Int'l Conf. on Data Mining. Society for Industrial & Applied, 2003. 47−58.

[11] Karypis G, Han E, Kumar V. CHAMELEON, a hierarchical clustering algorithm using dynamic modeling. IEEE Computer, 1999, 32:68−75.

[12] Hinneburg A, Keim DA, Wawryniuk M. HD-Eye: Visual mining of high dimensional data. IEEE Computer Graphics and Applications, 1999,19(5):22−31.

[13] Sprenger TC, Brunella R, Gross MH. H-BLOB: A hierarchical visual clustering method using implicit surfaces. In: Proc. of the IEEE Visualization. IEEE CS Press, 2000. 61−68.

[14] Seidman SB. Network structure and minimum degree. Social Networks, 1983,5:269−287.

[15] Zhang T, Ramakrishnan R, Linvy M. BIRCH: An efficient data clustering method for very large databases. In: Proc. of the ACM SIGMOD Conf on Management of Data. ACM Press, 1996. 103−114.

[16] Wong YF. Nonlinear scale-space filtering and multi-resolution system. IEEE Trans. on Image Proc., 1995,4(6):774−787.

[17] Qian Y, Zhang G, Zhang K. FACADE: A fast and effective approach to the discovery of dense clusters in noisy spatial data (demo abstract). In: Proc. of the ACM SIGMOD Conf on Management of Data. ACM Press, 2004. 921−922.

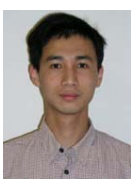
[18] Qian Y, *et al*. Visualization-Informed noise elimination and its application in proc. high-spatial-resolution remote sensing imagery. Computers and Geosciences, 2008,34:35−52.

[19] Batagelj V, Mrvar A, Zaversnik M. Partitioning approaches to clustering in graphs. In: Proc. of the Graph Drawing 1999. LNCS, 2000. 90−97.

[20] Ester M, *et al*. A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining (KDD-96). AAAI Press, 1996. 226−231.

[21] Hinneburg A, Keim DA. An efficient approach to clustering in large multimedia databases with noise. In: Proc. of the 4th Int'l Conf. on Knowledge Discovery and Data Mining (KDD-98). AAAI Press, 1998. 58−65.

[22] Qian Y, Zhang K. GraphZip: A fast and automatic compression method for spatial data clustering. In: Proc. of the 2004 ACM Symp. on Applied Computing (SAC 2004). ACM Press, 2004. 571−575.



QIAN Yu is currently a senior research associate at University of Texas Southwestern Medical Center at Dallas, USA. His research areas are data mining, data integration, bioinformatics and software visualization.