Spatial data clustering with boundary detection

Liu, Dong Quan

2008

Liu, D. Q. (2008). Spatial data clustering with boundary detection. Doctoral thesis, Nanyang Technological University, Singapore.

https://hdl.handle.net/10356/41852

https://doi.org/10.32657/10356/41852

SPATIAL DATA CLUSTERING WITH BOUNDARY DETECTION

LIU DONGQUAN

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University

in fulfillment of the requirement for the

degree of Doctor of Philosophy

2008

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor, Dr. Olga Sourina, for her patience, her great support and valuable advices throughout the course of this study.

I also would like to thank Dr. Gleb. V. Nosovskiy from Moscow State University, for his fascinating discussions and advices during his visit to NTU as Tan Chin Tuan Exchange Fellow. He helped me a lot in problem solving with his broad knowledge in mathematics.

Special thanks go to my family, who always supports me and encourages me with great patience whenever and wherever needed.

My appreciation also goes to my friends Wang Qijie, Shi Chao, and Xie Wei, for their generous assistance and sharing.

Thanks are also dedicated to all of my friends, faculties, and technicians in SWE Lab who made my life in NTU enjoyable.

Last but not least, scholarships granted by School of Electrical and Electronics Engineering, Nanyang Technological University are gratefully acknowledged.

Contents

Summary

To deal with various types of data from diverse areas, where the amount of data is still increasing dramatically, data mining has become one of the fastest growing fields in computer industry in last decade. Data mining aims at discovery of knowledge implicitly stored in information repositories. Among all data mining tasks, clustering is an approach to distinguish different groups of data (clusters) without given class labels. It is also named unsupervised learning in artificial intelligence (AI) research field. In this study, we focus on clustering methods for analyzing spatial data, especially on low dimensional data. Applications in large spatial databases, multimedia, point-based graphics, etc brought new requirements for spatial clustering such as automatic discovering of arbitrary shaped and/or non-homogeneous clusters, detecting various types of outliers and building cluster boundaries, etc. To fulfill such new requirements generated from different spatial applications we proposed and implemented three novel algorithms. Our research is concentrated on 2D algorithms implementation.

In the study when the connectivity between data points is the most concern, we propose novel clustering algorithms based on specially constructed adaptive functions. The adaptive function-based clustering method applies functions as

influence functions to simulate a relationship between data points in data set. Different features of the whole data set can be described by the field function which is built by the sum of influence functions of each data point. The clustering procedure is executed according to threshold value of the field function. The introduced adaptive parameters of influence functions are employed to localize the field function based on both local and global data distribution. By this way, the derived algorithms can handle the data sets with very complex shape clusters and outliers automatically, hence providing very good results. For special applications when the user has complete knowledge of the data set in advance, he/she can tune the algorithm with the introduced control parameters implying both global and local views on the data set. Another advantage of our algorithms is the ability to build cluster boundary which is important in spatial clustering applications. Because our spatial clustering model shares many principles with function-based solid model in computer graphics, the proposed algorithms allow integrated clustering process with visualization that is proven to be very useful for data analysis. Two types of influence functions are studied in our research such as Gaussian and piecewise linear functions. The Gaussian function is more suitable for data set with Gaussian distribution and for smooth visualization, and piecewise function is more efficient for general cases due to the low time complexity which is linear to the number of data points.

In the cases when statistical features of data point inside its proximity and of the whole data set are the most concern, we propose a novel algorithm TRICLUST based on triangulation. The neighborhood built by a graph generated

from Delaunary Triangulation is considered as the proximity for each data point. Statistical features of the data point in its neighborhood and statistical features for the whole data set are calculated. The criteria function is built for describing the statistical characteristics for each data point based on both global and local views. The clustering process is executed according to the criteria function threshold, which is chosen automatically. The clustering process is executed according to that threshold based on which all data points are classified into inner cluster points and boundary points so that the multiple dimensional clustering problem is converted to one dimensional classification problem. The main advantage of this algorithm is the ability to discover clusters of non-uniform densities and to handle the data set with "short bridges", which are referred to the connections formed by noises. This algorithm is also fully automatic and efficient for large data set. Moreover, it can also build the boundary for each cluster since all points on its boundary have already been detected. The time complexity of TRICLUST is $O(NlogN)$, where N is the number of data points.

The performance of our novel algorithms is analyzed in both theoretical and experimental way. We tested them with several computer simulated data sets and applicable benchmark data sets. The results are also compared with several well known clustering algorithms to show the advantages of our spatial clustering algorithms and to demonstrate the potential of the algorithms for real world applications. All those investigations have shown that the proposed algorithms have achieved superior performance and can discover clusters of arbitrary shapes and diverse densities, adequately capture clusters boundaries, and are robust to noise.

Due to the importance of visualization for spatial clustering, we also developed a visualization system based on our spatial clustering models. It not only provides the users with clearer and intuitive understanding of data set and clustering result, but also allows the users to be involved in the clustering process that can be important or even crucial in some applications.

List of Figures

List of Tables

Chapter 1

Introduction

During the last several decades, human capabilities of both generating and collecting data have been increasing rapidly due to the widespread use of barcodes and global internet, the computerization of transactions in many areas, and advances in data collection tools. This extraordinary data expansion provides both challenges and opportunities to the researchers and scientists. While the amount of data is growing drastically, the redundancy of data increases at the same time. This means to extract useful information from the vast amounts of data becomes more and more challenging. Hence, the great demand for new and effective techniques directly results in recent fast development of data mining techniques.

Data mining [3–12], the term which is originated from the artificial intelligence (AI) research field, is often set in the broader context of knowledge discovery in databases, or KDD. It is an automated or convenient extraction of patterns representing knowledge implicitly stored in large databases, data warehouses and other massive information repositories [4]. Data mining has become not only an important research area but also one with great potential for the real-world applications. The applications spread from business intelligence such as targeted marketing and

customer profiling, to science researches such as protein structure prediction and gene function analysis in biology, and/or image processing and system optimization in engineering.

There are several main approaches of data mining techniques such as prediction techniques that are used to find the trends of data [4], association rule mining techniques [13, 14] which are used to find the relationships between data, classification [8, 15] and clustering techniques [16–21] which are used to detect the patterns of data, etc. These methods have already been successfully applied in research areas including marketing, finance, biomedical engineering, geography, image processing, etc. This thesis focuses on the clustering techniques, which distinguish different groups of data (clusters) without given the class labels. According to the data characteristics of our potential applications such as spatial databases, geospatial data processing, image processing, and point-based graphics, etc, we mainly worked on the clustering methods for 2D spatial data. The spatial data clustering approach, which is introduced in detail in Chapter 2, could also be regarded as a general approach for clustering problem, since the representation of data as vectors in data space is a very natural way and could be a general representation for widespread real world applications. Most of the existing clustering problems can be classified as spatial data clustering. Therefore, our proposed methods also could be used as general clustering techniques to be applied in different areas with algorithm adaptation according to domain knowledge.

1.1 Motivation

Clustering is a classical problem in databases, warehousing, and artificial intelligence. Clustering algorithms can be applied for similarity search, customer segmentation, pattern recognition, trend analysis, etc. Unlike classification techniques, clustering techniques require the algorithm to distinguish data sets without knowing the labels of them in advance. To achieve it, first, we need to represent data properly which means to represent the data not only revealing the nature of them but also giving the possibility to compare them. For spatial data clustering [22–24], the data is represented as a vector in the data space. Second, we have to discriminate the dissimilar data sets according to defined criteria. This is the main topic for our research. Recently, many clustering algorithms for spatial data were developed based on different models or theories. The main categories (introduced in detail in Chapter 2) of these methods are partition methods, hierarchical methods, density-based methods, grid-based methods, and other hybrid methods [4]. But the effectiveness and the efficiency of these algorithms, however, are somehow limited if they are evaluated by requirements of spatial clustering. These requirements include the issues listed below. And these requirements also serve as the typical criteria for justifying a good general clustering method.

- **Scalability**

  The clustering method should be capable to deal with huge data sets and the complexity should increase nearly linearly with data set size increase.

  Although there is a steady improvement in performance of CPUs and other

hardware, it is still impossible to analyze huge data set using algorithm with high complexity, especially for the real time problems analysis.

The classic partition methods, such as CLARANS [25] or PAM [16], suffer from lack of efficiency. The overall time complexity of CLARANS is at least quadratic due to the calculation of distances between data points. The representative hierarchical methods, such as CURE [26] and BIRCH [27], also are not quite efficient to handle large data sets. The modern graph-based methods, such as AMOEBA [28] and AUTOCLUST [29] trade the computational cost for effectiveness by utilizing triangulation.

- **Ability to discover clusters with arbitrary shapes and different densities**

  If we cannot make any assumptions in advance or put constraints on the data set to be analyzed, the shape of expected clusters can be arbitrary. This fact results in the requirement of detecting all arbitrary shape clusters. Classic partition methods, such as K-MEANS [30] and K-MEDOIDS [16], also their improved descendant CLARANS, lack the ability to deal with the clusters with concave shape. Hierarchical methods can detect clusters with irregular shapes, but the linkage methods predispose to find clusters with convex shapes. Moreover, the improved hierarchical algorithms such as CURE [26] still cannot detect clusters with very complex shape because of the limitation of representation of clusters. The density-based algorithms such as DBSCAN [31] and DENCLUE [2] can find arbitrary shape clusters relatively

efficiently. But there are still limitations for these methods, especially for the cases when the distribution of the data varies greatly between clusters and inside clusters. In the real-world applications, for example, the point-based graphics problems [32, 33] and geographical image processing tasks [28, 29], the distribution of the data could be quite different not only between clusters but also inside the same clusters. But most of the existing methods, even the new algorithms like AMOEBA [28] and AUTOCLUST [29] which can work on different densities clusters, cannot deal with these kinds of problems very well. The main reason of that is that these methods can not distinguish the local variations of data distribution. The models or parameters used to describe the characteristics of data set lack flexibility. We will give examples and discuss density changing clusters in more detail in Chapter 3.

- **Adaptive local parameters together with predefined global parameters**

  In order to handle the complex real world problems, such as non-uniform data distribution, for example, sparse clusters adjacent to high-density clusters or density changing inside clusters, we need to propose a model with parameters which do not only provide an overview of data set but also the details of it. It means the adaptive local parameters rather than only predefined global parameter should be used to design a good clustering method. When only global parameters are applied, like in DBSCAN and DENCLUE, these global parameters cannot describe the local difference of data distributions

in different parts of data set clearly. Therefore, clusters should be the results of both large scale or global effects and small scale or local effects. Clusters may share global first order effects, but also may have their own local second order effects. Only when the whole data set has relatively uniform distribution, global parameters can perform well alone. For more general situations, the use of global parameters should be minimized not only to allow local variation of data distribution, but also to detect true clusters.

- **No constraints and prior-knowledge required**

  Openshaw in [34] emphasized: "let the data speak for themselves", it means the preconditions and constraints should be minimized in exploratory data analysis. But most of the existing clustering methods share their needs for user-specified parameters or prior knowledge to produce their best results. Partitioning methods such as K-MEANS and K-MEDOIDS face the problem of prescribing the number of clusters in advance. Classic hierarchical methods, such as single-linkage and complete-linkage methods, suffer of setting the merge/ split conditions, which define when to end the clustering process. The representative density-based methods DBSCAN and DENCLUE also generate their results highly dependent on the density threshold values choices. In probabilistic model-based clustering technique, data is considered to be a sample independently drawn from a mixture model of several probability distributions. To determine the distribution of data sets may be the most difficult task in those methods. Except the cases of specific applications, where

there is a complete knowledge of the data to ensure the validity of user supply information, these nonautomatic methods could be quite unstable and inefficient because of the probability of introducing human-generated bias and the time-consuming parameters tuning procedure. So, it is important to develop new clustering method, which can automatically generate its best result according to the criteria defined by application or generally accepted.

- **Robust with regard to outlier and noise**

  For clustering problems, detection of noises or outliers is also an important task. In some applications, such as point-based graphics, the definitions of noise and outlier are different. Noise is caused by environmental factor and is not part of data. On the other hand, outlier is an original data point which does not belong to any clusters. Here, we follow these definitions and classify real world data sets into two categories: data sets with outliers but no noises and data sets with both outliers and noises. For the first type of data set, many traditional methods can discover outliers in the case when the density of clusters does not vary greatly. But it becomes quite a challenge to detect outliers when the density of clusters changes not only between clusters but also inside clusters. It means the outliers also could be local. We need to distinguish clusters from outliers by considering both global and local information. For the situation where both outliers and noises exist, the most important issue is to find all clusters correctly rather than to detect all outliers and noises. Since the appearance of noises could cause

the problem named "short bridges" [29] or "chaining effect" [17, 35], which refers to the incorrect connection built by noises added between clusters, the original connectivity between clusters may be twisted. For some applications, whether a clustering method can settle this problem could be crucial.

- **Boundary detection**

  In common clustering algorithms, boundary detection usually is not incorporated. But, the problem of cluster boundary detection arose recently. For some applications of spatial clustering, for example, in computer geometry applications where we need to compute (and draw) an accurate boundary of a geometric object determined by the set of points, boundary detection or building should be part of the whole clustering task. Classification problems also give us another example when boundary detection of clusters is really important. Knowledge of cluster boundaries makes it possible to classify new data without repeating the clustering process [4]. This advantage will make clustering algorithm more efficient. In work [36], the application of boundary extraction is demonstrated. In work [37], a definition of cluster as a solid described by a set of points endowed with influence functions was introduced. Such definition is capable not only to describe granular property of a cluster but also its boundary.

- **Integration with visualization techniques**

  Another useful feature of spatial clustering algorithm is visualization. To give the user an easy understanding of both the data set and the results

becomes more and more important for modern clustering systems [4, 38]. Visualization techniques [36, 38–41] could improve interpretability of clustering process and usability of the data. First, the user is interested in not only the results of clustering shown as label of each data point, but also in characteristics of clusters such as shapes, or relationships between clusters. For example, if there is a hole inside a cluster, the user could not notice it only by the labels. Second, because of the special requirements of different applications, there is no clustering system that could deal with all possible problems well. At the current moment, the human could still be regarded as one of the best classifiers. The criteria of human spatial discrimination are quite complex and could combine many factors such as the distance between objects, density of the areas and the integrality of shapes, etc. Learning to simulate all these factors is still very difficult for machine. Visualization offers the user an intuitive way of analysis that can help to discover data patterns and structures. Therefore, sometimes, the user should be involved in the clustering process to make it more efficient and accurate. Thus, visualization techniques could be used not only for the interpretation of the results but also for the support of the whole process of clustering in order to improve algorithm performance. For instance, the user could select the projection directions [42] for high dimensional data set. Most of the existing visualization approaches use the result generated by clustering algorithm as the input for visualization system. It is costly and inefficient. The better solution is to combine two processes together, which means to use the same model in

clustering and visualization. The model should also allow the boundary of cluster be visualized conveniently together with clustering result.

Due to all new requirements described and limitations of existing clustering methods mentioned above, as well as due to the increasing needs for effective and efficient clustering methods for both real world applications and scientific researches, there is a great demand to study spatial data clustering with boundary detection and propose novel algorithms that could be successfully applied on a wide range of tasks.

1.2 Objectives and Major Contributions

By analyzing the requirements of spatial data clustering based on different real world applications and investigating the mechanisms of main existing clustering methods, we realize that there are still many problems to be addressed and settled in this area. Our objective is to study the problems in spatial data clustering, propose new approaches to solve the problems, and design and implement novel 2D algorithms. The major contributions of our study are summarized as follows:

1. By studying the important issues related to spatial data clustering and various clustering models together with different application requirements, we classified our spatial clustering problems into two main categories: connectivity concerned situation and statistical characteristics concerned situation. In the first case, the clustering investigation is based on the connectivity between data points. For the second case, the statistical features of the

proximity built by triangulation are the most concern. Both global and local information are considered during the clustering process in our two approaches.

2. For the situation when the connectivity between data points is the most concern, we proposed new clustering algorithms based on specially constructed adaptive functions. Here, cluster is defined as an object (solid), which includes not only the inside data points but also the surrounding data space. Since our model shares many principles with function-based solid modeling in computer graphics, the derived algorithms have advantages in building cluster boundary and in integrating them with visualization systems which can assist in generation of good results in many applications. The proposed adaptive function based clustering methods apply mathematical functions as influence functions to simulate the relationship between data points. The introduced adaptive parameters of influence functions are employed to localize the field function-based on both local and global data distribution. By this way, the derived algorithms can handle the data sets with very complex shape clusters and outliers efficiently without any predefined parameters and distribution assumptions. For special applications when the user has complete knowledge of the data set in advance, he/she can tune the algorithm with the introduced control parameters implying both global and local views on the data set. Two types of influence functions are studied in our research such as Gaussian and piecewise functions. The Gaussian function is more

suitable for data set with Gaussian distribution and for smooth visualization, and piecewise function is more efficient for general cases due to the low complexity. We improved the traditional Gaussian function-based algorithm by introducing adaptive local parameters. And we proposed and implemented a novel clustering algorithm ADACLUS which is based on adaptive piecewise function. For 2D spatial data, it allows to efficiently discover clusters of arbitrary shapes and of non-uniform densities, to detect boundaries of the clusters, and its time complexity is linear to the number of data points.

3. In the cases when statistical features of data point inside its proximity and of the whole data set are the most concern, we proposed a novel algorithm TRICLUST based on triangulation. The neighborhood built by a graph generated from Delaunary Triangulation [43–45] is considered as the proximity for each data point. Statistic features of the data point in its neighborhood and statistical features for the whole data set are calculated. The criteria function is built for describing the statistic characteristics for each data point based on both global and local views. For 2D spatial data, we choose the threshold of criteria function by applying K-MEANS algorithm with $k = 2$ on its distribution. The clustering process is executed according to that threshold based on which all data points are classified into inner cluster points and boundary points so that the two dimensional clustering problem is converted to one dimensional classification problem. The main advantage of this algorithm is the ability to discover clusters of non-uniform densities and

to handle the data set with "short bridges", which are referred to the connections formed by noises. This algorithm is able to perform automatically on many complicated data sets and it is efficient for large data set. Moreover, it can also build the boundary for each cluster since all points on its boundary have already been detected. The time complexity of TRICLUST is $O(NlogN)$, where N is the number of data points.

4. Based on the introduced geometric model that allows us to integrate clustering process with visualization techniques, we implemented our visualization system using Visualization Tool Kit, VTK [1]. The users could get more intuitive and thorough understanding of target data set and clustering result by visualizing clusters as solids with boundaries. They also can cluster their data set manually based on their judgment and knowledge via our visual interactive clustering GUI. Moreover, geometric querying function is provided in order to let the users apply geometric operations on their data set and investigate specific part of the data set according to their certain interests. This system also has the potential to be naturally incorporated with virtual reality tools which have been proved very useful for many real world applications.

1.3 Application Discussion

Based on the characteristics of proposed algorithms, the possible applications for them are discussed in this section and listed as following:

1. Geo-information in two dimensions is available in large amounts, at different scales and covering many disciplines (geology, climatology, epidemiology, population, etc). In GIS (Geographical Information System), data are often kept as spatial location and values of attributes collected by locations. Spatial data mining for GIS application is to discover meaningful patterns from large data sets where an important dimension of interest is geographical location of events. Currently, availability of large amount of Geo-data made spatial data mining a very active area of research. Clustering algorithms allow create spatially connected regions corresponding to attribute/feature values or combinations of attribute/feature values. In GIS, locations of collected data are usually distributed non-uniformly. Thus, to receive spatially connected regions corresponding to some feature values or range of the values we need algorithms that find clusters with changing gradually density of locations. Our algorithms showed its value in the application we implemented. For GIS data analysis, these algorithms are able to generate useful results according to distribution of data, especially when the density of clusters varies gradually and "short-bridge" noises exist. Users can study the distribution of data or spatial relation between data by studying the generated clustering data results. The focus of study could be the coverage of areas by clusters or the connectivity between data points. For example, in applications given in Section 3.4.7 of Chapter 3 and Section 4.9 of Chapter 4, we used our algorithms on air quality monitoring stations location maps. The user can investigate the coverage of existing stations by studying the

boundaries of clusters and the generated clustering results can be used for setting new stations in the future. The user could tune parameters based on his/her domain application knowledge. The proposed application has been interested by scientific community working in this area. Our algorithms can be used in future Earth Observation projects that recently attracted funds in many countries.

Spatial co-location pattern mining is a new direction in GIS application [46] [47]. The goal of spatial co-location pattern mining is to find subsets of spatial features frequently located together in spatial proximity. The applied association-rule mining framework usually includes 2D clustering by layers, where the layers correspond to targeting features. The final patterns are received by horizontal overlaying the layers of clusters. Overlaying includes intersection of the polygon clusters boundaries, union of cluster objects located inside the resulting boundaries, and studying features relations. In this application, our algorithms have the following advantages. First, building the cluster boundary is not a separate procedure, but it is the integrated part of our algorithm. Second, our algorithms allow find clusters with different densities and clusters with gradually changing densities so that provide clusters with more various distributions for people to investigate. So users could discovery useful co-location relationship based on their requirement.

2. The proposed clustering methods for gradually changing density could be used in point-based graphics applications. Point-based graphics is a recent

field which focuses on points as the fundamental representation of surfaces. Clustering of points with gradually changing density could be used in model simplification [48] and surface segmentation [49]. The data generated by 3D scanners is usually processed by classic tessellation methods such as triangulation and simplification methods such as vertex removal. After the processing, the suddenly changing density of data point (vertex) distribution are usually caused by discontinuity of smoothness of surfaces, and gradually changing density of data distribution usually implies continuity of smoothness of the surfaces. The reason is the parts of surface with big curvature value need more data points (vertices) to represent. It means the gradually changing of data point (vertex) density is related to gradually changing of curvature. So we could use our methods to separate the different patches of the object surfaces so that those surfaces will be cut into segments according to smoothness. The patches we got have gradually changing curvature so that those are smooth ones. We also could use our algorithms to achieve further simplification by representing each patch with fewer data points (vertices) based on clustering results. For example, we could use only the peaks of field function value in ADACLUS to represent the whole cluster (patch). Then, the number of data points (vertices) of achieved model will be decreased. These results will be useful for the rendering or re-modeling processes. Further study should be done to extend our algorithms.

Although these algorithms can be used on applications mentioned above, to

improve the performance of them, we should do further research of application area to customize our algorithms, such as utilize the domain knowledge to help parameters tuning, integration of the algorithms to the frameworks, etc..

1.4 Organization of this thesis

The rest of this thesis is organized as following:

In Chapter 2, an overview of spatial data clustering is given by classifying main existing algorithms into categories and briefly introducing the main ideas of them. Other background knowledges related to this research, such as concepts and techniques in computer graphics, are also described in this chapter.

In Chapter 3, new algorithms based on adaptive functions are proposed and verified both theoretically and experimentally. First, basic ideas and concepts are elaborated, followed by models and implementation descriptions. The theoretical analysis and experimental results are shown in the later parts. There are also comparisons with some classical algorithms in the experiment section to illustrate the advantages and new features of the proposed algorithms.

In Chapter 4, a new algorithm based on analysis of information extracted from the graph built by triangulation is introduced and verified in both theoretical way and experimental way. First, basic ideas and concepts are elaborated, followed by the algorithm description. The theoretical analysis and experimental results are shown in the later parts. There are also comparisons with some classic algorithms in the experiment results analysis section to illustrate the advantages and new

features of the proposed algorithm.

In Chapter 5, a system for integrating spatial data clustering and visualization is introduced. Visualization results of the developed system implemented based on implicit model are shown to demonstrate functions of our system.

In Chapter 6, conclusions of our work are drawn and some suggestions on the possible future work are given.

Chapter 2

Research Background

In this chapter, the research background of our work is reviewed by introducing and analyzing the related knowledge and techniques.

2.1 Basic Concepts

2.1.1 Spatial Databases and Spatial Data Mining

Spatial data describes information related to the space occupied by objects [22]. It represents the geometric information of the objects in the data space. In principle, any data point or so-called object which could be represented by coordinates under certain reference frame can be regarded as a spatial data point. The space which is spanned by all data points of the data set forms the data space. The coordinates which represent the geometric information of the data point can be either discrete or continuous. In spatial clustering applications, such as spatial databases, geo-spatial data processing, image processing, and point-based graphics, etc, the dimensionality of spatial data space is usually low, mainly not higher than three.

Spatial databases [50, 51] are the database systems that store and manage spatial data. The term "spatial database system" is associated with a view of database

as containing sets of objects in space, sometimes, it is different from image database or pictorial database which manipulates and retrieves raster images as discrete entities. The spatial database is a full-fledged database system with additional capabilities for handling spatial data, it offers spatial data types in its data model and query language, supports spatial data types in their implementation and provides at least spatial indexing and efficient algorithms for spatial join.

Spatial data mining [52] is the data mining approach which applies non-trivial search for interesting and unexpected spatial pattern in large spatial databases. Spatial pattern is referred to the frequent arrangement, configuration, composition, or regularity in the distribution of spatial data set. Some examples are: diseases clusters to investigate environment health hazards, crime hotspots for planning police patrol routes, location distribution study of different air stations for certain air quality features, etc. By now, because of the dramatic increase of the amount of spatial data, it is not possible for human to completely analyze the data being collected. The needs for automatic spatial algorithms are rising rapidly. Spatial data mining techniques have been widely used in the areas from geo-spatial data processing to bio-medical knowledge discovery, such as preparing land-use maps from satellite imagery, and micro-array biomedical data analysis [52, 53]. This approach bas been proved very useful in many application domains.

Similar to data mining techniques, spatial data mining techniques also include spatial classification, spatial trend detection, spatial clustering [52, 53], etc. Among them, the spatial clustering methods are the most interesting and well developed. Since, in general, clustering techniques are mainly developed for spatial data, we

will review the general clustering approaches in the following section.

2.1.2 Clustering Techniques

Clustering techniques can be also called as unsupervised machine learning [54] or hidden patterns recognition depending on application areas. It has been widely used in many fields, such as image processing, market research, medical diagnostics, computational biology, and many others. In the research area of computer science, clustering is one of data mining techniques. Unlike classification, it is a discrimination of data without knowing the label of each object in advance. The aim of clustering is to group the data into clusters so that objects belonging to one cluster have high similarity. On the other hand, the objects belonging to different clusters are very dissimilar to one another. The dissimilarities can be different according to the types of data or applications.

Clustering methods have been proven very useful in many areas. They have been employed in data analysis areas such as geographical data analysis and image processing. Or they also have been used as preprocess methods for other data mining techniques. For example, clustering algorithms usually serve as feature discovery or feature selection methods for classification problems [16] or as identifier of similar regions for local regression analysis [4, 16]. Here, we are trying to focus on the mechanisms and models of different clustering techniques to analyze their working principles.

For a general clustering problem, the first step is to represent data properly which means to represent the data not only by revealing the nature of them but also

giving the possibility to compare them. Next step is to discriminate the dissimilar data sets according to defined criteria.

According to different applications, it is always important to preprocess our data in the first step. The preprocessing usually includes suitable type of dissimilarity measurement selection and data transformation or data integration.

The dissimilarity measurement is also called distance which is used to describe relationship between data. For numerical data, the most straightforward and commonly chosen distance is Euclidean distance. Other widely used distances are Squared Euclidean distance, City-block distance, Chebychev distance and Power distance, etc. The exact formula of each distance is give in Table. 2.1. Squared Euclidean distance is able to place progressively greater weight on objects that are further apart. City-block distance is simply the average difference across dimensions. In most cases, this distance yields results similar to Euclidean distance. However, for this distance, the effect of single large differences (outliers) is dampened since they are not squared. Chebychev distance is the biggest absolute difference among all data dimensions. This measure is a good choice in cases when one wants to define two objects as "different" if they are different on any one of the dimensions. If sometimes one may want to increase or decrease the progressive weight that is placed on dimensions on which the respective objects are very different, Power distance will be the suitable one. The data transformation or integration is to preprocess data in order to make it ready for clustering. The most common data transformation is normalization where the attribute data are scaled so as to fall within a small specified range. For distance-based clustering methods, normaliza-

Table 2.1: Commonly used distances between objects $\vec{X}$ and $\vec{Y}$ for clustering algorithms

Distances	Formula
Euclidean distance	$dist(\vec{X},\vec{Y}) = \sqrt{\sum_{i=1}^{d}(x^i - y^i)^2}$ $\quad \vec{X},\vec{Y} \in \mathbf{R}^d$.
Spuared Euclidean distance	$dist(\vec{X},\vec{Y}) = \sum_{i=1}^{d}(x^i - y^i)^2$ $\quad \vec{X},\vec{Y} \in \mathbf{R}^d$.
City-block distance	$dist(\vec{X},\vec{Y}) = \sum_{i=1}^{d}\lvert x^i - y^i\rvert$ $\quad \vec{X},\vec{Y} \in \mathbf{R}^d$.
Chebychev distance	$dist(\vec{X},\vec{Y}) = \max\lvert x^i - y^i\rvert$ $\quad \vec{X},\vec{Y} \in \mathbf{R}^d$.
Power distance	$dist(\vec{X},\vec{Y}) = (\sum_{i=1}^{d}\lvert x^i - y^i\rvert^p)^{1/q}$ $\quad \vec{X},\vec{Y} \in \mathbf{R}^d$. where p and q are user-defined parameters

tion helps prevent attributes with initially large range from outweighing attributes with initially small ranges, since it is always necessary to avoid the variable of one dimension with large values dominate the results of clustering. Generally, it is good practice to transform the dimensions so they have similar scales. But, on another hand, normalization may sacrifice the way it represents the underlying objects so that lead to discover patterns that were not obvious on the original scale. Thus, users should apply this technique according their clustering context and domain knowledge.

For spatial data clustering, the data is represented as vectors in the data space. If we view these two steps separately, in principle, all general clustering algorithms can be used to handle spatial data. So we will review the main general clustering approaches for a complete and thorough view of this area in this section.

Although there are some special requirements of clustering in reference to the different application, such as the requirements for spatial data clustering we highlighted in Chapter 1, several typical requirements [4] [19] for general clustering should be mentioned here for generality:

- Assessment of results

  The clustering system can be assessed by users or by a particular automatic procedure. The results of the processes can be evaluated based on different rules. Traditionally, the assessment relates to two issues:

  - Interpretability

    It means the results of clustering must be interpretable and comprehensible to users. When clustering methods are applied to specific application, the results may have special interpretations. Generally, the distribution of data, density of data and distance between objects are often used to check the results of the clustering processes.

  - Cluster visualization [55]

    By now, visualization of both the results of the clustering and the processes of clustering is still very important for the users to understand the data, and to help the methods to perform more efficient and quickly.

- Ability to discover arbitrary shape clusters

  In different applications, the distribution of data may vary a lot, and the shape of the clusters could be arbitrary. Then, the algorithms should be able to find clusters with irregular shapes. Moreover, the densities of clusters may be not uniform inside clusters. So, an effective algorithm should be able to detect clusters in such cases.

- Ability to deal with noises and outliers

In different research areas, noise and outlier are defined differently. Here, as mentioned in Chapter 1, we define outlier as the data object belonging to the original data set, but which can not be grouped into any clusters. On the contrary, noise is the data object added by systems during data processing procedures. It is not a part of the original data set. Since most of the real-world data sets contain noises and outliers, especially in high-dimensional cases, it is important for a good clustering method to find correct clusters from them. The ability to be insensitive to noises and outliers becomes much crucial for the quality of the results.

- Ability to find the natural clusters with minimal parameters

  Considering requirements from different users and characteristics of different data sets, clustering algorithms should be able to detect the clusters without changing the natural shapes of them with minimal inputs. The meaning of what is "natural" for a clustering result has been mentioned in [17][25] by determining the "natural" number of clusters. This means a best separation of one data set according to a defined criteria. There are many criteria based on different clustering validation theories [56] to verify the separation of one data set. But the judgement of natural separation or natural shapes of clusters still depend on specific applications. Therefore, the clusters from one data set to another may be more granular and inseparable. For many clustering algorithms, the setting of parameters serves as answers to questions such as the number of clusters. The appropriate setting of parameters

always needs the users to have more knowledge on the data set, that makes those algorithms more difficult to use. For these clustering methods, the values of parameters also could be received from heuristic analysis or tuning procedures. But, it is always difficult to set each of the parameters with a clear meaning in mathematics, and it is time consuming to apply tuning procedures for the parameters setting. Therefore, input parameters are not only the burdens for users but also lead the quality of clustering harder to control.

- Insensitivity to the order of input data

  The order of input data should do nothing with the distribution of the data set, so the clustering methods should be insensitive to the order of input records.

- Ability to deal with high-dimensional data set

  The dimensionality of many real world data sets may be quite high. This requires that the algorithms can find clusters not only in low-dimensional cases but also in high-dimensional cases. For the sparseness and non-uniform distribution of data in high-dimensional cases, it becomes more difficult and costly to find clusters [57]. Therefore, accuracy and time complexity seems to be more important to methods applied on high-dimensional data sets.

- Ability to deal with dynamic data sets

  Sometimes, in many real world databases, the data changes frequently. There-

fore, the clustering algorithms should be able to redefine the clusters easily. It means the clustering algorithm should be able to check a new data object added and redefine the clusters after adding/ deleting data object effectively. The clustering algorithms with cluster boundary construction have the advantage for handling these situations, because the new separation of the data set can be achieved by checking the changed part of the data set with cluster boundaries rather than re-executing the algorithm on the whole updated data set.

The issues mentioned above are the main requirements for a good general clustering algorithm. Based on these requirements, we will review main existing clustering methods in Section 2.2 by classifying them into categories and analyzing the mechanism and performance for each of them.

2.2 Clustering Methods Review

There are a large number of clustering algorithms. Roughly speaking, the main clustering algorithms can be categorized into six groups: partition methods, hierarchical methods, density-based methods, grid-based methods, graph-based methods, and other methods. We will review the most important clustering methods of each category in the following sections.

2.2.1 Partition Methods

Because partition methods are straightforward and easy to implement, they are the most popular clustering tools used in scientific and industrial applications.

They require the users to input parameter k (number of clusters) in advance, and according to it partition the whole data set into k clusters. Generally, the clusters are not overlapped, and each data object must be assigned into exactly one cluster (except for fuzzy partition methods). They use iterative relocation to optimize the result. Since iterative improvements such as pair-wise computation would be very costly, they use unique cluster representations to solve the problem. By different ways of clusters representation, they can be classified into two main types of methods named K-MEANS [30, 58] and K-MEDOIDS [16]. The partition methods use various types of criteria to form the clusters. The distance is the widest used one. The main idea of partition method is to let the objects that are close to each other to be in one cluster. On the other hand, the objects of different clusters are far apart from one another. Because of the cost of achieving global optimality in partition-based algorithms, local optimality seems to be a good choice for most methods of this type, especially when applying to large data sets. Let us introduce and analyze the two main types of the methods based on partition scheme in detail.

The K-MEANS method and its later version [30, 58] use the mean to represent a cluster. The similarity is measured with regard to its value, and it also can be viewed as the cluster's center of gravity. The process starts by selecting the k means randomly. Then each object is labeled to the nearest cluster represented by its mean. After that, the algorithm re-computes the means of k clusters and reassigns all data objects, until the criterion function converges. Typically, the square-error criterion is used. The K-MEANS method is sensitive to noises and

not suitable for finding arbitrary shapes clusters and clusters which density varies much from one to another.

The K-MEDOIDS method and its subsequent version such as PAM [16] are designed to reduce the sensitivity of K-MEANS method. These algorithms use the data objects, which are actual data to represent the clusters. The medoid is defined as the object, which is the most centrally located data object in the cluster. Then, by iterative process, we can find the most appropriate medoids of the k clusters. In this way, the K-MEDOIDS method becomes more insensitive to noises comparing to K-MEANS method, but the time complexity grows at the same time.

These two methods are relatively scalable and efficient in dealing with large data set. Their computational complexity could be close to linear with respect to N (the number of objects) in the best situation. Especially the improved K-MEDOIDS method named CLARANS [25] is relatively effective and efficient for large databases. These algorithms are also interpretable to users. But in general cases, the time and memory cost of the partition methods is still huge. For CLARANS, the computational complexity is about $O(N^2)$. Moreover, they do not perform well in detecting arbitrary shape clusters and have the disadvantage that these methods need the input the number of clusters in advance. The quality of traditional partition methods also depends on the initial selection of means or medoids, thus the result of clustering could be unstable. For dynamic data set, the clustering results can only be updated by rerunning the algorithm, which gives traditional partition method another drawback.

There are many new works in this approach to overcome the drawbacks of

traditional partition methods.

Recent works on K-MEANS method include [59–66]. In [61], authors employed kernel principal component analysis technique to refine the initial point selection of K-MEANS. In [62], the problem of K-MEANS of incapability to discover outliers is solved. The work in [63] establishes the connection between K-MEANS method and Occam's razor. In [64], graph theory is employed to refine the traditional partition algorithms. And in [65], authors integrated K-MEANS method with relational database management system. In [66], an improved K-MEANS method for gene expression analysis is proposed.

2.2.2 Hierarchical Methods

Hierarchical methods initialize a cluster system as a set of singleton clusters (agglomerative case) or a single cluster of all objects (divisive case) and proceeds iteratively with merging or splitting of the most appropriate cluster(s) until the stopping criteria is reached. This kind of methods can be categorized into two classes.

The first one is the traditional hierarchical methods, which can also be called linkage methods [67] [68] or geometric clustering methods. For these methods, the linkage metric that keeps the distances between objects is used to determine whether to merge or split subsets of objects rather than individual objects. The major inter-clusters linkage metrics include single link, average link and complete link. Although the idea of linkage methods reflects the common concept of closeness and connectivity, the time complexity of them is unrealistic. Formally, it is $O(N^2)$.

The second category of classic hierarchical method is the one, which is integrated with data compression or sampling methods to achieve good scalability and speed, such as BIRCH [27] and CURE [26]. BIRCH, begins with representing objects hierarchically by building the CF (Clustering Feature) tree to summarize the information about subclusters instead of storing all objects, and refines the result by other methods, such as classic partition method. It introduces two concepts: clustering feature and clustering feature tree to squash the data set. Those data structure plays an important role to achieve good efficiency and scalability in large databases applications. The CF mainly includes statistical information for a given subcluster. A CF tree is a height-balanced tree which stores these clustering features hierarchically. A new data object descends along the tree to the closet CF leaf. If the leaf node is not overloaded, the CF information is incremented for all nodes from the leaf to the root. Otherwise, the leaf node or other leaf nodes are split. BIRCH applies a multiphase structure to produce the best clustering results with limited amount of main memory by minimizing the time required for I/O. Despite the time complexity that is linear to the number of objects, BIRCH has a disadvantage that it only can find spherical shaped clusters effectively due to the notion of radius to control the boundary of a cluster. Another important hierarchical algorithm is CURE. The major feature of CURE is that it represents a cluster by a fixed number of objects scattered around its center. Such technique improved the representative-object-based approaches for representing non-spherical shape clusters. By making those representative objects shrink to the center of the cluster according to a specified fraction, CURE overcomes the problem of BIRCH in de-

tecting cluster with irregular shape. CURE uses two devices to improve scalability. First one is data sampling, but, sometimes, it also makes the result of clustering more unstable. Second, it uses partition method on the sample first to get the partial clusters, then, it refines the partial clusters in the second pass to get the final result. Though the time complexity of CURE is not straightforward, experiments show that CURE is relatively efficient to produce high-quality clusters. It can handle different shapes and size clusters, and it is insensitive to outliers. But when the shapes of clusters become very complex, for instance, like winding snakes, to represent the cluster by a certain number of objects will be quite difficult, especially, when the number of representative objects is relatively small. Moreover, the sensibility to parameters is also another drawback of CURE. Analysis showed the different setting of parameters does have a significant influence on its results.

Another famous hierarchical method is CHAMELEON [69]. It applies graph partitioning techniques at its first stage to find tight sub-clusters and explores dynamic modeling when it aggregates genuine clusters at its second stage. The information of both inter-connectivity between two different clusters and closeness of two clusters are considered when the decision of merging is made. Only when the inter-connectivity and closeness between two clusters are comparable to the internal inter-connectivity and closeness within clusters, these two clusters are combined. By utilizing k-nearest neighbor graph and defining the inter-connectivity and closeness, CHAMELEON localizes the partition of the data set to detect more natural clusters. Thus, it has more power at discovering arbitrarily shaped clusters than CURE. The data sets used in the testing experiments of CHAMELEON

becomes one of the most important benchmarks in clustering research area. But CHAMELEON still suffers from its high time complexity, especially for high dimensional applications. Its agglomerative process depends on users provided threshold, and also on parameter k for building the nearest neighbor graph. Moreover, it lacks the ability to handle outliers.

Recent works in this approach include [68, 70–75]. Relative hierarchical clustering method [70] which considers both distance between two clusters and distance from the two clusters to the rest clusters when deciding merging. In [68, 71], to decrease the computational complexity of traditional hierarchical method parallel algorithms are implemented. The work in [72] extended classic hierarchical methods to handle both numeric and nominal data. In [73], authors proposed a hierarchical shape clustering algorithm to enable a statistical analysis of shapes. In [74], an approximate nearest neighbor graph is employed to effectively reduce the computational complexity of hierarchical clustering methods. In [75], authors designed a novel hierarchical clustering method which generates the hierarchy by removing one cluster at a time according to optimization strategy. This method also can be integrated with genetic algorithm to achieve good performance.

2.2.3 Density-based Methods

From the idea that an open set in the Euclidean space can be divided into a set of connected components, the clustering operation can be implemented by partitioning of a finite set of objects using the concepts of density, connectivity and boundary. This is the way how density-based methods work. A cluster defined

by density-connected components can grow in any directions that density leads. Therefore, the density-based methods can discover clusters with arbitrary shapes. Also they can perform well against outliers by the natural properties. There are two major approaches for density -based algorithms. The first one pins density to a training data object and is reviewed in the sub-section density based connectivity. DBSCAN [31] [76] and OPTICS [77] are the representative ones of this kind of methods. The second approach of density-based methods uses influence function to describe the whole attribute space, and by the analysis of density function which is the sum of influence functions, it can find the clusters. The algorithm named DENCLUE [2] is the one that works in this way. Below, these two types of methods will be introduced.

The crucial concepts of DBSCAN are density and density connectivity, by which the method defines the neighbors of objects. It uses the following concepts:

- The neighbors within a radius ϵ of a certain object are the ϵ-neighborhood of it.
- A core object is the one, which has more than $Minpts$ objects in its neighbor.
- Density-reachablility and density-connectivity define the connectivity between common object and core object, and the connectivity between common objects.

Density-reachability is the transitive closure of direct density reachability, and it is asymmetric. And the density-connectivity is a symmetric relationship between common objects. The illustration of these two concepts is shown in Fig.2.1 [76].

With the two major concepts, the DBSCAN defines cluster as the maximal set of density-connected objects. And any objects, which cannot be covered by any clusters, are regarded as outliers. DBSCAN also relies on R-tree indexation to optimize the computation. In low dimensional cases, the theoretical time complexity is slight super-linear to N. The concepts given by DBSCAN can be extended to more general cases. But there is no straightforward way to fit parameters ϵ and $Minpts$ to data. Moreover, to discover clusters with natural shape, the setting of the parameters should be different in different parts of the data set rather than applying global parameters like in DBSCAN. All this makes the results of DBSCAN not stable enough. The subsequent algorithms like OPTICS improved DBSCAN with respect to this challenge. It builds an augmented ordering of data set which is consistent with DBSCAN. This ordering represents the density-based clustering structure and contains the information about all clustering levels of the data set. The achieved ordering can be used automatically or interactively to generate the best clustering result. As an extension of DBSCAN, OPTICS also has the same time complexity $O(NlogN)$. And there is still one user-predefined parameter $Minpts$ which gives this method another potential drawback. Another descendent algorithm of DBSCAN named DBCLASD [78] tries to solve the parameters setting problem by making assumptions on data set. It assumes the data points inside clusters are uniformly distributed which is not true for many realistic situations. Although DBCLASD can discover clusters with different densities, it is still not able to detect clusters with non-uniform inner cluster densities. DENCLUE [2] is a famous density-based algorithm, which represents another approach of density-

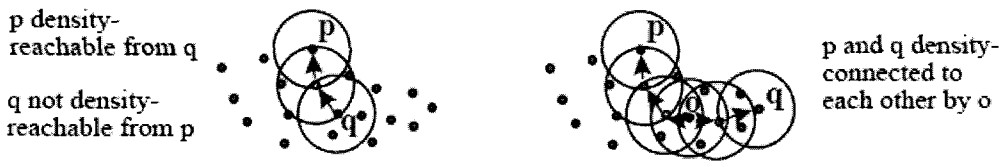


Figure 2.1: Illustration of density-reachability and density-connectivity.

based clustering methods. It is based on the idea that the influence of each object can be formally modeled using certain mathematic function which is called influence function. The distribution of data set in the whole data space can be described by density function [79] which is the sum of all influence functions of every single object. Thus, the clustering task can be converted into the mathematical analysis of the density function. Comparing with other clustering methods, DENCLUE has relatively more solid mathematical foundation and higher generality. By using different influence functions and parameters setting, DENCLUE can lead to other clustering methods such as DBSCAN and K-MEANS. The selection of influence function is also flexible; any monotone descendent function could be applied. Below there are two examples: Square wave function and Gaussian function.

$$f_{square}(\vec{X},\vec{Y}) = \begin{cases} 1, & \text{if} \quad 0 \le d(\vec{X},\vec{Y}) \le \sigma, \\ 0, & \text{if} \quad d(\vec{X},\vec{Y}) > \sigma. \end{cases}$$

$$f_{Gauss}(\vec{X},\vec{Y}) = e^{-\frac{d(\vec{X},\vec{Y})^2}{2\sigma^2}}$$

where $d(\vec{X},\vec{Y})$ is the distance between data point $\vec{X}$ and data point $\vec{Y}$.

The density function for data set D is defined as:

$$f^D(\vec{X}) = \sum_{i=1}^{N} f(\vec{P}_i,\vec{X}), \qquad \forall \vec{P}_i \in D$$

where $\vec{X}$ represents any space point, $\vec{P}$ represents any data point.

The cluster defined by DENCLUE is the area which includes the density attractors that represent local maxima of density and the sequential objects attracted by them and with the density values greater than a pre-set threshold ξ. To optimize the computation, DENCLUE applies the local maximal strategy, and uses a method of gradient hill-climbing technique for finding the density attractors. The main advantage of it is the good performance for finding arbitrary shape clusters and stability of the algorithm to outliers. In Fig.2.2 [2], arbitrary-shape clusters for different ξ are shown. By integrating with grid methods building a map of hyper-rectangle cubes, the time complexity of DENCLUE could be N sub-linearly. Thus, DENCLUE is classified into grid-based clustering method in some books. But there are still some problems in this algorithm. First, to check the connectivity of two density attractors could be difficult in high dimensional cases, because finding the proper path between them could be impossible in high dimension cases. Second, the gradient hill-climbing technique may be useless when the gradient of certain object is zero. Also, the distribution of data objects could affect the distribution of gradient field [80], it will cause problems both in time complexity and accuracy of DENCLUE. Moreover, the global parameters built in DENCLUE make it lack the ability to detect natural clusters with respect to local distribution. And the setting of those parameters still depends on the users, thus, high-quality clustering result cannot be generated by this algorithm automatically. The recent works in density-based clustering approach include [81, 82] which are descendent algorithms of DBSCAN. They extended DBSCAN to spatial-temporal data and considered density variation between clusters.

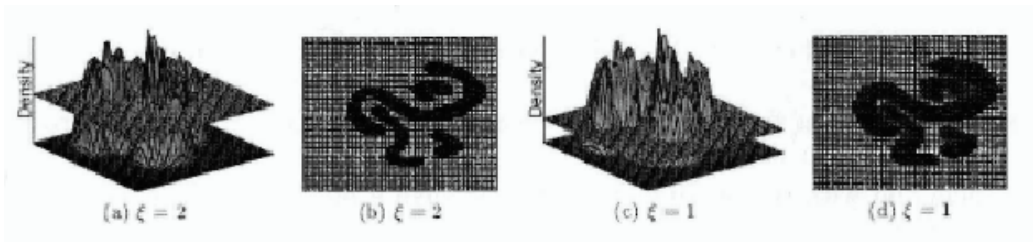


Figure 2.2: Arbitrary-shape clusters for different ξ.

2.2.4 Grid-based Methods

The grid-based clustering methods achieve the separation of one data set by applying the idea that the relationship between data objects can be inherited from the topology of underlying attribute space so that the partitioning of data objects is converted to the partitioning of data space. It first quantizes the data space into a finite number of cubes (cells) that form a grid structure on which all operations are performed. The main advantage of this approach is that the time complexity only depends on the resolution of grid, which refers to the number of cubes in each dimension in the data space, rather than the number of the data objects. It makes grid-based method more suitable for high-density data sets with a huge number of data objects in limited data space. The representative grid-based methods will be introduced in detail bellow.

The CLIQUE [83] clustering algorithm integrates grid-based method with density-based method. It is based on the fact that the data space of multidimensional data is not uniformly occupied by data objects. Thus, the cluster is formed by the maximal regions of connected dense cubes (hyper-cubes). The dense cell is defined as the unit in which the fraction of total objects exceeds an input parameter. In order to get a more accurate description of cluster, the CLIQUE generates a mini-

mal description of the maximal region that covers the connected dense cells. This algorithm uses projection scheme to find all dense cells, which generates candidates from subspaces and follows a bottom-up approach. The identification of the candidates is based on an apriori property used in mining frequent item sets for association rule mining. The property states the following: if a k-dimensional cell is dense, then its projections in $k-1$ dimensional space are dense as well. CLIQUE works in two steps: first, after partitioning the data space into non-overlapping cells, it finds all dense cells in subspaces as the candidates for higher dimensionality till all connected dense cubes are found in the whole data space. Second, CLIQUE makes the minimal description of the region form by connected dense cubes to build the final cluster. This algorithm is insensitive to the order of input data and does not need assumptions on the distribution of data set. It performs well for high-dimensional applications due to its good scalability as the dimensionality increases. But CLIQUE is sensitive to noise and the setting of parameters. It means this algorithm is not ideal for spatial data clustering.

The algorithm STING [84] is designed to facilitate "region oriented" queries. It divides the data space into rectangular cells and different levels of such cells corresponding to different levels of resolution. It assembles statistics in a hierarchical structure that each cell at a higher level can be partitioned to form a number of lower cells. The statistical information such as mean, maximum, minimum values and types of distribution are pre-computed and stored in the hierarchical tree. The statistical information of higher level cells can be calculated from the information of lower level cells. STING starts with building a hierarchical tree by going

through the whole data set to compute all statistical information for every cell. Except the types of distribution which is assigned by user if they are available or obtained by hypothesis tests, all information can be calculated directly from the data. After tree construction, the query process begins. The determination of the relevancy to the given query for a cell is made based on the confidence interval or estimated range of probability. After irrelevant cells are removed, process goes to the next lower level, until the bottom level is reached. STING has the advantages of efficiency that the time complexity is linear to the number of objects. And it supports the user defined specific query requirements due to its data summary independence for query. But the shapes of clusters boundaries are either horizontal or vertical because of the merging scheme of STING. And defining the appropriate granularity is not straightforward.

The WAVECLUSTER [85] is another successful method based on grid scheme. It applies the wavelet transformation to transform the original feature space, and finds the dense regions in the transformed space. The idea of WAVECLUSTER comes from signal processing technique. It regards the clustering system as a filter, and is based on the fact that the higher-frequency part of a signal correspond to boundaries, while the lower frequency high amplitude parts of the signal correspond to clusters' interiors. This algorithm is relatively efficient, insensitive to noise and can deal with cluster of arbitrary shape, but the complexity exponentially grows with the dimension.

2.2.5 Graph-based Methods

The graph-based clustering approach applies graph theory into data clustering area. By viewing the data set from a graph theory perspective, the division of data objects can be regarded as the investigation of the topology of the built graph. Thus, the clustering task is converted into the segmentation of the graph based on graphic features. Its early work is also referred to in the literature as so called minimum spanning tree or MST [86], which involves extracting the minimum spanning tree from the complete graph after certain edges removed. The most obvious drawback of MST is that the time complexity for forming the initial tree is $O(N^2)$. One alternative way to build the graph reflecting data topology is to employ Delaunay triangulation which is the dual graph of Voronoi diagram [43–45]. Each Delaunay edge is an explicit representation of a neighborhood relation between data points. The information extracted from this neighborhood can provide us an effective way to analyze the relationship between data points. More specific description of Delaunary triangulation will be given in Chapter 4.

The representative clustering methods based on triangulation include the work of C.Eldershaw and M.Hegland [87], the work of In-Soo Kang, etc.[88], the work of S. Hader, and F.A. Hamprecht [89], AMOEBA [28], and AUTOCLUST [29] [90].

The work of C.Eldershaw and M.hegland [87] focuses on how to cut off the connection between clusters while preserving the connection inside clusters. This means to remove all inter-cluster edges at the same time keeping the inner cluster edges. To achieve the ideal threshold to distinguish inter cluster edges from inner

cluster edges, C.Eldershaw and M.hegland utilize the idea from K-MEANS that the appropriate threshold can be found by minimizing the cost function. They effectively mapped the N clusters, multidimensional problem to a relatively trivial two clusters, one-dimensional problem. But, since only the length of edge is considered in this algorithm, it lacks the stability when the distribution of data set becomes more complicated. Thus, the quality of clustering result does depend on the characteristics of data distribution.

AMOEBA [28] also uses triangulation as the source of analysis for identifying clusters. Comparing to the previous work, AMOEBA applies a more complex edges study procedure by computing several statistical features for each edge. Clusters are defined as objects in a connected component. After the construction of the Delaunay diagram, the achieved planar plane-embedded graph is passed to the algorithm. Recursively, every edge is tested by the criterion function which is built by statistical features. All passive edges that are longer than the criterion defined by the function as well as noises are eliminated. The recursion ends while no new connected component is created. The active edges and their objects form the proximity sub-graph at each hierarchical level. AMOEBA does not require user input parameters to generate clustering result. And it is insensitive to outliers or ordering of data input. Its time complexity is $O(NlogN)$ due to the triangulation implemented. But AMOEBA still has limitation for the data set it is applied on. If the distribution of data is very complicated, such as dense cluster surrounded by sparse clusters, it can not perform well. This problem is solved by the derivative algorithm AUTOCLUST [29] which applies more sophisticate statistical investiga-

tion on edge discrimination. There are three main steps of this algorithm. First, after the construction of Delaunay Diagram, all edges in short-edge and long-edge categories are eliminated. The classification of edges is processed according to the pre-defined statistic criterion. This phase produces the boundaries of clusters perhaps with some bridges which are referred to the connections between clusters built by noises [29, 35]. Second, the short edges that do not merge non-trivial connected components are recovered. Finally, clusters are refined by looking for bridges in local regions. In AUTOCLUST, the data points linked by short-edges indicating closeness are grouped together. Other edges representing relation between clusters and noises are eliminated to isolate distinct components. This algorithm has the ability to deal with very complex data set with respect to both cluster shape and cluster density. The time complexity is dominated by computing Delaunay Diagram in $O(NlogN)$. But for the data set with non-uniform density inside clusters, this algorithm can hardly perform well. We will give analysis of these algorithms in more detail in Chapter 4.

Other graph-based clustering methods include [91–94], in which [91] constructs hypergraph and [93] employs probabilistic model in graph-theory based clustering algorithm. The method proposed in [94] is designed based on minimal spanning tree to handle data sets with large amount of outliers.

2.2.6 Other Methods

There are also many other approaches in clustering area. In this thesis, they will be just briefly introduced for the reason that those methods are not very close to

our research topic.

Model-based clustering methods [95–100] assume that data points are generated according to different probability distributions. By estimating the parameters of presumed distributions, clusters are detected, such as in EM [99] method. In [100], a model-based method is proposed to reduce the memory requirement of clustering algorithm.

Combinatorial search techniques based algorithms [101–106] view clustering problem from the optimization aspect. By applying different searching algorithms, the global or approximate global optimum are achieved to get the best partition of data. In work [101], a relatively low cost searching algorithm is proposed. In [103–106], the genetic algorithm is employed to improve the performance of traditional partition method in aspects such as achieving global optimum for K-MEANS method or making it insensitive to initial point selection.

Other famous clustering approaches include Artificial Neural Network for Clustering [107–111], Fuzzy Clustering [112–114], projective clustering [115, 116] which investigates high dimensional clustering problem, etc.

Among those approaches, there are several methods are widely used in many real-world applications for their good performance, especially, on-line clustering algorithms such as ECM (Evolving Clustering Method) [117] and ESOM (Evolve Self-Organizing Maps) [118]. These methods are suitable for dynamic data. The ECM algorithm performs an one-pass, maximum distance-based clustering process without optimization for data steam and it also can work on off-line data with constrained minimization. ESOM is designed based on Kohonen self-organizing map

[107]. It applies connectionist-based models for on-line construction of intelligent information systems. The advantages of it include fast incremental learning, evolvable network structure and visualization. The network systems based on these two methods are shown quite efficient and effective for on-line clustering applications.

There are also many hybrid clustering methods which combine the principles of different clustering approaches to maximize the advantages of them as well as minimizing the disadvantages of them.

In [119], authors proposed a framework to combine clustering results generated by different algorithms or by different parameter settings of the same algorithm by employing the idea of evidence accumulation (EAC). In [120], classification techniques are integrated to refine clustering algorithm by optimizing the objective function in the optimization domain while satisfying the constraint specified in the constraint domain. The work in [121] employed the idea from classification technique boosting algorithm to improve the classical clustering methods. In [122], the author employed search algorithm Simulated Annealing (SA) [123] to improve Fuzzy clustering method. In [124], the possibilistic C-MEANS (PCM) and Fuzzy CMEANS (FCM) are combined together to avoid the disadvantages of both. In [125, 126], authors combined the features of partition method and hierarchical method to achieve effectiveness and robustness. In [127], K-MEANS method is integrated with DBSCAN method to take the advantages of both algorithms. In [128], a method which utilizes the idea of hierarchical method and partitional method on vector quantization is proposed.

2.3 Solid Modeling Techniques

Since our aim is to combine clustering methods with visualization techniques to provide users a thorough understanding of clustering result and process and, even to provide users with the opportunity to assist clustering algorithm using their knowledge, it is necessary to introduce some basic concepts and techniques related to visualization in this section.

In computer graphics [129, 130], there are many techniques for modeling $2D$ and $3D$ geometric objects according to different requirements. The solid modeling techniques [131, 132] define a solid as a surface and its interior. An important characteristic of solid modeling is the emphasis on the geometric integrity of a physical object; this idea is certainly heuristic to clustering. All techniques of solid modeling are usually based on certain representation models. The representation models can be divided into two broad categories [130]: 1) boundary representations that describe $3D$ object as a set of surfaces that separate the object interior; 2) space-partitioning representations that are used to describe interior properties. Here, we focus on the methods of the first category, for their modeling of solid with the boundary description is very helpful for clustering research. Based on different mathematics foundations, there are two major classes of boundary representations which are parametric surface and implicit surface.

2.3.1 Parametric Surface

The parametric surface [129] is the representation model which uses parametric function to build three dimensional surface. This method has been known for a

relatively long time and has been developed by many scientists in computer graphics. There are many implementations based on this representation model, such as Bezier patches [129], B-spline patches [129], and so on. We give an example of parametric surface by introducing the basic idea of Bezier patch. Mathematically, the 3-dimensional surfaces are said to be generated from the Cartesian product of two curves. And a point on the surface patch is given by a bi-parametric function and a set of blending or basis functions are used for each parameter. A cubic Bezier patch is defined as follows:

$$P(u,v) = \sum_{i=0}^{3}\sum_{i=0}^{3} P_{ij}B_{i,3}(u)B_{j,3}(v)$$

where P_{ij} is the set of control points and B_{ij} is blending functions. Then, we can join the patches together to build the whole surface.

2.3.2 Implicit Surface

Implicit modeling techniques [133–135] are relatively new. This approach has become more sophisticated, generating new interest in computer graphics and related fields. It uses implicit function instead of parametric function or explicit function as its mathematical foundation. Although comparing with implicit methods, the traditional parametric methods are simpler to render and more convenient for some geometric operations, the implicit methods can naturally describe the interior of an object for their natural representation of solid objects and innate blend properties. This uniqueness is quite important to the idea of clustering. Since implicit modeling has become a general and widely used topic, here we will limit us to the concepts that are useful to our project.

Since it is very difficult to build solid of arbitrary shape, the idea of decomposing the solid into many primitive solids becomes useful and helpful. We can build all primitives by implicit function and blend them up to form the whole object. That is the basic idea of BLOBBY model [55, 135–137], which is a useful model to build objects with irregular shapes and smooth surfaces. The primitives of BLOBBY model are usually spheres and ellipsoids. We can use different functions to describe the primitive. The most used formulas are introduced below:

(i) BLOBBY model was first accomplished by Blinn [136], and now the term BLOBBY always includes other related models and is not limited only to the original model which is defined as:

$$D(r) = ae^{-br^2}$$

where r is the distance from the center point of a primitive solid, a is the scale factor of the function and b is the exponential factor related to the standard deviation of Gaussian distribution.

(ii) Metaball [137] is a computationally cheaper alternative to using BLOBBY model. The function of Metaball model is defined as 2.1.

$$D(r) \begin{cases} a(1 - \frac{3r^2}{b^2}), & 0 \leq r < \frac{b}{3} \\ \frac{3a}{2}(1 - \frac{r}{b})^2, & \frac{b}{3} \leq r < b \\ 0, & r \geq b \end{cases} \tag{2.1}$$

where r is the distance from the center point of a primitive solid, a is the value of $D(r)$ at $r = 0$, and b defines the intervals of this piecewise function.

For all $r \geq b$, the function returns 0.

(iii) Soft Object [137] is another implementation alternative to BLOBBY model. The function to describe primitive for Soft Object is defined in 2.2.

$$D(r) = 1 - (4/9)r^6/R^6 + (17/9)r^4/R^4 - (22/9)r^2/R^2 \tag{2.2}$$

where r is the distance from the center point of a primitive solid, R is the radius of influence.

For all sub-models of BLOBBY model, at any point of the surface, the iso-surface "potential" is equal to the sum of all the primitives' contributions using function 2.3:

$$S = \sum_{i=1}^{N} D(r_i) \begin{cases} S < T, & \text{Outside the iso-surface.} \\ S = T, & \text{On the iso-surface.} \\ S > T, & \text{Inside the iso-surface.} \end{cases} \tag{2.3}$$

where N is the total number of primitives and T is the threshold constant that determines the level of the iso-surface.

In the above models, the distance from the center point of the primitives describes the range that the primitive can influence on, the summary function adds up the influences of all primitives and the threshold determines the level of the iso-surface built. We will show that the concepts of implicit modeling are quite useful for clustering in the Chapter 3.

2.4 Visualization Techniques

Visualization techniques [39–41] are widely used in many areas including engineering, business, science, etc. Here we will introduce applications of visualization in

data mining.

Methods of visualization in data mining can be classified into data visualization [38, 41], data mining result visualization, data mining process visualization, and visual data mining [4, 55].

Data visualization techniques are methods that present data in various visual forms depending on applications [40], for example, such as boxplots, $3D$ cubes and data distribution charts, etc. Visualization can give the users an overview of data characteristics of the data sets.

Data mining result visualization and data mining process visualization can make the results and the processes more clear and easier to understand for users. With the pictures given by visualization methods, the users are not only provided with an intuitive view of the results received by data mining system but also know how the data are extracted and how the data are mined. For example, the method named H-BLOB suggested in [55], applies a physical-based multidimensional data visualization and analysis model which can quantify the similarity of related objects by transforming data from distance space to coordinate space. H-BLOB works in two steps: First, an agglomerative hierarchical algorithm computes a cluster tree. Second, a single enclosing shape (boundary), which is built by BLOBBY model, is computed for each cluster. The single shape approximates the outline of the including cluster as closely as possible to give an accurate description of the cluster. The advantage of H-BLOB is not only in the multi-level visualization of clusters but also in the visualization of boundaries of clusters, which can provide a more reasonable and intuitive view of cluster to speed up the process of understanding

clustering results, and to facilitate the decision making by the user.

Visual data mining is a little different from the methods mentioned above. It is interactive to users. It means the visualization tools can be used in the data mining process to help users make smart data mining decisions. By this way, the knowledge and experience of users could be useful and helpful to improve the accuracy and efficiency of data mining systems. The following examples show the feasibility and advantages of visual data mining, such as using colored columns to display the data distribution in order to select the attribute that should be first used for classification [40] or using different shape polygons to present data to select the projection directions for multidimensional data [42, 57], etc. The recent work in [138] designed a framework for clustering large data set through interactive visualization, and its adaptive ClusterMap labeling subsystem.

The advantage of visual data mining, besides the integration of the activity of the user, is also dealing with the problem of data sparseness in high dimensional data set. Due to the development of new display devices and visualization techniques, this approach can provide more choices and freedom to the user.

The visualization system we developed based on the visualization techniques will be elaborated in Chapter 5.

Chapter 3

Function-based Methods

After we analyzed the main clustering approaches according to the demands from both theoretic researches and real world applications and showed the limitations of the existing methods, it is necessary to develop new clustering algorithms that let us to overcome those problems. In this chapter, the clustering methods which focus on connectivity investigation will be elaborated. The major advantages of the proposed algorithms include good ability to find clusters with arbitrary shapes; applying different criteria of dissimilarity according to the density variation; low time complexity, automatic parameters setting, and insensitivity to outliers. In the next section, we will introduce the basic ideas of our work.

3.1 Basic Ideas

Our algorithms are developed based on observations of data distribution, limitations of existing clustering algorithms and heuristic ideas from geometric modeling.

(i) **Clusters with different densities**

In the related new research works in spatial clustering, in algorithms such as AMOEBA and AUTOCLUST [29], and in some classic clustering approaches

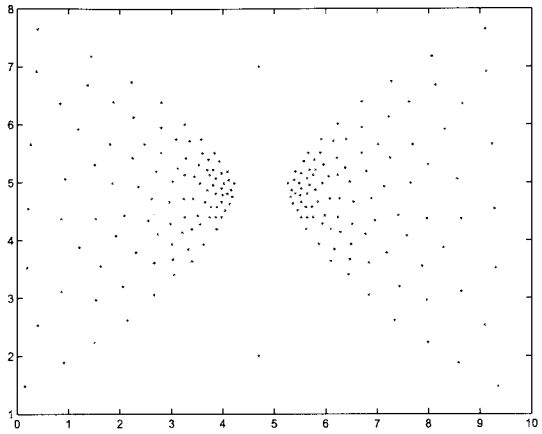


Figure 3.1: An example of non-uniform density data set

like DBSCAN as well, the problem of the clusters with different densities has been mentioned. All these methods are declared with the advantages to deal with those kinds of data sets. In DBSCAN, high density cluster apart from low density cluster can be correctly detected. In AMOEBA and AUTOCLUST, the cases when high density cluster is surrounded by low density cluster or two adjacent clusters with different densities are considered. Since only global parameters are used in DBSCAN, it lacks the ability to discover local variations of the data set. For the cases proposed in AMOEBA and AUTOCLUST, DBSCAN can not perform well. Generally, the data sets in those cases are quite challenging for all classic algorithms. But when we think of different data distribution even further, we can see the problem which has not been thought through yet. The data density could be different not only between different clusters but also between the different parts of the same cluster. We provide an example in Fig.3.1.

In Fig.3.1, although the density varies in the different parts of one cluster,

we can still recognize the two clusters just by visual inspection due to the gradual variation of density. Data set with the similar property could exist in many real world applications, such as geo-spatial data or pointed-based graphic data. The real world data examples will be shown in the Section 3.4.7. Thus, it is reasonable and valuable to extend our concern to the cases where data density can change both inter clusters and inner clusters for both research and application purpose. To meet this proposed challenge, the dissimilarity criteria should be adaptive to the local density distribution. In Fig.3.2, an illustration of dissimilarity criteria variation due to variation of density is shown. If we consider the variation of dissimilarity criteria based on density, in Fig.3.2(a), the red point should not belong to the cluster made up of black points. Because the distance from it to the neatest black point is much bigger than the smallest distances between black points, it means the density of the area with black points is higher than of the area with red point. But in Fig.3.2(b), the red point should belong to the cluster consisted of black points because the data density is relatively uniform, although the distance from the red point to the nearest black point is the same as the one in Fig.3.2(a). In this example, we demonstrated the importance to apply different dissimilarity criteria according to density variation when clustering data point.

(ii) **Limitation of traditional cluster definition**

Traditional clustering methods define clusters as subsets that consist of ob-

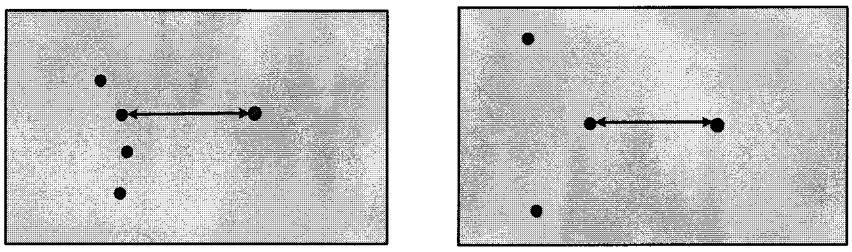


Figure 3.2: Illustration of dissimilarity criteria variation according to density

jects that are similar to one another. That definition only emphasizes the characteristics of objects inside the cluster, but not between clusters. For the boundary of a cluster could be a good description of the dissimilarity between clusters, we added it to our definition of cluster. One great advantage of adopting this concept is that we can classify new coming data points without repeating the whole clustering process. However, how to build the boundary of the clusters should be carefully considered. Here, we use the idea of DENCLUE that the distances between all objects can influence on the resulting clusters, and this influence can be simulated by functions which are called influence functions. Then, the boundary of one cluster can be described by functions as the furthest area that all objects inside the cluster can influence on, or the area with a certain value of influence from all objects. Therefore, the cluster could consist of the inside area and its boundary. From this point, to define cluster as a solid could be a more natural way. This definition of cluster also reflects the granule feature of it.

(iii) Importance of visualization

The visualization techniques play a significant role in clustering. As it was shown in Section 2.4, it is natural to integrate clustering techniques with solid modeling techniques. Then, we can take the advantages of both techniques such as efficient detection of clusters and interpretability of clustering process to users. Although we could use the results we have got from clustering systems as inputs to the visualization systems, it would be costly and inconvenient. A better solution is to combine these two processes together, which means using the same model for both clustering and visualization. By comparison of formulas in computer graphics with ones in clustering, we have noticed that the formulae of the density-based clustering method DENCLUE and the formulae of computer graphics solid model BLOBBY are quite similar. Moreover, the BLOBBY model could simplify the computation if we apply it to clustering. Based on these findings, we built a geometric model for both clustering and visualization.

(iv) Idea from Solid Modeling

Function-based shape modeling is becoming increasingly popular in computer graphics [133]. The idea that complex geometric objects could be produced from a "small formula" is applied in our work. We build the geometric model by using implicit function. It provides the possibility of further analysis such as the comparison of shapes of clusters and querying of clusters. In our model, every cluster can be expressed by a unique function. Let $P = \{\vec{P_1}, \vec{P_2}, \ldots, \vec{P_n}\}$

be a set of multidimensional points $\vec{P_i} = \{p_i^1, p_i^2, \ldots, p_i^d\} \in \mathbf{R}^d$ in the d-dimensional Euclidean space $\mathbf{R}^d$. Then, a solid reconstructed on these points can be described with function based representation as follows:

$$F(\vec{X}, P) \geq 0, \qquad \forall \vec{X} \in \mathbf{R}^d$$

The function can be defined analytically or by procedure. Such function defines closed d-dimensional geometric solid in $\mathbf{R}^d$ space under the following conditions:

$F(\vec{X}, P) > 0$, for the points inside the solid.

$F(\vec{X}, P) = 0$, for the points on the solid boundary (surface).

$F(\vec{X}, P) < 0$, for the points outside the object.

where $\vec{X} = \{x^1, x^2, \ldots, x^d\}$ is a position vector of a point in $\mathbf{R}^d$. The zero set of these functions provides surfaces, and the values that are greater or equal to zero define multidimensional geometric solid objects (solids).

Summing up the above ideas, we regard each cluster with uniform or non-uniform density as a different solid. To find out whether one object belongs to it, we use function-based representation of the solid. We build the solids by analyzing the density function which consists of all influence functions of objects. It means we add up all the influence of objects inside the data space. The density function can make a complete description of the whole data space. Parabolic function, square wave function and Gaussian function are some examples of influence function. The functions such as meta-balls or soft objects can also serve as the influence function

for efficiency. We build the solids on the objects whose values of density function are greater than a threshold, and then we get the function-based representation of clusters.

The model we employed to build clusters also can be conveniently used for visualization. The visualization system developed based on it will be described in Chapter 5.

3.2 Basic Definitions

- The d-dimensional domain (vector space) of all data points [1] is defined as $\mathbf{R}^d$.
- A data point or space point is the multidimensional point defined as: $\vec{X} = \{x^1, x^2, \ldots, x^d\} \in \mathbf{R}^d$.
- The data set of all data points is defined as: D.
- In our work, any metric can be chosen to define the distance between data points in data space. For most our algorithms, we apply the Euclidean Metric, which is defined in 3.1:

$$dist(\vec{X}_1, \vec{X}_2) = \sqrt{\sum_{i=1}^{d} (x_1^i - x_2^i)^2} \qquad \forall \vec{X}_1, \vec{X}_2 \in \mathbf{R}^d. \tag{3.1}$$

- The influence function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ which simulates the influence of a data point $\vec{P}$ on any space point $\vec{X}$ is defined as:

$$f(\vec{X}) = f_b(\vec{X}, \vec{P}) \qquad \vec{X} \in \mathbf{R}^d$$

[1] We will use data point instead of term object in the rest of this chapter and Chapter 4.

where f_b is the basic influence function. In principle, any monotonic decreasing function could serve as the basic influence function. The specific choices in our research will be given in the algorithm description section for each method.

- The field function [2] is defined as the summation of the influences of all data points on one space point $\vec{X}$:

$$F(\vec{X}) = \sum_{i=1}^{N} f_b(\vec{P}_i, \vec{X}), \qquad \forall \vec{P}_i \in D \tag{3.2}$$

3.3 Gaussian Function-based Algorithm

In this section, the work applying Gaussian function as the influence function will be elaborated. The relation between our algorithm and previous method DENCLUE, the algorithm description and testing results, the analysis and comparison will be given in the following sections.

3.3.1 Previous Works Related to Our Algorithm

We selected DENCLUE as our reference method because of similarity of density-based model to BLOBBY model and its relatively good performance. Then, we proposed an algorithm with the following difference from DENCLUE:

(i) Checking the connectivity of data points is always a hard task especially in high dimensional cases. The DENCLUE uses a path searching strategy to check the connectivity of two local maximum (density attractors). We

[2] We call the density function in DENCLUE field function in our methods.

used the idea of grid methods to partition the data space into cubes (hyper-cubes), because cubes can inherit the topology of data by indexing. And we also employed the connectivity reachable idea like in DBSCAN.

(ii) We proposed to apply different dissimilarity criteria according to the density variation in data set. Instead of using global parameter like σ in DENCLUE, a local parameter σ_i was set. The value of this parameter can reflect the dissimilarity criteria variation based on density.

3.3.2 Definitions

Besides the basic definitions introduced in Section 3.2, let us introduce the additional definitions that are used in our model:

- The set of all cubes (hyper-cubes) which cover the whole data space is defined as U. It is implemented by an indexed data structure in which the index of data points is stored.

- The neighborhood Ne of one data point $\vec{P}$ which is adjacent cubes to the cube it located in is defined as: $Ne_P \subset U$.

- Here, we use Gaussian function as the basic influence function. The influence function for one data point $\vec{P}$ to a space point $\vec{X}$ is defined as:

$$f_G(\vec{X}, \vec{P}) = e^{-\frac{dist(\vec{X},\vec{P})^2}{2\sigma^2}}, \quad \text{where } dist(\vec{X}, \vec{P}) \text{ is defined as in (3.1).} \qquad (3.3)$$

- Based on definition (3.2), the field density function is defined as:

$$F_G(\vec{X}) = \sum_{\vec{P}_i \in D} e^{-\frac{dist(\vec{X},\vec{P}_i)^2}{2\sigma^2}}, \quad \text{where } \vec{X} \text{ is a space point, } \vec{P}_i \text{ is a data point.} \tag{3.4}$$

- **Definition 3.1** *A cluster is a solid that reconstructed on data points, and in any part of it, the value of field function greater than a threshold* T. *In our model, we describe the function-based cluster as the area* $\mathbf{S}_c \subset \mathbf{R}^d$ *as follows:*

$$\mathbf{S}_c = \{\mathbf{S}_c \subset \mathbf{R}^d \mid F_G(\vec{X}) \geq T,\ \forall \vec{X} \in \mathbf{S}_c\}$$

The inside cluster area $\mathbf{S}_n \subset \mathbf{R}^d$ *is:*

$$\mathbf{S}_n = \{\mathbf{S}_n \subset \mathbf{R}^d \mid F_G(\vec{X}) > T,\ \forall \vec{X} \in \mathbf{S}_n\}$$

The cluster surface (boundary) area $\mathbf{S}_b \subset \mathbf{R}^d$ *is:*

$$\mathbf{S}_b = \{\mathbf{S}_b \subset \mathbf{R}^d \mid F_G(\vec{X}) = T,\ \forall \vec{X} \in \mathbf{S}_b\}$$

- Dissimilarity criterion T:

  The parameter T can control the level of clusters, since we can set a bigger value to get more dense clusters, on the contrary, we will get less dense ones for smaller values. Then, we could control the levels of clusters by T. In our algorithm, we set $T = e^{-4.5} \approx 0.01$ according to the 3σ-rule for Gaussian distribution. That means the influence of data points that are further than 3σ from the data point to be checked can be ignored. The value of T also can be set manually by users based on their knowledge.

- Dissimilarity control parameter σ:

  We use the parameter σ to control the range that a data point can influence on. By changing the value of this parameter, we can control the expansion of clusters, thus, can get different clustering results based on different requirements. Considering the density variation of data set, we design global and local parameters separately to control the expansion of the cluster. We use σ for the global one and σ_i for the local one. They can be used to get the best results based on different cases.

  Definition 3.2 *The local parameter σ_i is designed to describe the influence of a data point according to local information. For a data point $\vec{P}$, its σ_{iP} is defined as 3.5:*

$$\sigma_{iP} = \frac{\sigma}{\log_a \frac{(F_G(\vec{P})-1)}{T}}, \quad \textit{where } \mathrm{F}_G(\vec{P}) \textit{ is the field value for } \vec{P}. \tag{3.5}$$

  a is a parameter used to adjust the variation of σ_i against $F_G(\vec{P})$. The setting of it will be introduced in parameters discussion section.

  We illustrate the relationship between σ_i and $F_G(\vec{P})$ in Fig.3.3, where the X direction represents the value of $F_G(\vec{P})$ and the Y direction represents the value of σ_i. In function (3.5) we built, the value of σ_i is decreasing when the value of $F_G(\vec{P})$ is increasing. This means the data point with big field value should have small area where it can influence. The *log* operation is used since the relationship between field value and σ is exponential.

- **Definition 3.3** *Local influence function:*

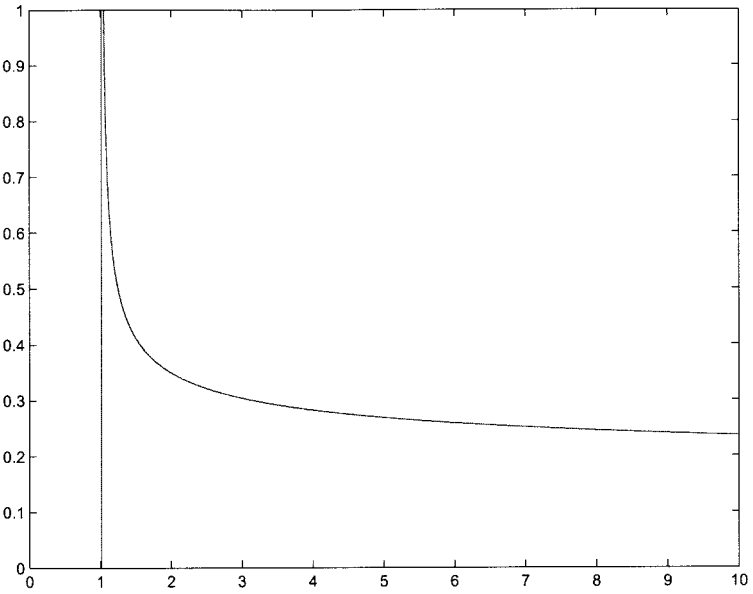


Figure 3.3: Function for computing σ_i

The influence function which applies local parameter σ_i will become local influence function which is defined as 3.6:

$$f_{Gi}(\vec{X}, \vec{P}) = e^{-\frac{dist(\vec{X},\vec{P})^2}{2\sigma_{iP}^2}}, \quad \textit{where } dist(\vec{X}, \vec{P}) \textit{ is defined as (3.1).} \tag{3.6}$$

- **Definition 3.4** *Local field function:*

  The local field function is the summation of local influence functions for all data points. We define the local field function as 3.7.

$$F_{Gi}(\vec{X}) = \sum_{\vec{P} \in D} e^{-\frac{dist(\vec{X},\vec{P})^2}{2\sigma_{iP}^2}}, \quad \textit{where } \vec{P} \textit{ represents any data point.} \tag{3.7}$$

3.3.3 Algorithm Description

According to Definition 3.1, the solid (cluster) can be reconstructed on data points with function-based representation. Any data point can be checked by the field function to determine whether it belongs to any cluster. But by using the function,

we cannot know to which cluster the data point belongs, since the function-based representation of solid cannot describe topology of data points. Here, we suggest to apply grid-based method to solve this problem.

This algorithm is designed to find clusters whose density varies through the data space. The dissimilarity criteria of clusters should vary according to density variation. Here we use σ_i to control the influence range of a data point. According to the definition of σ_i in (3.5), we can notice that the connectivity controlled by σ_i between two data points is not symmetry any more. It will be determined by the data point with higher density, that also means by the data point with the bigger σ_i. Below we introduced the main steps of our algorithm.

Let us consider 2-dimensional case:

Step 1: For all data points, calculate the minimal distance which is the distance to their nearest data point. Set the value of σ based on minimal distance distribution (details are given in parameter discussion section). Partition the data space into non-overlapping identical rectangular cubes. The length of each edge is l, the whole space is divided into M cubes. The neighborhood of one data point is composed of the cubes which are adjacent to the cube it located in. Let us set $l = 3\sigma$. By doing this, for each data point, we should only check the data points inside its neighborhood to calculate its value of field function, since the distance from this data point to any other data points is bigger than 3σ.

Step 2: We number the cubes from *1* to M, and map every data point to an appropri-

ate cube. At the same time, the parameter $count_i$ which records the number of data points in the ith cube increases when a data point is allocated into the cube.

Step 3: Find all cubes, which have more data points inside than their neighbors. These cubes are called core cubes. For each core cube, we find the data point whose value of field function is the biggest. Here, because of the setting of the length of cubes, we calculate the field value for $\vec{P}$ as following:

$$F_G(\vec{P}) = \sum_{\vec{Q} \in Ne_P} e^{-\frac{dist(\vec{Q},\vec{P})^2}{2\sigma^2}},$$

where Ne_P is the neighborhood of $\vec{P}$, $\vec{Q}$ represents any data point inside Ne_P. Other data points are ignored based on the 3σ rule of Gaussian distribution.

Step 4: After the data points with the biggest field values are found, the expansion process in all core cubes begins. For those data points, we build the local influence function by applying local parameter σ_i which is defined in (3.5) in stead of global σ as in (3.6).

We also update the neighborhood according to σ_i. We use parameter b : $b = \lceil \frac{\sigma_i}{\sigma} \rceil$, then, the area of new σ_i neighborhood is $NeN = b \cdot Ne$. In Fig.3.4, an example for $b = 2$ is shown.

We will check all data points in NeN area with the local influence function. For $\vec{P}$ which is the found data point with the biggest field value, if:

1. $f_{Gi}(\vec{Q}, \vec{P}) \geq T, \quad \vec{Q} \in NeN_P, \quad$ where $\vec{Q}$ is a data point,
2. $F_G(\vec{P}) \geq F_G(\vec{Q})$,

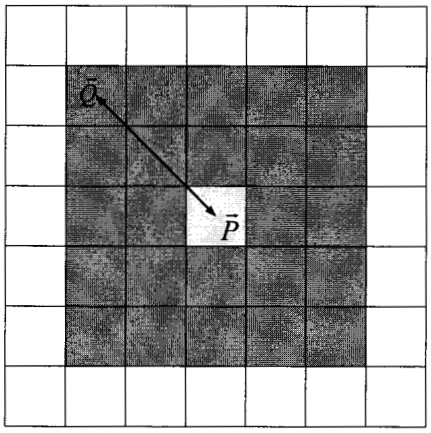


Figure 3.4: Illustration of σ_i neighborhood for $b = 2$

then $\vec{Q}$ can be included into the cluster (solid) which is initialized by $\vec{P}$.

Step 5: Let the solids grow by emerging more data points. First, we update the neighborhood of the solids constructed. For a certain solid C, the neighborhood NeC is composed by the neighborhoods of all data points inside: $NeC = \bigcup_{\vec{P} \in C} NeN_P$,where NeN_P is the neighborhood of the $\vec{P}$. Then we check all data points inside NeC.

For one data point $\vec{Q} \in NeC$ If:

1. $F_{Gi}(\vec{Q}) = \sum_{\vec{P} \in C} e^{-\frac{dist(\vec{Q},\vec{P})^2}{2\sigma_{iP}^2}} \geq T, \qquad \vec{P} \in C,$

2. $\exists \vec{P} \in C, \quad$ such that $F_G(\vec{P}) \geq F_G(\vec{Q}),$

then $\vec{Q}$ can be merged into solid C. During the expansion process of all solids, if two data points that belong to two different solids are checked to

be included by each other, the two solids are also emerged into one.

Step 6: After checking all data points in the data space, we can get all clusters (solids) by the formula (3.8) and outliers which are the solids with only one data point inside.

$$F_{Gi}(\vec{X}) = \sum_{\vec{P} \in C_j} e^{-\frac{dist(\vec{X},\vec{P})^2}{2\sigma_{iP}^2}} \geq T, \quad \text{where } C_j \text{ it the } jth \text{ cluster.} \tag{3.8}$$

Here, we only use the data points inside cluster to calculate the field value since the influences from other data points can be ignored for simplification. According to Definition.3.1, we can detect the internal, external areas and boundaries of all clusters.

Once a new data point is considered, we can conveniently check it by (3.8) without repeating the whole clustering process. For a new data point $\vec{R}$:

- if $F_{Gi}(\vec{R}) \geq T$, then, it belongs to cluster C_i.
- if $\forall C_i, \quad F_{Gi}(\vec{R}) < T$, then, it is an outlier.

Our method is general for n-dimensional clustering. But due to most spatial data applications are low dimensional, we only implemented our algorithm for low dimensional case.

3.3.4 Parameters Discussion

The values of parameter T and σ determine the result of clustering. Though we can set them manually and get the most appropriate values from testing, it could be costly and inefficient. We need some methods to predict the proper values in

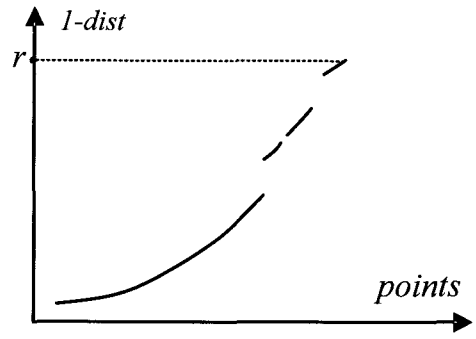


Figure 3.5: An example of 1-distance graph

advance. In order to let the values of parameters fit the data set, we must get some information from the data set about its uniqueness. The global parameter σ controls the range how one data point can influence others when we compute the local parameter σ_i. To set an appropriate value of σ, we find the maximal value in all minimal distances which are the distances from each data point to its nearest neighbor. To illustrate it, we can draw the named 1-distance graph which is plotted by the sorted minimal distances. The desired distance r will be the one which corresponds to the biggest value in the graph. There is an example of 1-distance graph in Fig.3.5. We set $\sigma = r/3$ according to the 3σ rule. By doing this, for most data points, the value of local parameter σ_i will be smaller than global parameter σ. The bigger field value based on global parameter σ, the smaller value of the local parameter σ_i will be.

The parameter a is to adjust the relationship between the value of field function and the value of σ_i. Since this relationship concerns to the characteristic of data distribution, we set it based on the distribution of the minimal distances of all data points.

Table 3.1: Choices for the value of parameter a

t	$t \leq 20\%$	$20\% < t \leq 30\%$	$t > 30\%$
a	2	3	5

Let t be equal to the nearest distance value of 90% of the nearest distance distribution divided by r. The 90% choice is for cutting out outliers. According to the value of t, we can select the value of a in Table.3.1. This heuristic method is based on our testing data sets, where for bigger value of t, we can set a with bigger value.

The parameter T can control the level of clusters. We can set $T = e^{-4.5} \approx 0.01$ according to the 3σ Rule for Gaussian distribution. Since these two parameter σ and T are not absolutely independent, we can adjust the value of one parameter and set another one with a fixed value to get the same result received by exchanging the roles of two parameters. We can also give more freedom to the user to set the value of T. By selecting different T, the user can get different clusters with different level just like hierarchical methods can provide. Here, we fix the value of T and use heuristic method to set the values of σ and a.

3.3.5 Comparisons and Experiment Results Analysis

The proposed algorithm was implemented and tested on the following data sets. We designed two data sets $D1$ and $D2$ with uniform and non-uniform distribution of data points inside clusters. The pictures of testing data set, statistical feature used in our algorithm, and clustering results are shown for each data set. The boundary built by our algorithm is also given to demonstrate its unique feature. To provide another view of functional representation of the data distribution, the

$3d$ picture of field value is given for data set $D1$.

We applied our algorithm and classic clustering methods such as K-MEANS, hierarchical Single Linkage and Complete Linkage methods, and DBSCAN on $D1$ - $D2$ data sets. We choose these classic clustering methods because they are representative in spatial clustering, and standard implementations of these methods are available. We can also avoid the potential problems causing by different implementations, which could lead us to bias in results comparisons. K-MEANS method is still the most widely used method, and it is based on the optimization principles which form a foundation for many well-known data mining techniques. We employed the squared Euclidean distance and random selection of initial cluster centers for K-MEANS in our testing. Linkage hierarchical methods have a good ability of handling clusters with complicated shapes. DBSCAN is another well known density-based clustering method for dealing with arbitrary shape clusters and outliers. The implementation of DBSCAN is provided by its authors and all parameter settings are based on the instruction given in the original paper [31] as $k = 3$ and $MinCard = 4$ for our 2-dimensional data sets. By comparing with those methods, we can have a more clear idea about performance of our algorithm. The characteristics of it such as discovering arbitrary shape clusters with different density, detecting outliers based on both global view and local view, have been well demonstrated. All results provided here are generated by automatic setting of parameters, which were described in Section 3.3.4 . For the user who has a very clear expectation of the data set, parameters also can be set based on his knowledge of data.

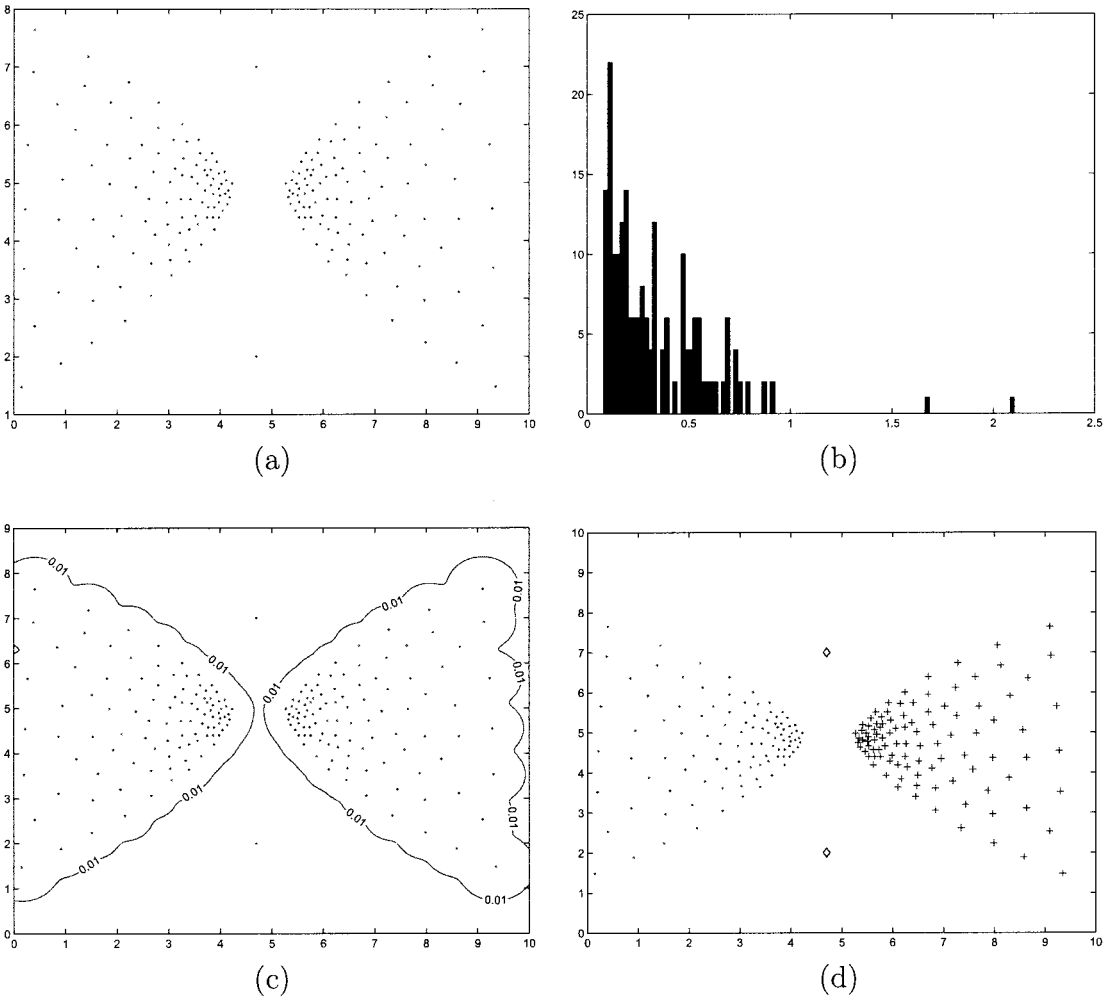


Figure 3.6: The testing on data set $D1$ of our algorithm: (a)The picture of data set $D1$; (b)The distribution of all minimal distances of data set $D1$; (c)The cluster boundaries of data set $D1$ built by our algorithm, $T = 0.01$; (d)The clustering result of data set $D1$ by our algorithm

In Fig.3.6, the picture of $D1$, statistic information, cluster boundaries and clustering result of $D1$ by our algorithm are shown.

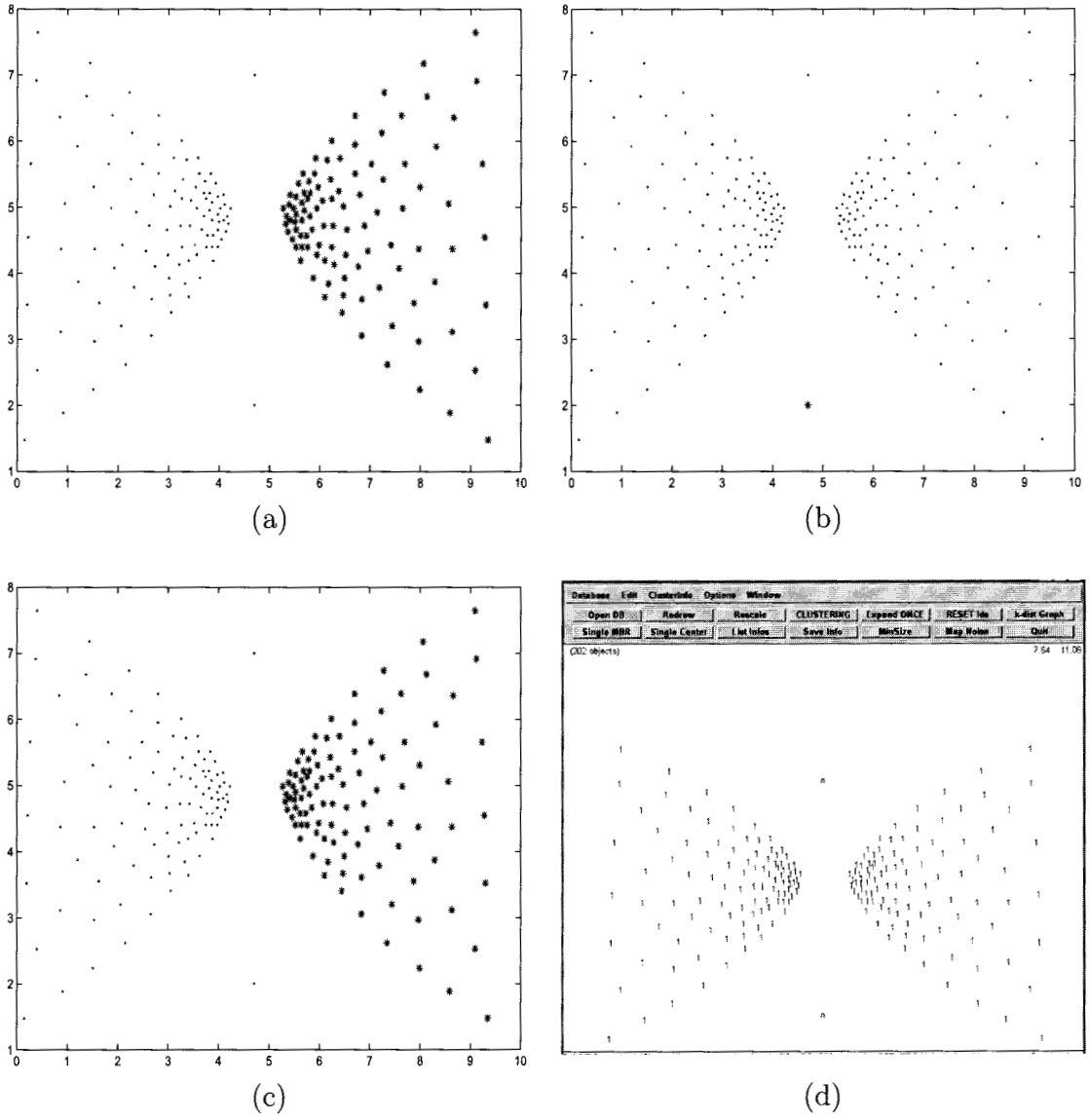


Figure 3.7: Clustering results of data set $D1$ by comparison methods: (a)Clustering result of testing data set $D1$ by K-MEANS; (b)Clustering result of $D1$ generated by Single-linkage algorithm; (c)Clustering result of $D1$ generated by Complete-linkage algorithm; (d)Clustering result of testing data set $D1$ generated by DBSCAN, where $EPS = 1.142$

In Fig.3.7, the clustering results of testing data set $D1$ by four classic clustering algorithms are shown.

There are two clusters and two outliers in this data set $D1$. The density of both clusters changes gradually from one side of the clusters to another. The maximal distance between data points inside clusters is comparable to the minimal distance between the data points belonging to different clusters and to the minimal distance from the outliers to cluster data points. Comparing the result of our algorithm with each of those four algorithms, we can clearly notice the advantage of our algorithm in dealing with non-uniform density data set. It can separate the two clusters as well as keep the integrity of them and detect the outliers. K-MEANS can separate the clusters but fail to detect outliers. For the hierarchical methods, Single-linkage method can not separate the two clusters due to the density distribution, Complete-linkage method performs quite similar to K-MEANS. DBSCAN also suffers from the problem of keeping the whole clusters together while discriminating outliers.

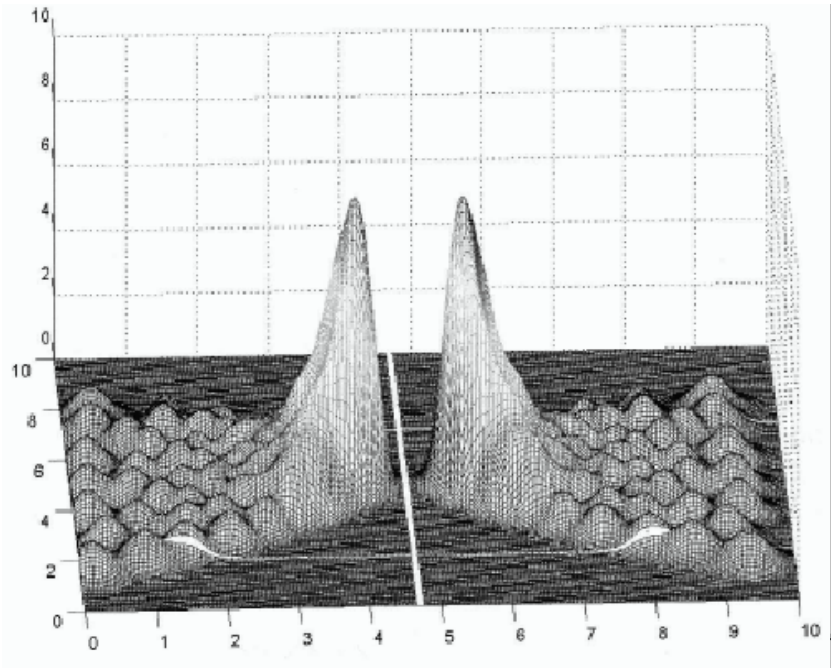


Figure 3.8: The $3d$ picture of the field value of data set $D1$

In Fig.3.8, the 3-dimensional picture of the distribution of field value for data set $D1$ is shown. We can see how the influences from all data points are simulated by function.

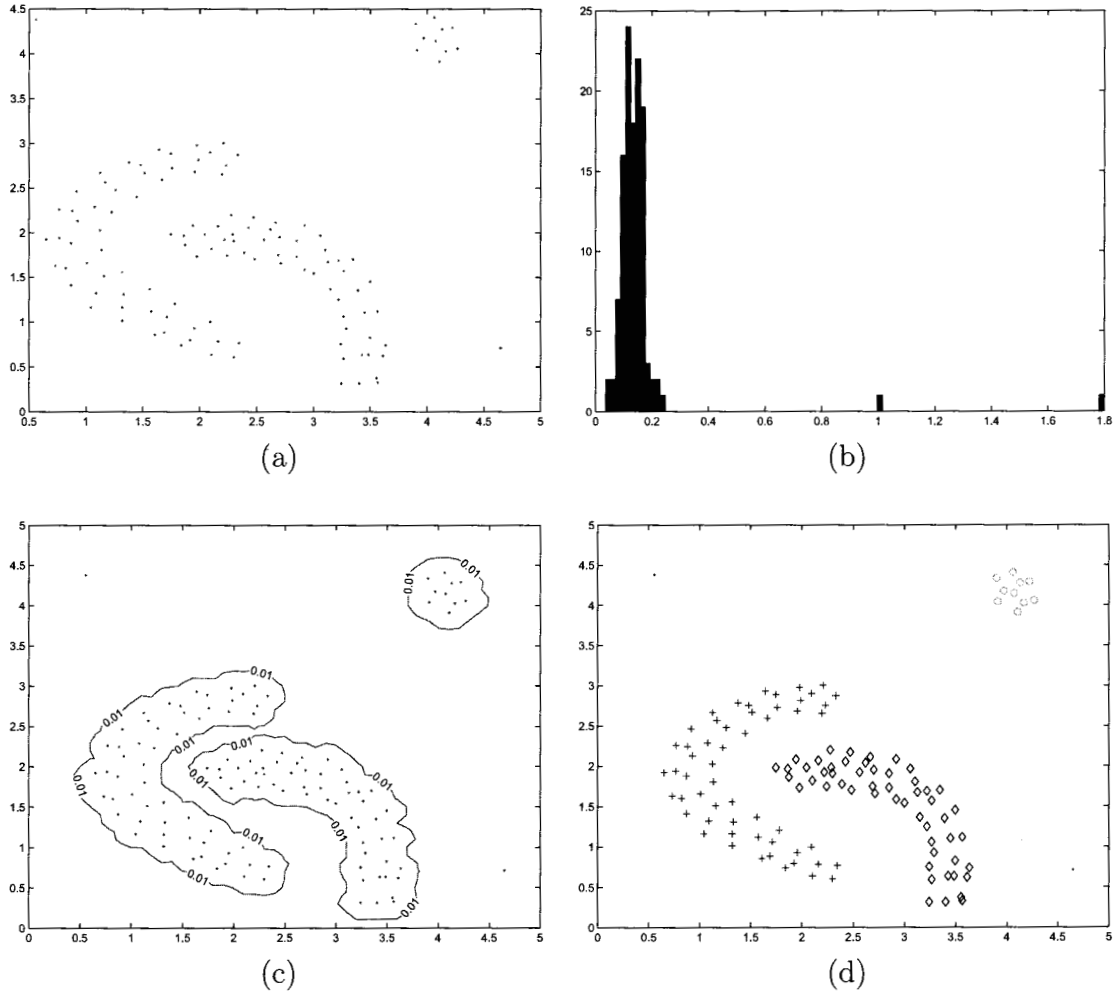


Figure 3.9: The testing on data set $D2$ of our algorithm: (a)The picture of data set $D2$; (b)The distribution of all minimal distances of data set $D2$; (c)The cluster boundaries of data set $D2$ built by our algorithm, $T = 0.01$; (d)The clustering result of data set $D2$ by our algorithm

In Fig.3.9, the picture of $D2$, statistic information, cluster boundaries and clustering result of $D2$ by our algorithm are shown.

In Fig.3.10, the clustering results of testing data set $D2$ by four classic clustering

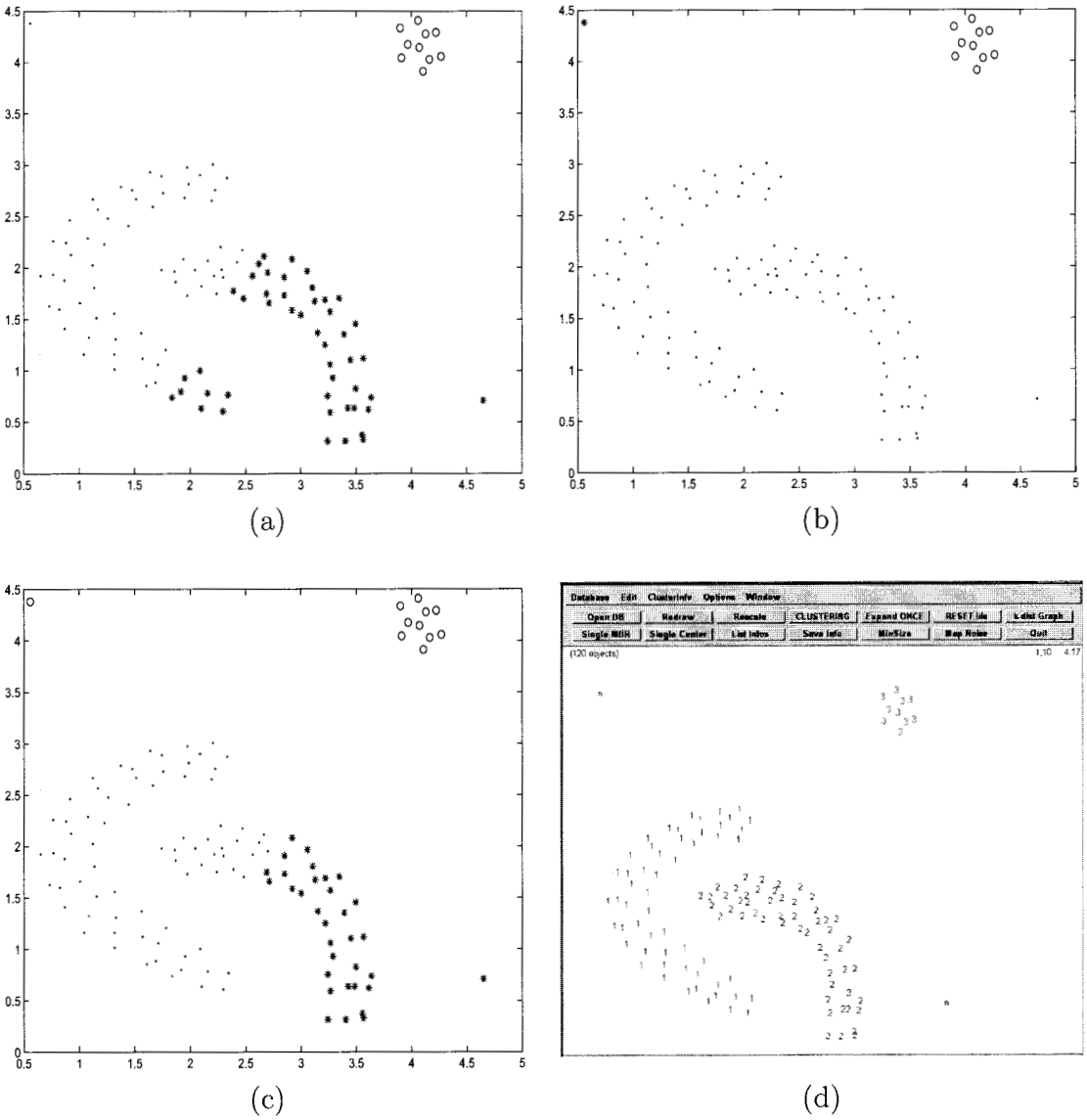


Figure 3.10: Clustering results of data set $D2$ by comparison methods: (a)Clustering result of testing data set $D2$ by K-MEANS; (b)Clustering result of $D2$ generated by Single-linkage algorithm; (c)Clustering result of $D2$ generated by Complete-linkage algorithm; (d)Clustering result of testing data set $D2$ generated by DBSCAN, where $EPS = 0.2874$

algorithms are shown.

$D2$ is one typical data set used in DBSCAN to demonstrate its ability to handle clusters with irregular shape. There are three clusters and two outliers inside. From the results, we can notice that our method can also deal with this kind of data set very well. Except DBSCAN, other three clustering algorithms all lack the capacity to discover clusters of complex shapes with outliers existing.

Although our Gaussian function-based method has the ability to detect clusters with non-uniform density, it is not very robust to the variation of data amount and outlier percentage. When we deal with large data set with high percentage of outliers, more accurate estimation method than proposed heuristic method is needed to generate good results. Also, when data distribution is unknown, applying Gaussian function has the disadvantage that the long tail of Gaussian distribution causes the difficulty of tuning parameters. The time complexity of this algorithm highly depends on data set, since the most time consuming procedure is neighborhood update.

3.4 Piecewise Function-based Algorithm

To fit the new applications [32, 33, 139–141] which have been mentioned in Chapter 1, clustering method should be able to discover clusters with arbitrary shape and non-uniform density automatically and efficiently, also should be able to detect outliers. The Gaussian-function based algorithm which has been introduced in Section 3.3, meets the new requirements for most cases according to our experiments. But there are still the limitations such as the computational cost is

relatively high for large data set and the setting of parameters is not very robust. For the cases where clustering results are sensitive to the values of parameters, the performance of Gaussian-function based algorithm is not very stable. The main reason for that problem is the function choice. Since the computation of Gaussian function is exponential, the time complexity of field value calculation is quite high. And the long tail of Gaussian distribution increases the difficulty in value tuning for parameters. Thus, in next section, we will introduce another function-based clustering algorithm ADACLUS (ADAptive CLUStering) which applies piecewise function to improve the algorithm performance in both speed and stability.

3.4.1 New Features of ADACLUS and its Relation to Previous Algorithms

ADACLUS is an adaptive density-based clustering algorithm. Similar to the pioneer DENCLUE algorithm, it is based on the general assumption that a "natural" clustering partition of a given data set situated in a metric space can be built through the use of the appropriate "field function" [3](i.e. data points integral influence function) which can be determined at any space point $\vec{X}$ as the sum of influences of all data points at $\vec{X}$. If the influence of any data point is restricted to a limited distance from it, or if it is rapidly decreasing with this distance, then it is enough to take into account only the influences of data points within some limited vicinity of the given point. This approach (first used in DENCLUE) results in a fast algorithm performance: the complexity of the algorithm could be made linear (or close to linear) with respect to the number of data points. It is

[3]We called the density function in DENCLUE field function here.

a very important advantage for computer geometry applications where real-time performance is essential. Another important (especially from geometrical point of view) advantage of field function clustering is that the corresponding algorithms have good capabilities to detect cluster boundaries [37, 142]. One more important advantage is that these algorithms can efficiently eliminate outliers.

ADACLUS possesses new important features and at the same time preserves all listed above advantages of function-based clustering algorithms. The main new feature of ADACLUS is that the parameters of influence function are not global and not predefined. Contrary to the previous density-based algorithms these parameters are not constant in ADACLUS. They are completely adaptive to the local situation in the vicinity of the point in consideration. The control parameters of ADACLUS could be set either automatically or manually. Generally, automatic settings work fine, but the user is provided also with the option to tune ADACLUS clustering rule from purely local-based to local-global-based one. In the latter case the influence function of ADACLUS takes in the account both local and global information combining them in certain proportion. The data points influence is modeled by adaptive piecewise linear function which ensure fast and efficient performance. The reasons of choosing such function and its advantages over other possible choices are described in the next section.

3.4.2 Definitions

- **Metric choice.**

  ADACLUS can work with any metric which defines the distance between

points in space. In ADACLUS we use another metric ρ_1 instead of Euclidean metric. Namely, we use the following definition of the distance between two points:

Definition 3.5

$$\rho_1(\vec{X}_1, \vec{X}_2) = \sum_{i=1}^{d} |x_1^i - x_2^i| \qquad \forall \vec{X}_1, \vec{X}_2 \in \mathbf{R}^d. \tag{3.9}$$

From topological point of view the metric ρ_1 is equivalent to ρ_{Eucl}, but results in faster calculations. It is easy to see that ρ_1 depends on the choice of coordinate system, but it is not a disadvantage in our case. If there is no "natural" coordinate system, associated with the data, this dependance will not affect significantly the results of clustering. And if there is such a coordinate system (which is the case for digital images or scanned data having intrinsic lattice structure), then the above coordinate-dependent form of metric ρ_1 can even result in slightly more accurate performance of the algorithm.

- **Influence function choice**.

  Similar to DENCLUE and other algorithms which use field functions for clustering, ADACLUS also needs a predefined rule for measuring "influence" of certain data point at any point in the space. Once such rule is chosen, field function is obtained by summation of all the influences of data points at the arbitrary space point.

In principle, ADACLUS, like DENCLUE, can work with any kind of influence function. But in order to create really fast clustering algorithm it is reasonable to choose the simplest (from the computational point of view) influence function which still preserves all necessary features for clustering. It is also reasonable to choose such influence function, which depends on meaningful parameters only. From both points of view, the Gaussian influence function $f_{Gauss}(\vec{P}, \vec{X}) = c \cdot e^{-\frac{\rho(\vec{P},\vec{X})^2}{2\sigma^2}}$, which is often used in density-based clustering to measure influence of the data point $\vec{P}$ at the space point $\vec{X}$, seems to be not the best choice. Indeed:

1) Gaussian function is not quite simple because it requires calculation of the exponent;

2) Except the case when distribution of data points is Gaussian (which is applicable to special kind of model-based clustering only), Gaussian function is not better than many other monotonic decreasing functions, which could be much simpler than Gaussian one;

3) The value of parameter σ in Gaussian function does not have any clear meaning if the distribution of data points is not Gaussian.

4) The long tail of Gaussian distribution could increase the difficulty in the control of field value. It will bring the problem of parameter setting.

Based on the expressed ideas, we propose the following simple, but flexible function which models the influence of data points in ADACLUS (see Fig.3.11)

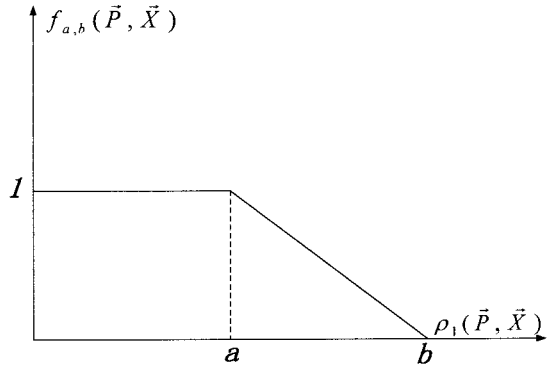


Figure 3.11: Parametric influence function used in ADACLUS

Definition 3.6 *The influence of the data point* $\vec{P}$ *at any point* $\vec{X}$ *in the space is defined as*

$$f_{a,b}(\vec{P},\vec{X}) = \begin{cases} 1, & \text{if } 0 \le \rho_1(\vec{P},\vec{X}) \le a, \\ \frac{b-\rho_1(\vec{P},\vec{X})}{b-a}, & \text{if } a < \rho_1(\vec{P},\vec{X}) \le b, \\ 0, & \text{if } \rho_1(\vec{P},\vec{X}) > b, \end{cases} \tag{3.10}$$

where the distance $\rho_1(\vec{P},\vec{X})$ *is calculated according to (3.5). Here parameter* a *denotes such maximum distance from the data point* $\vec{P}$ *which is still considered as "near" one. The parameter* b *denotes such minimum distance from* $\vec{P}$ *which is already considered as "far" distance.*

It is assumed that $b \ge a$. Generally, there exists a non-zero gap between these thresholds, i.e. $b > a$. Both a and b are fully adaptive in ADACLUS and their calculation is based on the mixture of global and local information extracted from the data set.

In the case $b = a$ the function $f_{a,b}(\vec{P},\vec{X})$ transforms to the Square Wave influence function which is used in DBSCAN:

$$f_{a,a}(\vec{P},\vec{X}) = \begin{cases} 1, & \text{if} \quad 0 \le \rho_1(\vec{P},\vec{X}) \le a, \\ 0, & \text{if} \quad \rho_1(\vec{P},\vec{X}) > a. \end{cases}$$

Based on the Definition 3.6 we define the field function at any point $\vec{X}$ in the space as:

$$F(\vec{X}) = \sum_{i=1}^{N} f_{a_i,b_i}(\vec{P}_i,\vec{X}), \tag{3.11}$$

where the values of parameters a_i, b_i which are used to calculate the influence of the data point $\vec{P}_i$ at any space point $\vec{X}$, are calculated for each $\vec{P}_i$ taking in account the mixture of information about the local distribution of data points in the $R-$neighborhood of $\vec{P}_i$ and the global distribution of data points in the whole screen (area of clustering). Proportion of this mixture is defined by the values of magnitude parameter G and radius R, which are calculated as described below.

By differentiating (3.11) with respect to all the components of coordinate system, the following formula can be easily obtained for the gradient $\text{grad}F = \left(\frac{\partial F(\vec{X})}{\partial x^1},\ldots,\frac{\partial F(\vec{X})}{\partial x^d}\right)$. We define here $\text{grad}F$ as zero in all points where $F(\vec{X})$ is not differentiable.

$$(\text{grad}F)^k = \frac{\partial F(\vec{X})}{\partial x^k} = \sum_{i=1}^{N} \frac{c_{i,k}(\vec{X})}{b_i - a_i}, \quad (1 \le k \le d), \tag{3.12}$$

where

$$c_{i,k}(\vec{X}) = \begin{cases} 1, & \text{if} \quad a_i < \rho_1(\vec{P}_i, \vec{X}) < b_i \text{ and } x^k > p_i^k, \\ -1, & \text{if} \quad a_i < \rho_1(\vec{P}_i, \vec{X}) < b_i \text{ and } x^k < p_i^k, \\ 0, & \text{in all other cases.} \end{cases} \tag{3.13}$$

On the basis of a field function and threshold T the ADACLUS algorithm determines *point-based clusters* and, in another version of the algorithm, *boundary-based clusters* which are defined as follows

Definition 3.7 *Given field function F and threshold parameter T, the cluster (or, strictly speaking, point-based cluster) is such a subset C of the set of data points that*

1) $F(\vec{P}) \geq T$ for each data point $\vec{P} \in C$;

2) $\forall \vec{P}_i, \vec{P}_j \in C$ there exists a continuous curve $x = x(s), 0 \leq s \leq 1$, such that $x(0) = p_i,\ x(1) = p_j,\ F(x(s)) \geq T\ \forall s \in [0,1]$;

3) it is impossible to add any more data points to C without violating properties 1) or 2).

Definition 3.8 *Given field function F and threshold parameter T, the boundary-based cluster D is a connected component (solid in computer graphical term) of the domain: $D = \{\vec{X} :\ F(\vec{X}) \geq T\}$.*

Definition 3.9 *The boundary of the boundary-based cluster $D = \{\vec{X} :\ F(\vec{X}) \geq T\}$ is simply it's boundary in the space. It is the domain (curve for 1-dimensional case, surface for 2-dimensional case, and hyper-surface for high dimensional case) $B = \{\vec{X} :\ F(\vec{X}) = T\}$ or a part of it.*

- **Control parameters of ADACLUS.**

  ADACLUS depends on 3 scalar parameters which could be either calculated automatically from the number of data points and their distribution on the screen (as described below), or defined by the user. All these parameters have clear meaning which gives the user real possibility to make ADACLUS more adapted for certain area of applications or to tune clustering results in a specific case. These parameters are:

  1) $G \in [0, 1]$ - *magnifier parameter.* If $G = 1$ the clustering is purely local-based (i.e. based on the "fully magnified view"). If $G < 1$, then clustering is based on the mixture of global and local information about the data set. Maximum portion of global information is added when $G = 0$, but even in this case the algorithm remains local-adaptive. One can imagine that G describes the "magnifying glass" through which algorithm "looks" at the data while determining clusters. If $G = 1$, then algorithm "looks through a microscope", and therefore performs clustering on a very local level. It results in the tendency to achieve small "micro"-clusters. When G decreases, the "microscope" is being continuously converted to a simple glass without any magnification. The smaller G becomes the more the tendency to glue neighboring small clusters into bigger ones increases. Parameter G must be chosen (automatically or manually) before the algorithm starts.

  One of the aims of ADACLUS is to serve computer-vision related problems, and therefore the general idea here is to follow a "human's eye approach" in

clustering. In automatic parameter setting, the value of magnifier parameter G is chosen according to two "magnifier factors": the number of data points and their distribution on the screen. In general, the rule is as follows.

Parameter G is chosen close to 1 (maximum magnification) when there is not too many data points in the screen and/or their distribution is approximately uniform inside the expected clusters. In both cases, the clustering is based on local information only, without taking in account global features of the whole picture. The reason is that if there is such a small amount of data points that one can notice any of the data points "at one glance", then the "natural eye-based" clustering should depend exclusively on the local details of the data points distribution. Moreover, the small number of data points is not enough to generate reliable global features of distribution which could be useful for clustering. Here, we denote G_0 to describe the relationship between G and number of data points.

In the case when the data points are distributed uniformly inside clusters, there is also no need to add global information to local one (for any number of points), because the global information in this case tells us nothing new and important about the data point distribution properties. So, in both cases clustering can be performed purely locally. Vice versa, if there are many data points and/or they are distributed quite irregularly with a lot of small details of distribution, then it is natural to look at the whole picture more globally to unite tiny clusters rather than separate them (if it is not the

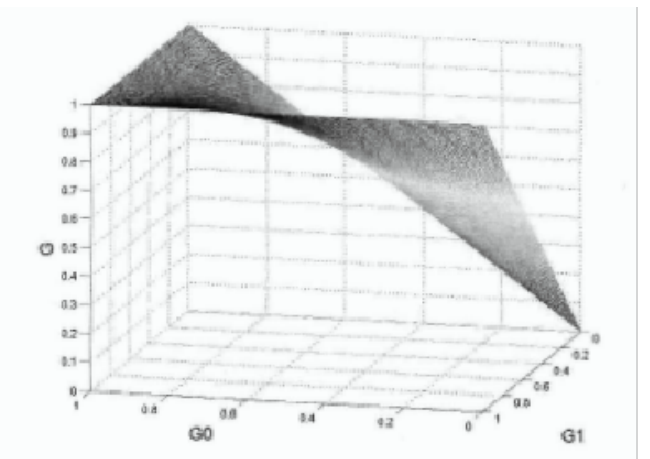


Figure 3.12: Parameter G determination based on the values of $G0$ and $G1$

case in a certain application, user can always set the magnifier parameter G manually). Here, we denote G_1 to describe the relationship between G and inner-cluster distribution of data points.

The calculations of G are performed as follows.

$$G = G_0 + G_1 - G_0 G_1, \tag{3.14}$$

the values of G_0 and G_1 model the "magnifier factors" due to the number of data points N and to the inner-cluster distribution. The final "magnification rate" G is calculated in (3.14) by means of bilinear surface, Fig.3.12 (see, for example [143]) determined by Table.3.2 of extreme G values, where G_0 and G_1 are the entries of the table. The choice of extreme values of G was discussed above.

Table 3.2: Extreme G Values

$G_1 \backslash G_0$	0	1
0	0	1
1	1	1

The exact formulas of G_0 and G_1 are given below in (3.15) and (3.16).

$$G_0 = \begin{cases} 1, & \text{if} \quad N \leq Smin, \\ 1 - \frac{N - Smin}{Smax - Smin} & \text{if} \quad Smin \leq N \leq Smax, \\ 0, & \text{if} \quad N \geq Smax. \end{cases} \tag{3.15}$$

$$G_1 = \begin{cases} 1, & \text{if} \quad t \leq 1/\sqrt{d}, \\ 1 - \frac{t - (1/\sqrt{d})}{1 - (1/\sqrt{d})} & \text{if} \quad 1/\sqrt{d} \leq t \leq 1, \\ 0, & \text{if} \quad t \geq 1. \end{cases} \tag{3.16}$$

In (3.15), $Smin$ is the number of data points, starting from which a human's eye starts missing individual points and therefore the manner of discrimination starts shifting from local to global one. $Smax$ is the number of points, starting from which global features of the data set become fully important for the human's eye. We will give our setting to these two parameters in Section 3.4.3, Step 3 and discuss more about them in Section 3.4.6.

To calculate the value of G_1 in (3.16), first, we compute the minimal distance $MD(i)$ for each data point $\vec{P}_i$, which is the distance to its nearest data point. Then, the values of mean $m_{MD} = \frac{1}{N}\sum_{i=1}^{N} MD(i)$ and standard deviation $std_{MD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(MD(i) - m_{MD})^2}$ are calculated. Let us denote

$$t = \frac{std_{MD}}{m_{MD}}.$$

Value of t is used in ADACLUS as an approximative measure of the deviation of data points density inside clusters from the "regular" one (i.e. uniform inside clusters). Let us show that, if the density of data points inside clusters is approximately uniform, then the value of t should be close to $\frac{1}{\sqrt{d}}$. First, we recall the fact that the uniform distribution of data points in clouds appears as a consequence of their Poisson distribution in data space [144, 145]. Next, according to (3.5), the minimal distance MD can be approximated by a sum of d independent one-dimensional minimal distances, taken along the certain coordinate directions. It is known from probability theory that in the case of Poisson distribution of data points in $\mathbf{R}$, the distance between neighboring points (i.e. minimal distance in our terminology) is distributed by the exponential law. Consequently, the law of distribution of minimal distances MD in the case of uniform density clusters in $\mathbf{R}^d$ can be well approximated by the law of distribution of a sum of d independent one-dimensional variables distributed exponentially with locally the same parameter λ. Of course, λ can vary from one cluster to another, but this does not create a problem because the following statement is valid.

Lemma 3.1 *Assume that* $\eta = \xi_1 + \ldots \xi_d$, *where* $\xi_1, \ldots, \xi_d$ *are independent one-dimensional random variables, distributed exponentially* $P(\xi_i < x) = 1 - e^{-\lambda x}, \quad 1 \leq i \leq d, \lambda > 0$. *Then, for any* $\lambda > 0$:

$$t_\eta = \frac{\sqrt{E(\eta - E\eta)^2}}{E\eta} = \frac{1}{\sqrt{d}},$$

where $E\zeta$ *denotes the mathematical expectation (mean value) of a random*

variable ζ.

Proof: For the exponential distribution,

$$E\xi_i = \frac{1}{\lambda}, \quad E(\xi_i - E\xi_i)^2 = \frac{1}{\lambda^2}, \quad 1 \leq i \leq d.$$

Consequently $E\eta = \frac{d}{\lambda}$ and therefore independence of ξ_i implies that $E(\eta - E\eta)^2 = \frac{d}{\lambda^2}$. Therefore, $\frac{\sqrt{E(\eta - E\eta)^2}}{E\eta} = \frac{1}{\sqrt{d}}$.

Lemma 3.1 implies that in the case of uniform (or approximately uniform) density of data points inside each cluster, the value of t will be close to $\frac{1}{\sqrt{d}}$. When the distribution of points inside clusters deviates from the uniform one, the value of t increases and t is a sensitive characteristic. For example, for $d = 2$, the value of t will be about $\frac{1}{\sqrt{2}} \approx 0.7$ in the case of uniform density clusters and about 1.0 for clusters which are already rather irregular in density.

Note that we do not make here any assumptions about generator of the data set. The proposed method of measuring density irregularity by the value of characteristic t is not model-based. It is applicable for any sort of data - stochastic or deterministic (chaotic, for example).

2) The second parameter, $T \geq 0$, is a *threshold parameter*. It determines a field function value threshold which divides inner-cluster and outer-cluster domains in the data space. T serves like a hierarchial parameter: big values of T result in more solid, dense inner-cluster domains and in treating relatively sparse areas as outer-cluster domain. Contrary to the first parameter,

parameter T may be chosen (or altered) at the end of the clustering process, because changing it's value requires repetition only of the final step of the algorithm.

3) S, *screen size,* - natural number which gives the upper bound for the "screen" linear size in pixels. The term "screen" is used here for the domain in data space where clustering is performed. Without lost of generality, we assume that this domain is a d-dimensional rectangular parallelogram which longest side does not exceed S pixels.

3.4.3 ADACLUS Algorithm Description

ADACLUS algorithm performs clustering in several steps. Without lost of generality we will assume in this section that the data set consists of N $d-$dimensional vectors: $\{\vec{P}_1, \vec{P}_2, \ldots, \vec{P}_N\}$, $\vec{P}_i = (p_i^1, \ldots, p_i^d) \in \mathbf{R}^d$, and $N > 1$.

If not specified by the user, the values of parameters T and S are set automatically to $T = 1$, $S = 1000$. Here, the value $T = 1$ means that value of the field function for cluster boundaries is set to 1. The value $S = 1000$ means that the distance between data points which is less than $1/1000$ of the side length of the screen is considered as a very small one and such points are glued together. Such value of S is quite natural for visual clustering and for computer geometry purposes.

The parameter G is calculated during STEP 3 if it was not pre-defined by the user.

STEP 1. Quantization and integer-coding.

First, for each of d coordinate axes the minimum and maximum values of corresponding data point component are determined: $m_k = \min\{p_i^k, 1 \leq i \leq N\}, M_k = \max\{p_i^k, 1 \leq i \leq N\}, 1 \leq k \leq d$. Then, for each $1 \leq k \leq d$ the interval $[m_k, M_k]$ is divided into S intervals ("quanta") $\Delta_j^k (1 \leq j \leq S)$ of equal length: $\Delta_j^k = [m_k + (j-1)\frac{M_k - m_k}{S}, m_k + j\frac{M_k - m_k}{S}]$, where S is the screen size parameter. Finally, each of d vector components of each data point is coded by a pair of integers (the number of the component and the number of the interval to which this component belongs):

$$p_i^k \rightarrow (k, \tilde{p}_i^k), \quad \text{if } p_i^k \in \Delta_{\tilde{p}_i^k}^k.$$

All further actions, related to all data points, are performed only with pairs of integers (k, j), such that the corresponding interval Δ_j^k is not empty (i.e. some data points components fall into it). Evidently, the total number of such pairs does not exceed dN.

STEP 2. Minimal distances (MD) calculation.

For each data point $\vec{P}_i$ the corresponding minimal distance $MD(i)$ from this data point to other data points is calculated in the sense of metric ρ_1. Since coordinates of data points were already converted to integers during STEP 1, all the minimal distances will be expressed by positive integers. It immediately follows from the definition of the metric ρ_1 (see Definition ??) and the assumption $N > 1$.

STEP 3. Parameter G automatic determination.

We calculate the value of G as mentioned in Section 3.4.2, here, we assume in (3.15) that $Smin = 200$ data points is a small enough number to observe each of

them separately and therefore, "natural" clustering in this case should be purely local-based. It is assumed that $Smax = 2000$ data points is a big enough number to make observation of individual points impossible, forcing therefore the "natural" clustering to be performed from more global point of view. Both numbers are very approximative, but the clustering results are quite robust to their variations, which is discussed in Section 3.4.6.

STEP 4. Calculation of the localization radius R.

In order to initiate the ADACLUS algorithm we first determine the size of the neighborhood in which the local considerations will be performed for each data point. Taking in account the value of G, which was calculated during STEP 3, the radius R of the "ball" representing this neighborhood is chosen as:

$$R = \min\{GR_{loc} + (1-G)R_{glob},\ 1/2\sqrt[d]{100}\, m_{MD}\}, \tag{3.17}$$

$$R_{loc} = \min\{x : x \geq R_{glob},\ g^{MD}_{emp}(x) = 0\}, \tag{3.18}$$

$$R_{glob} = \min\{x : P_{emp}(MD < x) = 0.9\}, \tag{3.19}$$

where $g^{MD}_{emp}(x)$ denotes the empirical density (frequency histogram) of the MD distribution calculated in STEP 2, R_{glob} represents a 90% of this distribution, and R_{loc} is it's first zero greater than 90%. See Fig.3.13. Note that according to Definition 3.5, the "R-ball" in $\mathbf{R}^d$ will be represented by the d-dimensional cube with the side length $2R$ in Euclidean metric. The radius R of this "ball" is restricted to $c \cdot m_{MD}$ (c times average minimal distance between data points) in order to keep the complexity of the calculations linear with respect to N for any kind of data set. The value of the coefficient $c = 1/2\sqrt[d]{100}$ is chosen in such a way

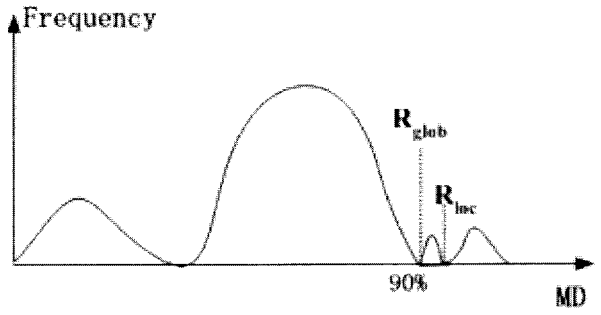


Figure 3.13: Localization radius choice in ADACLUS

that the average number of data points in a d-dimensional Euclidean cube with side length $2R$ is restricted to approximate 100 data points maximum.

According to (3.17)-(3.19) the localization radius R is computed as the restricted mixture of "global" and "local" values of the neighborhood radius. The mixture is taken in the proportion, determined by the parameter G (which was already calculated in STEP 3 or manually set by the user). The value of R_{glob} represents "10%-cut maximum" of the distribution of MD. 10% margin is cut off taking in account the possibility of the outliers and for algorithm stability. If the distribution of data points is approximately the same in different parts of the data set, then R_{glob} gives a suitable value for the neighborhood radius. Otherwise, it may be not so. One example is the situation when there are several dense parts in the data set which include majority of data points, but there are also sparse areas which are big enough but contain a relatively small amount of data, see Fig.3.14. If we are performing clustering from the global point of view, then, when we cluster in sparse parts we should remember that they are really sparse. This information could not be extracted locally - it comes from the global distribution of MD and

is summarized in the value of R_{glob}. So, for global-based clustering we can use this value. .

But if the aim is to perform clustering in sparse parts of the screen more locally, without taking in account far away density characteristics, then the appropriate value of the neighborhood radius should be determined by sparse areas not based on the whole MD distribution, but on it's right tail only. As we still do not want to be influenced by the outliers in this case, we choose R_{loc} - the first zero of MD histogram greater than 90%. It corresponds to the largest value of MD which still belongs to sparse component of MD distribution, cutting the outliers off, see Fig.3.13. Such value will be good for sparse areas, but it will also work for dense areas, because the final decisions are made based on the microanalysis performed inside the R-neighborhood of the data point. If this neighborhood was chosen too big, it will not spoil the results but rise the amount of calculations. However, in the situation which we consider now, the total area of dense parts is not very big in comparison to the whole screen, so the amount of extra calculations will be reasonable.

The balance between these two situations is calculated by the formula (3.17) in accordance with the value of parameter G.

STEP 5. Local microanalysis. Determination of a_i and b_i for each data point $\vec{P}_i$.

After localization radius R was determined in STEP 4, for each data point $\vec{P}_i$ $(1 \leq i \leq N)$, we mark all data points belonging to its neighborhood $Ne_R(\vec{P}_i) = \{\vec{P} : \rho_1(\vec{P}, \vec{P}_i) \leq R\}$, including $\vec{P}_i$ itself. And we consider the part of the whole

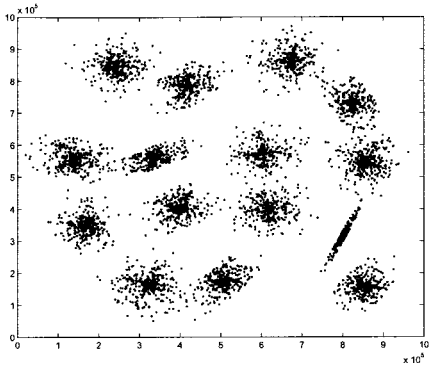


Figure 3.14: Example of a data set with non-uniform distribution of data points

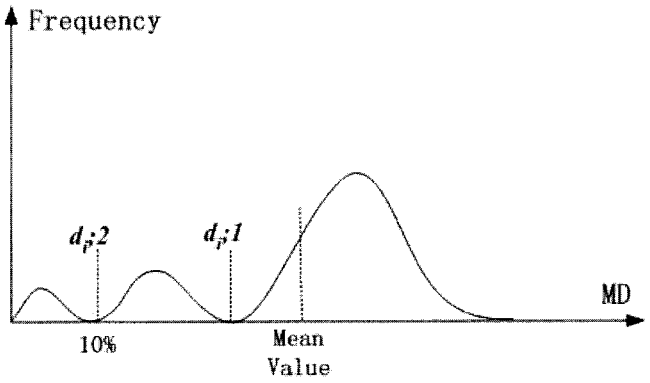


Figure 3.15: The choice of local parameters a_i, b_i in ADACLUS

MD distribution (built in STEP 2) which consists of only those minimal distances, which are associated with the data points from $Ne_R(\vec{P}_i)$. Let's denote the corresponding frequency histogram $g_{emp}^{MD;\,i}(x)$, it's mean value m_{MD}^i, and it's standard deviation std_{MD}^i.

Based on the histogram $g_{emp}^{MD;\,i}(x)$, we define a_i as the "typical minimum" of MD inside $Ne_R(\vec{P}_i)$ in order to eliminate the effect of local outliers and irregular dense area, which represents "typically small" value of MD in $Ne_R(\vec{P}_i)$. More precisely, we define a_i as following. (see Fig.3.15)

$$a_i = \frac{d_{i;\,1} + d_{i;\,2}}{2}, \tag{3.20}$$

where $d_{i;1}$ is the biggest zero of $g_{emp}^{MD;i}(x)$ which does not exceed the mean:

$$d_{i;1} = \max\{x : x \leq m_{MD}^{i},\ g_{emp}^{MD;i}(x) = 0\}, \tag{3.21}$$

and $d_{i;2}$ is 10%-fraction of the distribution $g_{emp}^{MD;i}(x)$:

$$d_{i;2} = \max\{x : P^{MD;i}(MD < x) = 0.1\}. \tag{3.22}$$

We use $d_{i;1}$ to eliminate the potential effect cased by local outliers, which are the outliers detected under local view. One example is the situation when the "R-ball" of a data point in sparse area includes several data points near to a dense cluster. We should not count these data points in, since they are local outliers for the dense cluster. For the same reason, when the "R-ball" of a data point in sparse area includes several data points belonging to a dense cluster, we also need to cut these data points off. This is why we set $d_{i;2}$.

Finally, taking in the account "$3\sigma-$rule", we define b_i based on the value of magnifier parameter G as:

$$b_i = a_i + (4 - 2G)std_{MD}^{i}. \tag{3.23}$$

So, the gap between a_i and b_i varies around "3σ" ranging from 2 times of standard deviation in the case of local clustering to 4 times of standard deviation in the case of maximum portion of global information added.

STEP 6. Computation of the field function in all data points.

Based on the values a_i, b_i $(1 \leq i \leq N)$ which are determined in STEP 5, we calculate the field function $F(\vec{P}_i)$ for all data points $\vec{P}_1, \ldots, \vec{P}_N$ by means of the formula (3.11).

STEP 7. Determination of the point-based clusters.

To determine point-based clusters (Definition 3.7), ADACLUS employs the standard hill-climbing algorithm and applies the similar method for clustering procedure as [2]. The direction of the hill-climbing is determined by the vector of the field function gradient, given in (3.12). After clustering process, data points which can not be assigned to any clusters are labeled as outliers.

STEP 8. Determination of the boundary-based clusters and boundary detection.

Direct drawing exact cluster boundaries $\{\vec{X} : F(\vec{X}) = T\}$ according to the Definition 3.8 may be time-consuming in the case of big screen, because it requires additional calculation of field function $F(\vec{X})$ in many space points. (Note that in STEP 6 the field function was calculated for data points only.) The total number of space points in the screen after quantization (see STEP 1) equals to S^d, which could be quite big number in comparison with N. But the amount of calculations still can be reduced to $O(N)$ by preliminary marking the space points, such that the field function is greater than zero for them. This can be done together with calculation of a_i, b_i during STEP 5. Namely, after b_i was determined for a certain i, we mark all space points in the ρ_1-ball of b_i-radius with the center in $\vec{P}_i : Ne_{b_i}(\vec{P}_i) = \{\vec{X} : \rho_1(\vec{X}, \vec{P}_i) < b_i\}$. This will reduce the amount of calculation to $O(N)$, where N is the number of space points in which the field function is to be calculated during this step.

3.4.4 Complexity Analysis: Linear Complexity of ADACLUS

The time complexity of ADACLUS is analyzed theoretically by steps below.

STEP 1. First step required dN operations for determining m_k, M_k $(1 \leq k \leq d)$ and another dN operations for the coding procedure. The total complexity of this step is $O(dN)$.

STEP 2. In order to make the complexity of this step linear with respect to N, the tables of correspondence between quantum numbers (i.e. integers from 1 to S) and the data points which have this number as the code of their $k-$th coordinate, are created for each $1 \leq k \leq d$. These tables are computed simultaneously with the coding procedure during STEP 1, so it requires dN additional operations only. Based on the mentioned tables, the minimal distance (MD) is calculated for each data point $\vec{P}_i$ by searching the nearest data point among only such data points $\vec{P}_j$ that their coordinate codes $\widetilde{p}_j^k$ are neighboring to the coordinate code $\widetilde{p}_i^k$. It requires $O(dN)$ operations, so the total complexity of this step is $O(dN)$, too.

STEP 3. In this step, the calculations are performed according to the formulas (3.14), (3.15), and (3.16). The complexity of these calculations does not depend neither on N nor on d. It is constant.

STEP 4. Calculations in this step are performed according to the formulas (3.17), (3.18), and (3.19). They are based on MD distribution which is obtained together with MD values during STEP 2 (without rising it's complexity). Complexity of the STEP 4 does not depend neither on N nor on d. It linearly depends on S only.

STEP 5. The complexity of this step equals to the complexity of calculation of the histograms $g_{emp}^{MD;\,i}(x)$ for all data points $\vec{P}_i$. All other calculations here consist in direct evaluation of the expressions (3.20), (3.21), (3.22), and (3.23). In order to calculate the histogram $g_{emp}^{MD;\,i}(x)$ for $\vec{P}_i$, it is enough to determine all data points which belong to the vicinity $Ne_R(\vec{P}_i)$. This could be done based on the tables of correspondence between quantum numbers and the data points (see complexity analysis for STEP 2). Therefore, the total complexity of the STEP 5 equals to $O(cN)$, where c is the average number of data points in the vicinity $Ne_R(\vec{P}_i)$ for an arbitrary data point $\vec{P}_i$. According to the definition of R in (3.17), there are not more than 100 data points in $Ne_R(\vec{P}_i)$ in average, so the complexity of the step does not exceed $O(100N)$ which is linear with respect to N and can be written as $O(N)$.

STEP 6. According to (3.11), the complexity of this step in $O(kN)$, where k is the average number of data points influencing the arbitrary data point $\vec{P}_i$ (i.e. such $\vec{P}_j$ that $f_{a_j,b_j}(\vec{P}_i) > 0$). It is easy to see that the same k represents also the average number of data points, which are influenced by a randomly chosen data point $\vec{P}_i$. The data points, influenced by $\vec{P}_i$ are such and only such data points $\vec{P}_j$, that their distance to $\vec{P}_i$ is less than b_i : $\rho_1(\vec{P}_i, \vec{P}_j) < b_i$. According to (3.20),(3.23), b_i does not exceed $a_i + 4std_{MD}^i$, where a_i is the typical minimum of the minimal distance between data points in the R-neighborhood of $\vec{P}_i$ and std_{MD}^i is the standard deviation of this minimal distance in this neighborhood. The worst case, giving the largest number of data points in the neighborhood of $\vec{P}_i$, is evidently the case when typical minimum of MD equals to typical MD in the neighborhood

of $\vec{P}_i$, i.e. when the distribution of data points in this neighborhood is uniform (because the more there are MD's larger than minimal ones, the less will be the number of data points in the neighborhood of $\vec{P}_i$). But in the case of approximately uniform data points distribution in the neighborhood of $\vec{P}_i$, according to Lemma 3.1, the value of std^i_{MD} will equal approximately to $\frac{1}{\sqrt{d}}m^i_{MD}$, and according to (3.20)-(3.22), the value of a_i for exponential distribution of MD (corresponding to uniform distribution of data points) will not exceed m^i_{MD}. Consequently in the uniform case we have:

$$b_i \leq \left(1 + \frac{4}{\sqrt{d}}\right) m^i_{MD} < 5m^i_{MD},$$

where m^i_{MD} is average MD in the neighborhood of $\vec{P}_i$. Consequently, in the case of uniform density of data points around $\vec{P}_i$, the average number of them in b_i-neighborhood of $\vec{P}_i$ will be approximately equal to 10^d. For any other distribution, it will be smaller. Consequently,

$$k \leq 10^d.$$

It is theoretical upper bound for k. In real calculations, it is much smaller, because for uniformly distributed data points, the magnifying parameter G is set to 1 (see(3.14),(3.16)) and in this case, according to (3.23), $b_i = a_i + 2std^i_{MD} < 3m^i_{MD}$. So, the inequality for k turns to $k < 6^d$, which is still not a typical value but an upper bound.

We conclude that the complexity of this step is typically lower than $O(6^d N)$, which means $O(36N)$ in the case of $\mathbf{R}^2$ screen. Theoretical upper boundary for

the complexity of this step is $O(10^d N)$. This complexity is linear with respect to N.

STEP 7. The hill-climbing algorithm has linear complexity with respect to the number of data points [2], so the complexity of this step is $O(N)$.

STEP 8. The complexity of this step is $O(N)$ as explained in the description of this step.

Therefore, ADACLUS has linear complexity. The total complexity is $O(N)$ with respect to the number of data points N. It is $O(dN)$ with respect to both data points number and dimension d of the data space.

3.4.5 Comparison and Performance Analysis

In this study, the proposed algorithm ADACLUS was designed for 2D data. It was implemented and tested on the following 2-dimensional data sets. We designed four testing data sets with non-uniform distribution of data points inside clusters. Generally, there are clusters of arbitrary shapes in the data sets. To simulate general situations for clustering of non-uniform density data, we differ two shape features of clusters in our data sets: circular/non-circular, and concave/non-concave and we also build these data sets with outliers. The data sets names, the shape and statistic features of these data sets are listed in Table.3.3 and Table.3.4. ADACLUS can also get the correct results for data sets $D1$ and $D2$. Here, only clusters's boundaries of $D1$ and $D2$ detected by ADACLUS are shown in Fig.3.16 to avoid duplicating. Moreover, ADACLUS is able to cluster more challenging data sets such as $D3$ to $D6$.

Table 3.3: Shape Features of Testing Datasets

Data-set name	Shape feature 1	Shape feature 2	Data points distribution	Number of Outliers
D3	Circular	Non-concave	Non-uniform	3
D4	Non-circular	Non-concave	Non-uniform	3
D5	Non-circular	Concave	Non-uniform	3
D6	Circular	Concave	Non-uniform	3

Table 3.4: Statistical Features of Testing Datasets

Data-set name	Number of data points	Mean m_{MD}	Standard deviation std_{MD}	$\frac{std_{MD}}{m_{MD}}$
D3	76	0.4714	0.3793	0.8045
D4	127	0.3038	0.2799	0.9213
D5	191	0.3242	0.3262	1.0062
D6	289	0.2980	0.2842	0.9537

We applied ADACLUS and classic clustering methods such as K-MEANS, hierarchical Single Linkage and Complete Linkage methods, and DBSCAN on $D3$ - $D6$ data sets. We choose these classic clustering methods because they are representative in this area, and standard implementations of these methods are available. We can avoid the potential problems causing by different implementations, which could lead us to bias in results comparisons. K-MEANS method is still the most widely used method, and it is based on the optimization principles which form a foundation for many well-known data mining techniques. We employed the squared Euclidean distance and random selection of initial cluster centers for K-MEANS in our testing. Linkage hierarchical methods have a good ability of handling clusters with some complicated shapes. DBSCAN is another well known density-based clustering method for dealing with arbitrary shape clusters and outliers. The implementation of DBSCAN is provided by the author and all parameter settings are based on the instruction given in the original paper ($k = 3$ and $MinCard = 4$ for our 2-dimensional data sets).

By comparing with those methods, we can have a more clear idea of the performance of ADACLUS. The characteristics of ADACLUS such as extracting both global and local information from data set and detecting outliers based on both global view and local view, have been well demonstrated. All results provided here are generated by automatic setting of parameters, which was described in Section 3.4.3. For the user who has a very clear expectation of the data set, parameters also can be set based on his knowledge of data.

In Fig.3.17, the picture of $D3$, statistic information, cluster boundaries and

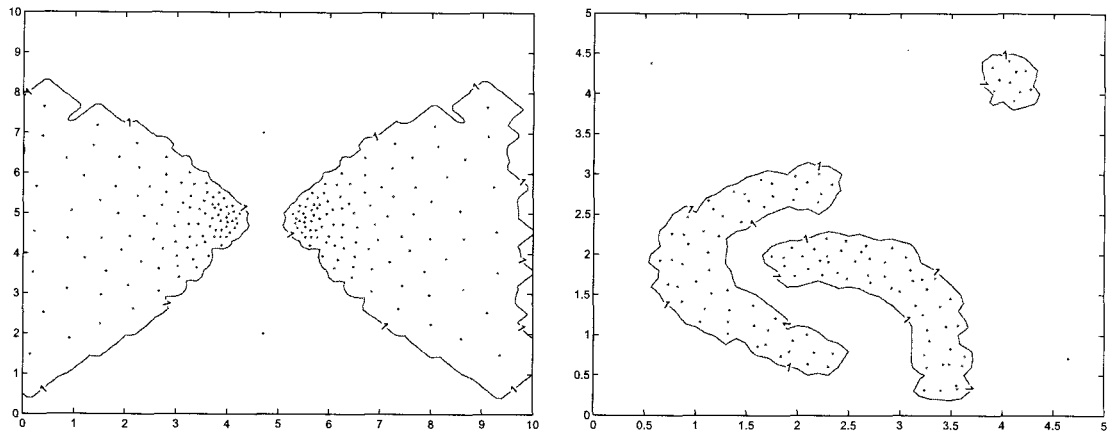


Figure 3.16: The testing on data sets $D1$ and $D2$ of ADACLUS: (a)Clusters' boundaries of data set $D1$ detected by ADACLUS; (b)Clusters' boundaries of data set $D2$ detected by ADACLUS

clustering result of $D3$ by ADACLUS are shown. The three clusters are well separated as well as three outliers.

In Fig.3.18, the clustering results of testing data set $D3$ by four classic clustering algorithms are shown.

Data set $D3$ is a challenging data set with non-uniform density clusters and outliers. There are three outliers and three clusters among which two clusters have relative uniform inner cluster density but different inter cluster density and the biggest cluster has non-uniform inner cluster density. The difficulty here is that the smallest cluster is very close to the biggest one. From the clustering results, the ability of ADACLUS for discovering clusters based on both global and local information is well demonstrated. K-MEANS and Single-linkage method can not separate the clusters correctly. Complete-linkage method fails to detect the outliers. DBSCAN successes in the outliers detection, but fails in separation of the

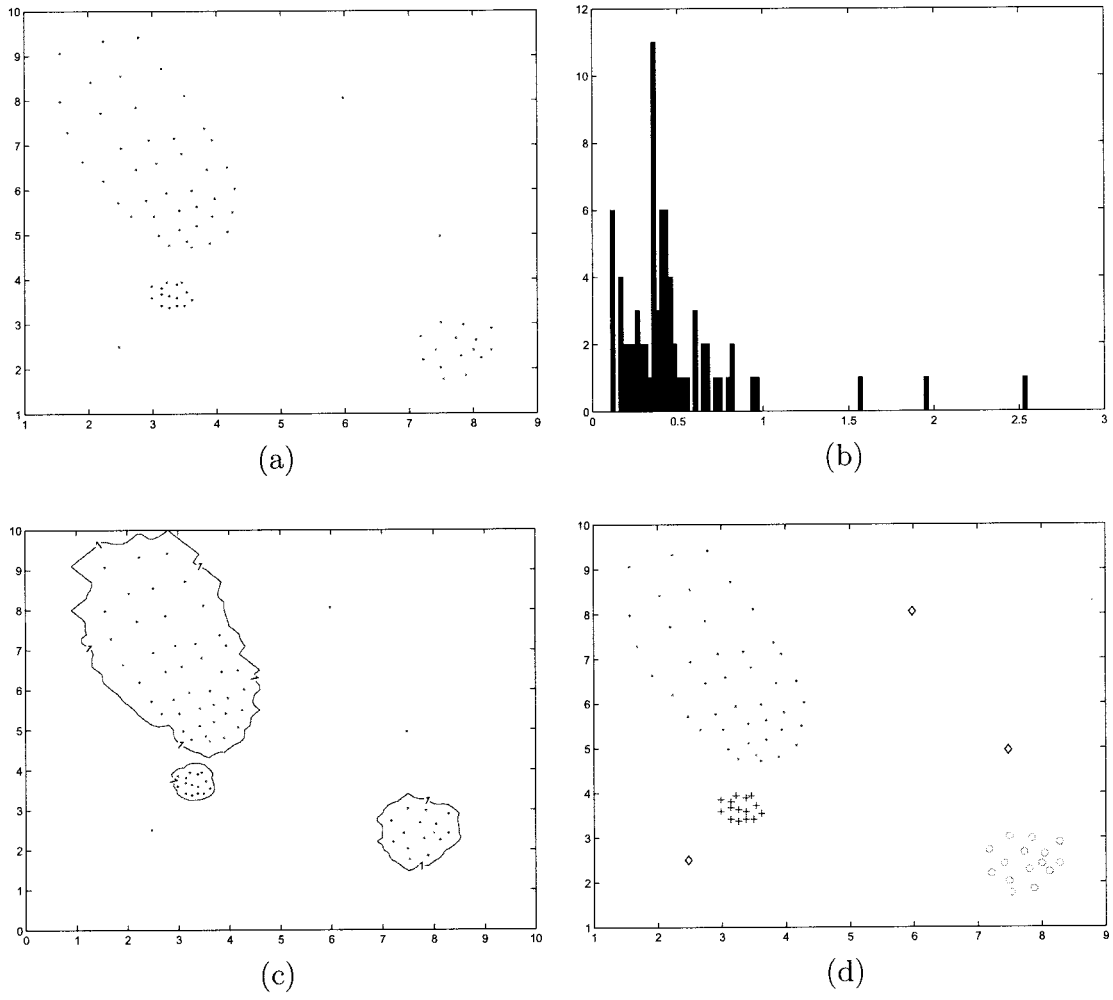


Figure 3.17: The testing on data set $D3$ of ADACLUS: (a)The picture of data set $D3$; (b)The distribution of all minimal distances of data set $D3$; (c)The boundary of data set $D3$ built by ADACLUS, $T = 1$; (d)Clustering result of $D3$ by ADACLUS

nearby clusters.

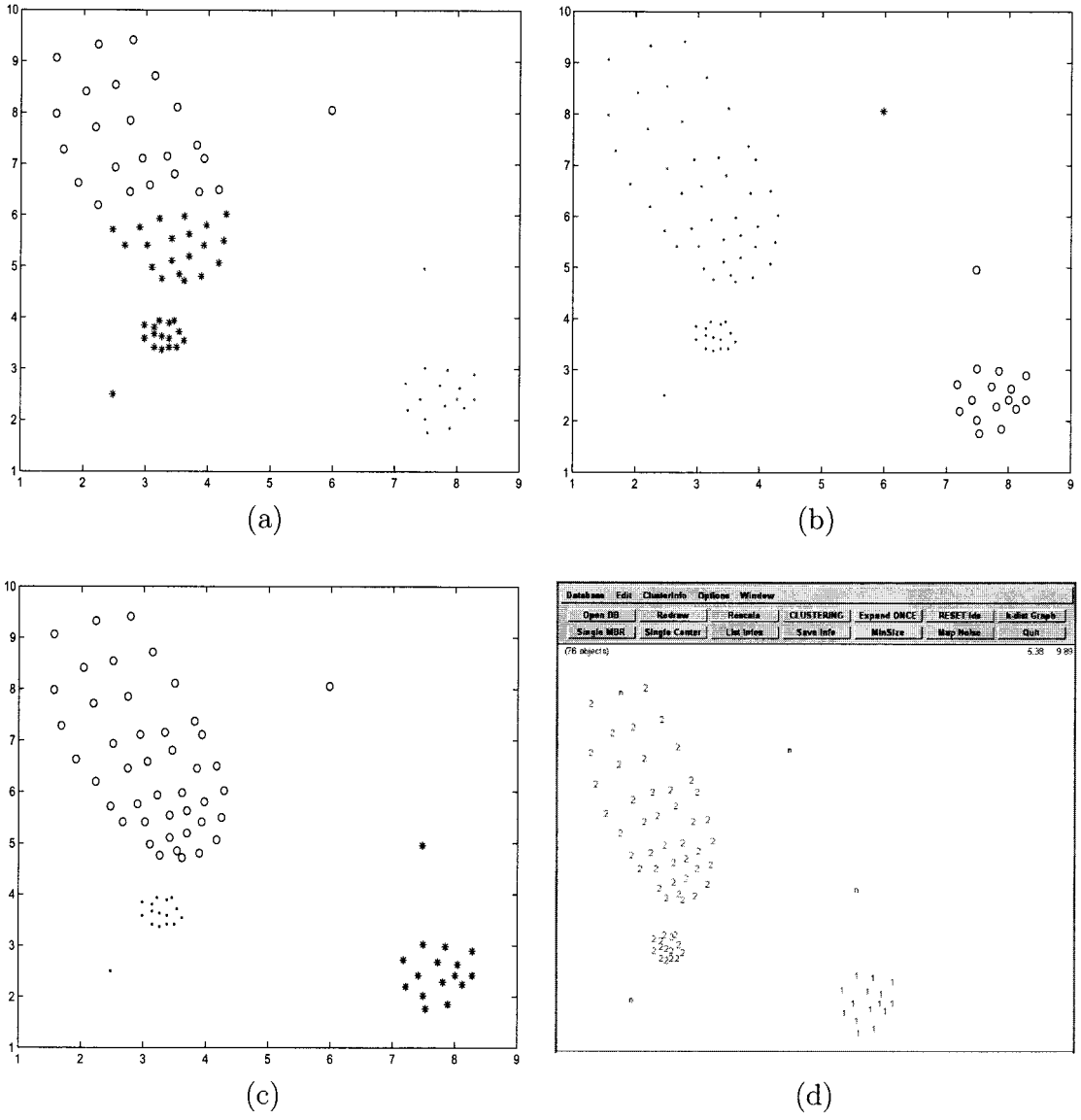


Figure 3.18: Clustering results of $D3$ by comparison methods: (a)Clustering result of testing data set $D3$ by K-MEANS; (b)Clustering result of $D3$ by Single-linkage algorithm; (c)Clustering result of $D3$ by Complete-linkage algorithm; (d)Clustering result of testing data set $D3$ generated by DBSCAN, where $EPS = 0.8095$

In Fig.3.19, the picture of data set $D4$, statistic information, cluster boundaries and clustering result of $D4$ by ADACLUS are shown.

In Fig.3.20, the clustering results of testing data set $D4$ by four classic clustering

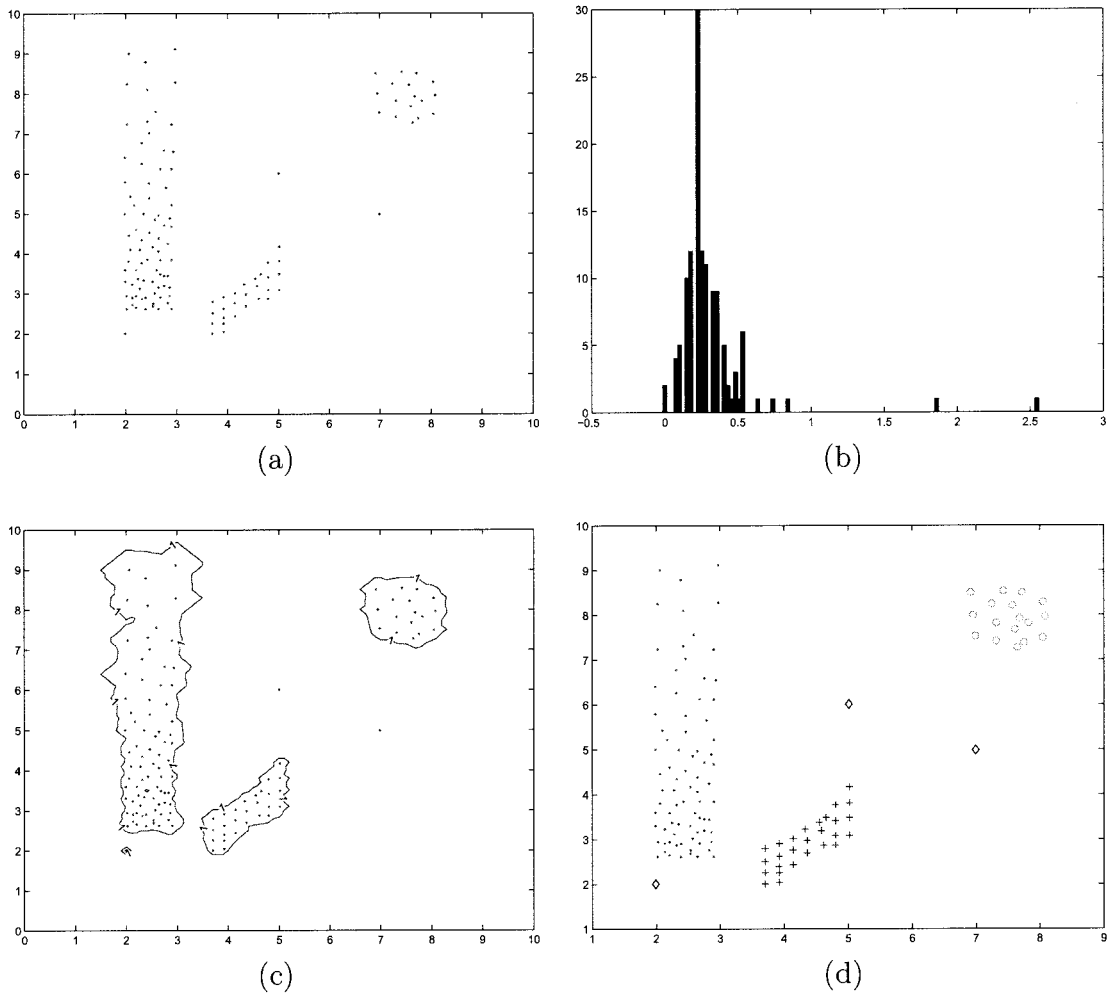


Figure 3.19: The testing on data set $D4$ of ADACLUS: (a)The picture of data set $D4$; (b)The distribution of all minimal distances of data set $D4$; (c)The boundary of data set $D4$ built ADACLUS, $T = 1$; (d)The clustering result of $D4$ by ADACLUS

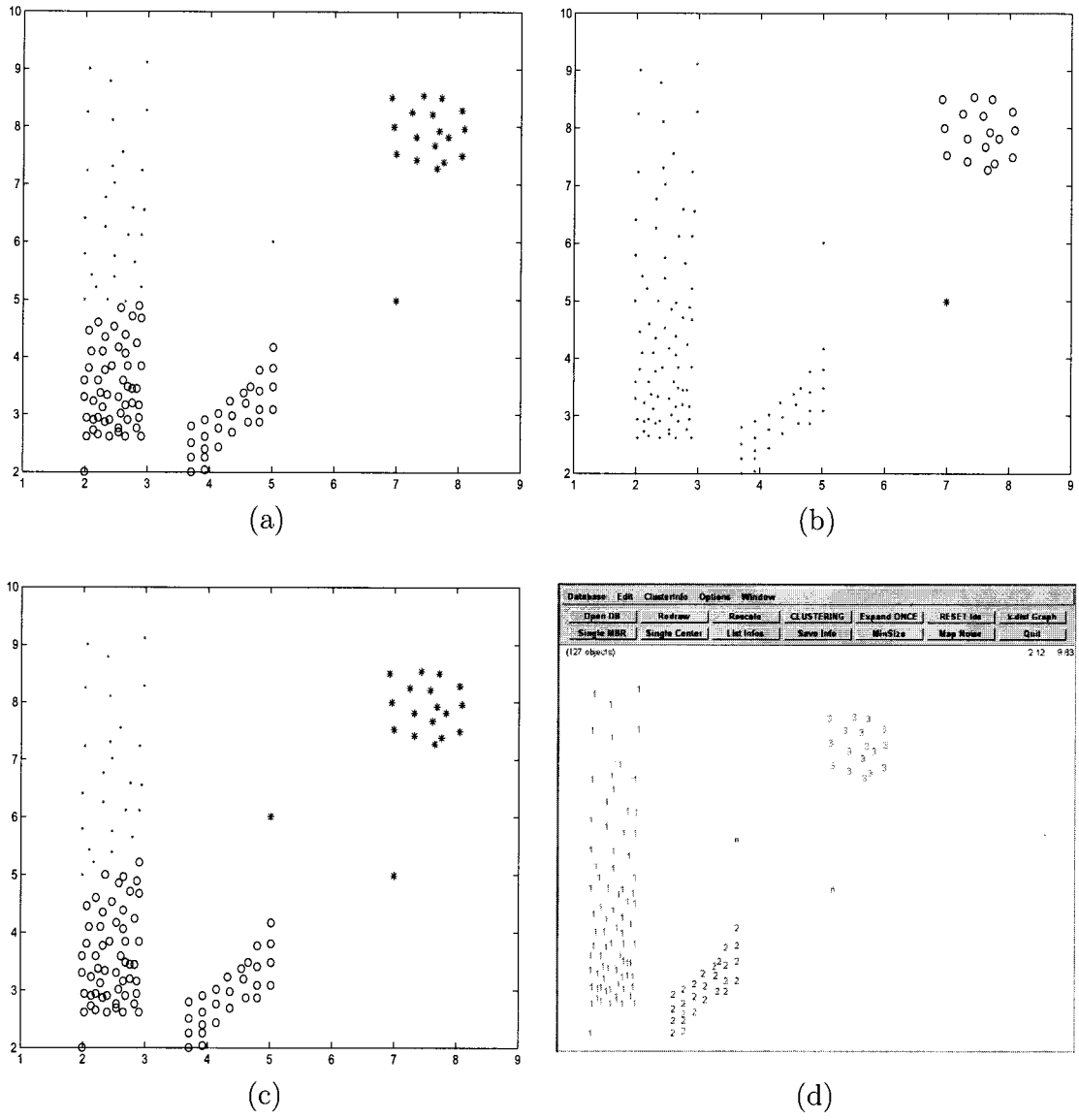


Figure 3.20: Clustering results of $D4$ by comparison methods: (a)Clustering result of testing data set $D4$ by K-MEANS; (b)Clustering result of $D4$ generated by Single-linkage algorithm; (c)Clustering result of $D4$ generated by Complete-linkage algorithm; (d)Clustering result of testing data set $D4$ generated by DBSCAN, where $EPS = 0.7687$

algorithms are shown.

In Data set $D4$, there are three clusters of which two have relative uniform density, one has non-uniform density. There are also three outliers in this data set. Based on the results given, we can see that K-MEANS and Complete-linkage method can not detect the non-circular clusters. Single-linkage method keeps the integrity of clusters but fails to separate them from each other. It also can not discover the outlier which is nearby the cluster. It seems that DBSCAN performs well on this data set, but if we check carefully, we can notice that it misses the local outlier which is close to the biggest cluster. Only ADACLUS is fully successful in cluster and outlier detection.

In Fig.3.21, the picture of data set $D5$, statistic information, cluster boundaries and clustering result of $D5$ by ADACLUS are shown.

In Fig.3.22, the clustering results of testing data set $D5$ by four classic clustering algorithms are shown.

Data set $D5$ is a very challenging one for most clustering algorithms. There are three clusters together with three outliers among which one is very close to one cluster. There are two clusters with non-uniform inner density and concave shape. Based on visual inspection, we can easily notice the density differences between clusters and outliers, although the smallest distance between one outlier to the nearby cluster is smaller than the biggest distance between data points inside that cluster. Except ADACLUS, no method can separate all clusters and detect all outliers correctly at the same time.

In Fig.3.23, the picture of data set $D6$, statistic information, cluster boundaries

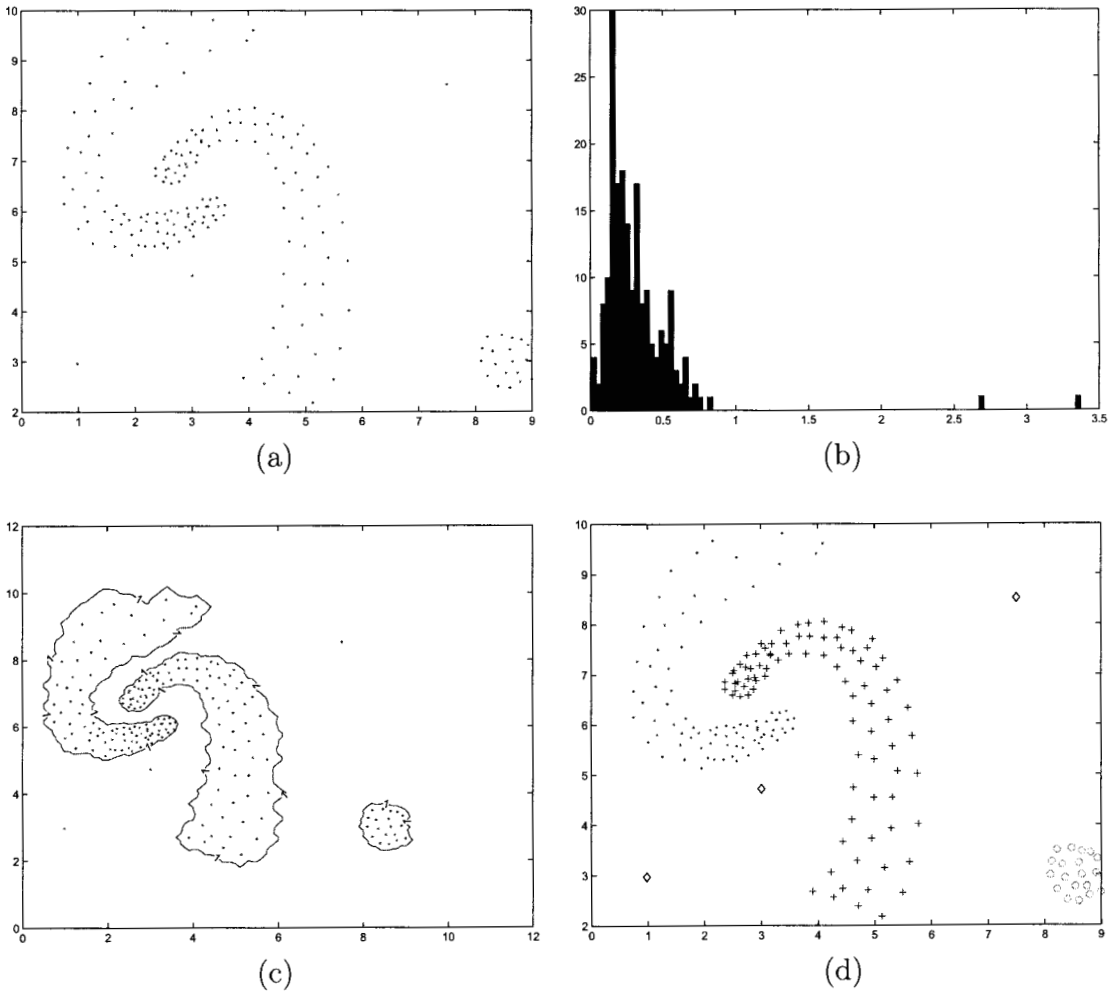


Figure 3.21: The testing on data set $D5$ of ADACLUS: (a)The picture of data set $D5$; (b)The distribution of all minimal distances of data set $D5$; (c)The boundary of data set $D5$ built ADACLUS, $T = 1$; (d)The clustering result of data set $D5$ by ADACLUS

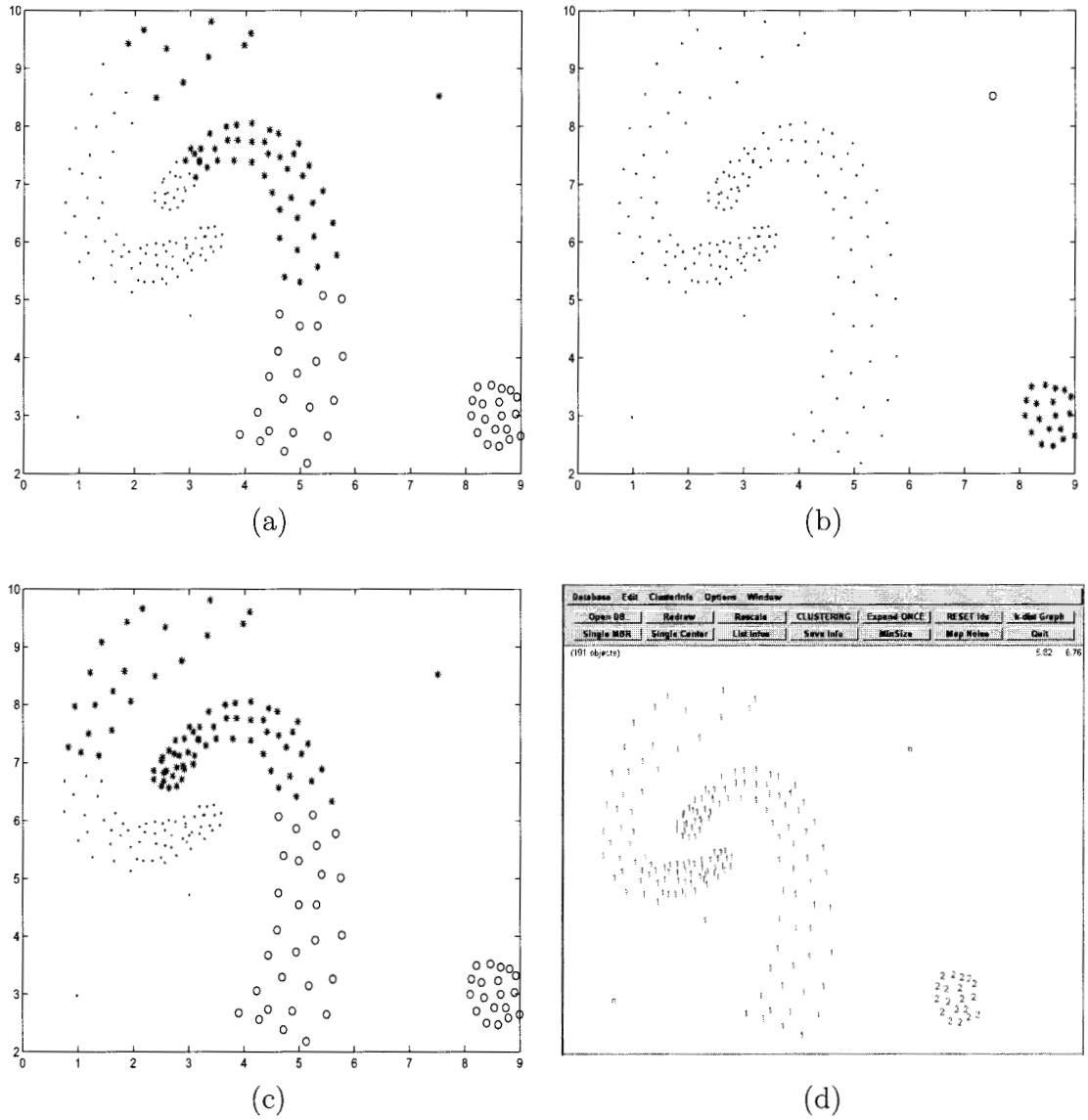


Figure 3.22: Clustering results of data set $D5$ by comparison methods: (a)Clustering result of testing data set $D5$ by K-MEANS; (b)Clustering result of $D5$ generated by Single-linkage algorithm; (c)Clustering result of $D5$ generated by Complete-linkage algorithm; (d)Clustering result of testing data set $D5$ generated by DBSCAN, where $EPS = 0.7432$

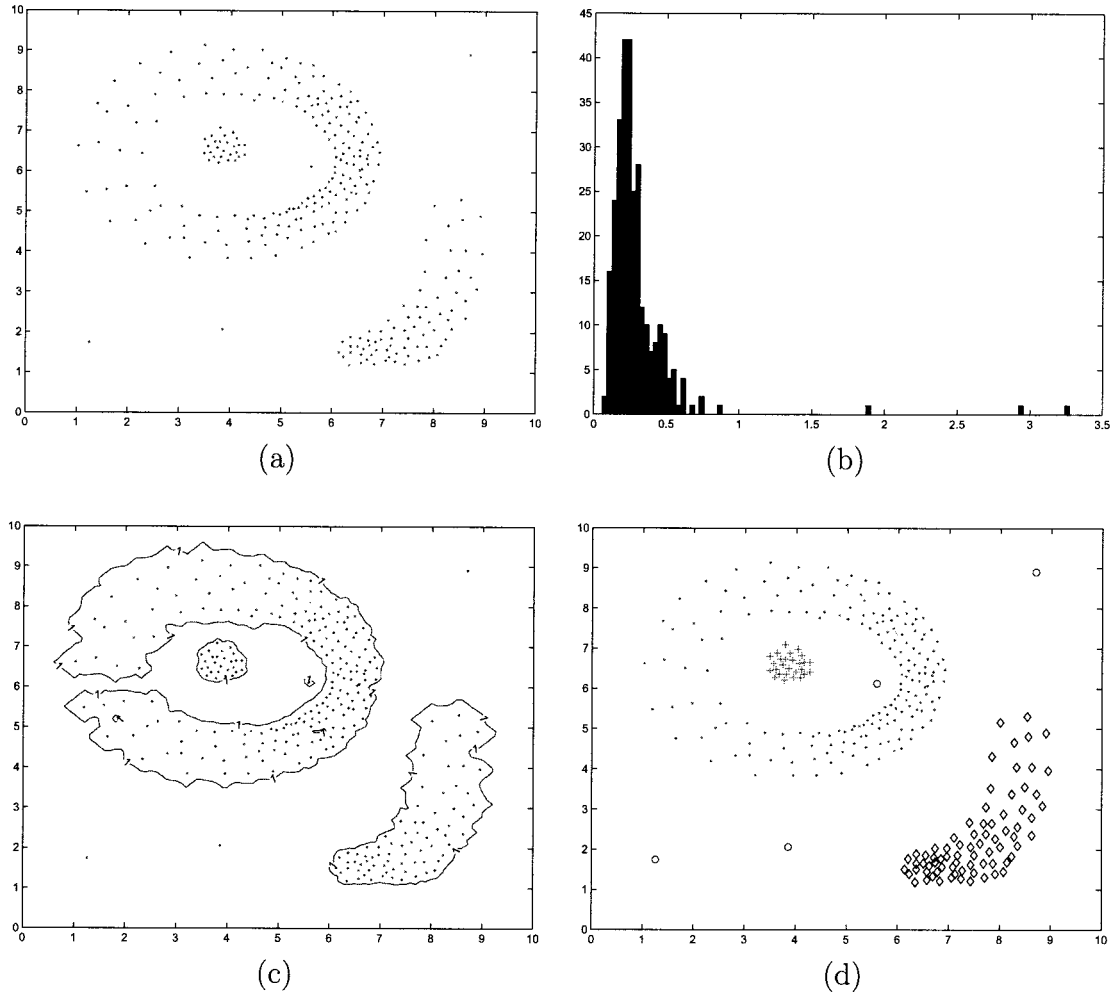


Figure 3.23: The testing on data set $D6$ of ADACLUS: (a)The picture of data set $D6$; (b)The distribution of all minimal distances of data set $D6$; (c)The boundary of data set $D6$ built ADACLUS, $T = 1$; (d)The clustering result of data set $D6$ ADACLUS

and clustering result of $D6$ by ADACLUS are shown.

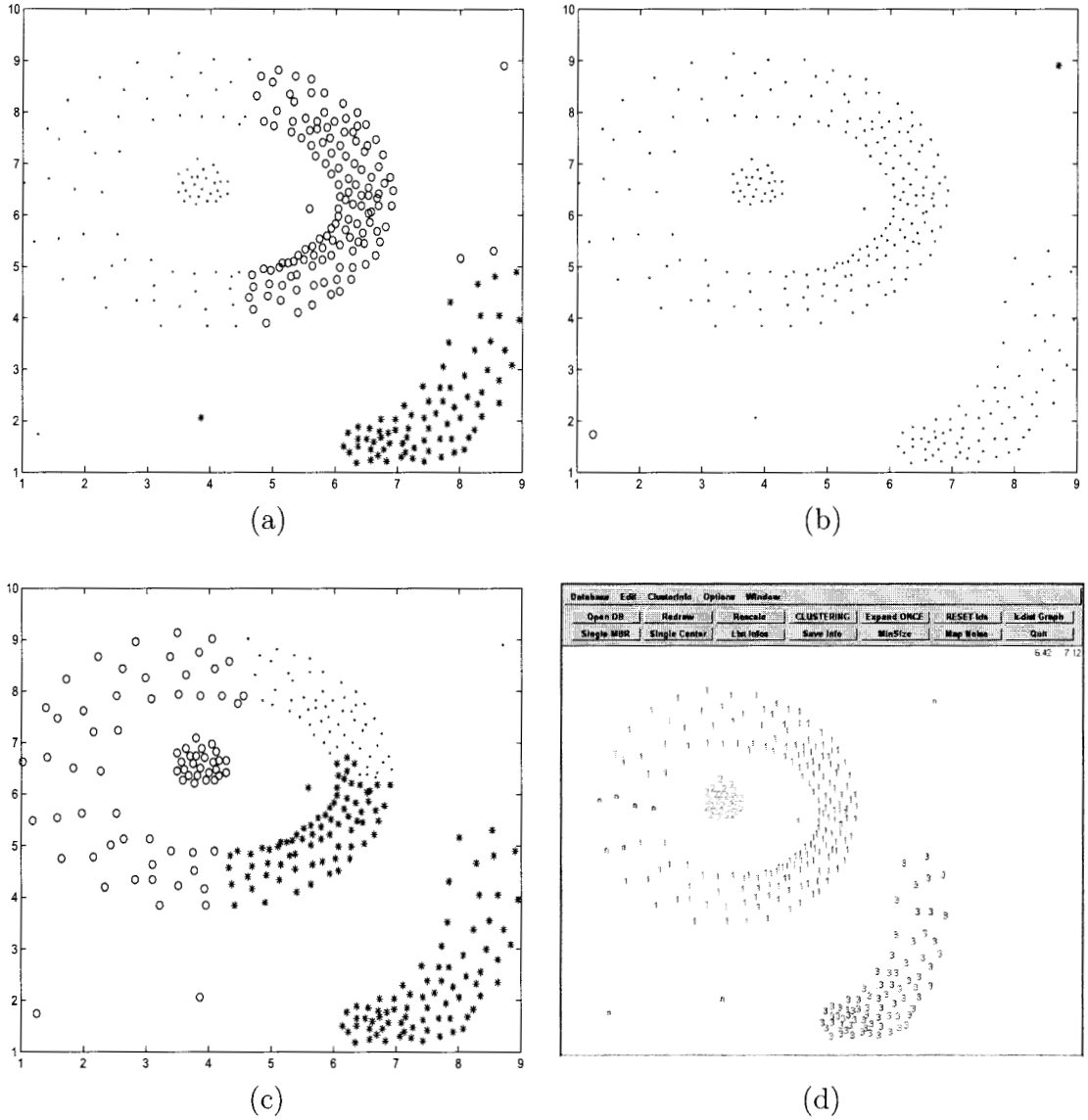


Figure 3.24: Clustering results of data set $D6$ by comparison methods: (a)Clustering result of testing data set $D6$ by K-MEANS; (b)Clustering result of $D6$ generated by Single-linkage algorithm; (c)Clustering result of $D6$ generated by Complete-linkage algorithm; (d)Clustering result of testing data set $D4$ generated by DBSCAN, where $EPS = 0.6737$

In Fig.3.24, the clustering results of testing data set $D6$ by four classic clustering algorithms are shown.

Data set $D6$ which is another very challenging data set to all well-known clus-

tering methods. In the data set $D6$, there is one small cluster inside another cluster which is a big ring shape cluster with non-uniform density. There is an outlier located inside the ring as well. It can be observed from the results that the ADACLUS detects natural clusters even in this complicated case. As it is seen from the Fig.3.24, other algorithms can not cluster this data set in the expected way. K-MEANS method and Complete-linkage method can not discover the ring shape cluster since they lack the ability to handle clusters with concave shapes. Single-linkage method suffers from correct separation of clusters. DBSCAN fails in keeping the integrity of the ring shape cluster and detecting one local outlier.

In Fig.3.26, we tested ADACLUS on the DENCLUE bench-mark data set $DD1$. The results of ADACLUS and DENCLUE are shown. The clustering result of DBSCAN on this data set is the same as DENCLUE. All three methods can detect all clusters accurately.

In Table.3.5, the accuracy of ADACLUS and all comparison methods on all testing data sets are given. The definition of accuracy is given below. For a given data point, let α_i and l_i be the cluster label and the label pre-defined.

$$AC = \frac{\sum_{i=1}^{N} \delta(\alpha_i, MAP(l_i))}{N} \tag{3.24}$$

where, $\delta(x, y)$ is the delta function which equals one if $x = y$ and equals zero otherwise. $MAP(l_i)$ is the best mapping that maps each cluster label l_i to the pre-defined label.

From Table.3.5, we can see that except for DBSCAN bench-mark data set (data set $D2$) and DENCLUE bench-mark data set (data set $DD1$), ADACLUS is able

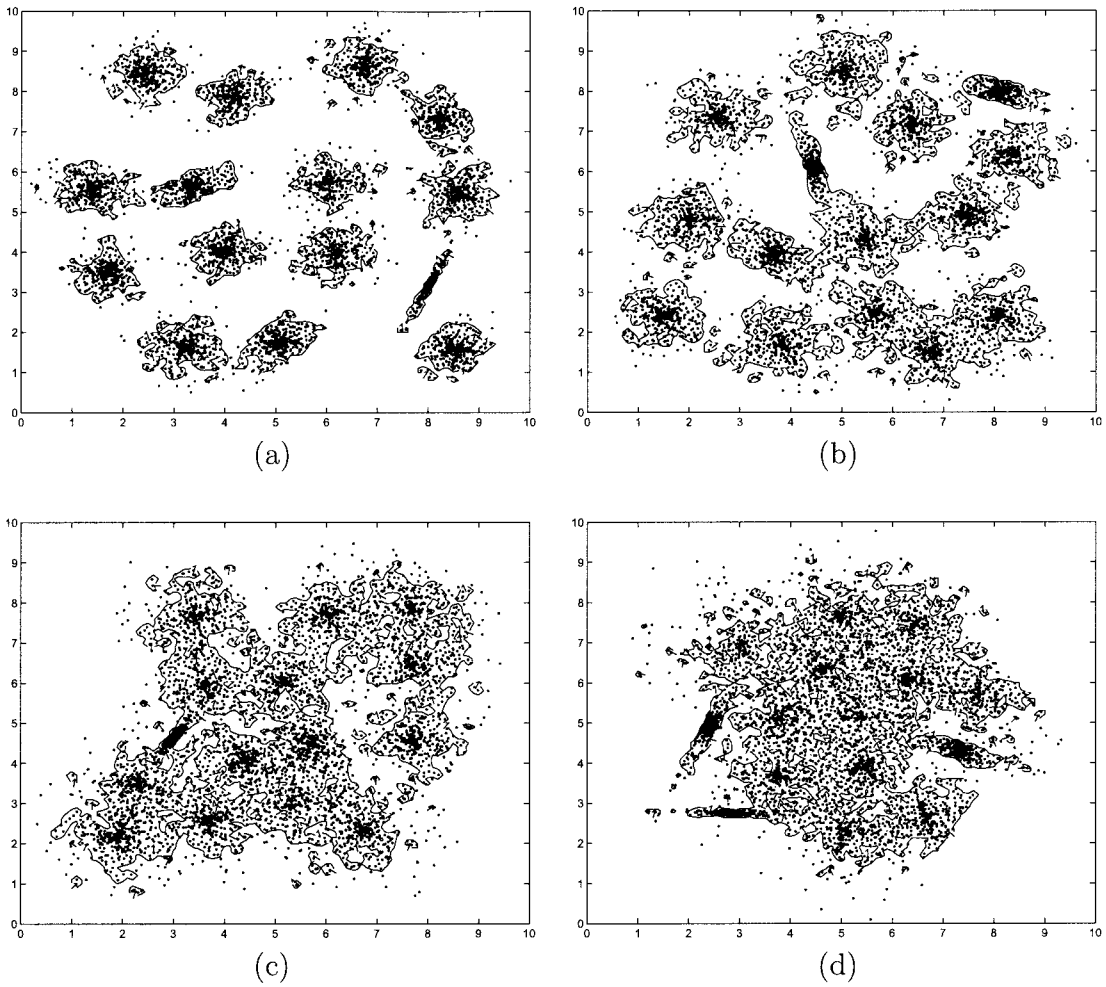


Figure 3.25: Clustering results of ADACLUS for data sets $S1 - S4$, from (a) to (d) correspondingly

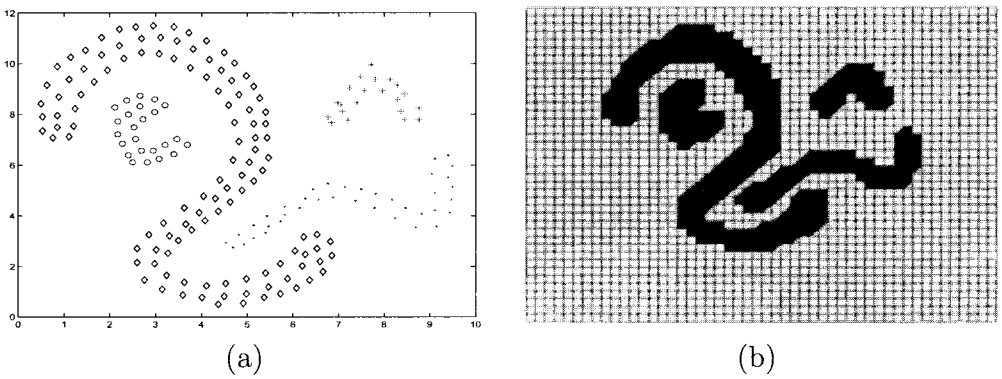


Figure 3.26: Comparison of ADACLUS and DENCLUE on $DD1$. (a) Clustering result of ADACLUS by automatic parameter setting(b) Clustering result of DENCLUE [2].

Table 3.5: Clustering accuracy in percentage

	KMEANS	Single-linkage	Complete-linkage	DBSCAN	ADACLUS
D1	99.00%	49.50%	99.00%	49.50%	100%
D2	80.00%	54.17%	82.50%	100%	100%
D3	51.32%	77.63%	96.05%	80.26%	100%
D4	55.12%	77.17%	54.33%	99.21%	100%
D5	71.73%	53.93%	72.77%	54.97%	100%
D6	78.89%	64.01%	59.86%	97.58%	100%
DD1	58.79%	100%	59.82%	100%	100%

Table 3.6: Statistic Features of Data sets $S1 - S4$

Data-set name	Number of data points	Mean m_{MD}	Standard deviation std_{MD}	$\frac{std_{MD}}{m_{MD}}$
S1	5000	0.0459	0.0497	1.0818
S2	5000	0.0547	0.0530	0.9677
S3	5000	0.0591	0.0550	0.9311
S4	5000	0.0534	0.0557	1.0426

to achieve better results than comparison methods. For data sets with concave clusters and outliers, KMEANS and Complete-linkage method are not able to perform well. DBSCAN and Single-linkage method are better for such cases. But when the density of each cluster changes gradually, DBSCAN and Single-linkage method still lack the ability to achieve ideal results.

We also tested ADACLUS with some large data sets $S1 - S4$ proposed in work [101]. We showed the results of clustering on the data sets with clusters boundaries. Fig.3.25 shows the results of ADACLUS for testing data sets $S1 - S4$. The details of the data sets are listed in Table.3.6.

The $S1 - S4$ data sets are two-dimensional data sets with 15 predefined clusters.

Table 3.7: Statistic Features of large data sets $LD1 - LD2$

Data-set name	Number of data points	Mean m_{MD}	Standard deviation std_{MD}	$\frac{std_{MD}}{m_{MD}}$
LD1	4760	1.5252	0.7191	0.4714
LD2	8170	1.8511	1.2844	0.6939

The distances between the clusters keep decreasing from data set $S1$ to $S4$. Fig.3.25 shows the results of ADACLUS. For data set $S1$, ADACLUS separated 15 clusters very well. Since the information of local distribution of each cluster is used, outliers are detected by ADACLUS as well. Parts of the data set inside boundaries are formed by data points with continuously changing density. The areas where the field density changes suddenly are cut by ADACLUS. From data set $S1$ to $S4$, the number of clusters keeps decreasing because of the decreasing distances between clusters. For data set $S4$, we have only one big cluster for the whole data set. But based on the local information, there are still many small isolated parts surrounding the big cluster which can be regarded as outliers.

To test the capacity of ADACLUS in handling large data set with irregular shape and non-uniform density distribution. We applied ADACLUS on two large data sets $LD1$ and $LD2$. The statistic information of those data sets are list in Table.3.7. From the result, we can draw the conclusion that ADACLUS is able to deal with quite large and complex data set, especially, when data density inside clusters are changing as long as local outliers existing.

We also applied ADACLUS to the well-known CHAMELEON bench-mark data sets [69]. Fig.3.29 and Fig.3.30 present clustering results for data set $C1$ and $C2$

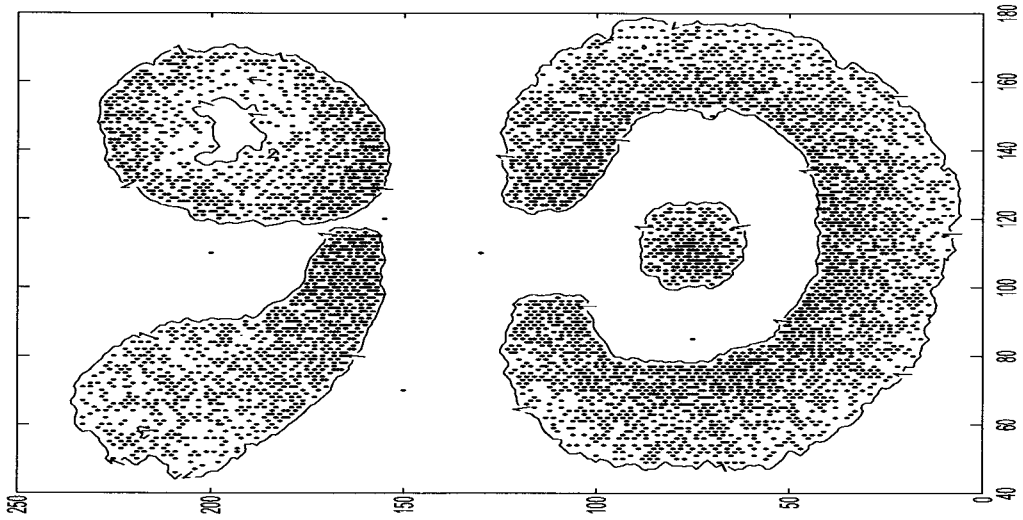


Figure 3.27: Clustering result of large data set $LD1$ by ADACLUS

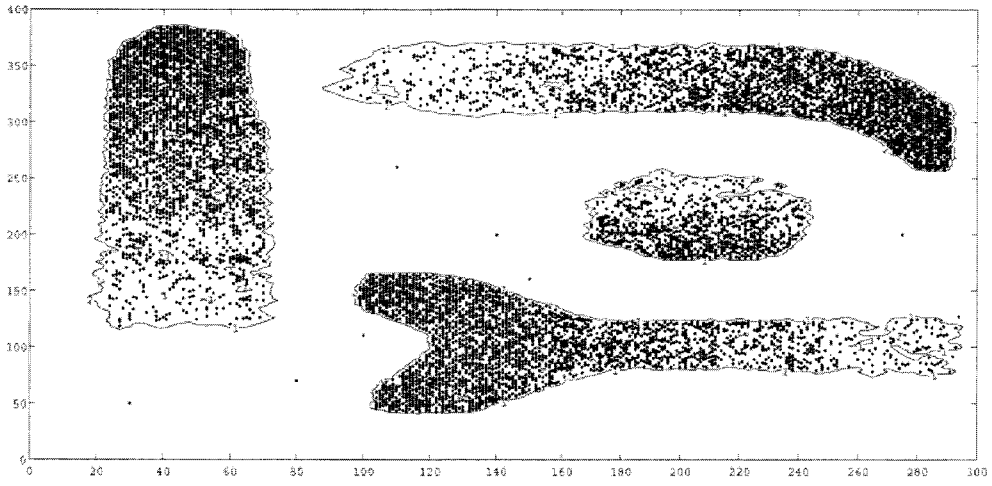


Figure 3.28: Clustering result of large data set $LD2$ by ADACLUS

by showing clusters' boundaries. In Fig.3.29, several clusters are separated into parts comparing with clustering results given in CHAMELEON paper [69]. The reason is the dramatic changes of cluster density. In Fig.3.30, several clusters are glued together. This is because of the existing of links formed by noises. The characteristics and limitations of our connection based method are demonstrated.

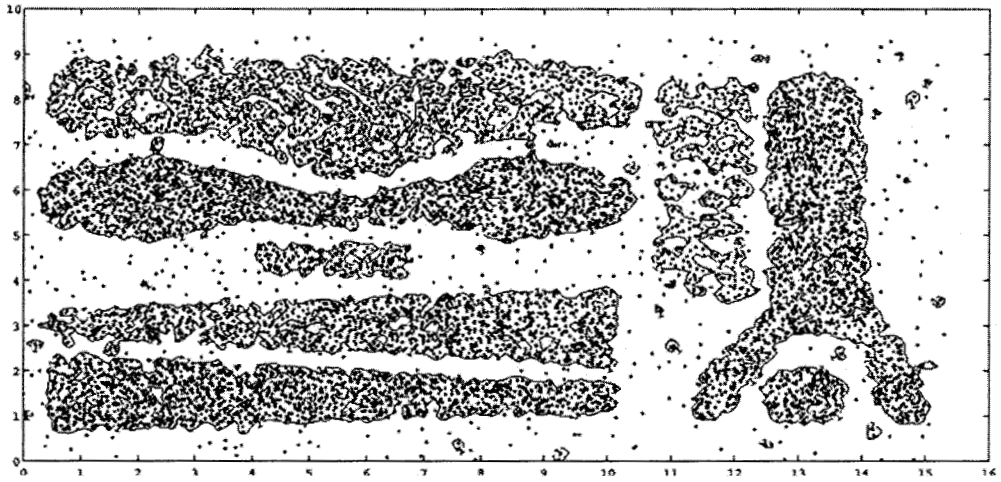


Figure 3.29: Result of ADACLUS for data set $C1$

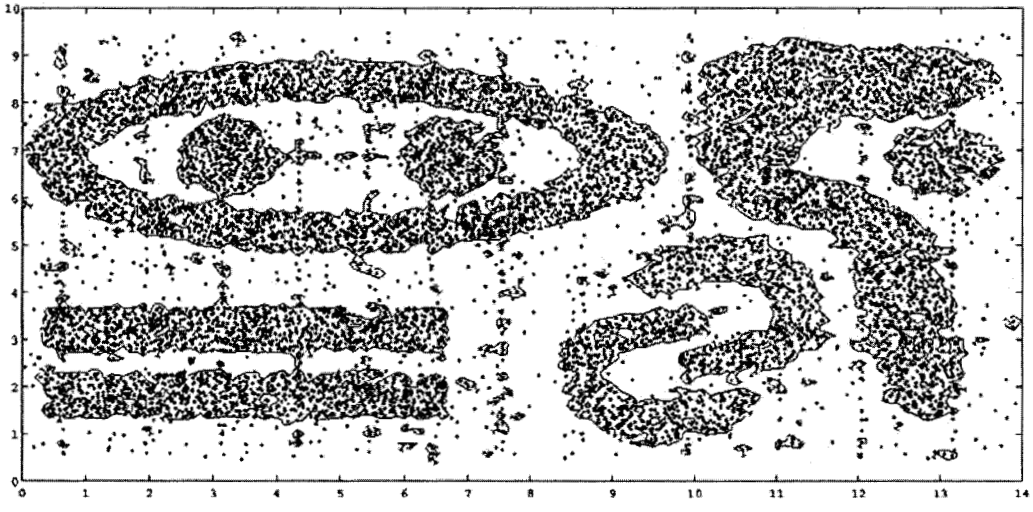


Figure 3.30: Result of ADACLUS for data set $C2$

3.4.6 Stability of ADACLUS to deviation of intrinsic parameters

Although there are three intrinsic parameters in ADACLUS: $Smin$, $Smax$ and c, the clustering results are quite robust against their variations.

Parameters $Smin$ and $Smax$ are designed to simulate a human vision clustering feature: the more data points are present in the data set the less amount of particular details in relatively small parts of data one can notice. In principle, these two parameters could be estimated by research in human vision. But the ADACLUS algorithm appears to be so robust to their deviation that we just set them to 200 and 2000 values. The robustness of ADACLUS to variation of these parameters is illustrated below by tests on data set $D2$ and large data set $LD1$, where $Smin$ varies from 50 to 500 and $Smax$ from 500 to 10000. In Table.3.8, we provide corresponding values of G and R. The clustering results are the same as the ones shown in previous section.

The last intrinsic parameter is parameter c in Step 4. It is just a "cutting" parameter which is used to keep the complexity of the algorithm linear against the number of data points. Value of c is chosen in such a way that the average number of data points in the local area which is examined during algorithm adaptation process, is restricted by 100 points. Variations of this parameter practically do not influence the clustering results.

Table 3.8: Variation of values of intrinsic parameters

		G_0		G_1		G		R	
$Smin$	$Smax$	$D2$	$LD1$	$D2$	$LD1$	$D2$	$LD1$	$D2$	$LD1$
50	500	0.6867	0	0	1	0.6867	1	0.7997	5.5100
	2000	0.9277	0	0	1	0.9277	1	0.8769	5.5100
	10000	0.9858	0.5266	0	1	0.9858	1	0.8955	5.5100
200	500	1	0	0	1	1	1	0.9000	5.5100
	2000	1	0	0	1	1	1	0.9000	5.5100
	10000	1	0.5347	0	1	1	1	0.9000	5.5100
500	500	1	0	0	1	1	1	0.9000	5.5100
	2000	1	0	0	1	1	1	0.9000	5.5100
	10000	1	0.5516	0	1	1	1	0.9000	5.5100

3.4.7 Real World Application of ADACLUS

To illustrate the practical value of ADACLUS, we applied it on the real world data set collected from European Topic Center on Air and Climate Change (ETC/ACC), which focuses on the EU-ECCP (European Climate Change Programme), CAFE (Clean Air for Europe) and related legislation, such as the Greenhouse Gas Monitoring Mechanism and the Air Quality Framework Directive. To achieve those objectives, ETC/ACC established the European air quality database system which contains next to multi-annual time series of measurement data and their statistics for a representative selection of stations throughout Europe. We chose the data set of the locations of stations. From our clustering result shown in Fig.3.31 and Fig.3.32, the coverage of the stations over Europe and the relationship between stations can be discovered especially by the building cluster boundaries. With ADACLUS, people who work in this area can conduct investigations based on distribution of station locations for particular values of different air quality features.

This result is generated automatically by ADACLUS. We can also set the parameters according to the knowledge of data set, such as the expected coverage of each station, to fit the specific application requirements. The statistic features of this data set is shown in Table.3.9. In Fig. 3.33, the clustering result of DBSCAN on the same data is shown for comparison.

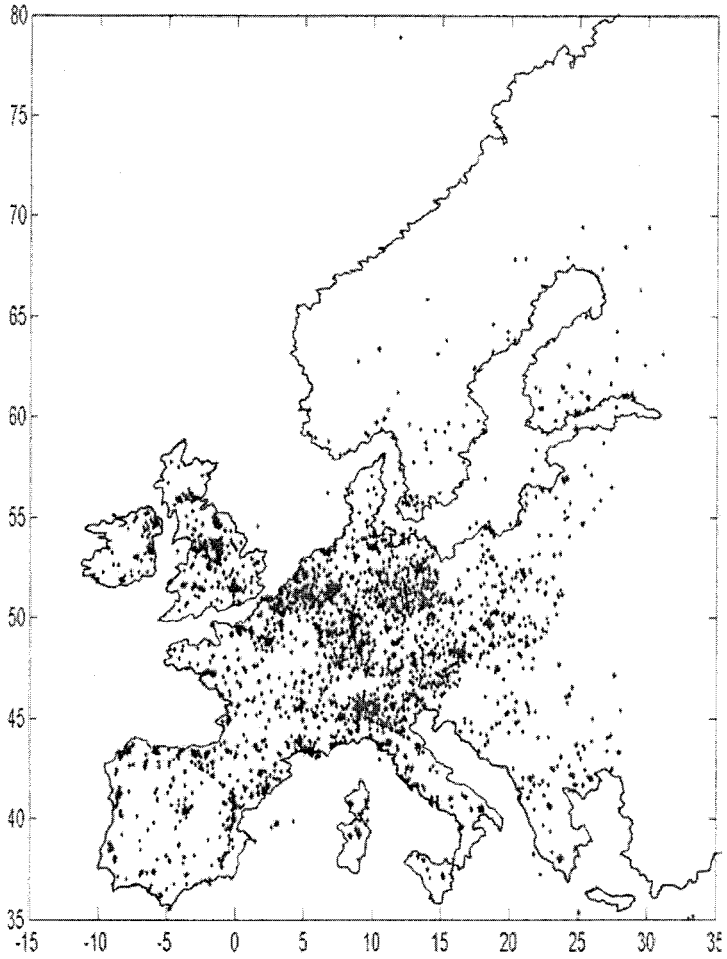


Figure 3.31: The locations of stations throughout Europe

ADACLUS allows to automatically discover clusters of arbitrary shape, different in density and clusters of non-uniform density, to detect boundary of the

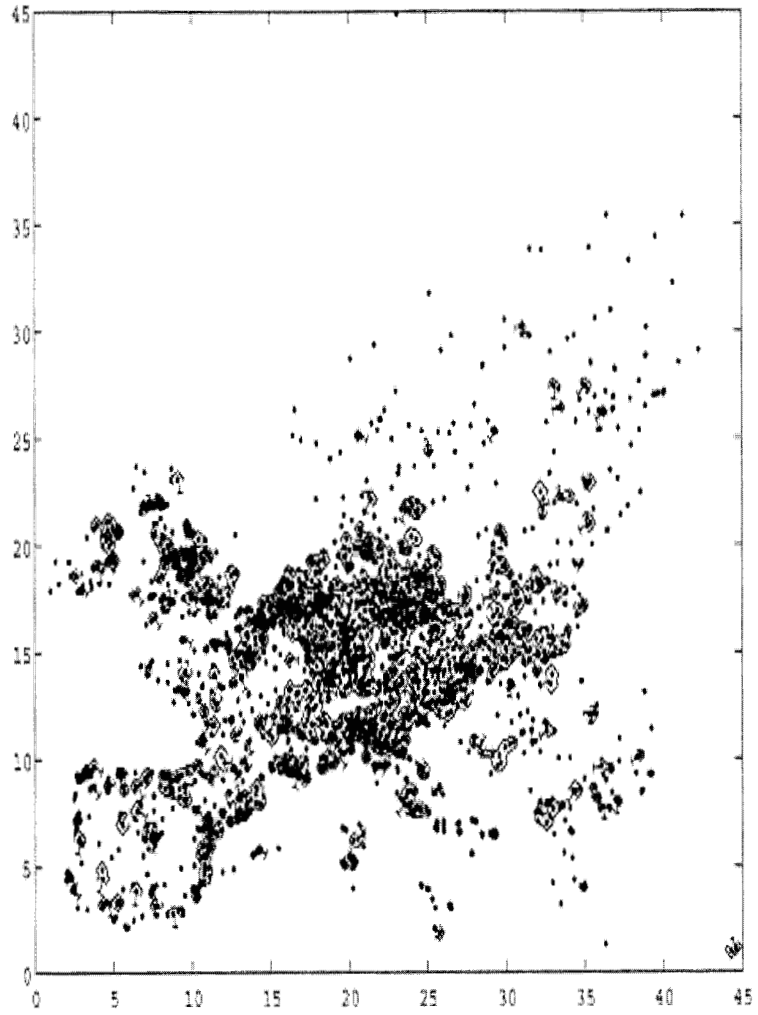


Figure 3.32: Clustering result of ADACLUS

Table 3.9: Statistic Features of our real world data set

Data-set name	Number of data points	Mean m_{MD}	Standard deviation std_{MD}	$\frac{std_{MD}}{m_{MD}}$
Real world data	5456	0.1061	0.2959	2.7903

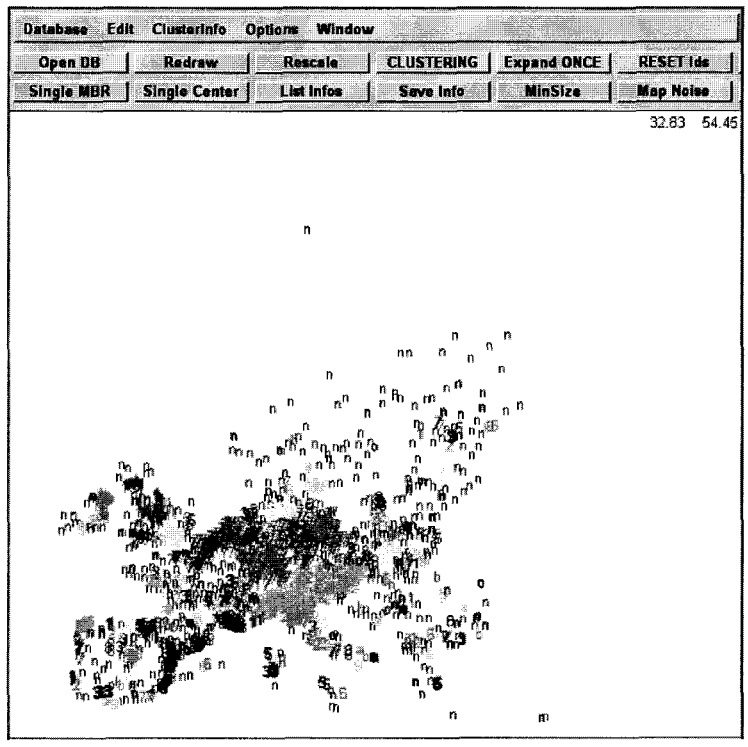


Figure 3.33: Clustering result of DBSCAN on our real-world data, $EPS = 0.4028$

clusters, and it is robust to outliers. It was demonstrated that the algorithm has linear performance that is very important for real-time applications, such as large scale geo-spatial data sets. Although the proposed algorithm is generally automatic, the user is given the possibility to tune the algorithm to his/her application by choosing values of three parameters that have clear meaning.

Chapter 4

Algorithm based on Triangulation

For real applications, in addition to the variety of data distributions, there are always noises introduced by processing systems to data sets at the stages of data collection or data transformation. Distribution of noise also significantly influences the result of clustering. If the noises are sets of isolated points, the effect of them may be relatively trivial. Many methods such as DBSCAN and Single-linkage method, can deal well with this type of noises. But if the noise forms a chain which connects two clusters that should be separated from each other, by using the above methods, the result will be totally unexpected. This kind of problems is the so-called "short bridges" problem [86] [29] or "chaining effect" problem [35] [17], or if there are several chains formed by noises, it is called "multiple bridges" [29] problem. The well-known solution is cut-point finding which only works for single bridge cases.

Thus, to meet the demands coming from both real world applications and theoretical researches, it is necessary to develop a new algorithm which can cut off the linkage brought by noise to solve those problems. Methods, such as Single-linkage method, DBSCAN and ADACLUS, which mainly focus on connectivity investiga-

tion always have difficulties to deal with these, since there are false connections between clusters which are not easy to identify. In principle, those methods are designed to deal with data sets without noises or with isolated noises. Therefore, we should develop clustering method based on statistical features of data rather than mainly on connectivity investigation.

In this chapter, we treat clustering task by analyzing statistical features of data and design a novel automatic clustering algorithm TRICLUST using triangulation. The proposed method TRICLUST is able to automatically detect not only clusters with non-uniform inner density, the case which has been introduced in Chapter 3, but also with short bridges formed by noises. By developing this algorithm, we can effectively deal with a wider range of data sets which have different distributions of data density and noises. TRICLUST also can build clusters boundaries by identifying the data points on them.

4.1 Introduction

Before introducing TRICLUST, there are following several issues which are very important for our algorithm to be elaborated.

- **Proximity Modeling**

  The spatial data is always described by traditional vector model or raster model. Despite of the fact that data point is the prime concern, the proximity which can reveal the topology of data is also very important. The proximity of one data point characterizes the adjacent relation to other data

points, so that determines the neighbors of it. For many spatial data analysis approaches, the proper proximity modeling can lead to effective algorithms. But the conventional vector model and raster model are not able to define data proximity that can hold a spatial adjacency properly for discrete data points [43]. The topology in vector model is always defined by intersections. For discrete data points, there is no intersection. Thus, all of them are not considered as each other's neighbors. To overcome this problem, the relationship between data points is usually described by distance. But, how to decide on the threshold which serves as the closeness criteria becomes another problem which results in the distance-based proximity definition inconsistency. Obviously, vector approach does not model data points with a good proximity relationship. On the other hand, the raster approach implies adjacency relationship between data points by its grid-like (cell) structure. The data space is separated into cells, and data points belonging to adjacent cells are neighbors. However, this proximity definition so heavily depends on the size of cell (grid) that makes it lack enough consistency as well so that derived algorithm may not be very robust to parameter setting.

An alternative way to define proximity of data point is by using Delaunay Triangulation. The Delaunay Triangulation or Delaunay Diagram [43–45] is a method used to build topology of data set. It is widely used in computer graphics and solid modeling. Delaunay Triangulation or Delaunay Diagram represents proximity relationships of data sets by building the connected

graph. It has the important properties [88] [87] that the nearest data points to a given spatial data point are always connected by edges, which provide us a good description of proximity. And the circumcircle (circumsphere for higher than 2-dimension cases) of every triangle does not contain any other data points of the triangulation. In addition, the triangulation is as equilateral as possible, so that unexpected effect of extraordinary long edges can be minimized. It overcomes the limitations of conventional data models by building the graph which represents data topology explicitly. Moreover, triangulation can be processed in only $O(NlogN)$ time. The number of edges is linear to the number of data points and the expected number of edges incident to one data point is limited to a constant value. Based on the achieved succinct graph, we can conveniently extract information such as statistical features from the graph. This kind of information can effectively characterize the relationship between data points so that it can lead to good clustering method.

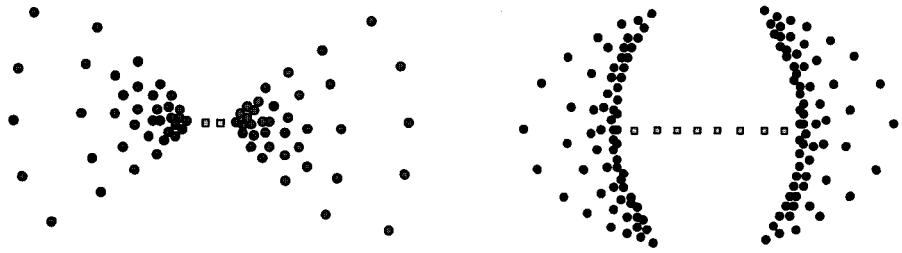
- **Classification of Short-bridges**

  The "short-bridge" problem, or "chaining effect" problem has been mentioned in several works [17] [29]. But it is still necessary to put further investigation on this problem. The noises are not part of the original data but the extra data which are generated during data collection or transformation by systems. In general, the distribution of noises can be regarded as Gaussian. However, the effect of added noises is much different with regard

to the location of their appearance. Let us consider the 2-dimensional situations shown in Fig.4.1, there are two clusters which are built up by dots for each situation given in the corresponding picture Fig.4.1(a) and Fig.4.1(b). There are also noises which are represented as yellow cubes in those data sets. In Fig.4.1(a), the short bridge connected the closest parts of two clusters is very short and there are quite a few data points in those parts. The distances between other data points belonging to different clusters are much bigger than the length of the short bridge. It means, in such case, it is very difficult to tell whether this linkage built by noises is valid, since from both global and local view, the separation of these two clusters becomes a fuzzy problem. This situation is not what we are going to focus on. In Fig.4.1(b), the short bridge formed by noises is not located in the most narrow channel between clusters. And there are many data points belonging to different clusters with the comparable distance with the length of the short bridge. Thus, from the global view, it is easy to identify that linkage as an invalid connection which should be cut off to isolate the two clusters. This kind of short bridges is the one we are going to study in our work.

4.2 Relationship to Previous Works using Triangulation

There are a few methods developed based on Delaunay Triangulation, such as the method proposed by Eldershaw and Hegland [87], AMOEBA [28], AUTOCLUST [29, 146] and the method proposed by Hader and Hampercht [89]. The approach



(a) (b)

Figure 4.1: Two different types of short bridges

of Hader and Hampercht applies Delaunay Triangulation as a path building tool. After the graph is generated, along the paths found, the value of the density function is checked to find the local maxima in order to start the density-based clustering process. The approach proposed by Eldershaw and Hegland as well as AMOEBA and AUTOCLUST mainly focus on classifying the edges of the graph built by Delaunay Triangulation or Delaunay Diagram into several groups. Then, by removing the inter-cluster edges, clusters are isolated. Among these previous approaches, AUTOCLUST is the latest and the most effective one. It applies Delaunay Diagram on the data set, then by analyzing the statistic information extracted from the graph built by Delaunay Diagram, unusually long and short edges are removed. After the edges recovery and bridges looking phases, the data set is separated into clusters and outliers[1]. This method has an advantage that the values of the parameters can be found from the statistical information of the edges of the graph. It makes AUTOCLUST an automatic clustering method. Comparing to its prior method AMOBEA, AUTOCLUST has the ability to deal with more

[1]In this chapter, when we analyze clustering results, we call a point which does not belong to any clusters an outlier no matter it is a data point or a noise point.

complicated data sets, such as in which clusters are connected by multiple bridges and/or nearby clusters are with different density, but the density inside these clusters should still be relatively uniform.

The proposed algorithm TRICLUST is also based on Delaunay Triangulation. But our method has the following new features:

(i) **Data set with non-uniform density distribution**

First, we extended our concern on data distribution to non-uniform inner cluster density cases, which have been introduced in Chapter 3. Based on the needs risen recently in both research and application areas, we developed a new algorithm which can be applied on wider range of data sets. Thus, the variation of distribution of data density between and inside clusters is one of our main concerns.

Second, if the density of data set varies not only between clusters but also inside clusters, the classification of edges of the graph will be more difficult, since the long edges could connect data points of the thin part of the same cluster, and short edges could connect local outliers with nearby clusters. The discrimination of edges in the way how it was done in AMOBEA and AUTOCLUST, would be much more difficult in such situation, moreover, for some complex data distribution, it could just fail. Therefore, we use data point investigation instead of edge investigation in our algorithm. The data points are classified into different categories according to the statistic information which is extracted from their neighborhood which are built by

triangulation. An advantage of studying on data points rather than edges is that we can minimize the effect of density variation and achieve a more clear description of data distribution. In addition, the time complexity will decrease during the checking and searching procedures, since, in general, there are more edges than data points in a graph built by triangulation . The details of data point investigation will be given in Section 4.4. Our algorithm will cluster one data set according to the classification of data points. The corresponding process of clusters building is done by including more data points rather than removing edges. Details of our algorithm implementation will be given in Section4.5.

(ii) **New statistical features**

In order to characterize data points based on data distribution, we designed a few statistic features. Different from the ones used in previous methods, we consider data point not edges together with the effect of density changing both inter clusters and inner clusters. The target which we study on and extract statistic information from is the length of all edges inside the neighborhood of each data point. The neighborhood for each data point is the sub-graph of Delaunay Triangulation. The statistic features we applied here are mean of the length, standard deviation of the length divided by mean and the positive part of the derivative of mean. The details of definitions of these statistic features are given in Section 4.3.

4.3 Definitions

Here, we regard each data point as a point P in the n-dimensional data space $\mathbf{R^n}$. The data set D is a set of n-dimensional points. $D = \{P_1, \ldots, P_N\}$. N is the number of data points.

Definition 4.10 *Neighborhood:*

The neighborhood Ne of one data point P is the sub-graph of the Delaunay Triangulation of the data set. This neighborhood is constructed by all data points which are directly connected to P based on Delaunay Triangulation and all edges between those data points. We call two data points directly connected if they are linked by the same edge of Delaunay Triangulation.

Definition 4.11 *The length of an edge:*

The length of an edge in the Delaunay Triangulation graph is denoted as L. Any metric could be used to calculate the length. Here, we use the Euclidean distance (which is defined in 3.1) between the two data points P_i and P_j of this edge as its L which is defined in 4.1.

$$L_i = dist(P_i, P_j) \tag{4.1}$$

Definition 4.12 *Mean of edge length for one data point:*

The Mean of edge length for one data point P is the mean of the length of all edges inside its neighborhood Ne. $Mean(P)$ is defined in 4.2.

$$Mean(P) = \sum_{L_i \in Ne} L_i / M_e = \sum_{i=1}^{M_e} L_i / M_e \tag{4.2}$$

where M_e denotes the number of edges in Ne.

Definition 4.13 *Standard deviation (STD) of edge length for one data point:*

The standard deviation of edge length for one data point P is the standard deviation of the length of all edges inside P's neighborhood Ne. It is defined in 4.3.

$$STD(P) = \sqrt{\sum_{i=1}^{M_e} (Mean(P) - L_i)^2/(M_e - 1)} \tag{4.3}$$

where $Mean(P)$ is the mean of data point P.

Definition 4.14 *The quotient of standard deviation (STD) divided by Mean of one data point:*

The quotient of STD divided by Mean of one data point P, which we denoted as DM, is defined in 4.4.

$$DM(P) = STD(P)/Mean(P) \tag{4.4}$$

Definition 4.15 *The PDM value of one data point:*

The PDM value of one data point P_i is the mean of positive parts of the derivative of the Mean along all edges connected P_i to its neighboring data points in its neighborhood Ne_i. The PDM is defined in 4.5.

$$PDM(P_i) = \sum_{j=1}^{M_p} PD(P_i, P_j)/M_p \tag{4.5}$$

where $PD(P_i, P_j)$ is the positive part of the derivative of Mean along the edge connected P_i and P_j. PD is defined in 4.6; M_p is the number of data points in Ne_i with smaller Mean value than $Mean(P_i)$.

$$PD(P_i, P_j) = \begin{cases} 0, & if \quad Mean(P_i) \leq Mean(P_j) \\ \frac{Mean(P_i) - Mean(P_j)}{L_{ij}}, & if \quad Mean(P_i) > Mean(P_j) \end{cases} \tag{4.6}$$

where L_{ij} is the length of the edge connected P_i and P_j.

4.4 Basic Ideas

The basic idea of TRICLUST, which is a clustering algorithm based on Delaunay Triangulation, is to cluster data set by classifying all data points into two categories. These two categories are inner cluster data points and boundary data points. By using proposed statistical features extracted from the neighborhood of data points, we build the criteria function (the exact formula of criteria function is introduced below) according to both global view and local view for different situations. The value of criteria function for each data point is calculated. We use K-MEANS method as threshold detecting method to choose a threshold of our criteria function values. The classification of data points is executed by applying this threshold. Thus, the n-dimensional clustering problem can be transformed to one dimensional classification problem. We will give the description of our statistical features in more detail in this section with 2-dimensional examples. The description of TRICLUST algorithm, analysis of testing results and performance comparisons with other clustering methods will be given in Section 4.5 to Section 4.7.

Let us elaborate on basic ideas of TRICLUST.

(i) **New statistical features calculated based on triangulation**

The statistic features we employed in TRICLUST to classify one data point are the $Mean$ of it, DM which is the quotient of standard deviation of it divided by its $Mean$ and PDM which is the mean of the positive parts of the derivative of $Mean$.

To illustrate the meaning of the value of each statistic feature, we consider the different locations of one data point inside its data set. In Fig.4.2, we show all possible locations for one data point in a 2-dimensional data set. The data point drawn with circle represents the situation when the data point is located in the dense part of the cluster. The data point drawn with triangle represents the situation when the data point are located in the thin part of the cluster. The data point drawn with diamond represents the situation when the data point are located on the cluster boundary. The data point drawn with square represents the situation when the data point is an outlier.

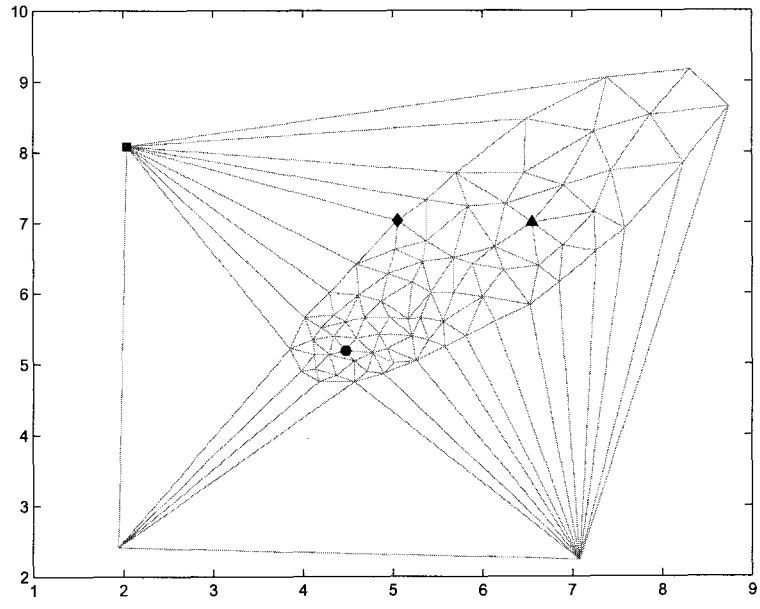


Figure 4.2: Illustration of all possible locations of one data point in the data set

To give a clearer view of the meaning of statistic feature values in situations shown in Fig.4.2, in Fig. 4.3, we listed all situations with the corresponding neighborhood graph of the data point. From the sub-picture (A) to (H), the

neighborhood graphs of the center red points which are the data points we are investigating in are shown. In each situation, the red center point corresponds to the data point drawn with different symbols shown in Fig.4.2. The neighborhood graphs are built by Delaunay Triangulation. The situations of (A) and (B) represent the situations when the center data points are located in the dense part of a cluster. The situations of (C) and (D) represent the cases when the data points are in the thin part of a cluster. The situations of (E) and (F) represent the cases when the center data points are on the boundary of one cluster. The centered data points correspond to the diamond point in Fig. 4.2. And (G) and (H) represent the cases when they are outliers. The centered data points correspond to the square point in Fig. 4.2. In these situations, the length of all edges inside the data point's neighborhood are much bigger than other cases. We provided two sub-pictures for each situation to illustrate that, as the center data point is in the same situation, the different positions of the neighboring data points will not affect our study on the statistic features extracted from its neighborhood.

From the four types of situations shown in Fig.4.3, we can notice the differences between those neighborhood graphs so that we are able to study the variation of values of the statistic features defined in Section 4.3. For situations in sub-picture (A) and (B), since the center data points are in the dense part of a cluster, the $Mean$ of them and the standard deviation STD of them are both with small values, also the value of their DM and the value

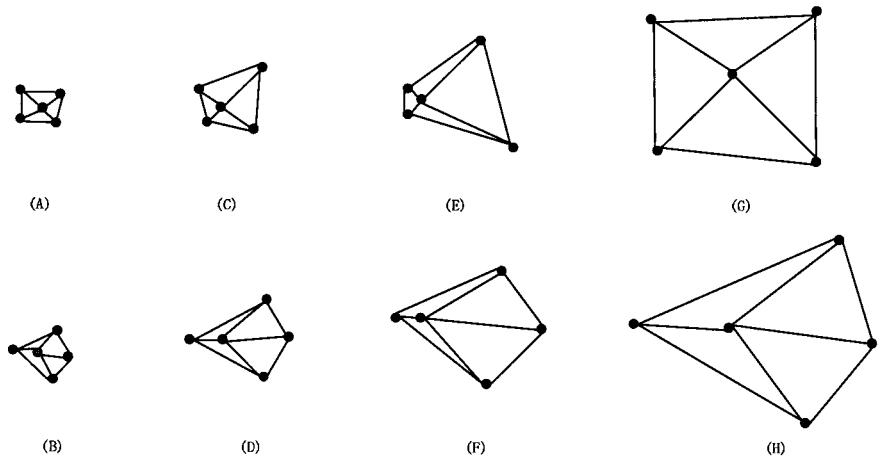


Figure 4.3: Different neighborhood graphs of the center data point regarding its different locations in the data set

of their PDM. For the situations in sub-picture (C) and (D), since the center data points are located in the thin part of one cluster, the values of $Mean$ and STD of them become bigger than the ones in situation (A) and (B). But the values of DM will not increase with the same extent, because the scale of the increase of STD and $Mean$ is the same. The values of PDM of the center points in these situations will also increase. For the situations shown in (E) and (F), the center data points are on the boundary of one cluster, thus, values of the $Mean$ and the STD of them will both increase. But they will not increase to the same extent, there will be more increase of the values of STD than of $Mean$. Therefore, the values of DM will increase greatly because of the different scales between increase of DM and STD. The values of PDM of the center data points will also increase dramatically since the sudden change of the derivative of $Mean$ occurs to these data points.

For situations (G) and (H), since the center data points represent outliers, the values of $Mean$ of them will be much bigger than the values in previous situations. In general, the values of STD and DM will also have big values. The values of PDM in these situations will be also big. In general, they will be quite similar to the values in situation (E) and (F), since the values of the $Mean$ of both the center data points and their neighbors which are data points on cluster boundaries are big.

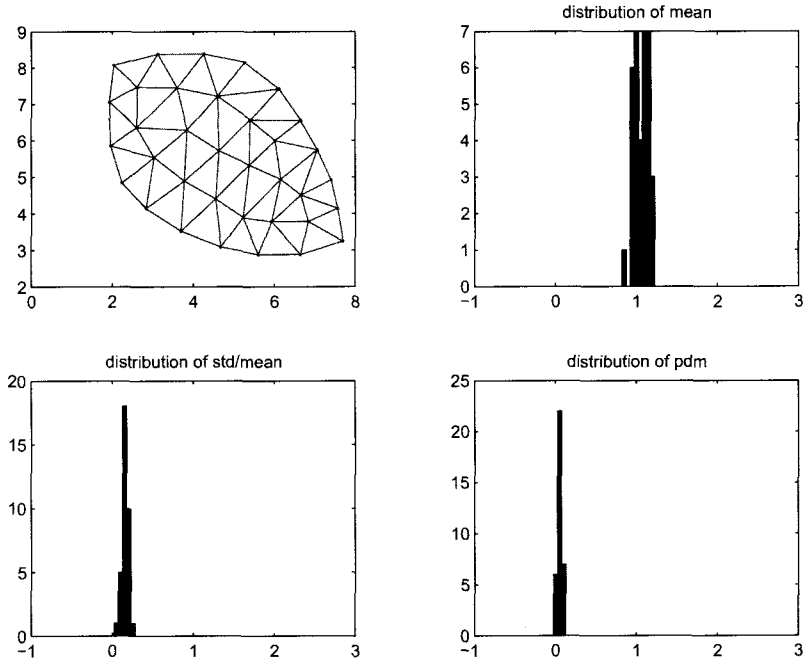


Figure 4.4: Data simulation of inner cluster data points in uniform small cluster

We provided more intuitive explanations of the meanings of values of our statistic features by data simulations. In Fig.4.4 to Fig.4.6, we simulated the situations when data points are inner cluster data points of clusters with different cluster densities. In Fig.4.4, the density of cluster is relatively uniform and the number of data points is small. In Fig.4.5, the density of

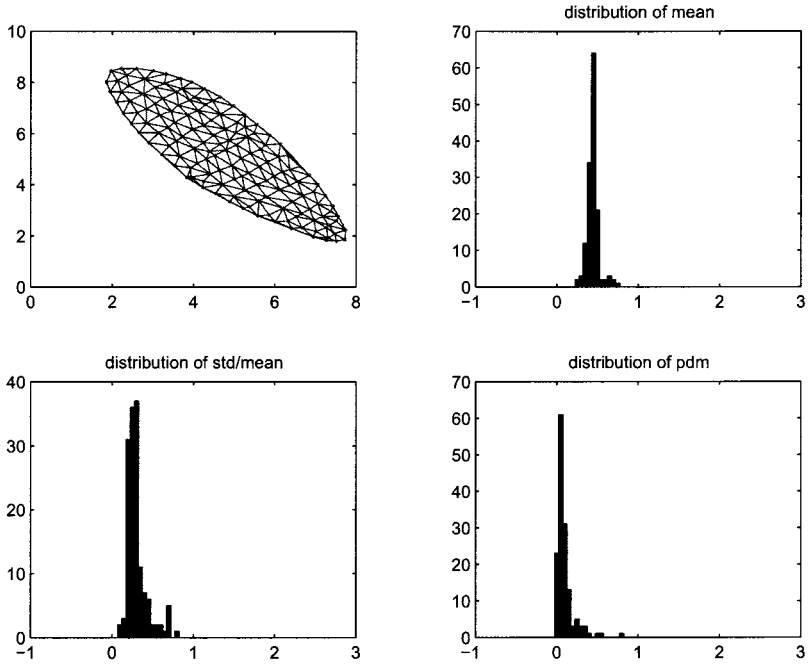


Figure 4.5: Data simulation of inner cluster data points in uniform large cluster

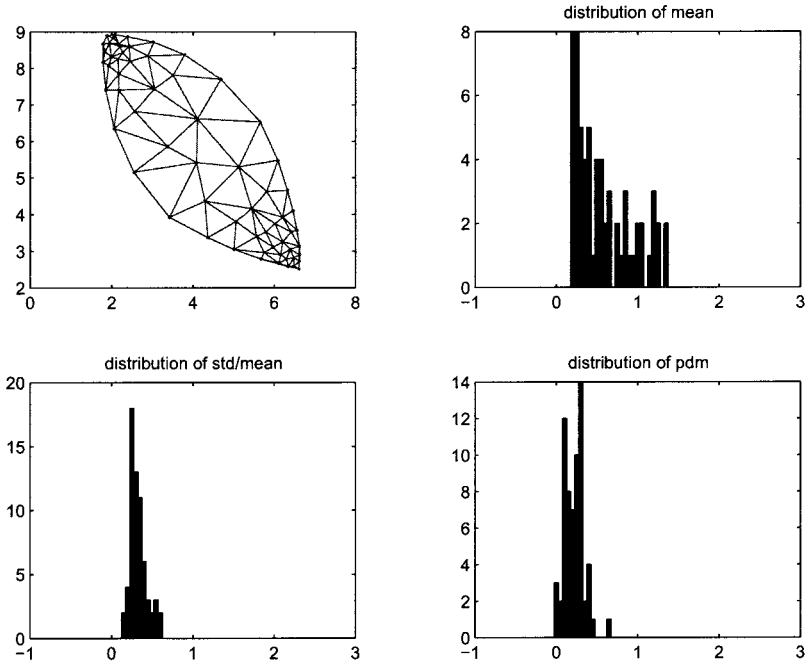


Figure 4.6: Data simulation of inner cluster data points in non-uniform small cluster

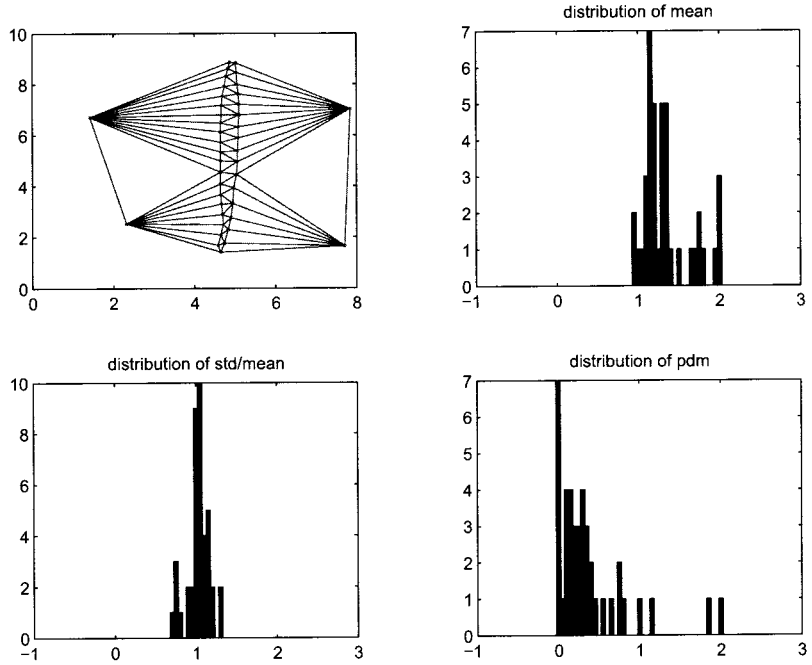


Figure 4.7: Data simulation of boundary data points

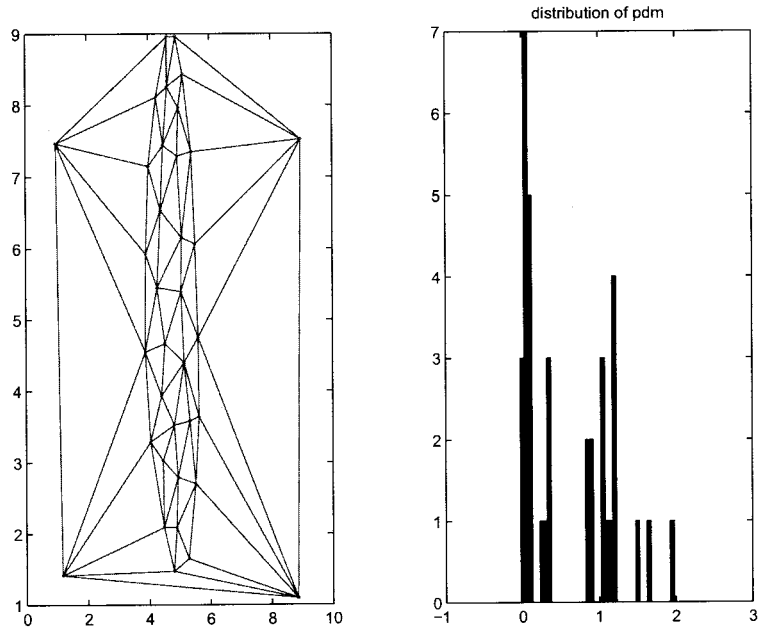


Figure 4.8: Data simulation of boundary data points for PDM

cluster is still uniform but the number of data points increases. We can see the distributions of the values of DM and PDM in these two cases are quite similar. The distributions of $Mean$ are different due to the different cluster densities. In Fig.4.6, the cluster density is not uniform any more, there are dense and thin parts of the same cluster. Comparing with the distributions in Fig.4.4 and Fig.4.5, we can discover that the distributions of the values of DM are still quite similar to one another. It means the value of DM will not change according to the gradual variation of cluster density. This property makes the DM a suitable statistic feature for clustering clusters with non-uniform densities. In Fig.4.6, the distribution of PDM is also similar to previous cases that all of them are with relatively small values. In Fig.4.7, we simulated the distribution situation for boundary data points. It is easy to notice that the distribution of DM is much different from the ones in Fig.4.4 to Fig.4.6. The boundary data points have much bigger values of DM than inner cluster data points, since the variations of STD and $Mean$ are not under the same scale. The $Mean$ value in this case is also the biggest one among these four situations. To illustrate the distribution changes of PDM in boundary data point case better, we used the data simulation shown in Fig.4.8, in which there are both inner cluster data points and boundary data points, and also outliers. From this figure, we can notice the sudden change of the value of the derivative of $Mean$ in the PDM distribution. The PDM values of boundary data points are much bigger than the values of inner cluster data points. It means that PDM is also able to distinguish

boundary data points from inner cluster data points.

Based on the observations on the distributions of our three statistic features in all situations given above, we can discover that by choosing proper thresholds of the distributions of $Mean$, DM and PDM, it is possible to classify all data points into two categories which are inner cluster data points and boundary data points. Our novel clustering algorithm TRICLUST is designed based upon this idea so that the n-dimensional clustering problem can be transferred to one dimensional classification problem.

(ii) **Criteria Function**

The three statistic features we used in TRICLUST have different characteristics with respect to their capacities to reflect the data distribution. $Mean$ can distinguish inner cluster data point from boundary point in the case when the densities of clusters are uniform and similar to each other. But for non-uniform density clusters or clusters with different densities, such as when sparse clusters are near to dense clusters, it can not work well. On the other hand, DM is a good statistic feature for detecting non-uniform density clusters. It has the strong ability to identify boundary data points when both uniform and non-uniform density clusters are existing. But it is very sensitive to the irregular data distribution inside clusters. It means when there are several data points inside the same cluster with quite different distributions from others, such as when they are more close to each other than to other data points, these data points may be identified as boundary points

by their DM value. PDM is more robust to the irregular data distribution inside clusters than DM. But for detecting boundary data points, it is not as effective as DM, especially when "short-bridges" exist.

Another important issue which has been mentioned in Chapter 2 and Chapter 3 for modern clustering research is applying different criteria on different parts of data set according to both global and local information of it. It means that clustering data set under global view or local view only cannot achieve ideal result. We should combine the global and local information based on certain data distribution and clustering requirement.

To maximize the advantage of each statistic feature and consider both global and local information when we cluster our data set, we build the criteria function by combining our three statistic features with different weights. The values of these weights reflect the degree to which we want to cluster the data set under global view or local view. In general, the bigger is the size of the data set, the more important is the global information; the smaller is the size of the data set, the more important is the local information.

Since the $Mean$ of data point can serve as a good criterion under global view, we let it play a more significant role when the size of our data set is big. On the contrary, DM is a local and very sensitive criterion, when the size of our data set is small, it should be in charge. PDM is a good complement of DM with regard to dealing with irregular inner cluster data points. And it is better than $mean$ to detect boundary data points. So

the importance of PDM should increase when DM is not in charge. Thus, by combining the three statistic features into one criteria function, we can have good discrimination in classifying data points according to both global and local views so that our algorithm TRICLUST is able to automatically detect cluster with non-uniform density and is robust to noises, even when the "short bridges" are existing.

The criteria function f_c is defined as in 4.7, the value of it becomes the final feature for every data point. For data point P, $f_c(P)$ is:

$$f_c(P) = a * Mean(P) + b * DM(P) + c * PDM(P) \tag{4.7}$$

where a, b, c are the weights which are defined in 4.8.

$$a = \begin{cases} \frac{N}{2000}, & N < Smin \\ \frac{1.9N + Smax/10 - 2Smin}{Smax - Smin}, & Smin \leq N < Smax \\ 2, & N \geq Smax \end{cases}$$

$$b = \begin{cases} 1, & N < Smin \\ 1 - \frac{N - Smin}{2(Smax - Smin)}, & Smin \leq N < Smax \\ 0.5, & N \geq Smax \end{cases} \tag{4.8}$$

$$c = \begin{cases} 0.5, & N < Smin \\ \frac{0.5N + 0.5Smax - Smin}{Smax - Smin}, & Smin \leq N < Smax \\ 1, & N \geq Smax \end{cases}$$

where N is the number of data points inside the data set.

The values of parameter a, b and c reflect the level to which we want to cluster our data set under local or global view. If there is a small amount

of data points that one can notice any of the data points "at one glance", then the natural clustering should depend exclusively on the local details of the data points distribution. Moreover, the small number of data points is not enough to generate reliable global features of distribution which could be useful for clustering. We denote $Smin$ as the number of data points, starting from which human visual inspection starts missing individual points and therefore the manner of discrimination starts shifting from local to global one. Another threshold is $Smax$ the number of points, starting from which global features of the data set become fully important for human visual inspection. These two thresholds should be estimated by biological research of human vision for eye simulation cases or be selected based on requirements of certain applications. Here, they are very approximatively set to $Smin = 200$ and $Smax = 10000$, but the clustering results are quite robust to their variations, which will be shown in Section 4.8.

4.5 Algorithm Description

The main steps of TRICLUST are elaborated in this section.

Step 1. Apply the Delaunay Triangulation on our data set to get the triangulation graph of it. This triangulation graph indicates the neighborhood of each data point. After the triangulation graph is built, for each data point, its neighboring data points has been achieved and been stored as matrix. The length of all edges in the neighborhood of each data point is also calculated and stored as matrix.

Step 2. Calculate the three statistic features for each data point according to the definitions introduced in Section 4.3.

Step 3. Set the values of parameters a, b, c of the criteria function based on the method proposed in Section 4.4. For every data point P_i, we calculate the value of its final feature $f_c(P_i)$ which is the value of criteria function on the data point P_i. After that, we get the distribution of the final feature values for all data points by drawing the frequency histogram. We find the value R_c of the frequency histogram. R_c is defined in 4.9.

$$\begin{aligned} R_c &= \begin{cases} R_1, & \text{if} \quad N \leq 5000 \\ min\{R_2, R_1\}, & \text{if} \quad N > 5000 \end{cases} \\ R_1 &= \min\{x : g_{emp}^{f_c}(x) = 0\} \\ R_2 &= \min\{x : P_{emp}(f_c < x) = 0.97\} \end{aligned} \tag{4.9}$$

where $g_{emp}^{f_c}(x)$ denotes the empirical density (frequency histogram) of the f_c distribution. R_1 is the first zero. R_2 is the 97% of the distribution. The 3% cut is for algorithm robustness when dealing with large data set. We select the final feature values less than R_c for threshold deciding in next step. In Fig.4.9, we provided an illustration of the frequency histogram of final feature.

Step 4. By applying K-MEANS algorithm (the parameters setting for K-MEANS: number of clusters equal to 2; initial cluster centers are the minima and mean of the selected final feature values) on the final feature values achieved from Step 3, we can get the threshold th which is used to distinguish

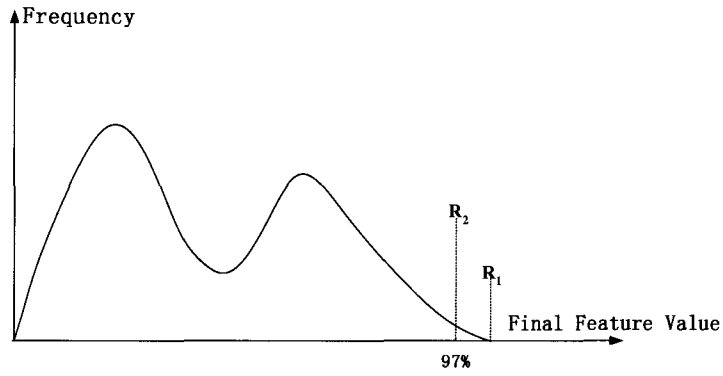


Figure 4.9: An illustration of frequency histogram of final feature

boundary data points from inner cluster data points. The data points with the final feature value bigger than th are labeled as boundary data points (this category also includes outliers). The rest data points are labeled as inner cluster points.

Step 5. We start the clustering process by selecting the first inner cluster data point P_i based on indexing. A new cluster C_j is generated by assigning this data point P_i to it as its first element.

Step 6. The cluster expansion progress is executed by continuously adding the inner cluster data points in the neighborhoods of the assigned data points. After the first data point P_i is assigned to the new cluster C_j, we initialize the cluster neighborhood Ned_j which is a set of data points. Initially, it includes the data points in the neighborhood of P_i. We find all the inner cluster data points inside Ned_j. We assign these data points to C_j and update the cluster neighborhood Ned_j by adding data points in the neighborhoods of all newly added data points. The achieved cluster neighborhood Ned_j does not include

all data points which have already been assigned to cluster C_j. After that, we add the inner cluster data points in Ned_j to cluster C_j and update Ned_j until there is no suitable data point to be assigned to cluster C_j. During the cluster expansion progress, we update the index of one data point when it is assigned to a cluster.

In Fig.4.10, for 2-dimensional data, how a cluster neighborhood is updated and how a cluster is expanding are demonstrated. The shown cluster neighborhoods in sub-picture (a) to sub-picture (d) are the initial cluster neighborhood, which includes the data points in the neighborhood of the initial data point, to the third updated cluster neighborhood. The red dots represent data points belonging to the cluster. The black asterisks represent updated cluster neighborhood for each iteration.

Step 7. Repeat Step 5 and Step 6 until no new cluster can be generated. Till now, the clustering process complete. All data points have been assigned to clusters or labeled as outliers.

Step 8. If we want to build complete cluster boundaries for boundary detection applications, besides the data points that have been already labeled as boundary points, the boundary data points which only connect to their own clusters are also labeled. In 2-dimensional case, the data points which are the ends of such edge that belongs to only one triangle are also labeled as boundary points.

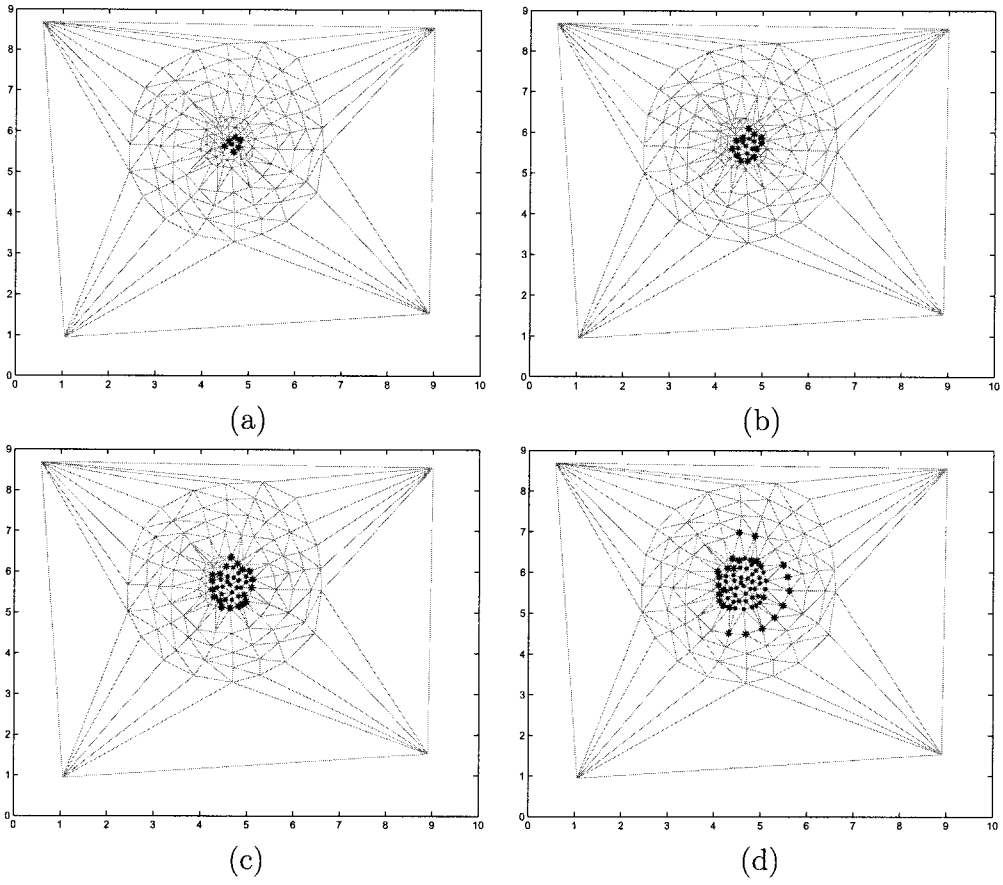


Figure 4.10: An illustration of cluster neighborhood updating and cluster expansion

4.6 Complexity Analysis

The time complexity analysis of TRICLUST is given in this section. Since in this research work we designed and implemented our algorithm for 2D data, the analysis here is for 2-dimensional cases. The theoretical time complexity analysis is given bellow by steps:

step1. The time complexity of constructing Delaunay Triangulation graph is $O(NlogN)$, where N is the number of data points. The time complexity of finding neighboring data points and calculating the length of edges in the neighborhood for all data points is linear to the number of triangles. Since for N data points, the number of edges is $3N-6$ and the number of triangles is $2N-4$ at most [88], the totally time complexity of this step is $O(NlogN)$.

step2. The time complexity of this step is linear to the number of data point N, since average number of neighboring data points to one data point is bounded. For 2-dimensional cases, if we denote the number of neighboring data points to data point P_i as $\|Ne_i\|$, $E(\|Ne_i\|) = 6$ [29].

step3. The time complexity of this step is constant. It only depends on the number of intervals which are the segments achieved by dividing the range from the minimal value to the maximal value of the criteria function by S, where $S = min(N, 100)$.

step4. The time complexity of this step is linear to N, since we set $k = 2$ and the number of iteration to 1 for K-MEANS method.

step5. The time complexity of step 5 is N in the worst case.

step 6. and step 7. The most time consuming procedure in this step is cluster

neighborhood updating. We have stored the neighborhood for each data point. We just combine the neighborhoods of data points inside the cluster to update the cluster neighborhood. The time complexity is $O(NlogN)$ originally, but, when we keep the information of checked data points, we can make almost each data point be examined only once. Because average number of neighboring data points to one data point is bounded, then the time complexity of this procedure can be linear to N.

step8. The time complexity of this step is linear to N.

To sum up, the total time complexity of TRICLUST is $O(NlogN)$.

The experimental testing of the time complexity of TRICLUST is shown in Fig.4.11. The X direction is the number of data points. The Y direction is the CPU time required by TRICLUST in second.

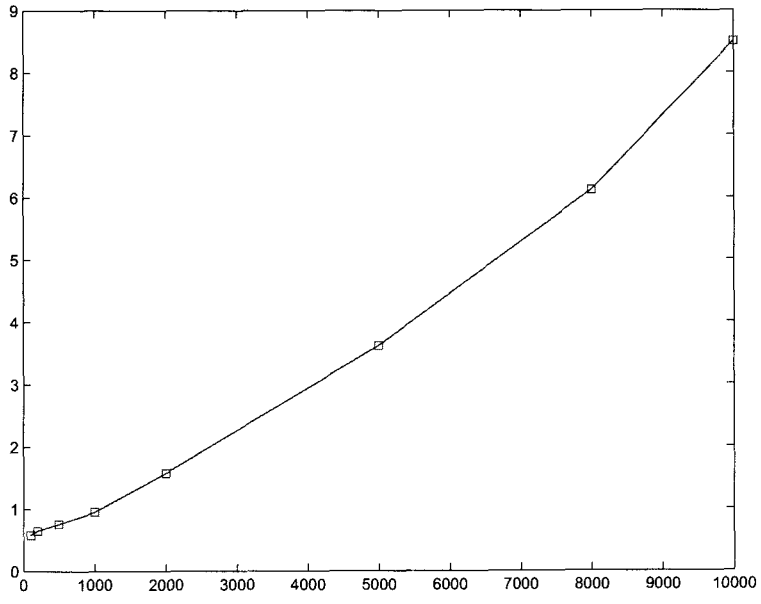


Figure 4.11: Time required by TRICLUST in seconds

4.7 Comparisons and Experiment Results Analysis

To show the capability of TRICLUST to handle data with complex distribution, we designed four 2-dimensional testing data sets $D1$ to $D4$. The new challenges for clustering algorithms from the distribution of data, such as non-uniform cluster density, complicated cluster shape and short bridges built by noises, are implemented in these testing data sets. In them, we define each cluster with gradually changing inner density and smooth boundaries. We implemented our algorithm in 2-dimension and applied it on these four data sets with automatic parameters setting. By showing clustering results, the cluster boundaries built by TRICLUST, and the distributions of final feature values, the performance and characteristics of TRICLUST are well demonstrated.

For comparison, we also applied classic clustering methods such as K-MEANS, hierarchical Single Linkage method, DBSCAN and AUTOCLUST on $D1$ - $D4$ data sets. We choose these classic clustering methods because they are representative in this area, and standard implementations of these methods are available. We can avoid the potential problems causing by different implementations, which could lead us to bias in results comparisons. K-MEANS method is still the most widely used method, and it is based on the optimization principles which form a foundation for many well-known data mining techniques. We employed the squared Euclidean distance and random selection of initial cluster centers for K-MEANS in our tests. Single Linkage hierarchical method has a good ability of handling

clusters with complicated shapes. DBSCAN is another well known density-based clustering method for dealing with arbitrary shape clusters and outliers[2]. The implementation of DBSCAN is provided by the author and all parameter settings are based on the instruction given in the original paper ($k = 3$ and $MinCard = 4$ for our 2-dimensional data sets). AUTOCLUST is one famous clustering algorithm based on Delaunay Triangulation. The implementation of AUTOCLUST is obtained from GEOTOOLS package (http://geotools.codehaus.org/).

In Fig.4.12, the Delaunay Triangulation graph of testing data set $D1$, the boundary data points detected by TRICLUST, distribution of the final feature values, and clustering result of $D1$ by TRICLUST are provided. There are two clusters inside this data set together with four outliers. The small round cluster is surrounded by the large ring shape cluster. The densities of these two clusters are quite different that the round cluster is denser than the ring shape cluster. Moreover, the density of ring shape cluster is non-uniform comparing with the density of another cluster. These two clusters are also very close to each other so that the smallest distance between data points belonging to different clusters smaller than the largest distance between data points of the ring shape cluster. From Fig.4.12, we can notice that TRICLUST separated the two clusters very well and also found the outliers.

In Fig.4.13, the clustering results of $D1$ by K-MEANS, Single Linkage method, DBSCAN and AUTOCLUST are shown. None of them can get the ideal result. K-

[2]In this chapter, when we analyze clustering results, we call a point which does not belong to any clusters an outlier no matter it is a data point or a noise point

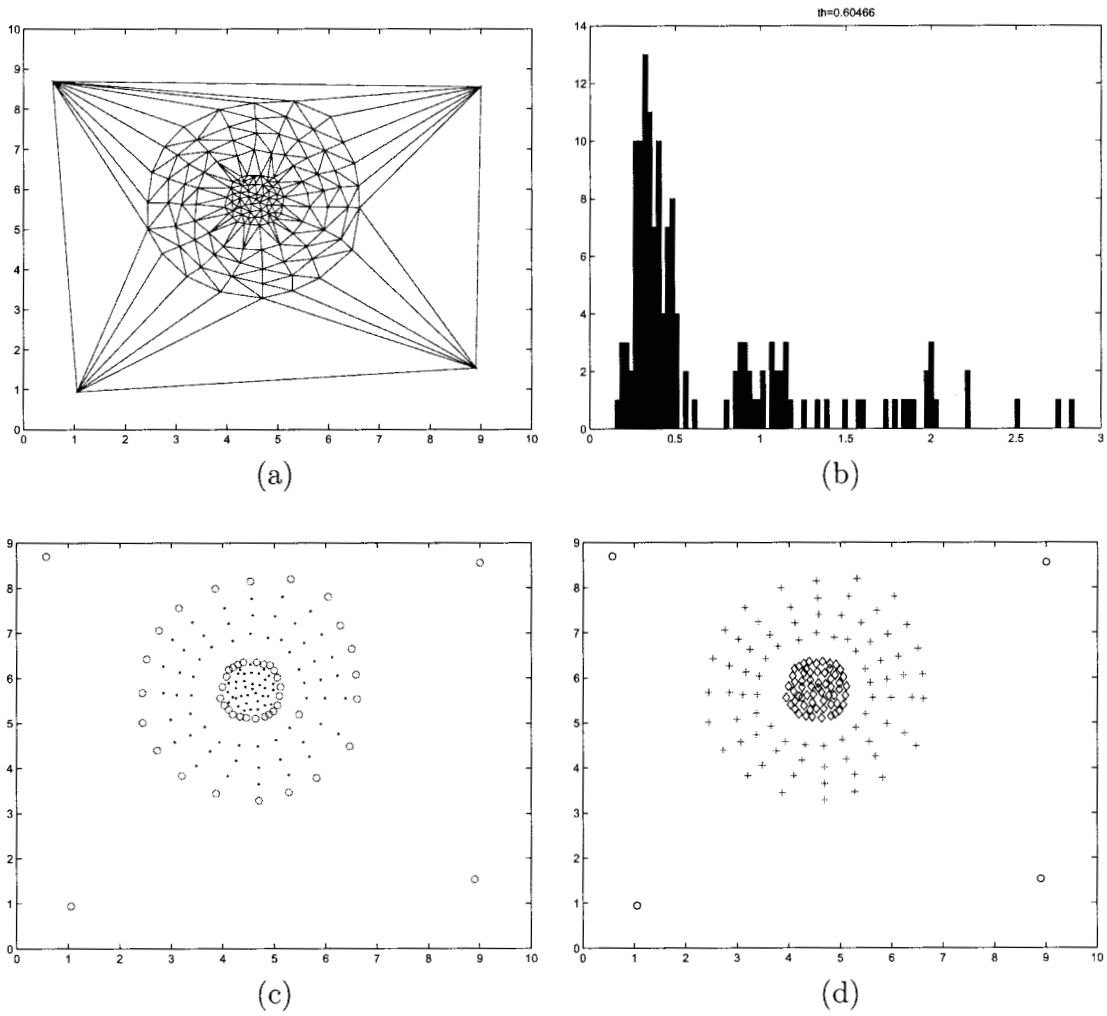


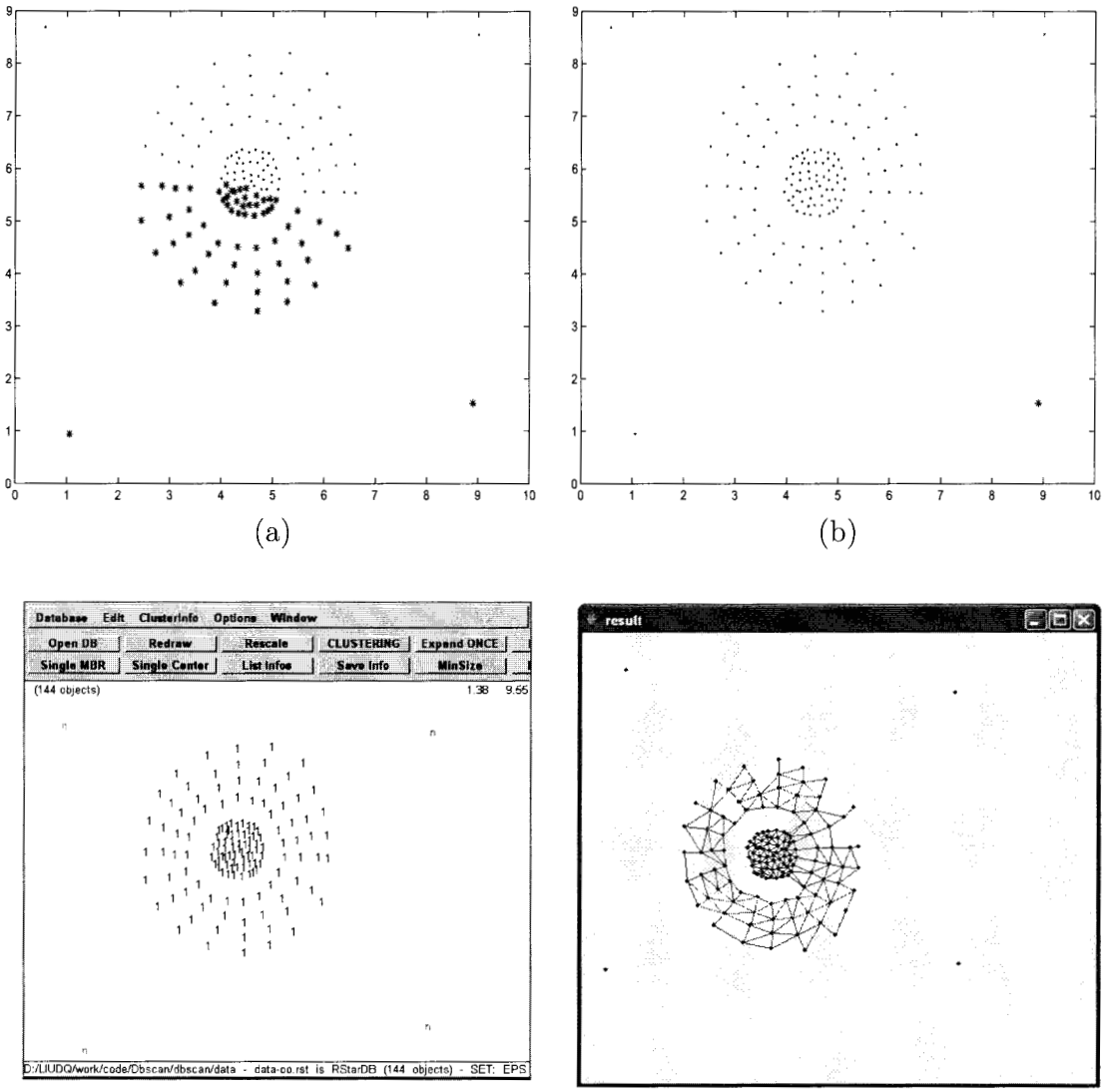
Figure 4.12: The testing on data set $D1$ of TRICLUST: (a)The graph built by triangulation of $D1$; (b)The distribution of final feature values of $D1$; (c)The boundary data points of $D1$ detected by TRICLUST; (d)The clustering result of $D1$ by TRICLUST

MEANS separated the clusters into half and joined parts of them together. Single Linkage method, AUTOCLUST and DBSCAN are not able to distinguish these clusters from each other at all. Comparing with the results shown in Fig.4.12, the advantage of TRICLUST to handle clusters with non-uniform density is well demonstrated.

In Fig.4.14, the clustering result of testing data set $D2$ by TRICLUST, boundary data points detected by TRICLUST and distribution of the final feature values are shown. There are three clusters in $D2$ together with three short bridges and three isolated outliers. Although the distances between points in short bridges are quite similar to the ones between data points inside clusters, we can see that the short bridges are correctly cut off to separate the three clusters. The robustness of TRICLUST to noises is shown.

In Fig.4.15, the clustering results of $D2$ by K-MEANS, Single Linkage method, DBSCAN and AUTOCLUST are provided. K-MEANS can not deal with clusters with such irregular shapes. Single Linkage method and AUTOCLUST suffered from the short bridges. And DBSCAN failed to separate two of the three clusters.

In Fig.4.16, the clustering result of testing data set $D3$ by TRICLUST, boundary data points detected by TRICLUST and distribution of the final feature values are given. There are two clusters in this data set together with two short bridges. But different from $D2$, the densities of the clusters in $D3$ are not uniform inside. One of the two short bridges connects the dense parts of those two clusters, and another one connects them with their thin parts so that we can test the effect of



(a) (b) (c) (d)

Figure 4.13: Clustering results of data set $D1$ by comparison methods: (a)Clustering result of $D1$ generated by K-MEANS; (b)Clustering result of $D1$ generated by Single-linkage algorithm; (c)Clustering result of $D1$ generated by DBSCAN, where $EPS = 0.7937$; (d)Clustering result of $D1$ generated by AUTOCLUST algorithm

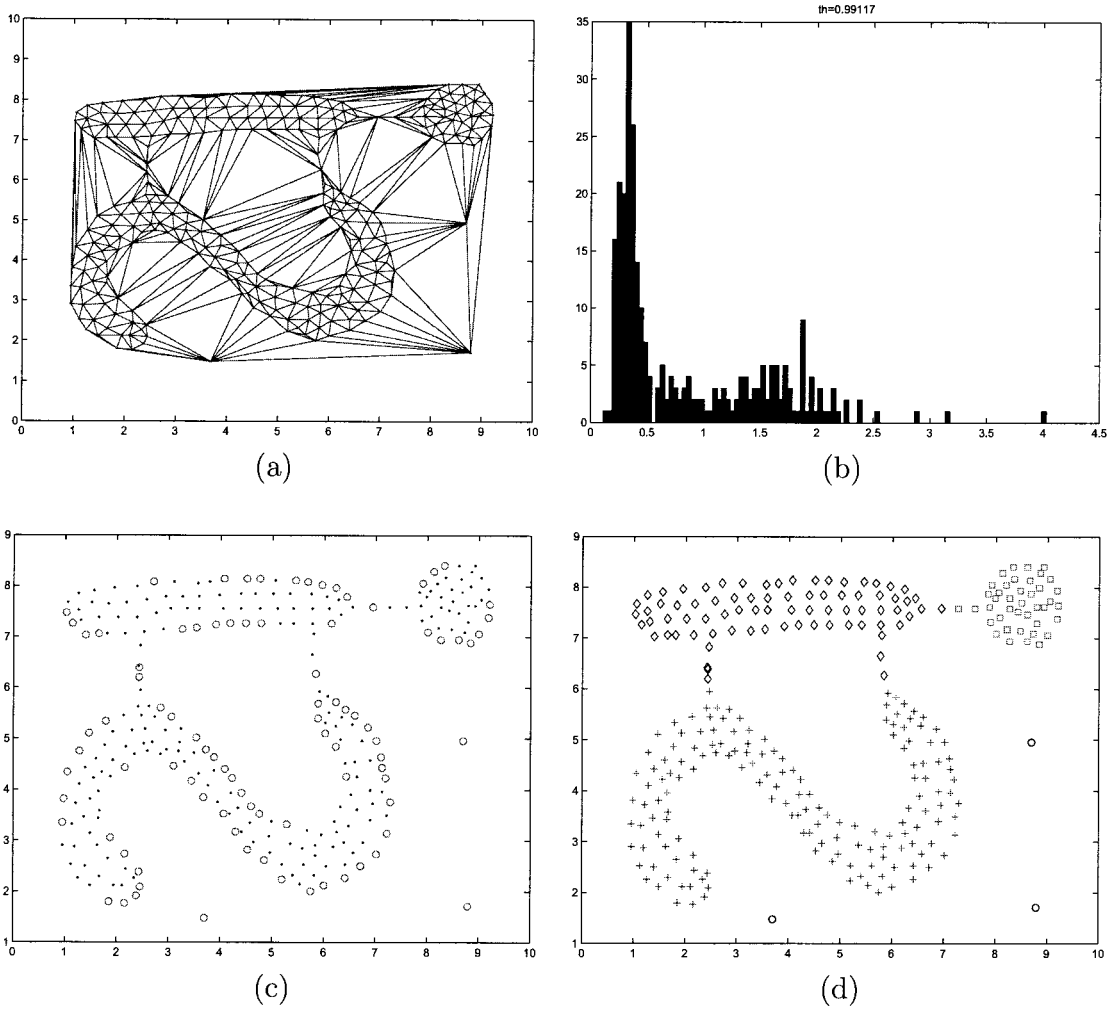


Figure 4.14: The testing on data set $D2$ of TRICLUST: (a)The graph built by triangulation of $D2$; (b)The distribution of final feature values of $D2$; (c)The boundary data points of $D2$ detected by TRICLUST; (d)The clustering result of $D2$ by TRICLUST

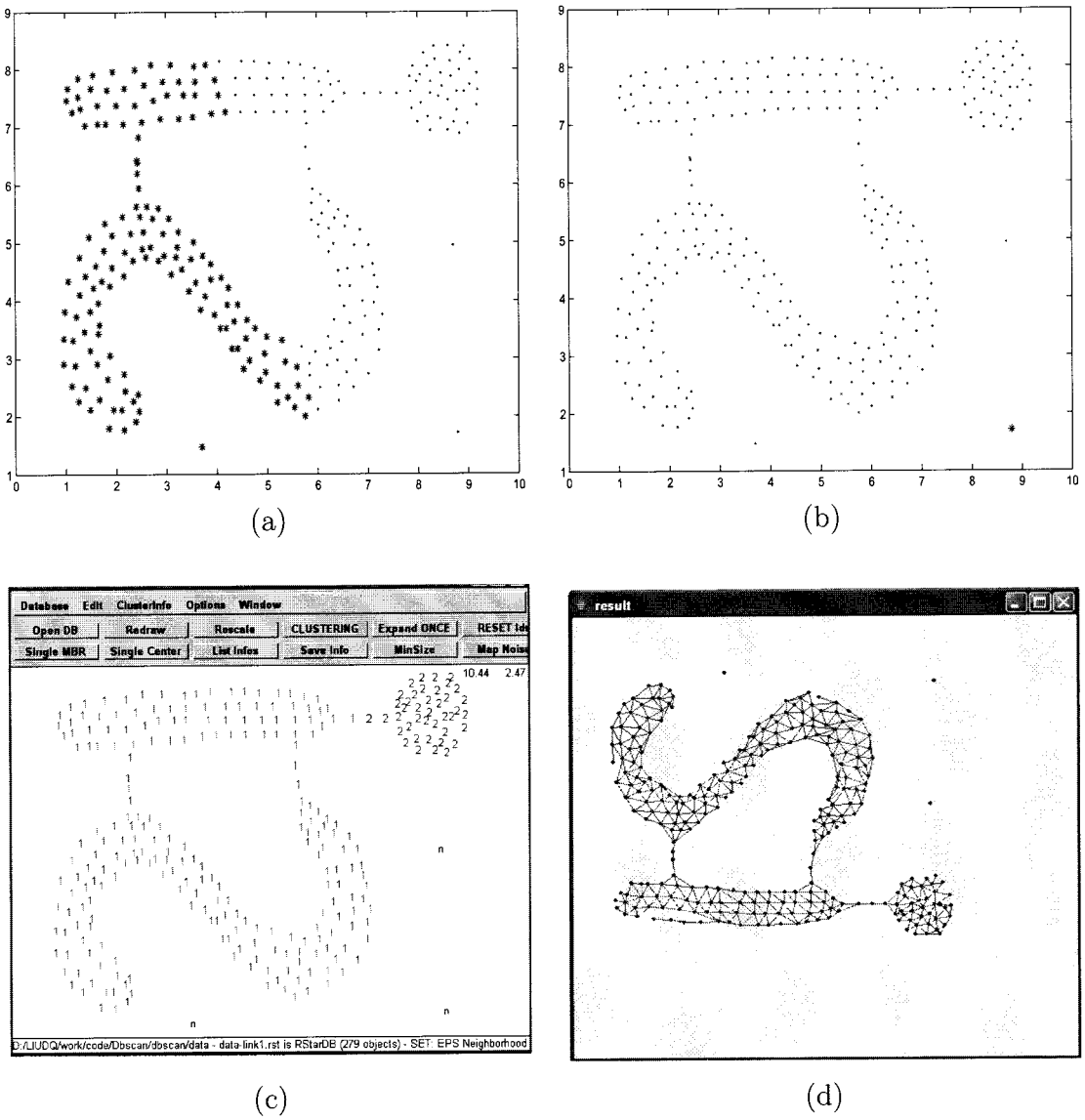


Figure 4.15: Clustering results of data set $D2$ by comparison methods: (a)Clustering result of $D2$ generated by K-MEANS; (b)Clustering result of $D2$ generated by Single-linkage algorithm; (c)Clustering result of $D2$ generated by DBSCAN, where $EPS = 0.4312$; (d)Clustering result of $D2$ generated by AUTO-CLUST algorithm

different locations of short bridges upon clustering result by this data set. From Fig.4.16, we can see that TRICLUST performed well in this case.

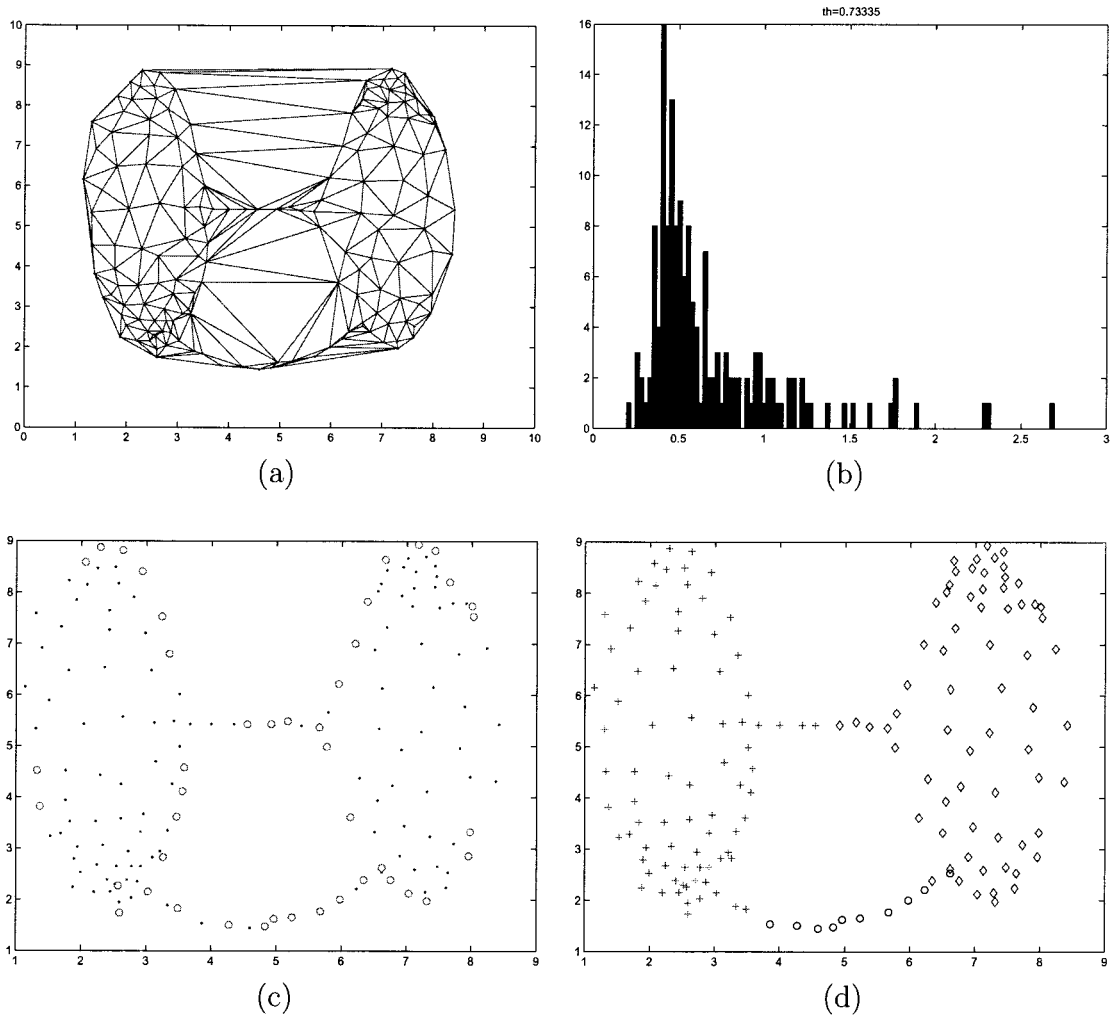


Figure 4.16: The testing on data set $D3$ of TRICLUST: (a)The graph built by triangulation of $D3$; (b)The distribution of final feature values of $D3$; (c)The boundary data points of $D3$ detected by TRICLUST; (d)The clustering result of $D3$ by TRICLUST

In Fig.4.17, the clustering results of $D3$ by K-MEANS, Single Linkage method, DBSCAN and AUTOCLUST are given. K-MEANS method cut both two clusters into half and joined each half of them together. Single Linkage method and DBSCAN glued one cluster and part of another cluster due to the non-uniform cluster

density. AUTOCLUST separated two clusters correctly, but all "short bridge" data points have been included into clusters.

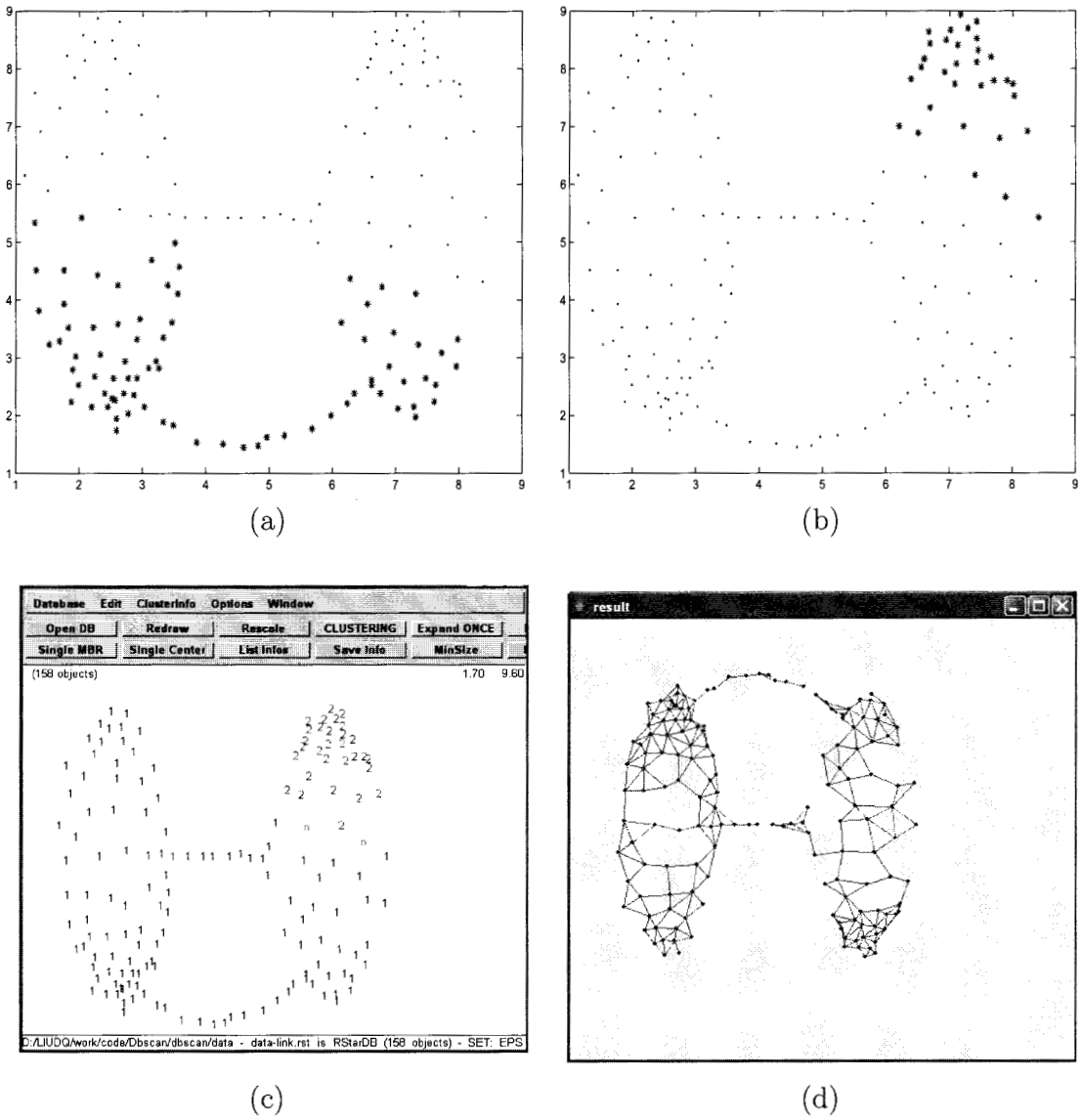


Figure 4.17: Clustering results of data set $D3$ by comparison methods: (a)Clustering result of $D3$ generated by K-MEANS; (b)Clustering result of $D3$ generated by Single-linkage algorithm; (c)Clustering result of $D3$ generated by DBSCAN, where $EPS = 0.7655$; (d)Clustering result of $D3$ generated by AUTOCLUST algorithm

In Fig.4.18, the clustering result of testing data set $D4$ by TRICLUST, boundary data points detected by TRICLUST and distribution of the final feature values

are provided. There are three clusters in this data set together with three short bridges and two isolated outliers. Comparing with the previous data sets, data set $D4$ is more challenging because not only short bridges exist but also the clusters are with non-uniform inner density and complex shapes. From the clustering result by TRICLUST, the good performance of it on $D4$ is clearly demonstrated.

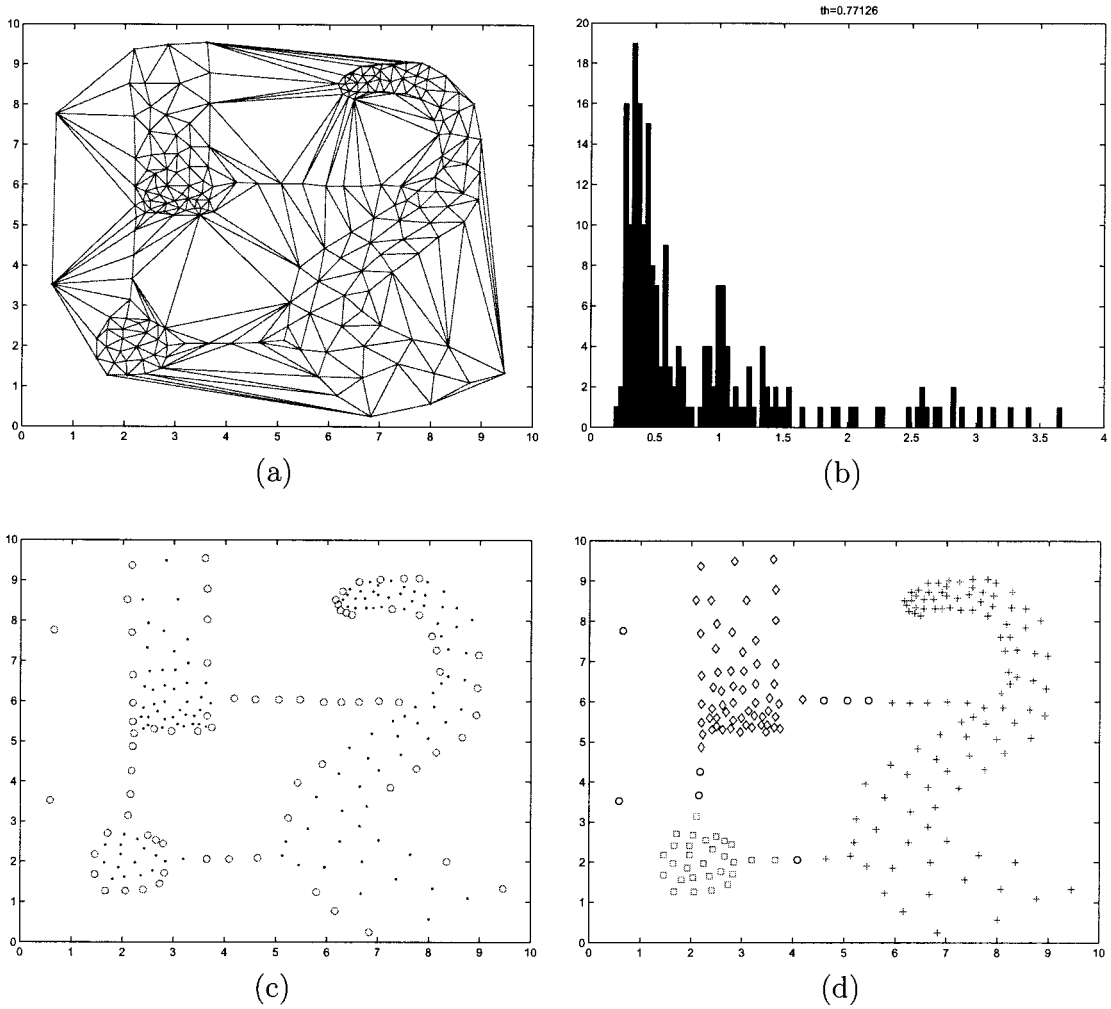


Figure 4.18: The testing on data set $D4$ of TRICLUST: (a)The graph built by triangulation of $D4$; (b)The distribution of final feature values of $D4$; (c)The boundary data points of $D4$ detected by TRICLUST; (d)The clustering result of $D4$ by TRICLUST

In Fig.4.19, the clustering results of $D4$ by K-MEANS, Single Linkage method,

Table 4.1: Clustering accuracy in percentage

	KMEANS	Single-linkage	DBSCAN	AUTOCLUST	TRICLUST
D1	57.64%	43.06%	45.14%	45.13%	100%
D2	46.45%	54.08%	69.79%	54.84%	96.06%
D3	61.37%	71.76%	70.16%	88.93%	93.04%
D4	62.23%	52.01%	89.22%	94.31%	94.61%

DBSCAN and AUTOCLUST are shown. K-MEANS method can not discover two of the three clusters. Single Linkage method joined all three clusters together because of the short bridges. DBSCAN did separate the three clusters but did not keep the integrity of them at the same time. AUTOCLUST separated three clusters correctly, but all "short bridge" data points have been included into clusters.

In Table.4.1, the accuracy of TRICLUST and all comparison methods on testing data sets are given. The definition of accuracy is given in 3.24. From this table, we can see, KMEANS lacks the ability to handle arbitrary shape clusters; Single-linkage method and DBSCAN can not deal with clusters with non-uniform density and "short-bridges"; AUTOCLUST performs better than other three methods, but it is not able to handle clusters with gradually changing density very well.

We also tested TRICLUST with the famous CHAMELEON [69] data sets which have been regarded as benchmark data sets in this area. From Fig.4.20 to Fig.4.22, the pictures of three CHAMELEON data sets $C1$ to $C3$ and clustering results of them by TRICLUST are provided. In order to illustrate the results better, we drew clusters with different symbols without outliers in those figures of clustering results. All clusters in these CHAMELEON data sets have been correctly detected by TRICLUST automatically, even there are short bridges between clusters. The

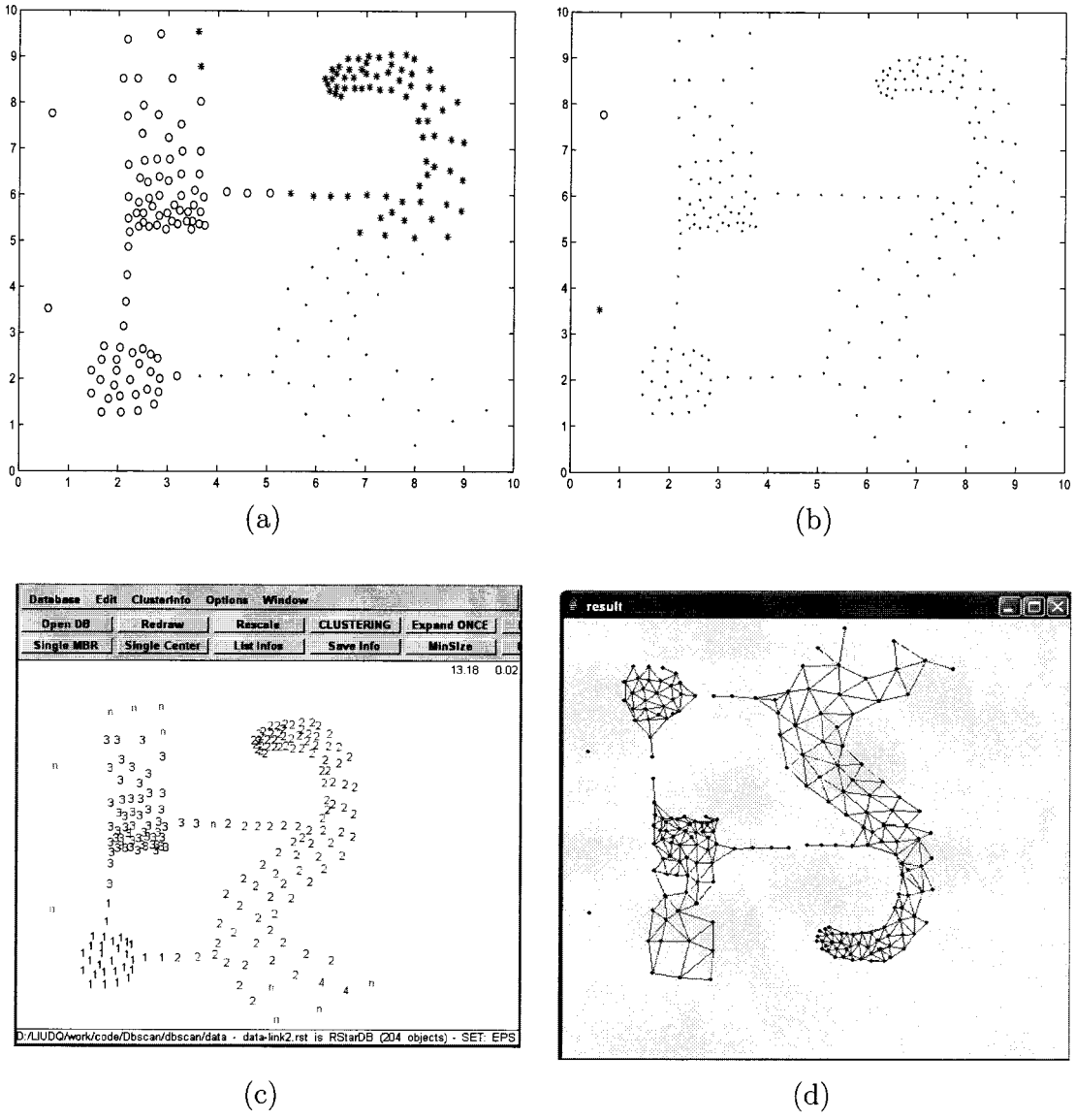
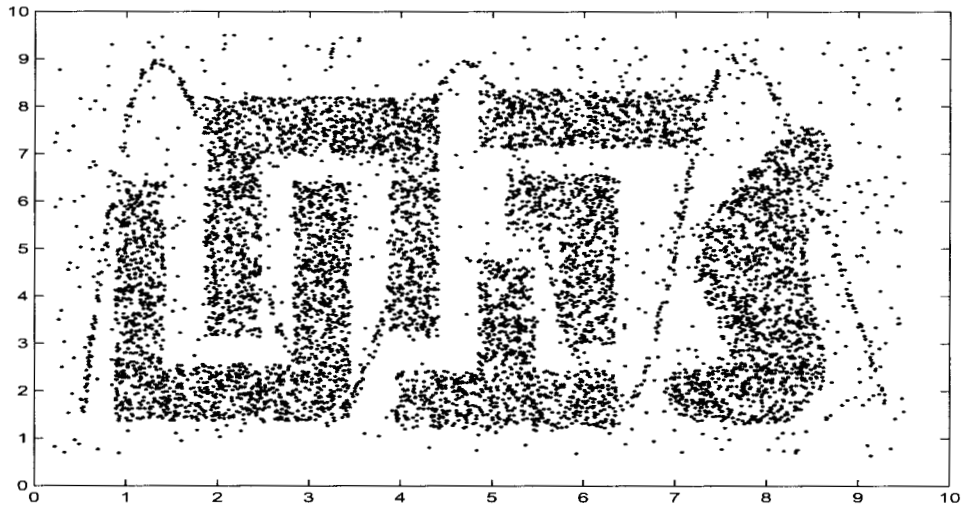
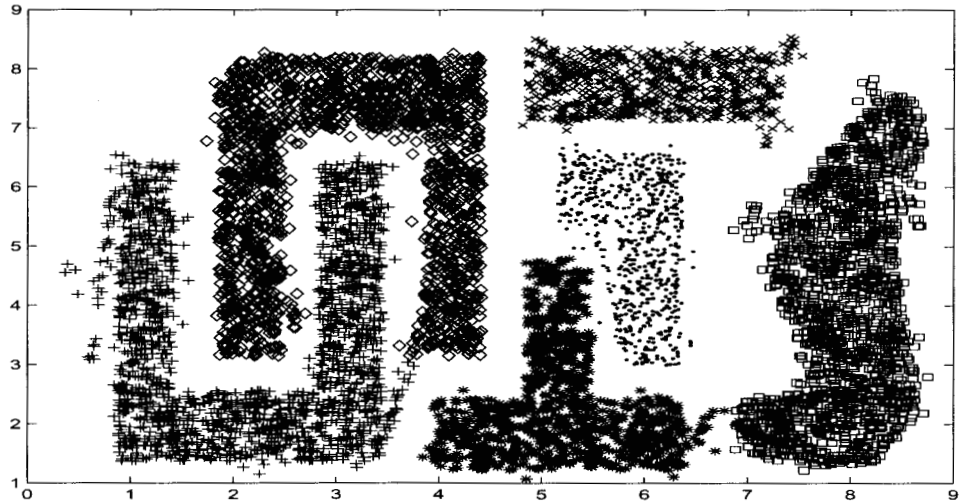


Figure 4.19: Clustering results of data set $D4$ by comparison methods: (a)Clustering result of $D4$ generated by K-MEANS; (b)Clustering result of $D4$ generated by Single-linkage algorithm; (c)Clustering result of $D4$ generated by DBSCAN, where $EPS = 0.7598$; (d)Clustering result of $D4$ generated by AUTO-CLUST algorithm

ability of TRICLUST to deal with large complex data sets with masses of noises is clearly shown. For comparison, in Fig.4.23, the clustering results of CHAMELEON bench-mark data sets by DBSCAN are given. In each picture, clusters found by DBSCAN were labeled with different colors and different symbols. We can notice that DBSCAN is not able to separate clusters correctly when clusters' density varies greatly and "short-bridges" exist.

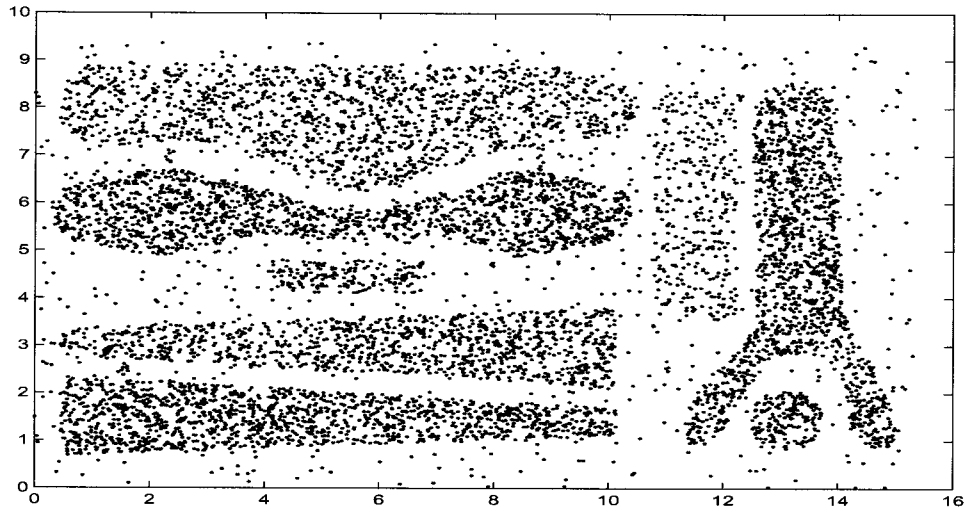


(a)

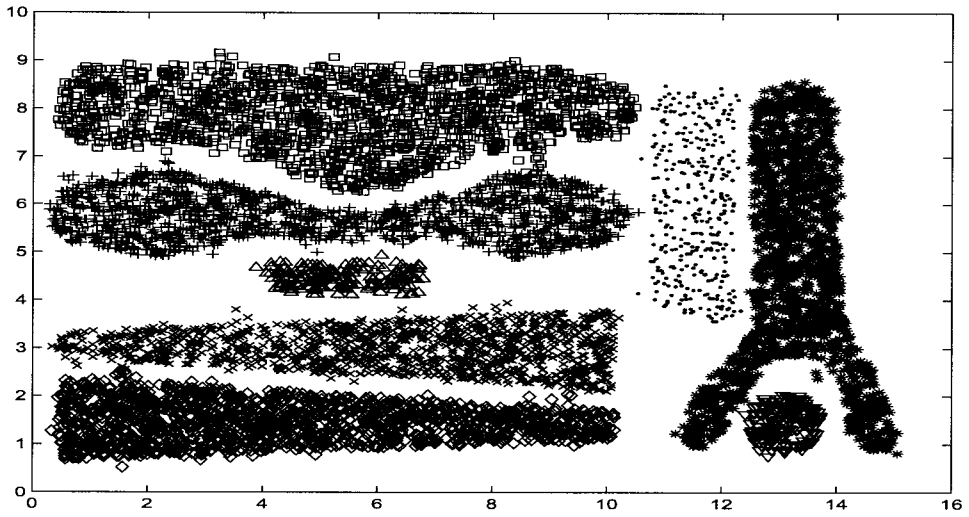


(b)

Figure 4.20: (a)The picture of CHAMELEON data set $C1$; (b)The clustering result of CHAMELEON data set $C1$ by TRICLUST



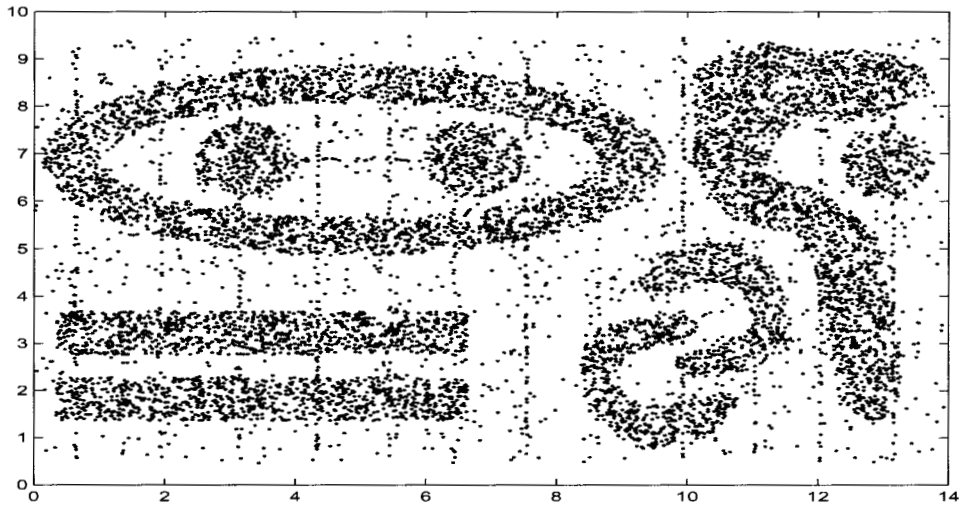
(a)



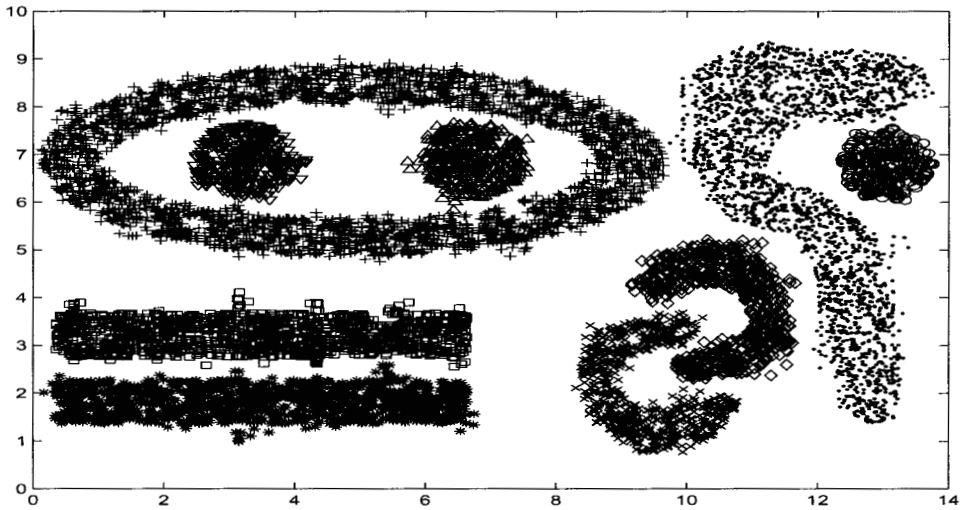
(b)

Figure 4.21: (a)The picture of CHAMELEON data set $C2$; (b)The clustering result of CHAMELEON data set $C2$ by TRICLUST

In this section, all clustering results are generated automatically according to the parameters setting method proposed in Section 4.4. Users can also set the values of parameters based on their knowledge or special requirements.

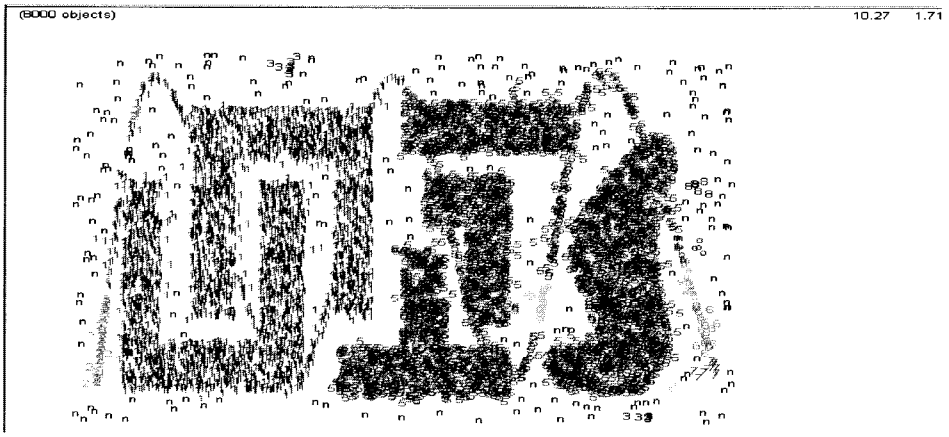


(a)

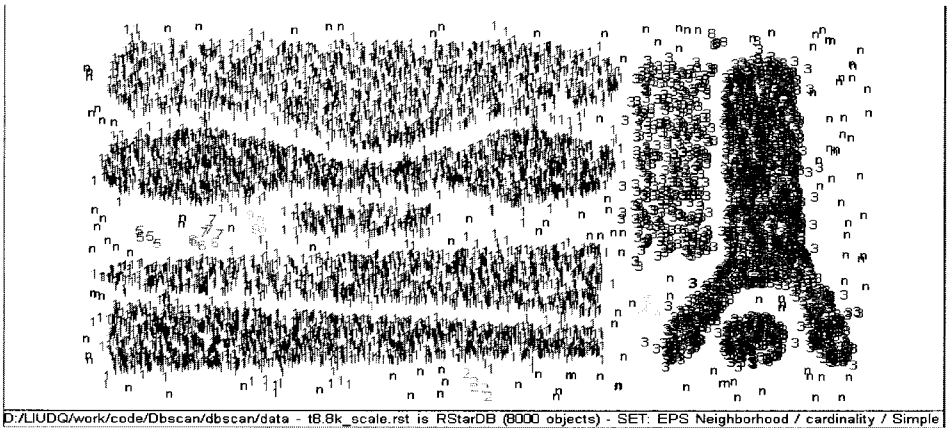


(b)

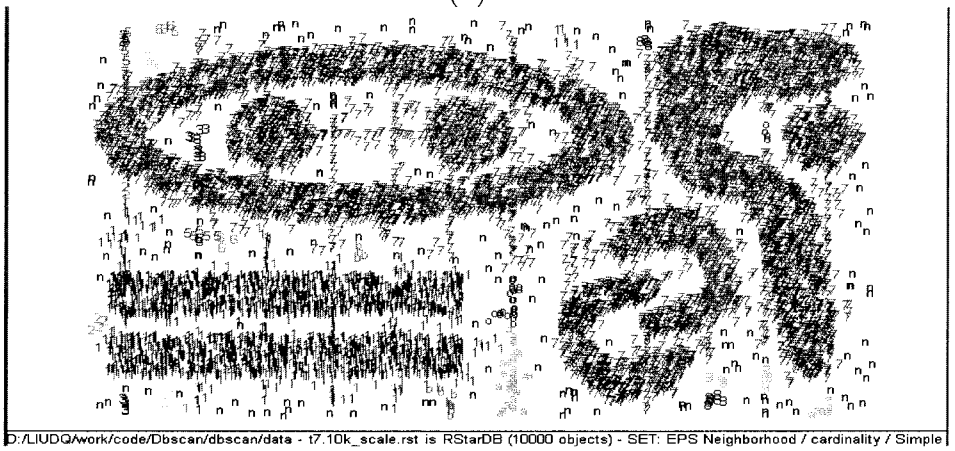
Figure 4.22: (a)The picture of CHAMELEON data set $C3$; (b)The clustering result of CHAMELEON data set $C3$ by TRICLUST



(a)



(b)



(c)

Figure 4.23: (a)The clustering result of CHAMELEON data set $C1$ by DBSCAN(EPS=0.1795); (b)The clustering result of CHAMELEON data set $C2$ by DBSCAN(EPS=0.2907);(c)The clustering result of CHAMELEON data set $C3$ by DBSCAN(EPS=0.2382)

4.8 Stability of TRICLUST to deviation of intrinsic parameters

In this section, we will discuss the stability of TRICLUST to three intrinsic parameters: *Smin*, *Smax* in 4.8 and 5000 in 4.9.

Parameters *Smin* and *Smax* are designed to simulate clustering feature of human vision: the more data points are inside the data set the less particular details in relatively small parts of data one can notice. In principle, these two parameters should be estimated by research in human vision. But TRICLUST algorithm appears to be robust to their deviation. The robustness of TRICLUST to variation of these parameters is illustrated below by tests on data set $D2$ and large data set $C1$, where *Smin* varies from 100 to 400 and *Smax* from 5000 to 20000. In Table.4.2, we provided corresponding values of a, b and c. The clustering results are the same as the ones shown in previous section.

Table 4.2: Variation of values of intrinsic parameters

		a		b		c	
Smin	*Smax*	$D2$	$C1$	$D2$	$C1$	$D2$	$C1$
100	5000	0.1694	2	0.9817	0.5	0.5183	1
	10000	0.1344	1.6162	0.9910	0.6010	0.5090	0.8990
	20000	0.1171	0.8453	0.9955	0.8015	0.5045	0.6985
200	5000	0.1313	2	0.9918	0.5	0.5082	1
	10000	0.1153	1.6122	0.9960	0.6020	0.5040	0.8980
	20000	0.1076	0.8485	0.9980	0.8030	0.5020	0.6970
400	5000	0.1395	2	1	0.5	0.5	1
	10000	0.1395	1.6042	1	0.6042	0.5	0.8958
	20000	0.1395	0.8367	1	0.8061	0.5	0.6939

The number 5000 in 4.9 is large data set threshold. For large data set, we cut

3% to increase algorithm robustness. This is a standard statistical procedure. The choice of large data set threshold highly depends on application and, in general, the variation of it would not affect the performance of TRICLUST.

4.9 Real World Application of TRICLUST

We also applied TRICLUST on real world data from European Topic Center on Air and Climate Change (ETC/ACC) as we did for ADACLUS in Section. 3.4.7. Here, the data set is about the locations of stations with more than 95% coverage for ozone value monitoring. The clustering result is shown in Fig.4.24. All 1181 stations have been divided into four clusters according their location distribution under both global and location view. The black dots represent stations that are far from others or with irregular locations. With TRICLUST, people who work in this area can conduct further investigations on the relationship between station positions and the coverage of stations. This result is generated automatically by TRICLUST. We can also set the parameters according to the knowledge on data set to fit the specific application requirements.

By the theoretical analysis, experimental testing and comparisons with several classic clustering algorithms, the capability of TRICLUST to efficiently deal with non-uniform density clusters and the robustness of it to noises and outliers are demonstrated. Since all boundary data points can be found by TRICLUST, it is convenient to build the cluster boundary for special applications.

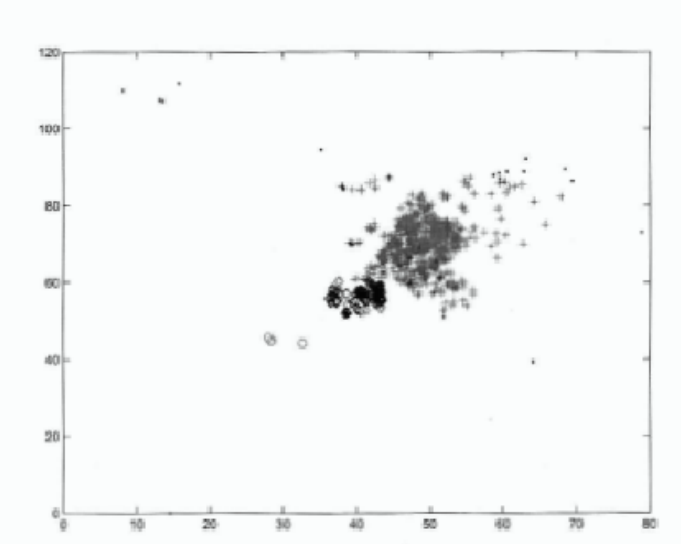


Figure 4.24: An example of real world application of TRICLUST

Chapter 5

Clustering Visualization System

Visualization could be said is the most intuitive way to understand and even to observe clusters, especially when the clusters are of irregular shape. This allows the users to be more involved in the clustering process via interactive visualization. The user can visually validate defined clusters via interactive operations. The interactive operations also allow the users to refine the clusters and to produce better results by incorporating their domain knowledge.

The clustering models introduced in Chapter 3 and Chapter 4 are based on the geometric concepts in computer graphics, such as Blobby model, or geometric topology representation, such as triangulation. One additional natural advantage of them, besides the good algorithm performance shown already in previous chapters, is the simplicity of invoking visualization. Sharing the same data representation and modeling functions provides a convenient way for the information transformation from clustering system to visualization system. Thus, a more efficient and compact system which integrates clustering process with visualization process can be built, moreover, not only the result of clustering could be easily visualized to users but also the knowledge and activity of users could enhance the

clustering algorithm performance. In this chapter, the basic visualization models, software platform, and the demonstrations of the visualization system we developed are described. Our system is designed and implemented for 2D and 3D data.

5.1 Implicit Modeling

In Chapter 2, we reviewed the basic concepts of implicit modeling of solid objects. Here, we briefly introduce properties of implicit modeling and the models we used in our system.

5.1.1 Properties of Implicit Modeling

Implicit modeling uses implicit function instead of parametric function or explicit function as its mathematical foundation. Implicit functions are functions of the form $f(x_1, x_2, \ldots, x_n) = c$, where c is any constant and n is the dimension of variables. Implicit function has three important properties:

- *Simple geometric description.* Implicit functions are convenient to describe primitive geometric objects [129, 130] such as planes, spheres, cylinders, cones, ellipsoids, and etc.

- *Region separation.* Implicit functions separate the n-dimensional space into 3 distinct regions. These regions are inside, on, and outside the implicit function. These regions are defined as $f(x_1, x_2, \ldots, x_n) < c$, $f(x_1, x_2, \ldots, x_n) = c$, $f(x_1, x_2, \ldots, x_n) > c$, respectively.

- *Scalar generation.* Implicit function converts a position in space into a scalar

value. If we know the implicit function, we can get the scalar value at any point by sampling the implicit function at that point.

The basic approach of implicit modeling is to evaluate implicit functions on a regular array of points, or volume, and then to generate scalar values at each point in the volume. We can use single implicit function to model simple geometric objects, however, because of the limitation of simple implicit function in modeling complex geometries, we could use boolean or set-theoretical operations over primitives to model complex geometries.

Let f_1, f_2 and f_3 be three implicit functions, G_1, G_2 and G_3 be geometric objects, E^n be n-dimensional space, then the set-theoretical operations union, intersection, complement, difference and cartesian product can be mathematically defined as [147]:

- *Union*: $G_3 = G_1 \cup G_2$ of two objects $G_1 \subset E^n$ and $G_2 \subset E^n$ with the descriptive functions f_1 and f_2 will be defined as function $f_3 = f_1 \vee f_2 = max(f_1, f_2) = 0$, where $G_3 \subset E^n$.

- *Intersection*: $G_3 = G_1 \cap G_2$ of two objects $G_1 \subset E^n$ and $G_2 \subset E^n$ with the descriptive functions f_1 and f_2 will be defined as function $f_3 = f_1 \wedge f_2 = min(f_1, f_2) = 0$, where $G_3 \subset E^n$

- *Complement*: $G_2 = \neg G_1$ of object $G_1 \subset E^n$ with the descriptive function $f_1$will be defined as function $f_2 = -f_1 \geq 0$

- *Difference*: $G_3 = G_1 \setminus G_2$ between objects $G_1 \subset E^n$ and $G_2 \subset E^n$ with

descriptive functions f_1 and f_2 will be defined as function $f_3 = f_1 \wedge (\neg f_2) = min(f_1, -f_2) \geq 0$, where $G_3 \subset E^n$.

- *Cartesian product*: $G_3 = G_1 \times G_2$ of objects $G_1 \subset E^n$ and $G_2 \subset E^n$, $k + m = n$, with descriptive functions f_1 and f_2 will be defined as function $f_3 = f_1 \wedge f_2 = min(f_1, f_2) \geq 0$, where $G_3 \subset E^n$.

The most frequently used geometric object for our project is ellipsoid. Mathematically, an ellipsoid can be defined as $f(\vec{X}) = r^2 - ((x_1 - x_{0,1})/a_1)^2 - ((x_2 - x_{0,2})/a_2)^2 - \ldots - ((x_n - x_{0,n})/a_n)^2 = 0$ where $x_{0,1}, x_{0,2} \ldots x_{0,n} \in R$ and $a_1, a_2 \ldots, a_n \in R$.

5.1.2 Models in our System

There are two specific models mainly used in the implementation of our visualization system—Blobby model and Metaball model.

- **Blobby Model**

  Blobby models are originally proposed for simulation of electron density filed. However, they are commonly used for $3d$ modeling, especially in scientific research areas. The basic idea is to build iso-surface around a data point. The surface is defined by function 5.1 which is the influence function for Blobby model:

  $$D(r) = ae^{-br^2} \tag{5.1}$$

  r is the distance from a data point, a is the scale factor of the function and b is the exponential factor of the distribution curve, which is Gaussian distribution. Compared with the influence function defined by 3.3 in Chapter

3, for that case, the parameter values are as $a = 1$ and $b = 1/2\sigma^2$. The potential (field value) for each iso-surface point is equal to the sum of all data points' contributions. Thus, the field function for Blobby model is an implicit function defined as 5.2:

$$S = \sum_{i=1}^{N} D(r_i) \begin{cases} S < T, & \text{Outside the iso-surface.} \\ S = T, & \text{On the iso-surface.} \\ S > T, & \text{Inside the iso-surface.} \end{cases} \tag{5.2}$$

where N is the total number of data points and T is a threshold constant that determines the value of the iso-surface. As T increases, the surface gets more compact and smoother between adjacent data points. This is identical to the definitions of field function and cluster in Chapter 3, and one iso-surface represents a cluster boundary.

An example of blobby model is given in Fig.5.1[148]. If there are two identical data points (in the same position and with the same parameter values), different threshold values will result in spheres of different radius; if the two data points move apart, the combined fields will interact, the resulting iso-surface will plunge between the two data points depending on their relative distance.

- **Metaball Model**

Metaball model is an alternative to implementation of blobby models. Metaball model is computationally cheaper in creating implicit surfaces. The influence function of metaball model is defined in 5.3.

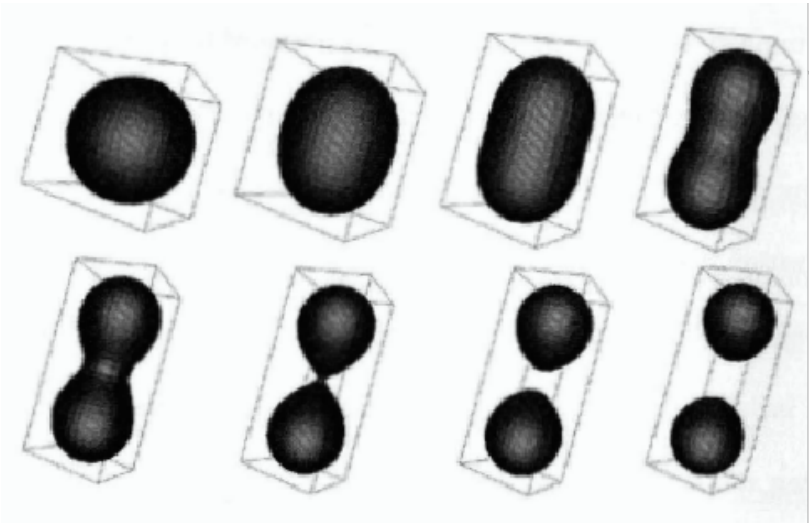


Figure 5.1: An example of Blobby model regarding different distances between two data points

$$D(r)\begin{cases} a(1-\frac{3r^2}{b^2}), & 0 \le r < \frac{b}{3} \\ \frac{3a}{2}(1-\frac{r}{b})^2, & \frac{b}{3} \le r < b \\ 0, & b \le r \end{cases} \tag{5.3}$$

where a is the value of $D(r)$ at $r = 0$, and b defines the intervals of this piecewise function. For all $r \geq b$, the function returns 0.

The corresponding field function for Metaball model is defined as same as 5.2.

5.2 Visualization System Implementation

Visualization offers the user an intuitive way of analysis that can help to understand clustering results and discover data patterns and structures. By applying implicit modeling techniques in data visualization, we can not only efficiently visualize clusters with complicated shapes but also we are able to integrate more interactive functions into the visualization system because of the unique proper-

ties of implicit modeling which we have introduced in Section 5.1.1. It provides more flexibility to users and a more efficient way to utilize users activity. We developed our visualization system based on implicit models and implemented it with Visualization Toolkit (VTK). Introductions of VTK and visualization algorithm Marching Cubes are provided in Appendix. The functions of the implemented system include visualization of data set and clustering results, visual interactive clustering and cluster querying.

In order to help users getting a clear view, we have the coordinate system in rendering window together with visualization result and, by dragging the mouse, users can rotate and zoom in/out the view. Our system allows users to have different views (front, top, left side, and right side) in the render windows. In Fig.5.2, an example of different views of the same solid is provided.

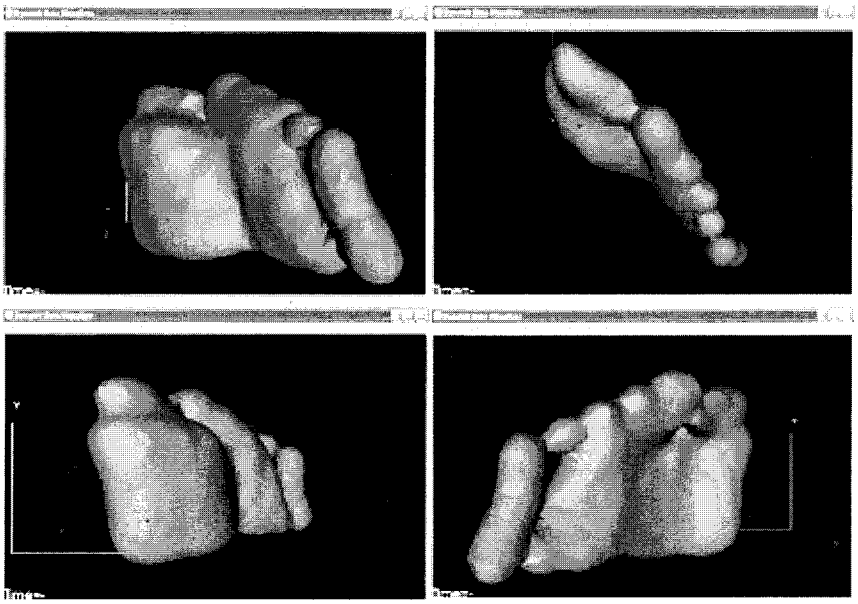


Figure 5.2: View one solid from different directions

5.2.1 Clustering Result Visualization

The visualization functions of our system include visualization of data points in the data set and visualization of clustering result. The data set visualization provides a prime look at the data as points clouds. The user can choose the number of data points to be visualized. And the whole data set or any part of it can be shown on the screen. In Fig.5.3, an example of data set visualization is given.

The clustering result visualization is based on our clustering algorithms proposed in Chapter 3 and Chapter 4. The data points can be visualized with different colors according to their cluster index so that the clustering result will be shown as colored point clouds. For visualizing the clustering results of algorithms introduced in Chapter 3, there are more advantages because of the same model used for both clustering and visualization. It means the implicit solids (clusters) are already available after clustering processes, and all values of parameters as well. Although the clustering results can be shown directly by our system without any user's assistance, we still designed the GUI to let users change the values of parameters to fit their special requirements. In Fig.5.4, the clustering result of one $3d$ data set is shown. In Fig.5.5, the same data set viewing with different parameters setting is provided. In Fig.5.6 and Fig.5.7, clustering results of two $3d$ data sets with uniform and non-uniform inner-cluster density correspondingly are given.

Another visualization function of our system is to view time-dependent data set. By indicated the time frames, data set at any time can be visualized so that the time-dependent features of it can be investigated, such as the movement directions

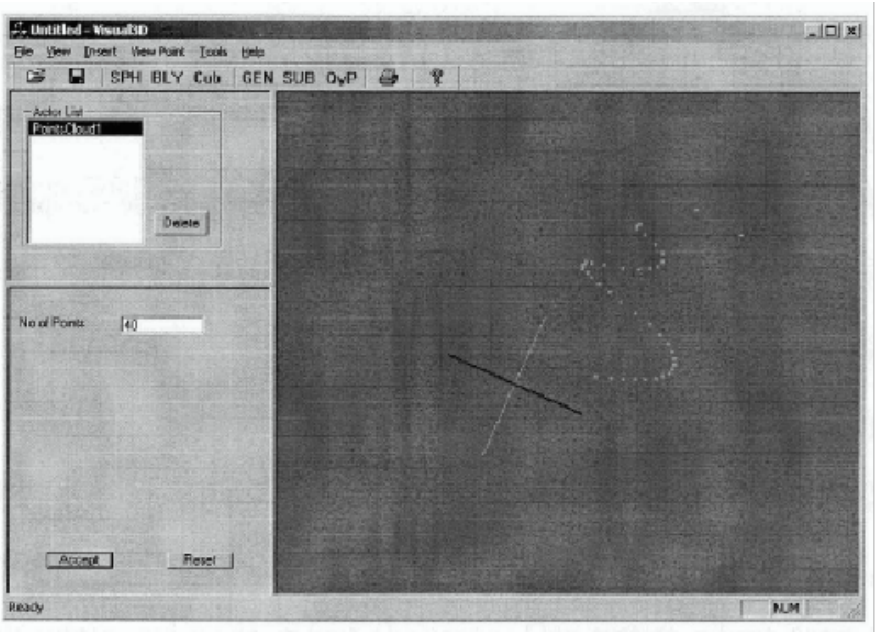


Figure 5.3: Visualization of data points as points cloud

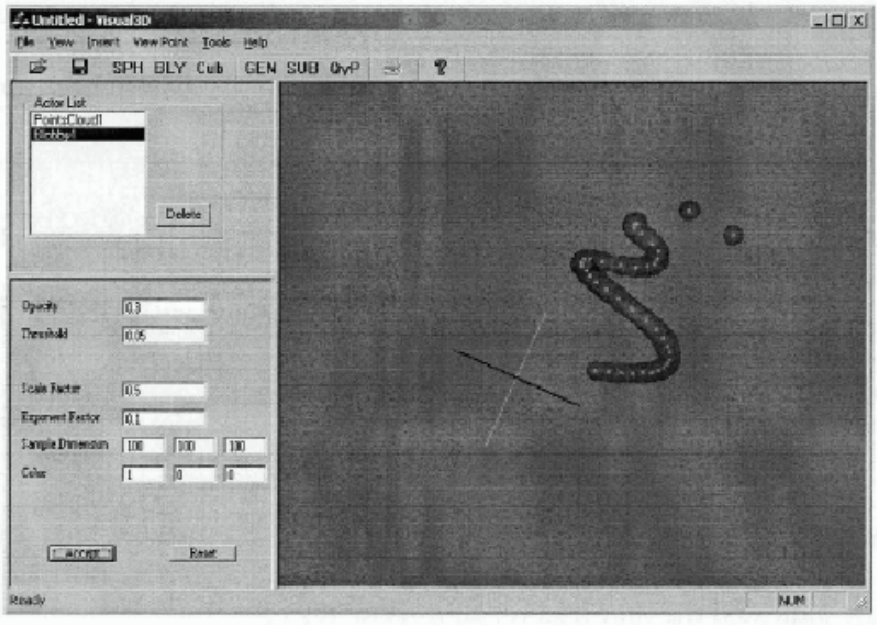


Figure 5.4: Visualization of clustering result

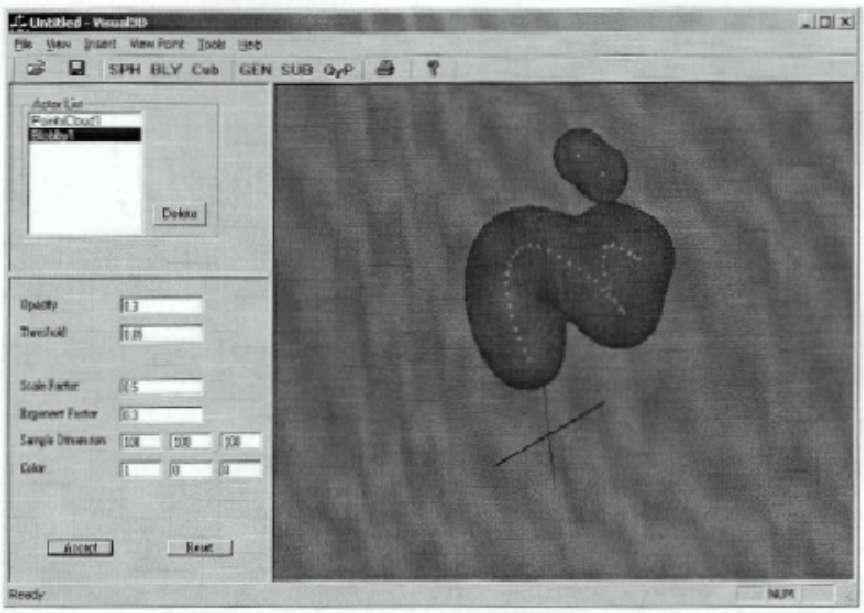


Figure 5.5: Visualization with different parameters setting

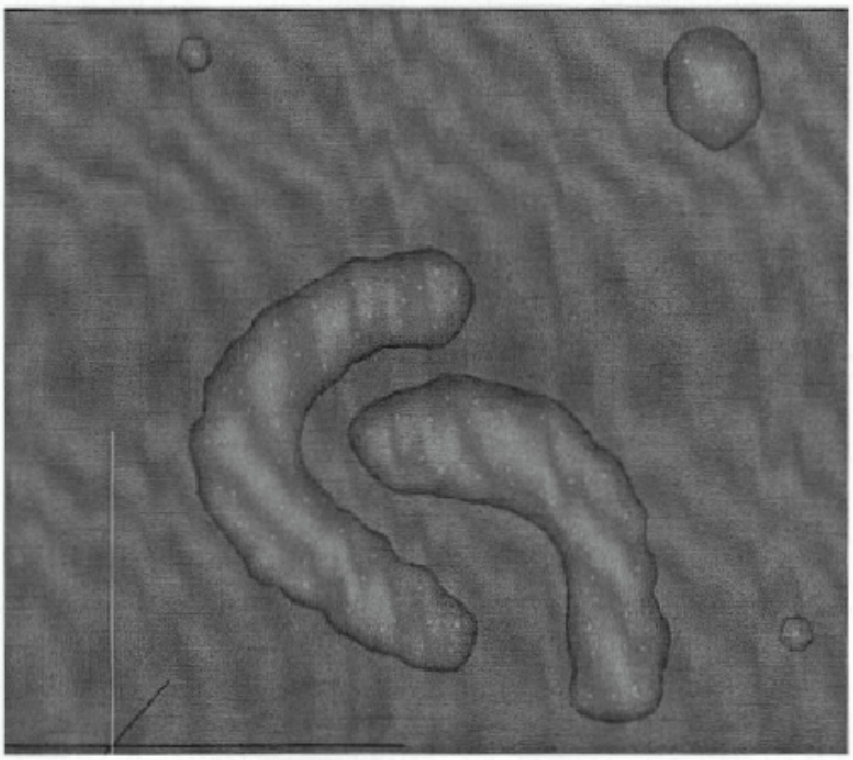


Figure 5.6: Visualization of clustering result of data set with uniform inner-cluster density

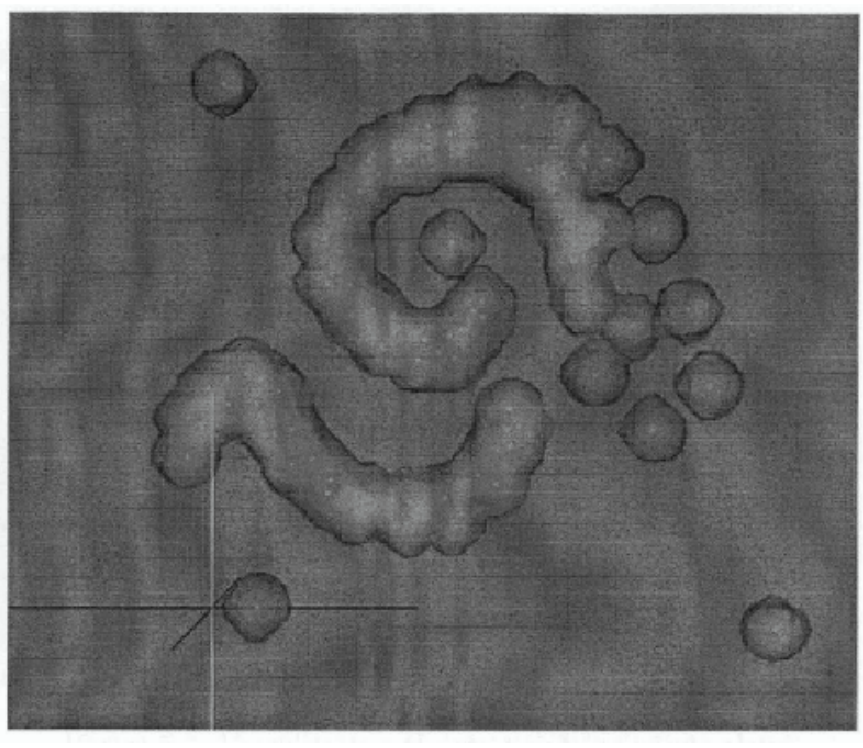


Figure 5.7: Visualization of clustering result of data set with non-uniform inner-cluster density

of data points or cluster shape changing tendency against time. In Fig.5.8, an example of visualizing one time-dependent data set is shown. In Fig.5.9, the time points of this data set are given.

5.2.2 Visual Interactive Clustering

Visualization techniques could be used not only for the interpretation of the results but also for the interpretation of the whole process of clustering in order to help the user to come up with hypothesis and to set the values of parameters, especially for 3-dimensional clustering. In our system, we provided the option that users can cluster their data set manually via the interactive clustering GUI. They are able to view their data set from different directions and select the most suitable im-

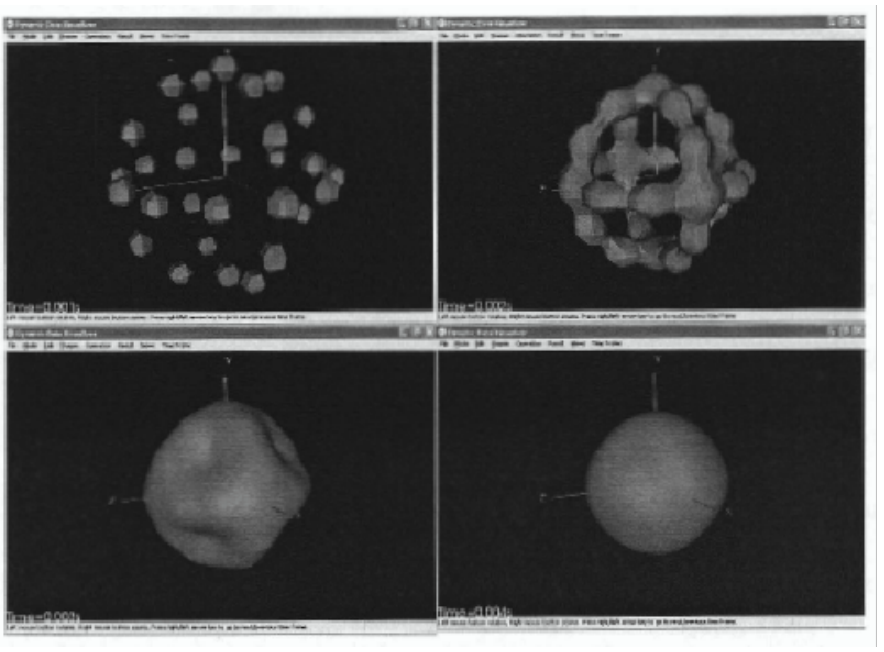


Figure 5.8: Visualization of time-dependent data set

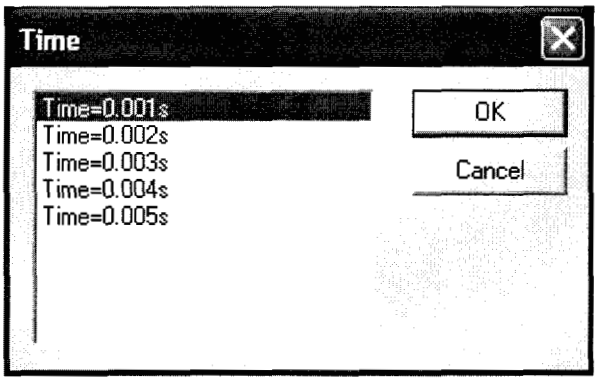


Figure 5.9: Time frame

plicit primitives and parameter values based on their knowledge and requirements. This interactive clustering function maximizes the user's activity by providing the flexible choices of implicit primitives and good visual assistance.

The candidate primitive solid objects are sphere, cube, ellipsoid and blobby. Once the user selects what kind of primitive solid he/she is going to use and sets all values of parameters, the primitive solid will be displayed in the rendering window. When the user is creating complex solid, the system applies the set-theoretic operations over primitive solids to form the final solid (cluster). As soon as the user finishes his/her work of initializing enough primitive solids to wrap up the expected cluster, the implicit function of this cluster is also generated by system so that the data set is well separated based the user's judgment. After, the system evaluates every data point according to the formed implicit functions of clusters, a complete list of data point status (inside the wrapping solid, on its boundary or outside it) is generated. The clustering result is finally gotten. In Fig.5.10 and Fig.5.11, the interactive clustering process of the same data set shown in Fig.5.3 is given to illustrate the visual clustering procedure of our system. In this example, we can separate this data set by wrapping either the cluster or the outliers. If we wrap the helical cluster, the outliers will be tested as located outside our solid (cluster), and vice versa. Both approaches are shown in Fig.5.10 and Fig.5.11.

5.2.3 Cluster Querying

The development of query methods with graphical user interfaces is a new trend in data mining [4]. Querying of data is a classical problem in spatial clustering,

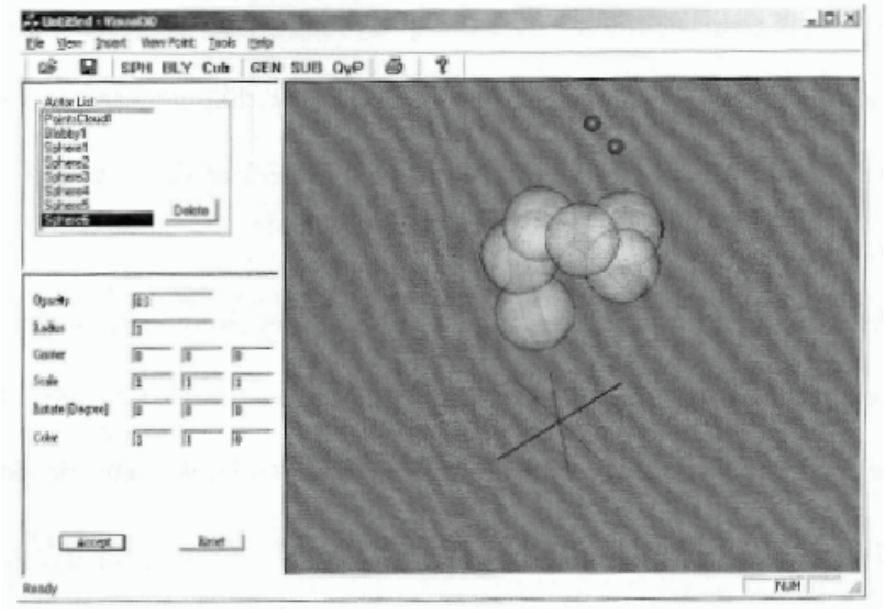


Figure 5.10: Visual interactive clustering by wrapping cluster

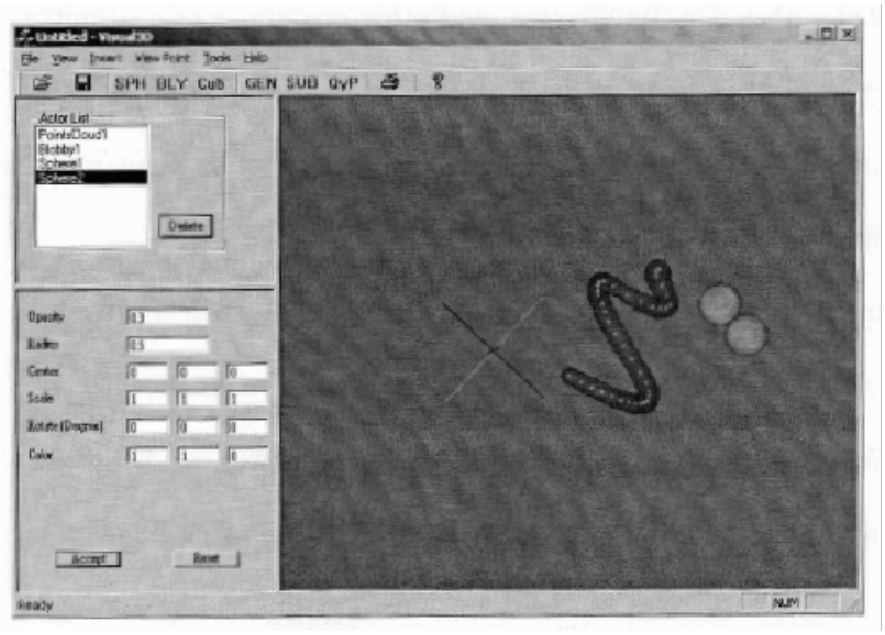


Figure 5.11: Visual interactive clustering by wrapping outliers

spatio-temporal databases, and warehousing. The goal of works in this area is to propose data representation model and query model able to handle time-dependent geometries including those changing continuously that describe moving objects [149] [150]. In our system, to deal with time-dependent data, we can also extend the model with time dimension.

After we visualize clusters using interactively set parameters, we can query the clusters using geometric objects. In this section, we describe our geometric query model consisting from geometric objects and operations. The proposed model is an extension of the model that was introduced first in work [147]. As it was shown there, geometric interpretation of relational algebra selection operation can be phrased as follows: "find out the points that belong to the solid." In our model, the query solid can be a complex geometric solid. The complex query solid can be created with union, intersection, and other operations over primitive solids that are generally hyperhalfspaces, hypercuboids, hyperellipsoids, etc. Selection operation of relational algebra can be found in geometry as point/solid classification predicate.

In general, the point/solid classification predicate can be described as follows. Let $\vec{P}$ be a data point in Euclidean space $\mathbf{E^n}$, $G1$ be a query solid described with implicit function $F1$, $bG1$ be a boundary of $G1$, and $iG1$ be an interior of $G1$. Then a point/solid predicate is described with the implicit function representation

of the geometric object $G1$ by a 3-valued predicate $S(P, G1)$:

$$S(\vec{P}, G1) = \begin{cases} 0 & (\vec{P} \in \neg G1), \quad \text{if} \quad F1(x_1, ..., x_n) < 0 \\ 1 & (\vec{P} \in bG1), \quad \text{if} \quad F1(x_1, ..., x_n) = 0 \\ 2 & (\vec{P} \in iG1), \quad \text{if} \quad F1(x_1, ..., x_n) > 0 \end{cases} \tag{5.4}$$

In our 3-dimensional visualization system, query solid can be defined analytically or by procedure. Thus, the geometric query model consists of the following geometric objects:

- 3-dimensional points: $\vec{P} = \{x_1, x_2, x_n\}$.
- 3-dimensional primitive geometric objects for the construction of a query solid using geometric operations.

The following are implicit function representations of the primitive 3-dimensional geometric solids that could be used for construction of geometric criteria $G1$:

- **Halfspace**

$$G1: \quad F1(\vec{X}) = F1(x_1, x_2, x_3) = (x_i - a_i) \geq 0$$

where a_i is real number ($a_i \in \mathbf{R}$).

- **Sphere**

$$G1: \quad F1(\vec{X}) = r^2 - (x_1 - x_{0,1})^2 - (x_2 - x_{0,2})^2 - (x_3 - x_{0,3})^2 \geq 0$$

where $x_{0,1}, x_{0,2}, x_{0,3}, r \in \mathbf{R}$.

- **Ellipsoid**

$$G1 : F1(\vec{X}) = 1 - ((x_1 - x_{0,1})/a_1)^2 - ((x_2 - x_{0,2})/a_2)^2 - ((x_3 - x_{0,3})/a_3)^2 \geq 0$$

where $x_{0,1}, x_{0,2}, x_{0,3} \in \mathbf{R}$, and $a_1, a_2, a_3 \in \mathbf{R}$.

- **Cone**

$$G1: \quad F1(\vec{X}) = ((x_1 - x_{0,1})/a_1)^2 - ((x_2 - x_{0,2})/a_2)^2 - ((x_3 - x_{0,3})/a_3)^2 \geq 0$$

where $x_{0,1}, x_{0,2}, x_{0,3} \in \mathbf{R}$, and $a_1, a_2, a_3 \in \mathbf{R}$.

- **Cylinder**

$$G1: \quad F1(\vec{X}) = ((x_1 - x_{0,1})/a_1)^2 - ((x_2 - x_{0,2})/a_2)^2 \geq 0$$

where $x_{0,1}, x_{0,2}, x_{0,3} \in \mathbf{R}$, and $a_1, a_2, a_3 \in \mathbf{R}$.

By further declaring that our model is open to any type of objects that can be defined implicitly with any functions $F(x1, x2, x3) \geq 0$, we could avoid the problem of a minimum set of primitives and to change this set depending on the application problem to be solved.

Geometric operations which have been introduced in Section 5.1.1 are applied on primitive geometric objects to obtain complex geometric solid. The analytical definition of set-theoretic operations is realized in the form proposed by Ricci [151], where operations over implicit functions are considered. Affine transformations (translation, rotation and scaling) are also used to increase an expressive power of the proposed geometric model. Geometric operations include set-theoretic union, intersection, difference and orthographic projection.

From Fig.5.12 to Fig.5.14, the geometric operations union, intersection, and substraction over primitive query solids such as cone, cylinder, tube, and cuboid

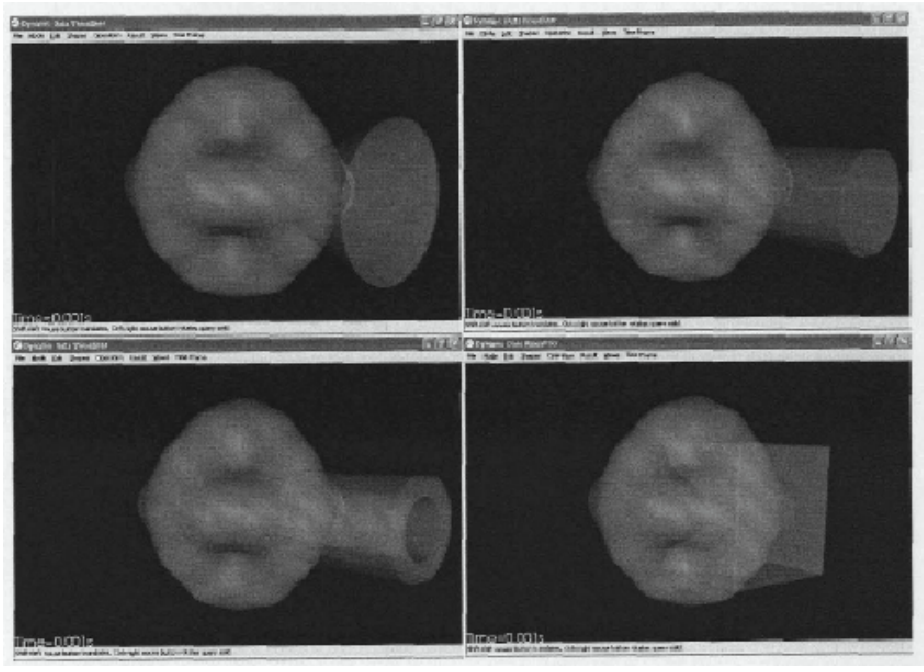


Figure 5.12: Geometric operations: Union

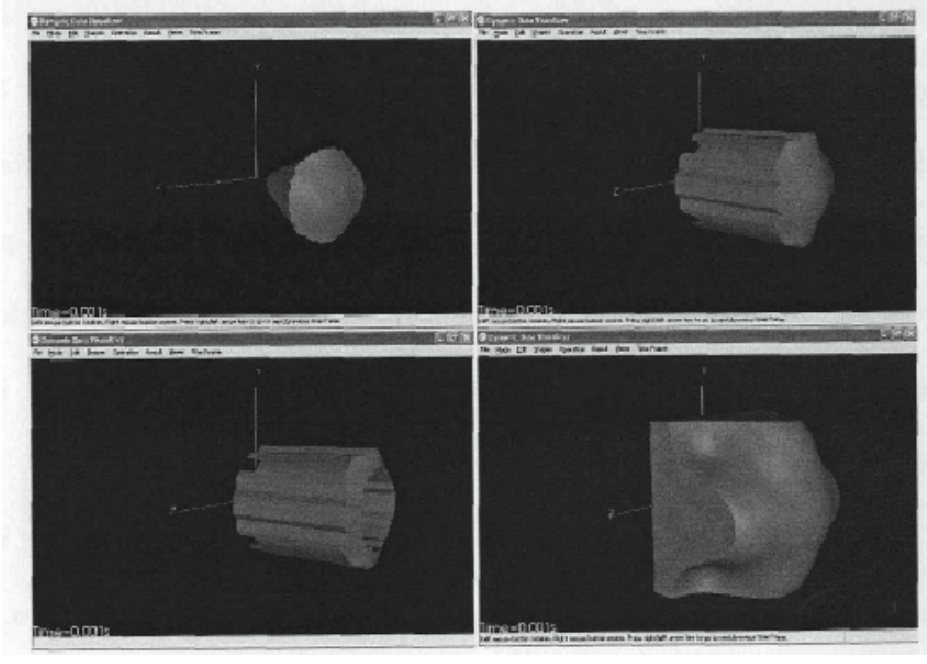


Figure 5.13: Geometric operations: Intersection

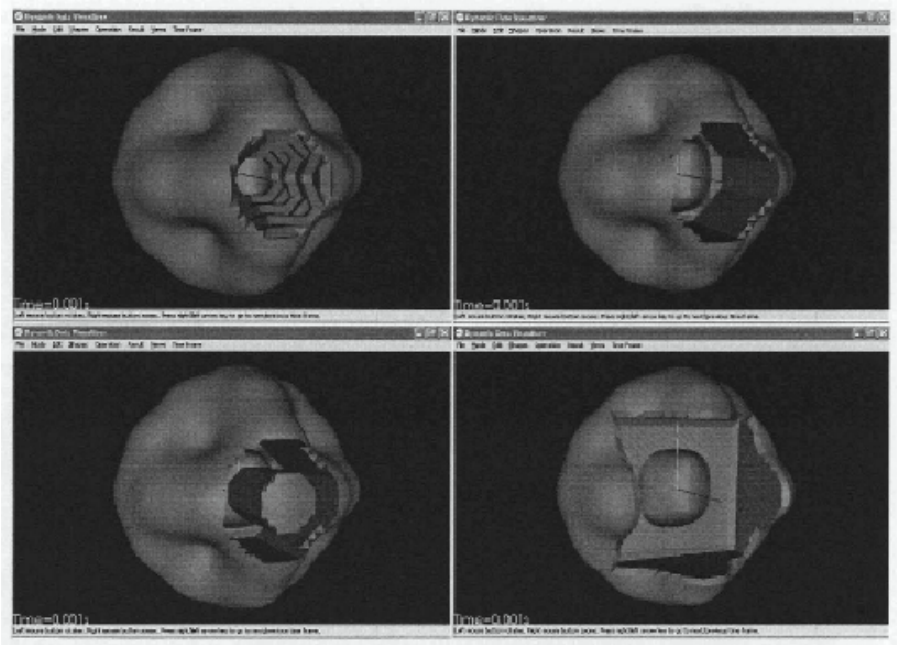


Figure 5.14: Geometric operations: Subtraction

are shown. With the proposed query model, users could identify the shapes of clusters or the shapes of certain parts of clusters based on visual inspection, then, they can find the specific data points inside them by applying geometric querying. This function provides an efficient way to investigate geometric characteristics of a particular part of data set. Moreover, the user can apply geometric operations, such as cut off a part of cluster with certain shape, on clustering results to fit their interests. In Fig.5.15, the querying result with a cylinder query solid is shown. Geometric objects can be drawn opaque or transparent.

In this chapter, based on the proposed implicit model, Clustering visualization system is designed and implemented in C++/ MFC/ VTK libraries. we have demonstrated the functions of our visualization system. It provides users a clear image of clustering results especially by showing the clusters' boundaries. More-

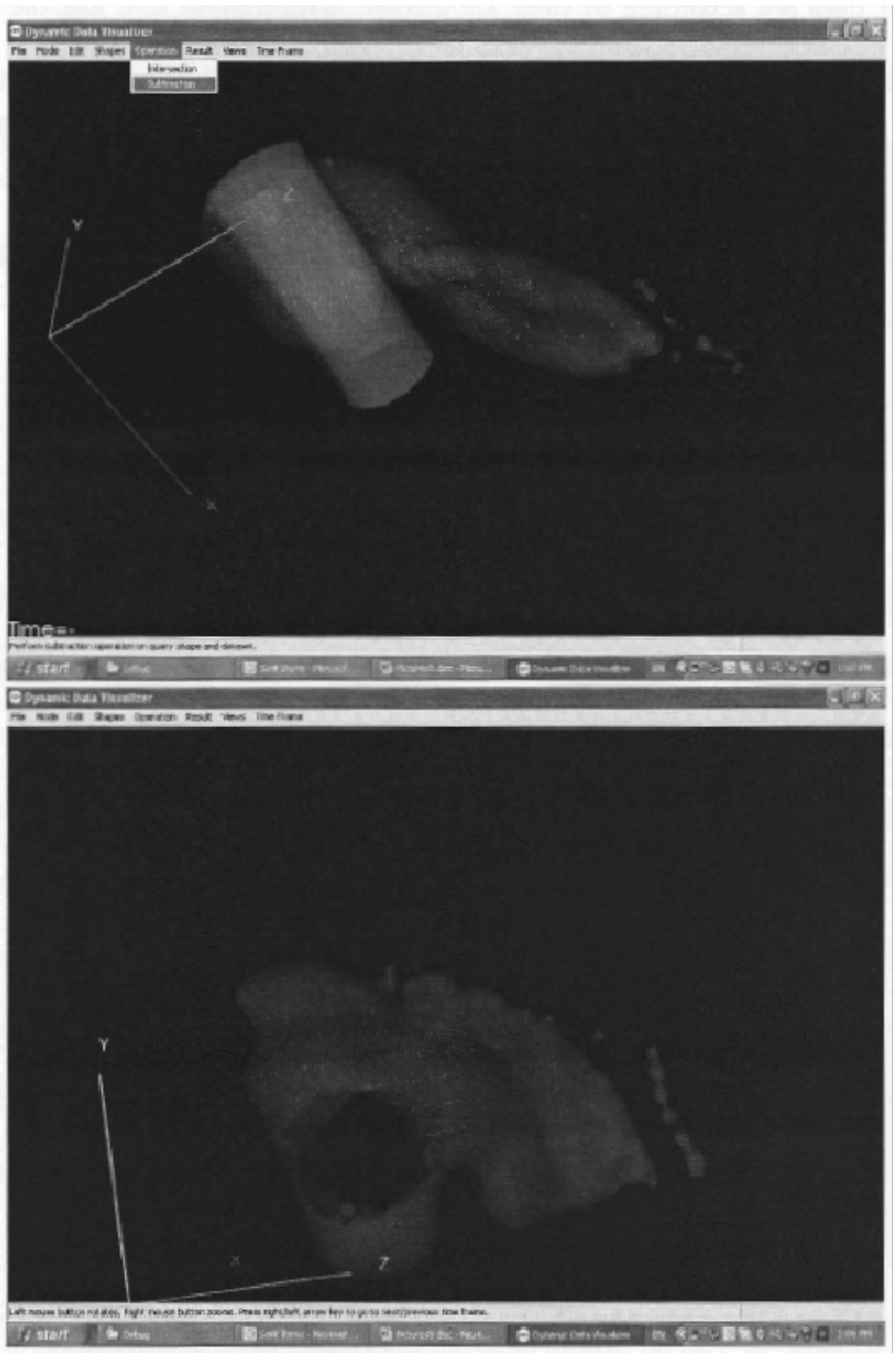


Figure 5.15: Geometric query result by using a cylinder query solid

over, the interactive visual clustering function gives users another options to cluster their data set according to their knowledge. This system also has the potential to be naturally integrated with virtual reality tools [152] which have been proved very useful for many real world applications.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we have introduced our work in the field of spatial data clustering. By the theoretical analysis and experimental tests, the capacity of our algorithms to deal with complex data set automatically has been demonstrated. The proposed algorithms fulfill the requirements for good spatial data clustering methods designed for general purpose. These requirements include the ability to detect clusters with irregular shapes, ability to discovery outliers, ability to achieve good result without user input parameters and preset assumptions, robustness to noises, easiness to integrate with visualization system, ability to handle large data set, and insensitivity to the order of data input. Moreover, the work of this thesis has contributed the clustering techniques in the following two aspects.

First, we considered complicate data distribution cases when data density are different not only between clusters but also inside the same cluster, especially when data density varies gradually inside clusters. Data sets with this kind of clusters widely exist in real world applications. The testing data sets we designed for this

situation can simulate many general cases of real world data. The two types of algorithms we developed in Chapter 3 and 4 are capable to handle these data sets efficiently and effectively including discovering outliers under both global and local views.

Second, we integrated boundary detection in clustering by building clusters' boundaries during the clustering process. It is not an additional procedure which would cost more in algorithm complexity but a natural function of our methods by applying geometric model and proximity building method. The built boundary provides the advantages not only in sound interpretability of our algorithms by integrating visualization but also in the efficiency of dealing with dynamic data sets.

In despite of the similarities which have been described above, there are also differences between our two clustering approaches.

The algorithms proposed in Chapter 3 are designed based on study on connectivity of data points, and this connectivity is described and measured by specially constructed adaptive functions. The clustering procedure is executed according to threshold value of the field function. The introduced adaptive parameters of influence functions are employed to localize the field function based on both local and global data distribution. The boundary built based on these influence function is similar to the definition of surface of solid in computer graphics so that our algorithms can be naturally integrated with visualization system. Since the connectivity between data points is the most concern, our algorithms can deal with the noises under global view which means the noises are relatively far from

clusters.

The algorithm proposed in Chapter 4 is developed based on the investigation on the statistical features of data point inside its proximity and of the whole dataset. The proximity for each data point is built by the graph generated from Delaunary Triangulation. The clustering process is executed according to the criteria function threshold based on which all data points are classified into inner cluster points and boundary points so that the multiple dimensional clustering problem is converted to one dimensional classification problem. The boundary generated by this algorithm includes just the data points but not the data space like the in algorithms described in Chapter 3. Since statistical features which reflect data distribution are employed, this algorithm can deal with the "short bridge" problem. This means that not only the noises can be found under global view but also wrong connections formed by noises between clusters can be discovered.

Besides the clustering algorithms development for 2D cases, the visualization system is also implemented in our work. It not only provides the users with clearer and more intuitive understanding of data set, but also allows the users to be involved into the clustering process so that their knowledge and activity can be utilized to improve the clustering results. Due to our applications, implementations of our algorithms are mainly for 2-dimensional problems, especially for Triangulation-based algorithm.

6.2 Future Work

Although theoretical study of our clustering algorithms has been done in this thesis, when we apply them to real world applications, there are still some issues to be considered.

First, the parameters should be adjusted according to the meaning of real world application, such as the value of piecewise function parameter a for ADACLUS. Based on the knowledge and requirements from applications, the threshold setting method such as the value setting of threshold th for TRICLUST could also be improved. More sophisticated method like density estimation method could be employed so that our algorithms can work more effectively.

Second, research on possible extension of our methods to 3D and high dimensional clustering problem could be done in the future. There are also issues of algorithm efficiency and effectiveness to be considered in such cases. Since high dimensional data are always sparse, the parameters setting methods for our algorithms need to be improved to estimate high dimensional data more effectively. Moreover, when dimensionality increases, the complexity of our algorithms will increase rapidly due to the hill-climb method in the case of ADACLUS and due to triangulation method in the case of TRICLUST. This means more efficient searching and proximity building techniques is needed.

Third, the implementation of our algorithms could be improved even further by employing some advance data structure and by implementing under other software platforms. Currently, our algorithms are mainly implemented in Matlab except for

the clustering visualization system. It could be more efficient to implement the algorithms in C++ or other programming language for real world applications. Moreover, it would be much better to develop the clustering and visualization methods as an entire software with interactive GUI and database interface, even with virtual reality devices. It will give our system a good prospect in applications such as geo-spatial data processing. The author was invited to contribute to X3D Earth project that is an initiative aimed at building a standards-based 3D Earth usable by governments, industry, scientists, academia and the general public. X3D mappings of world terrain, cartography and imagery will be made available for use in any scene, making it easy to geospatially reference.

The outcome of our work will be used in the environmental research for spatial and temporal analysis of large-scale data sets originating from satellite imagery and ground-based sensors.

Publication List

[1] Olga Sourina and Liu Dongquan, "Geometric Approach to Clustering and Querying in Databases and Warehouses," in *Proc. of Cyberworlds 2003*, Singapore, 2003, pp. 326–333.

[2] Liu Dongquan and Olga Sourina, "A Solid-based Clustering Method with Implicit Functions," in *Proc. of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS)*, Singapore, 2003, pp. ps07–5–04.

[3] Olga Sourina and Dongquan Liu, "Visual Interactive 3-dimensional Clustering with Implicit Functions," in *Proc. of IEEE Conference on Cybernetics and Intelligent Systems (CIS) 2004*, 2004, pp. 382–386.

[4] Dongquan Liu and Olga Sourina, "Free-parameters Clustering of Spatial Data with Non-uniform Density," in *Proc. of IEEE Conference on Cybernetics and Intelligent Systems (CIS) 2004*, 2004, pp. 387–392.

[5] Olga Sourina and Dongquan Liu, "Visual Interactive Clustering and Querying of Spatio-Temporal Data," in *Proc. of the International Conference on Computational Science and its Applications (ICCSA). O. Gervasi et al.*

(Eds), 2005, pp. 968–977.

[6] Olga Sourina and Dongquan Liu, "Visual Interactive 3-dimensional Clustering," *Computer Graphics & Geometry*, vol. 7, no. 1, 2005.

[7] Olga Sourina, Dongquan. Liu and Gleb V. Nosovskiy, "Visual Clustering and Boundary Detection of Time-Dependent Datasets," in *Proc. of IEEE 2007 International Conference on Cyberworlds*, Germany, 2007, pp. 54–60.

[8] Gleb V. Nosovskiy, Dongquan Liu, and Olga Sourina, "Automatic Clustering & Boundary Detection Algorithm Based on Adaptive Influence Function," *Pattern Recognition*, vol. 29/9, pp. 1261–1273, 2008.

[9] Dongquan Liu, Gleb V. Nosovskiy, and Olga Sourina, "Automatic Clustering & Boundary Detection Algorithm Based on Delaunay Triangulation," *Pattern Recognition Letters*, vol. 41/9, pp. 2757–2776, 2008.

Bibliography

[1] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3-D Graphics*, Prentice Hall, 2nd edition, 1998.

[2] A. Hinneburg and D.A. Keim, "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," in *Proc. of the Fourth International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 58–65.

[3] H. A. Edelstein, *Introduction to Data Mining and Knowledge Discovery*, Two Crows Corporation, 3rd edition, 1999.

[4] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 1st edition, 2000.

[5] T. Y. Lin, S. Ohsuga, C. Liau, and X. Hu, *Foundations and Novel Approaches in Data Mining*, Springer, 2005.

[6] P. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Addison Wesley, 2006.

[7] D. T. Larose, *Data Mining Methods and Models*, John Wiley & Sons Inc, 2006.

[8] S. K. Halgamuge and L. Wang, *Classification and Clustering for Knowledge Discovery*, Springer, 2005.

[9] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining (Adaptive Computation and Machine Learning)*, The MIT Press, 2001.

[10] M. Kantardzic, *Data Mining: Concepts, Models, Methods, and Algorithms*, John Wiley & Sons Inc, 2002.

[11] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*, Springer-Verlag New York Inc., 2005.

[12] D. T. Larose, *Discovering Knowledge in Data: an Introduction to Data Mining*, John Wiley & Sons Inc, 2004.

[13] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," in *Proc. of the 20th Int. Conf. Very Large Data Bases, VLDB*, 1994, pp. 487–499.

[14] D. Wai-Lok Cheung, S. D. Lee, and B. Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," in *Proc. of the Fifth International Conference On Database Systems For Advanced Applications*, 1997, pp. 185–194.

[15] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, 2nd edition, 2000.

[16] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley-Interscience, 2rev ed edition, 2005.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[18] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "Clustering Algorithms and Validity Measures," in *Proc. of the Thirteenth International Conference on Scientific and Statistical Database Management*, Washington, DC, USA, 2001, pp. 3–22.

[19] P. Berkhin, "Survey Of Clustering Data Mining Techniques," Tech. Rep., Accrue Software, San Jose, CA, 2002, http://citeseer.nj.nec.com/berkhin02survey.html.

[20] O. R.Zaïane, A.Foss, C.-H. Lee, and W. Wang, "Data Clustering Analysis - from Simple Groupings to Scalable Clustering with Constraints," Tech. Rep., Department of Computing Science, University of Alberta, Alberta, Canada, 2002.

[21] R. Xu and II. D. Wunsch, "Survey of Clustering Algorithms," *IEEE Transactions on Neural Networks*, vol. 16, pp. 645 – 678, 2005.

[22] E. Kolatch, "Clustering Algorithms for Spatial Databases: A Survey," Tech. Rep., University of Maryland, 2001, http://www.cs.umd.edu/ kolatch/index.html.

[23] T. Bailey and T. Gatrell, *Interactive Spatial Data Analysis*, Prentice Hall, 1996.

[24] M. H. Dunham, *Data Mining: Introductory and Advanced Topics*, Prentice Hall, 2002.

[25] R. T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," in *Proc. of the 20th International Conference on Very Large Data Bases VLDB 94*, Santiago, Chile, Sept 1994, pp. 144–155.

[26] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," in *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 1998, pp. 73–84, ACM Press.

[27] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an Efficient Data clustering method for very large databases," in *Proc. of the 1996 ACM SIGMOD international conference on Management of data*, 1996, pp. 103–114.

[28] V. Estivill-Castro and I. Lee, "AMOEBA: Hierarchical Clustering Based on Spatial Proximity Using Delaunay Diagram," in *Proc. of the 9th International Symposium on Spatial Data Handling*, 2000, pp. 7a.26–7a.41.

[29] V. Estivill-Castro and I. Lee, "AUTOCLUST: Automatic Clustering via Boundary Extraction for Mining Massive Point-Data Sets," in *Proc. of the 5th International Conference on Geocomputation*, 2000.

[30] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. of the 5th Berkeley Symp. Math. Statist. Prob.*, 1967, pp. 281–297.

[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[32] H. Pfister and M. Gross, "Point-Based Computer Graphics," *IEEE Computer Graphics and Applications*, vol. 4, pp. 22–23, 2004.

[33] E. Gobbetti and F. Marton, "Layered Point Clouds," in *Proc. of Eurographics Symposium on Point-Based Graphics*, Switzerland, Jun 2004, pp. 113–120.

[34] S. Openshaw, "Two Exploratory Space-time Attribute Pattern Analysers Relevant to GIS," in *Book named Spatial Analysis and GIS*, S. Fotheringham and P.Rogerson, Eds. 1994, pp. 83–104, Taylor & Francis, London.

[35] G. Nagy, "State of the Art in Pattern Recognition," *Proceedings of the IEEE*, vol. 56, pp. 836– 863, 1968.

[36] I. Lee and V. Estivill-Castro, "Polygonization of Point Clusters through Cluster Boundary Extraction for Geographical Data Mining," in *Proc. of the 10th International Symposium on Spatial Data Handling SDH*, Ottawa, Canada, Jul 2002, pp. 27–40.

[37] O. Sourina and Dongquan Liu, "Geometric Approach to Clustering and Querying in Databases and Warehouses," in *Proc. of Cyberworlds 2003*, Dec 2003, pp. 326–333.

[38] U. M. Fayyad, G. G. Grinstein, and A. Wierse, *Information Visualization in Data Mining and Knowledge Discovery*, Morgan Kaufmann Publishers Inc,US, 2001.

[39] T. C. Sprenger, M. H. Gross, A. Eggenberger, and M. Kaufmann, "A Framework for Physically-Based Information Visualization," in *Proc. of Eurographics Workshop on Visualization '97*, Boulogne sur Mer, France, 1997, pp. 77–86.

[40] D. A. Keim, "Information Visualization and Visual Data Mining," *IEEE Trans. on Visualization and Computer Graphics*, vol. 8, no. 1, pp. 1–8, 2002.

[41] C.R. Rao, E. J. Wegman, and J. L. Solka, *Handbook of Statistics, Volume 24: Data Mining and Data Visualization*, North Holland, 2005.

[42] A. Hinneburg, D. A. Keim, and M. Wawryniuk, "HD-Eye: Visual Mining of High-Dimensional Data," *IEEE Comput. Graph. Appl.*, vol. 19, no. 5, pp. 22–31, 1999.

[43] C. M. Gold, "Problems with Handling Spatial Data-The Voronoi Approach," *CISM Journal ACSGC*, vol. 45, pp. 65–80, 1991.

[44] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, John Wiley & Sons, 2nd edition, 2000.

[45] Lee D.T. and B.J. Schacter, "Two Algorithms for Constructing a Delaunay Triangulation," *International Journal of Computer and Information Sciences*, vol. 3, pp. 219–241, 1980.

[46] Yan Huang and Pusheng Zhang, "On the Relationships between Clustering and Spatial Co-location Pattern Mining," in *Proc. of 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, 2006, pp. 513–522.

[47] Vladimir Estivill-castro and Ickjai Lee, "Data Mining Techniques for Autonomous Exploration of Large Volumes of Geo-referenced Crime Data," in *Proc. of the 6th International Conference on Geocomputation*, 2001, pp. 783–789.

[48] Zhiwen Yu and Hau san Wong, "An Efficient Local Clustering Approach for Simplification of 3D Point-based Computer Graphics Models," *Multimedia and Expo, 2006 IEEE International Conference on*, pp. 2065–2068, Jul 2006.

[49] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva, "Point Set Surfaces," in *VIS '01: Proc. of the conference on Visualization '01*, Washington, DC, USA, 2001, pp. 21–28.

[50] R. H. Güting, "An Introduction to Spatial Database Systems," *The VLDB Journal*, vol. 3, pp. 357 – 399, 1994.

[51] P. Rigaux, M. Scholl, and A. Voisard, *Spatial Databases: With Application to GIS*, Morgan Kaufmann, 2nd edition, 2001.

[52] U. M. Fayad, *Advances in Knowledge Discovery in Databases*, MIT Press, 1996.

[53] J. F. Roddick and K. Hornsby, Eds., *Temporal, Spatial, and Spatio-Temporal Data Mining*, Springer, 1st edition, 2001.

[54] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical Pattern Recognition: A Review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[55] T. C. Sprenger, R. Brunella, and M. H. Gross, "H-BLOB: A Hierarchical Visual Clustering Method Using Implicit Surfaces," in *Proc. of the 11th IEEE Visualization 2000 Conference VIS 2000*, Washington, DC, USA, 2000, IEEE Computer Society.

[56] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On Clustering Validation Techniques," *J. Intell. Inf. Syst.*, vol. 17, no. 2-3, pp. 107–145, 2001.

[57] A. Hinneburg, *Mining for High Dimensional Clusters using Projections and Visualizations*, Ph.D. thesis, Institute for Informatics,Martin-Luther University, Halle/Saale, Germany, 2000.

[58] J. HARTIGAN and M. WONG, "Algorithm AS136: A K-means Clustering Algorithm," *Applied Statistics*, vol. 28, pp. 100*108, 1979.

[59] M.C. Su and C.H. Chou, "A Modified Version of the K-Means Algorithm with a Distance Based on Cluster Symmetry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 674–680, 2001.

[60] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained K-means Clustering with Background Knowledge," in *Proc. of the Eighteenth International Conference on Machine Learning*, San Francisco, CA, USA, 2001, pp. 577–584, Morgan Kaufmann Publishers Inc.

[61] M. Xu and P. Fränti, "A Heuristic k-means Clustering Algorithm by Kernel PCA," in *Proc. of IEEE Int. Conf. on Image Processing (ICIP'04)*, 2004, pp. 3503–3506.

[62] S. Cherednichenko, V. Hautamäki, T. Kinnunen, I. Kärkkäinen, and P. Fränti, "Improving k-means by Outlier Removal," in *Proc. of Scandinavian Conf. on Image Analysis (SCIA'05)*, 2005, pp. 978–987.

[63] J. Yu, "General C-Means Clustering Model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1197–1211, 2005.

[64] H. Brás Silva, P. Brito, and J. P. da Costa, "A Partitional Clustering Algorithm Validated by A Clustering Tendency Index Based on Graph Theory," *Pattern Recognition*, vol. 39, pp. 776–788, 2006.

[65] C. Ordonez, "Integrating K-Means Clustering with a Relational DBMS Using SQL," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 2, pp. 188–201, 2006.

[66] Z. S. H. Chan, L. Collins, and N. K. Kasabov, "An Efficient Greedy K-means Algorithm for Global Gene Trajectory Clustering," *Expert Syst. Appl.*, vol. 30, pp. 137–141, 2006.

[67] F. Murtagh, *Multidimensional Clustering Algorithms*, Compstat Lectures, Vienna: Physika Verlag, 1985.

[68] C. Olson, "Parallel Algorithms for Hierarchial Clustering," *Parallel Computing*, vol. 21, pp. 1331–1325, 1995.

[69] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical Clustering Using Dynamic Modeling," *IEEE Computer*, vol. 32, no. 8, pp. 68–75, Aug 1999.

[70] R. A. Mollineda and E. Vidal, "A Relative Approach To Hierarchical Clustering," in *Pattern Recognition and Applications, Frontiers in Artificial Intelligence and Applications*. 2000, pp. 19–28, IOS Press.

[71] E. Dahlhaus, "Parallel Algorithms for Hierarchical Clustering and Applications ot Split Decomposition and Parity Graph Recognition," *J. Algorithms*, vol. 36, pp. 205–240, 2000.

[72] C. Li and G. Biswas, "Unsupervised Learning with Mixed Numeric and Nominal Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 4, pp. 673–690, 2002.

[73] S. H. Joshi, W. Mio, A. Srivastava, and Xiuwen Liu, "Statistical Shape Analysis: Clustering, Learning, and Testing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 4, pp. 590–602, 2005.

[74] P. Fränti, O. Virmajoki, and V. Hautamäki, "Fast Agglomerative Clustering Using a k-Nearest Neighbor Graph," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 11, pp. 1875–1881, 2006.

[75] P. Fränti and O. Virmajoki, "Iterative Shrinking Method for Clustering Problems," *Pattern Recognition*, vol. 39, pp. 761 C 775, 2006.

[76] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "Density-Connected Sets and their Application for Trend Detection in Spatial Databases," in *Proc. of the Third International Conference on Knowledge Discovery and Data Mining*, 1997, pp. 10–15.

[77] M. Ankerst, Markus M.Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," in *Proc. of ACM SIGMOD Int. Conf. on Management of Data SIGMOD'99*, 1999, pp. 49–60.

[78] X. Xu, M. Ester, H. Kriegel, and J. Sander, "A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases," in *Proc. of the Fourteenth International Conference on Data Engineering*, Washington, DC, USA, 1998, pp. 324–331, IEEE Computer Society.

[79] K. Fukunaga and L. Hostetler, "The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition," *IEEE Trans. on Information Theory*, vol. 21, pp. 32– 40, 1975.

[80] F. KOWALEWSKI, "A Gradient Procedure for Determining Clusters of Relatively High Point Density," *Pattern Recognition*, vol. 28, no. 12, pp. 1973–1984, 1995.

[81] D. Birant and A. Kut, "ST-DBSCAN: An Algorithm for Clustering Spatial-temporal Data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[82] L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A Local-density Based Spatial Clustering Algorithm with Noise," *Information Systems*, vol. 32, pp. 978–986, 2006.

[83] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," in *Proc. of the 1998 ACM SIGMOD International Conference on Management of Data*, New York, NY, USA, 1998, pp. 94–105, ACM Press.

[84] W. Wang, J. Yang, and R. Muntz, "STING: A Statistical Information Grid Approach to Spatial Data Mining," in *Proc. of the 23rd VLDB Conference*, 1997, pp. 186–195.

[85] G. Sheikholeslami, S. Chatterjee, and A. Zhang, "WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases," in *Proc. of the 24th Int. Conf. Very Large Data Bases, VLDB*, 1998, pp. 428–439.

[86] C. T. Zahn, "Graph-theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Trans. on Computers*, vol. 20, no. 1, pp. 68–86, 1971.

[87] C. Eldershaw and M. Hegland, "Cluster Analysis using Triangulation," in *Proc. of Computational Techniques and Applications: CTAC97*, Singapore, 1997, pp. 201–208.

[88] I. Kang, T. Kim, and K. Li, "A Spatial Data Mining Method by Delaunay Triangulation," in *Proc. of the 5th ACM international workshop on Advances in geographic information systems*, New York, NY, USA, 1997, pp. 35–39, ACM Press.

[89] S. Hader and F.A. Hamprecht, "Efficient density clustering using basin spanning trees," in *Proc. of the 26th Annual Conference of the Gesellschaft für Klassifikation (GfK1)*, 2003, pp. 39–48.

[90] V. Estivill-Castro and I. Lee, "AUTOCLUST+: Automatic Clustering of Point-Data Sets in the Presence of Obstacles," in *Proc. of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining-Revised Papers*, London, UK, 2001, pp. 133–146, Springer-Verlag.

[91] J. Cherng and M. Lo, "A Hypergraph Based Clustering Algorithm for Spatial Data Sets," in *Proc. of the 2001 IEEE International Conference on Data Mining*, Washington, DC, USA, 2001, pp. 83–90, IEEE Computer Society.

[92] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering Gene Expression Patterns," *Journal of Computational Biology*, vol. 6, no. 3/4, pp. 281–297, 1999.

[93] R. Sharan and R. Shamir, "CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis," in *Proc. of the Eighth International*

Conference on Intelligent Systems for Molecular Biology (ISMB). 2000, pp. 307–316, AAAI Press, Menlo Park, CA.

[94] Y. Li, "A Clustering Algorithm Based on Maximal θ-distant Subtrees ," *Pattern Recognition*, 2006, In press.

[95] C. Fraley and A.E. Raftery, "Model-Based Clustering, Discriminant Analysis, and Density Estimation," *Journal of the American Statistical Association*, vol. 97, pp. 611–631, Jun 2002.

[96] C. Fraley and A.E. Raftery, "MCLUST: Software for Model-Based Clustering, Density Estimation and Discriminant Analysis," Tech. Rep. 415, Department of Statistics, University of Washington, 2002.

[97] C. Fraley and A. Raftery, "Mclust: Software for Model-based Cluster Analysis," *Journal of Classification*, vol. 16, pp. 297–306, 1999.

[98] C. Pizzuti and D. Talia, "P-AutoClass: Scalable Parallel Clustering for Mining Large Data Sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 3, pp. 629–641, 2003.

[99] G. J. McLachlan and T. Krishnan, *The EM Algorithm and Extensions*, Wiley-Interscience, 1st edition, 1996.

[100] I. Kärkkäinen and P. Fränti, "Gradual Model Generator for Single-pass Clustering," in *Proc. of IEEE Int. Conf. on Data Mining (ICDM'05)*, Houston, Texas, 2005, pp. 681–684.

[101] P. Fränti and J. Kivijärvi, "Randomized Local Search Algorithm for the Clustering Problem," *Pattern Analysis and Applications*, vol. 3, pp. 358–369, 2000.

[102] L. O. Hall, I. B. Özyurt, and J. C. Bezdek, "Clustering with a Genetically Optimized Approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 2, pp. 103–112, 1999.

[103] G. Patanè and M. Russo, "The Enhanced LBG Algorithm," *Neural Netw.*, vol. 14, no. 9, pp. 1219–1237, 2001.

[104] M. Laszlo and S. Mukherjee, "A Genetic Algorithm Using Hyper-Quadtrees for Low-Dimensional K-means Clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 533, 2006.

[105] P. Fränti, J. Kivijärvi, T. Kaukoranta, and O. Nevalainen, "Genetic Algorithms for Large Scale Clustering Problem," *The Computer Journal*, vol. 40, pp. 547–554, 1997.

[106] J. Kivijärvi, P. Fränti, and O. Nevalainen, "Self-adaptive Genetic Algorithm for Clustering," *Journal of Heuristics*, vol. 9, pp. 113–129, 2003.

[107] T. Kohonen, "The Self-organizing Map," *Proceedings of the IEEE*, vol. 9, pp. 1464–1479, 1990.

[108] Y.J. Zhang and Z.Q. Liu, "Self-Splitting Competitive Learning: A New Online Clustering Paradigm," *IEEE Trans. Neural Networks*, vol. 13, pp. 369–380, 2002.

[109] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by An Adaptive Resonance System," *Neural Netw.*, vol. 4, no. 6, pp. 759–771, 1991.

[110] N. Kasabov and R. Kozma, *Neuro-Fuzzy Techniques for Intelligent Information Systems*, Springer-Verlag Telos, 1999.

[111] D. Deng and N. K. Kasabov, "On-line Pattern Analysis by Evolving Self-organizing Maps," *Neurocomputing*, vol. 51, pp. 87–103, 2003.

[112] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Kluwer Academic Publishers, Norwell, MA, USA, 1981.

[113] S. Mitra, H. Banka, and W. Pedrycz, "RoughCFuzzy Collaborative Clustering," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 36, pp. 795– 805, 2006.

[114] N. K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering (Computational Intelligence)*, The MIT Press, 1996.

[115] E. Ka Ka Ng, A. W. Fu, and R. C. W. Wong, "Projective Clustering by Histograms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 369–383, 2005.

[116] C. C. Aggarwal and P. S. Yu, "Redefining Clustering for High-Dimensional Applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 2, pp. 210–225, 2002.

[117] Q. Song and N. Kasabov, "ECM - A Novel On-line, Evolving Clustering Method and Its Applications," in *Proc. of the Fifth Biannual Conference on Artificial Neural Networks and Expert Systems (ANNES2001)*, 2001, pp. 87–92.

[118] D. Deng and N. Kasabov, "ESOM: An Algorithm to Evolve Self-Organizing Maps from On-Line Data Streams," in *Proc. of the IJCNN'2000 on Neural Networks Neural Computing: New Challenges and Perspectives for the New Millennium*, 2000, vol. VI, pp. 24–27.

[119] A. L. N. Fred and A. K. Jain, "Combining Multiple Clusterings Using Evidence Accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, 2005.

[120] C. R. Lin and K. H. Liu, "Dual Clustering: Integrating Data Clustering over Optimization and Constraint Domains," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 5, pp. 628–637, 2005.

[121] R. Nock and F. Nielsen, "On Weighting Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1223–1235, 2006.

[122] S. Bandyopadhyay, "Simulated Annealing Using a Reversible Jump Markov Chain Monte Carlo Algorithm for Fuzzy Clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 479–490, 2005.

[123] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

[124] N.R. Pal, K.Pal, J.M. Keller, and J.C. Bezdek, "A Possibilistic Fuzzy c-Means Clustering Algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 13, pp. 517– 530, 2005.

[125] C. R. Lin and M. S. Chen, "Combining Partitional and Hierarchical Algorithms for Robust and Efficient Data Clustering with Cohesion Self-Merging," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 145–159, 2005.

[126] M. S. Yang and K. L. Wu, "A Similarity-Based Robust Clustering Method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 434–448, 2004.

[127] M. Dash, H. Liu, and X. Xu, "'1 + 1>2': Merging Distance and Density Based Clustering," in *Proc. of the 7th International Conference on Database Systems for Advanced Applications*, 2001, pp. 32–39.

[128] O. Virmajoki, P. Fränti, and T. Kaukoranta, "Practical Methods for Speeding-up the Pairwise Nearest Neighbor Method," *Opt. Eng.*, vol. 40, pp. 2495–2504, 2001.

[129] A. Watt, *Fundamentals of Three-dimensional Computer Graphics*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[130] D. Hearn and M. P. Baker, *Computer Graphics*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986.

[131] V. Shapiro, "Real Functions for Representation of Rigid Solids," *Comput. Aided Geom. Des.*, vol. 11, no. 2, pp. 153–175, 1994.

[132] C. M. Hoffmann, *Geometric and Solid Modeling: An Introduction*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[133] G. Turk, H. Q. Dinh, J. F. O'Brien, and G. Yngve, "Implicit Surfaces that Interpolate," in *Proc. of the seventh International Conference on Shape Modeling and Applications*, Italy, 2001, pp. 62–71.

[134] L. Velho, L. H. de Figueiredo, and J. Anthony Gomes, *Implicit Objects in Computer Graphics*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[135] J. Bloomenthal and B. Wyvill, Eds., *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[136] J. F. Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Trans. Graph.*, vol. 1, no. 3, pp. 235–256, 1982.

[137] J. Menon, "An Introduction to Implicit Techniques," SIGGRAPH Course Notes on Implicit Surfaces for Geometric Modeling and Computer Graphics, 1996.

[138] K. Chen and L. Liu, "iVIBRATE: Interactive Visualization-based Framework for Clustering Large Datasets," *ACM Trans. Inf. Syst.*, vol. 24, no. 2, pp. 245–294, 2006.

[139] A. Adamson and M. Alexa, "Approximating Bounded, Non-orientable Surfaces from Points," in *Proc. of Shape Modeling International 2004*, 2004, pp. 243–252.

[140] M. Andersson, J. Giesen, M. Pauly, and B. Speckmann, "Bounds on the k-Neighborhood for Locally Uniformly Sampled Surfaces," in *Proc. of the 1st Symposium on Point-Based Graphics*, 2004, pp. 167–171.

[141] M. Pauly, M. Gross, and L. Kobbelt, "Efficient Simplification of Point-Sampled Surfaces," in *Proc. of the conference on Visualization '02*, 2002, pp. 163–170.

[142] O. Sourina and D. Liu, "Visual Interactive 3-Dimensional Clustering with Implicit Functions," in *Proc. of IEEE CIS 2004*, 2004, pp. 382–386.

[143] N. N. Golovanov, A.T. Fomenko, D.P. Il'ytko, and G.V. Nosovskiy, *Computer Geometry*, Moscow: ACADEMIA, 2006.

[144] S. Karlin, *A First Course in Stochastic Processes*, NY:London, Academic Press, 1968.

[145] K. Matthes, J. Kerstan, and J. Mecke, *Infinitely Divisible Point Processes*, NY: Akademie-Verlag/John Willey & Sons, 1978.

[146] V. Estivill-Castro and I. Lee, "Multi-Level Clustering and its Visualization for Exploratory Spatial Analysis," *Geoinformatica*, vol. 6, no. 2, pp. 123–152, 2002.

[147] O. Sourina and S. H. Boey, "Geometric Query Types for Data Retrieval in Relational Databases," *Data Knowl. Eng.*, vol. 27, pp. 207–229, 1998.

[148] P. Bourke, "Implicit Surfaces," Tech. Rep., University of Western Australia, 1997, http://local.wasp.uwa.edu.au/ pbourke/modelling/implicitsurf/.

[149] R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," *ACM Trans. Database Syst.*, vol. 25, pp. 1–42, 2000.

[150] M. Erwig and M. Schneider, "Developments in Spatio-Temporal Query Languages," in *Proc. of the 10th International Workshop on Database & Expert Systems Applications*, Washington, DC, USA, 1999, pp. 441–449.

[151] A. Ricci, "A Constructive Geometry for Computer Graphics," *Computer Journal*, vol. 16, pp. 157–160, 1973.

[152] J. Vince, *Essential Virtual Reality Fast: How to Understand the Techniques and Potential of Virtual Reality*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[153] W. E. Lorensen and H. E. Cline, "Marching cubes: A High Resolution 3D Surface Construction Algorithm," in *Proc. of the 14th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1987, pp. 163–169.

[154] T. Disz, "Introduction to Scientific Visualization with VTK," Tech. Rep., Mathematics and Computer Science Division, Argonne National Laboratory, 1998, http://www-unix.mcs.anl.gov/ disz/cs-341/colorvis/index.htm.

Appendix A

Visualization algorithm: Marching Cubes [1]

To visualize the solid (object) we built by implicit functions, we need to apply visualization algorithm on our model. The latest one is called Marching Cubes algorithm.

Marching Cubes algorithm invented by [153] in 1987 is used in volume rendering in computer graphics. It produces a triangle mesh by computing iso-surfaces from discrete data. By connecting the patches from all cubes on the iso-surface boundary, a surface representation is built.

The basic principle behind the Marching Cubes algorithm is to divide data space into a series of small cubes. Then, the algorithm instructs us to 'march' through each of the cubes by testing the corner points and replacing the cube with an appropriate set of polygons. The sum total of all polygons generated will build a surface that approximates the one the data set describes.

Marching Cubes algorithm is designed for visualizing $3D$ scalar data which is single valued at each location in a data set, such as density, pressure and temperature. Building iso-surface is one of the most natural ways to visualize scalar data. To explain the working principle of Marching Cubes algorithm, we start with iso-surface building of scalar data set in $2D$ case, which is usually called contouring.

In Fig.A.1[1], an example of contouring a $2D$ structured grid is shown. Scalar values are labeled next to the points that define the grid. The contour line is with value 5. First, we use linear interpolation to generate the points on the contour. Second, we connect those points to build the contour.

There are two approaches to connect points with the same scalar value. One is by detecting the intersections and tracking the movement of the contour across all cell boundaries. Another approach uses a divide and conquer strategy. This is the technique employed by Marching Cubes algorithm.

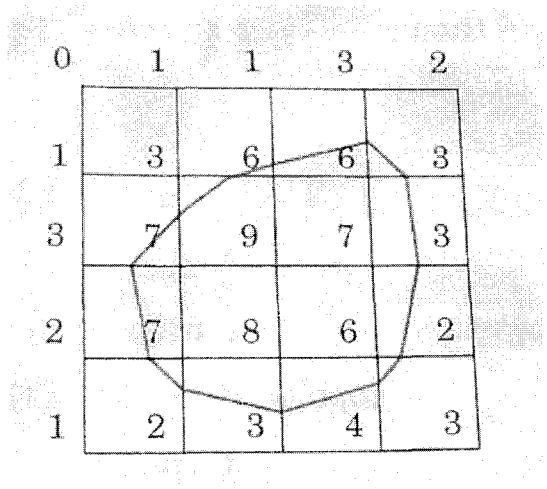


Figure A.1: Contouring a 2D structured grid with contour line value = 5

The basic assumption of Marching Cubes algorithm is that a contour can only pass through a cell[1] in a finite number of ways. A case table is constructed that enumerated all possible topological states of a cell, given combinations of scalar values at the cell points. The number of topological states depends on the number of cell vertices, and the number of inside/ outside relationships a vertex can have with respect to the contour values. A vertex is considered inside a contour if its scalar value is larger than the scalar value of the contour line. Vertices with scalar values less than the contour value are said to be outside the contour.

The marching algorithms proceed as follows [1]:

[1]For simplicity, we an consider a cell as a square in 2D or a cube in 3D

1. Select a cell.
2. Calculate the inside / outside state of each vertex of the cell.
3. Create an index by storing the binary state of each vertex in a separate bit.
4. User the index to look up the topological state of the cell in a case table.
5. Calculate the contour location (via interpolation) for each edge in the case table.

This procedure will construct independent geometric primitives in each cell. At the cell boundaries duplicate vertices and edges may be created. These duplicates can be eliminated by using a special coincident point-merging operation. Note that interpolation along each edge should be done in the same same direction. If not, numerical round-off will likely cause points to be generated that are not precisely coincident, and will not merge properly.

For Marching Cubes for $3D$ cases, there are 256 different combinations of scalar value, given that there are eight points in a cubical cell (i.e., 2^8 combinations). Fig.A.2 shows these combinations reduced to 15 cases by using arguments of symmetry. We use combinations of rotation and mirroring to produce topologically equivalent cases.

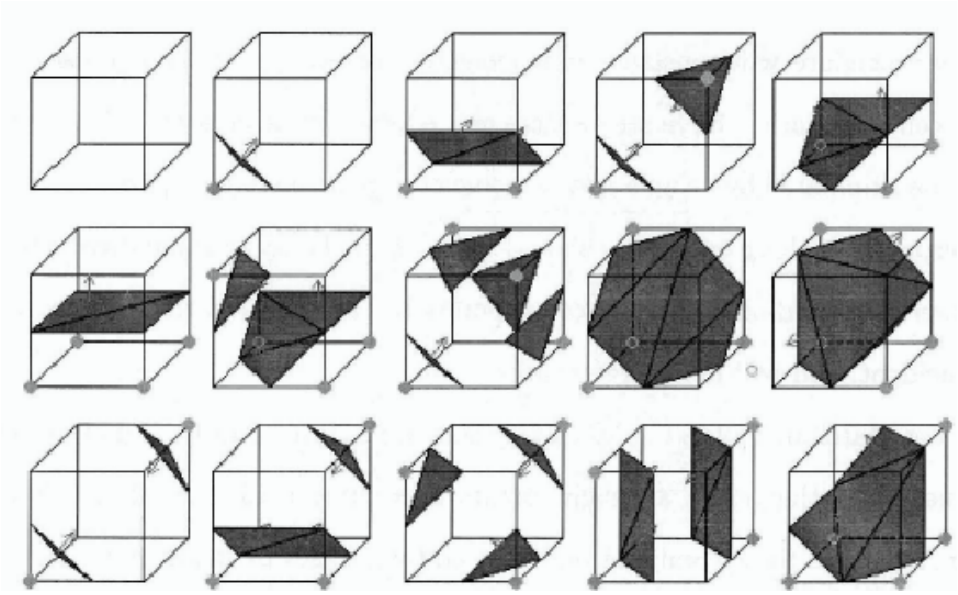


Figure A.2: Marching cubes cases for 3D iso-surface generation. The 256 possible cases have been reduced to 15 cases using symmetry. Green vertices are greater than the selected iso-surface value

Appendix B

Visualization Toolkit

To implement our system, we chose the visualization software system named Visualization Toolkit (VTK) [1] [154].

The Visualization Toolkit (VTK) is an open source, freely available software system for 3D computer graphics and visualization used by thousands of researchers and developers all over the world. VTK consists of a C++ class library, and several interpreted interface layers including Tcl/Tk, Java and Python. The programming language used our project is C++, therefore the C++ class library is the most relevant to this project. VTK supports a wide variety of visualization algorithms including scalar, vector, tensor, texture and volumetric methods; and advanced modelling techniques such as implicit modeling, polygon reduction, mesh smoothing, cutting, contouring, and Delaunay triangulation.

The Marching Cubes algorithm is the default visualization algorithm for implicit modeling in VTK system.

B.1 The Visualization Pipeline [1]

Visualization transforms data into images that efficiently and accurately represent information about the data. Transformation is the process of converting data from its original form into computer images. The pipeline in VTK system consists of objects to represent data (data objects), objects to operate on data (process objects),

an dan indicated direction of data flow (arrow connections between objects). To form a pipeline means to connect data and process objects to form a visualization network.

Data objects represent information. Data objects also provide methods to create, access, and delete this information. Direct modification of the data represented by the data objects is not allowed except through formal object methods.

Process objects operate on input data to generate output data. A process object either drives new data from its inputs, or transforms the input data into a new form. Process objects are further characterized as source objects, filter objects, or mapper objects. This categorization is based on whether the objects initiate, maintain, or terminate visualization data flow.

Source objects interface to external data sources or generate data from local parameters. they don't have any input. Filter objects require one or more input data objects and generate one or more output data objects. Mapper objects correspond to the sinks in the functional model. Mapper objects require one or more input data objects and terminate the visualization pipeline data flow. Usually mapper objects are used to convert data into graphical primitives, but they may write out data to a file or interface with another software system or devices.

B.2 Basic Data Representation in VTK [1]

Data objects in the objects in the visualization pipeline are called datasets. A dataset consists of one or more cells. Cells (Fig.B.1 [1]) are the fundamental building blocks of visualization systems.

- **Vertex** The vertex is a primary zero-dimensional cell. It is defined by a single point.
- **Polyvertex** The polyvertex is a composite zero-dimensional cell. The polyvertex is defined by an arbitrarily ordered list of points. Line The line is a primary one-dimensional cell. It is defined by two points.

- **Polyline** The polyline is a composite one-dimensional cell consisting of one or more connected lines.
- **Triangle** The triangle is a primary two-dimensional cell. The triangle is defined by a list of three points.
- **Triangle Strip** The triangle strip is a composite two-dimensional cell consisting of one or more triangles.
- **Quadrilateral** The quadrilateral is a primary two-dimensional cell. It is defined by a list of four points lying in a plane.
- **Pixel** The pixel is a primary two-dimensional cell defined by a list of four points. Each edge of the pixel is perpendicular to its adjacent edges and lies parallel to one of the coordinate axes $x - y - z$.
- **Polygon** The polygon is a primary two-dimensional cell. The polygon is defined by a list of three or more points lying in a plane.
- **Tetrahedron** The tetrahedron is a primary three-dimensional cell. The tetrahedron is defined by a list of four nonplanar points.
- **Hexahedron** The hexahedron is a primary three-dimensional cell consisting of six quadrilateral faces, twelve edges and eight vertices.
- **Voxel** The voxel is a primary three-dimensional cell. The voxel is equivalent to the hexahedron with additional geometric constraints. Each face of the voxel is perpendicular to one of the coordinate $x - y - z$.

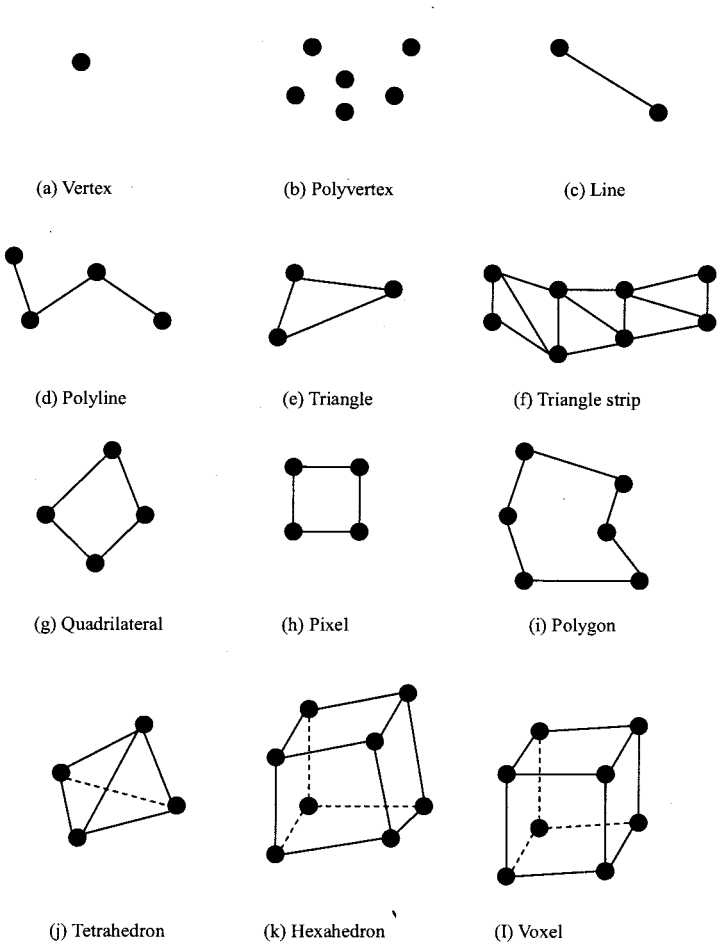


Figure B.1: VTK data representation