Validating and Refining Clusters via Visual Rendering

Keke Chen   Ling Liu
College of Computing, Georgia Institute of Technology
{kekechen, lingliu}@cc.gatech.edu

Technical Report, June, 2003

ABSTRACT

Clustering is an important technique for understanding of large multi-dimensional datasets. Most of clustering research to date has been focused on developing automatic clustering algorithms and cluster validation methods. The automatic algorithms are known to work well in dealing with clusters of regular shapes, e.g. compact spherical shapes, but may incur higher error rates when dealing with arbitrarily shaped clusters. Although some efforts have been devoted to addressing the problem of skewed datasets, the problem of handling clusters with irregular shapes is still in its infancy, especially in terms of dimensionality of the datasets and the precision of the clustering results considered. Not surprisingly, the statistical indices works ineffective in validating clusters of irregular shapes, too. In this paper, we address the problem of clustering and validating arbitrarily shaped clusters with a visual framework (VISTA). The main idea of the VISTA approach is to capitalize on the power of visualization and interactive feedbacks to encourage domain experts to participate in the clustering revision and clustering validation process. The VISTA system has two unique features. First, it implements a linear and reliable visualization model to interactively visualize multi-dimensional datasets in a 2D star-coordinate space. Second, it provides a rich set of user-friendly interactive rendering operations, allowing users to validate and refine the cluster structure based on their visual experience as well as their domain knowledge.

Keywords: Data Clustering, Cluster Validity, Information Visualization, Human Factor in Clustering

1. INTRODUCTION

Over the past decades most of the clustering research has been focused on automatic clustering algorithms and statistical validity indices. The automatic methods are known to work well in dealing with clusters of regular shapes, e.g. compact spherical shapes, but incur high error when dealing with arbitrarily shaped clusters. Concretely, problems with the automatic clustering/validation algorithms can be briefly summarized as follows:

- It is hard to handle the arbitrarily shaped clusters, which are common in applications. Some new algorithms like CURE [3], WaveCluster [20] and DBSCAN [15], have addressed this problem and try to solve it in restricted situations, such as in low dimensional datasets, or the cluster shapes are elongated/enlarged regular ones. Yet it is still considered as an unsolved hard problem due to the complexity in multi-dimensional (>3D) space and the unpredictable skewed cluster distributions.
- The arbitrarily shaped clusters also make the traditional statistical cluster validity indices ineffective [18], which leaves it difficult to determine the optimal cluster structure. For example, the compactness index of an elongated shape is not high but the quality of cluster could be considered as good in practice.
- In applications, some irregularly shaped clusters may be formed by combining two regular clusters or by splitting one large cluster with the incorporation of domain knowledge. However, it is inconvenient to incorporate domain knowledge in or allow the user to steer the clustering process with automatic algorithms.

One feature of the automatic clustering algorithms is that it almost excludes human from the clustering process, which is good in terms of reducing user's workload, but which is not so good since the user cannot easily manipulate the process. What the user can do is usually setting the parameters before the clustering algorithm running, waiting for the algorithm producing the results, validating the results and repeating the entire process if the results are not satisfactory. Once the clustering algorithm starts running, the user cannot monitor or steer the cluster process, which also makes it hard to incorporate domain knowledge into the clustering process and especially inconvenient for large-scale clustering since the iterative cycle is long. This exclusion makes the existing clustering framework inefficient and unintuitive for the user to deal with application-specific clustering.

Since clustering is an unsupervised process, cluster validity indices are used to evaluate the quality of clusters (the compactness or density of clusters, and the dissimilarity between clusters, etc.[]) Particularly, cluster validity indices are used to decide the optimal number of clusters. The typical indices includes root-mean-square standard deviation (RMSSTD), R-squared (RS), and S_Dbw[18, 25]. Although these indices were proved effective in determining the optimal number of compact well-separated spherical clusters, they do not work well for arbitrarily shaped clusters. A simple example in Figure 1 shows that the perfect clustering result of irregular clusters is often not consistent with the evaluation result. The 2D synthetic dataset consists of two clusters, both containing the same number of items and one of which is an elongated ellipse. With Euclidean distance the indices RMSSTD, RS and S_Dbw all suggest the k-means clustering result (Figure 1-2) is better, which is, however, not recommended intuitively. A partition of three clusters

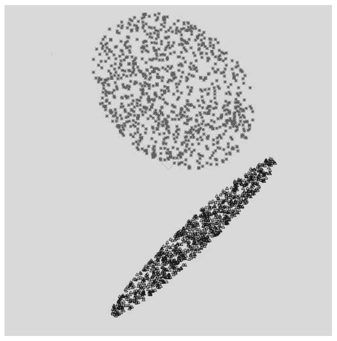
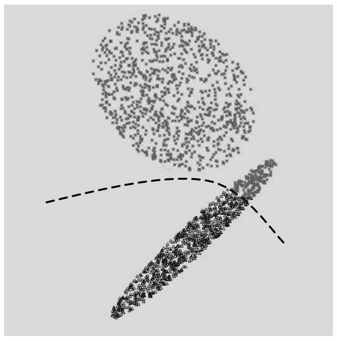
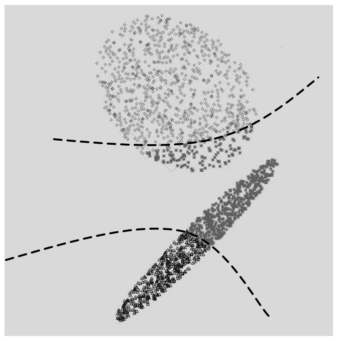


Figure 1-1 correct clustering.
RMSSTD = 4.6516, RS = 0.5923
S_Dbw = 1.5870


1-2 k-means result
RMSSTD=4.5536, RS= 0.6093
S_Dbw = 1.4605


1-3 k-means result (3 clusters)
RMSSTD= 3.4837, RS= 0.7715
S_Dbw = 1.2022

*The smaller RMSSTD, larger RS or smaller S_Dbw means the better partition in [18]

(Figure 1-3) even gives better index values than partitions of two, which is not correct at all. The indices simply do not work in evaluating irregular clusters.

Since the geometry and density features of clusters derived from the distance (similarity) relationship, determines the validity of clustering results, no wonder that visualization is the most intuitive method for validating clusters, especially the clusters in irregular shape. Many clustering algorithms in literature employ the 2D-plot of the clustering results to validate their effectiveness on 2D experimental datasets. However, the cluster visualization is not commonly used in practice because of the difficulty in visualizing multi-dimensional (>3D) datasets.

Clustering algorithms and validity indices have to answer the two problems: “how to recognize the special structure of each particular dataset?” and “how to refine a given imprecise cluster definition?” In this paper, we propose a visual framework that allows the user to be involved into the clustering process via interactive visualization. The core of the visual framework is the visual cluster rendering system VISTA. VISTA can work with any algorithmic results – at the beginning, VISTA imports the algorithmic clustering result into the visual cluster rendering system, and then lets the user participate in the following “clustering-evaluation” iterations interactively. With the reliable mapping mechanism employed by VISTA system, the user can visually validate the defined clusters via interactive operations. The interactive operations also allow the user to refine the clusters or incorporate domain knowledge to define better cluster structure.

Combining with the algorithmic clustering results, VISTA works well in improving the understanding of the cluster structure and the performance of validating and refining the arbitrarily shaped clusters. We will demonstrate the power of VISTA with two concrete examples – one is about how to validate and refine the algorithmic results with visual cluster rendering and the other is how to incorporate domain knowledge into the clustering process via visualization.

We organize the paper as following. The visual framework and VISTA system are introduced in section 2; in section 3, two empirical examples are demonstrated in details to show the power of VISTA in validating and refining clusters for real datasets. The related work is discussed in section 4. Finally, we conclude our work and give some of the future work.

2. VISTA VISUAL FRAMEWORK

Most frequently, the clustering is not finished when the computer/algorithm finishes unless the user has evaluated, understood and accepted the patterns or results, therefore, the user has to be involved in the “clustering – analysis/evaluation” iteration. In many cases, a simplified process that employs automatic algorithms is like the following:

1. Run the algorithms with initial parameters.
2. Evaluate the cluster quality and analyse the clustering results with statistical indices and domain knowledge.
3. If the result is not satisfactory, adjust the parameters and re-run the clustering algorithms, then do step 2 again until the satisfactory result is found.
4. If the result is satisfactory, do post-processing, which may label all of the items in the entire dataset or just output the cluster description.

Concrete discussion can be found in [14]. Our discussion will focus on steps 2 and 3. In step 2, it is often ineffective to validate the arbitrarily shaped clusters with the traditional cluster validity indices. And it is also difficult for human to verify the result with the domain knowledge. In step 3, it is usually very time-consuming to find appropriate parameters for a new run. The user often has to try several sets of parameters and find the relations between the clustering results. For example, CURE [3] requires the parameter of the number of representative points and shrink factor and DBSCAN [15] needs a proper Eps and MinPts to get satisfactory clusters.

We observed that with automatic clustering algorithms steps 2 and 3 can only be done in sequence. The user can only tune the parameters before the algorithm running and then wait for and evaluate the results. We propose that if we can interweave these two steps, e.g. the user can participate in the clustering process, monitoring and steering the process, the entire process would be more efficient. Instead of achieving this interweaving by improving the existing automatic algorithms – which could be very hard – we develop an interactive cluster visual rendering system to get human involved in. The entire visual framework is like Figure 2.

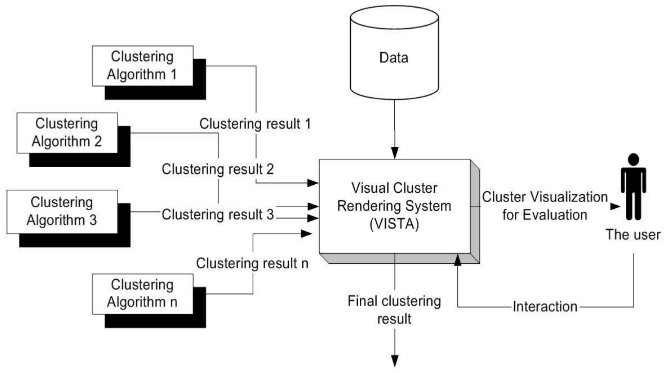


Figure 2. Visual framework for validating and refining clusters

Former studies [4] in the area of visual data exploration support the notion that visual exploration can help in cognition. Visual representations, especially interactive visualization, can be very powerful in revealing trends, highlighting outliers, showing clusters, and exposing gaps. Previous research shows that, with the right coding, human pre-attentive perceptual skills can enable users to recognize patterns, spot outliers, identify gaps and find clusters in a few hundred milliseconds [17]. For example, in a scatter-plot based visualization, the human visual ability is adept at finding the clusters – the point-dense area very quickly, and the shape of the cluster is identified at the same time too. All of the advantages make the interactive cluster visualization systems very attractive.

However, there are some challenges for cluster visualization techniques, among which the most challenging one is cluster preserving– the clusters appearing in the 2D/3D visualization should be the real clusters in k-D (k>=3) space. Since a k-D to 2D/3D mapping inevitably introduces visual bias, such as broken clusters, overlapping clusters or fake clusters formed by outliers, static visualization is not sufficient and additional rendering techniques are needed to improve the visual quality.

In VISTA cluster rendering system, we use a linear (or affine) mapping [24] – α-mapping to avoid the breaking of clusters after mapping, but the overlapping and fake clusters may exist. The compensation technique is interactive dynamic visualization. The interactive operations are used to change the projection plane, which allows the user to observe the datasets from different perspectives. Continuously changed visualization usually provides important clues for the user to discriminate the overlapping and the fake clusters.

While the visual cluster rendering system is combined with the algorithmic result, the two can improve each other. The coloured algorithmic result in visualization provides visual clustering clues – the points in same colour, i.e. the same cluster, should be grouped in the same area, which can guide the user to find a satisfactory visualization. On the other side, the satisfactory cluster visualization after rendering can validate the algorithmic results by visually checking the match of the visual cluster distribution and the algorithmic distribution. Therefore, the better way for visual cluster rendering is to combine the algorithmic results with the interactive visualization system.

The basic methodology employed in visual cluster validating and refining follows the steps:

Step1. Load the dataset, (and algorithmic result if available)

Step2. Use the interactive operations to find a satisfactory visualization,

Step3. Import domain knowledge if available, make the visual boundaries between clusters and refine the algorithmic result if applicable.

Step4. Output the refined result.

To illustrate how the VISTA works, we will briefly introduce the α-mapping and some interactive operations. The initial version of VISTA is used to render Euclidean datasets, where the similarity is defined by Euclidean distance, since the Euclidean datasets are the most common datasets in applications. By default, we will not mention this again in the following discussion.

α-mapping

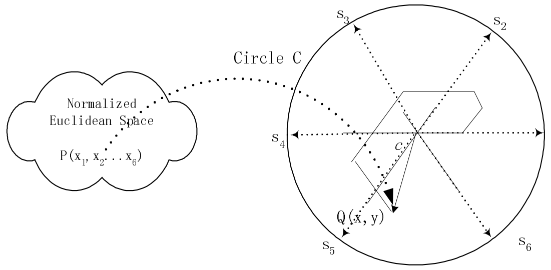


Figure 3. Illustration of α-mapping with *k*=6

We invent a linear mapping α-mapping that partially preserves *k*-dimensional (*k*-D) information in 2D space and is used to build a *k*-parameter-adjustable interactive visualization system. α-mapping maps *k*-D data to 2D star coordinates [9] and normalizes the visualization into the designated display area.

A *k*-axis 2D star coordinates is defined by an origin $\tilde{o}$ (x_0, y_0) and *k* coordinates $S_1, S_2, \ldots, S_k$, which represent the *k* dimensions in 2D spaces. The *k* coordinates are equidistantly distributed on the circumference of the circle *C*, as in Figure 3, where the unit vectors are $\tilde{S}_i = (\hat{u}_{xi}, \hat{u}_{yi})$, $i = 1..k$, $\hat{u}_{xi} = \cos(2\pi / i), \hat{u}_{yi} = \sin(2\pi / i)$. The radius *c* of the circle *C* is the scaling factor, which determines the size and the detail level of the visualization.

α-mapping is a parameterized mapping that utilizes star coordinates to establish the visualization. We describe α-mapping as follows. Let a 2D point *Q* (*x*, *y*) represent a *k*-dimensional max-min normalized [10] (with normalization bounds [-1, 1]) data point $P(x_0, x_1, \ldots x_i \ldots, x_k)$ mapped onto the 2D star coordinates. *Q*(*x*, *y*) is determined by the average of the vector sum of *k* vectors $\tilde{S}_i \cdot x'_i$ ($i = 1..k$) adjusted by *k* parameters ($\alpha_1, \alpha_2, \ldots, \alpha_k$) and scaled by the radius *c*.

α-mapping:

$$A(x_1, \ldots, x_k, \alpha_1, \ldots, \alpha_k) = (c / k) \sum_{i=1}^{k} \alpha_i x_i \vec{s}_i - \vec{o} \quad (1)$$

i.e. the position of *Q* (*x*, *y*)is determined by,

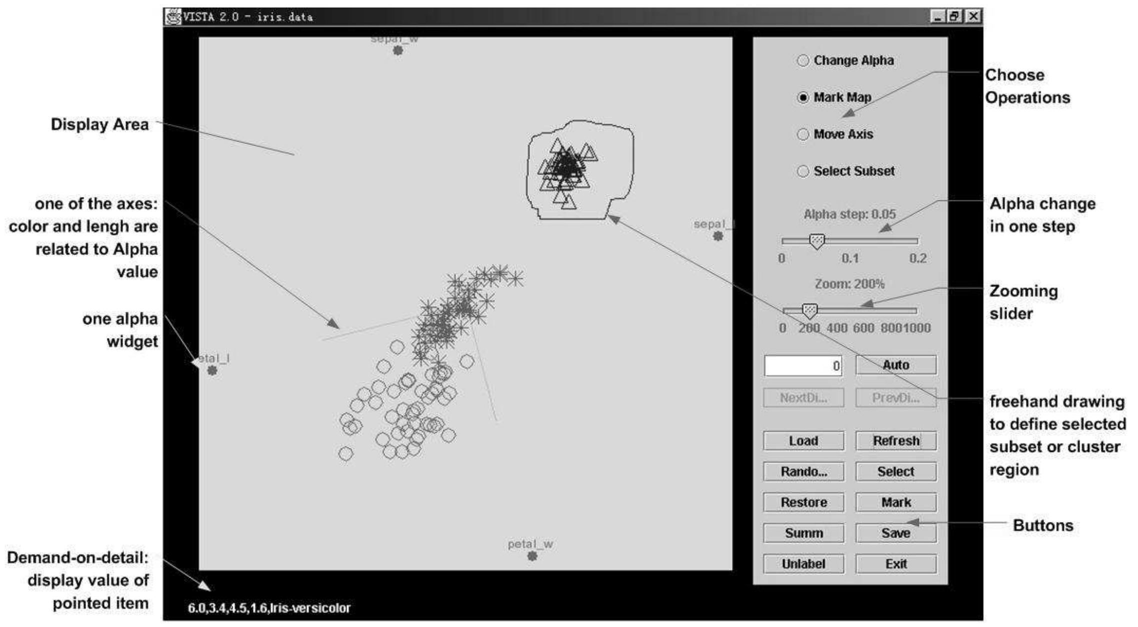


Figure 4 VISTA system

$\{x, y\}$ =

$$\left\{ (c/k)\sum_{i=1}^{k}\alpha_i x_i \cos(2\pi/i) - x_0, \quad (c/k)\sum_{i=1}^{k}\alpha_i x_i \sin(2\pi/i) - y_0 \right\}$$

The α_i (i = 1,2,…k, $-1 \le \alpha_i \le 1$) in the definition are dimensional adjustment parameters, one for each of the k dimensions – that is where the name "α-mapping" comes from. α_i is set to 0.5 initially.

The α-mapping has two properties:

- The mapping is linear. Without loss of generality, we set $\tilde{o}$ to (0, 0). It is easy to see the α-mapping is a linear mapping, given the constants α_i. It is known that the linear mapping does not break clusters but may cause overlapping clusters [24, 9], and sometimes, overlapping outliers to form fake clusters. Given that the α-mapping is linear and there is no "broken clusters" in the visualization, each gap seen in visualization confirms a real gap in the original space. What we need to do is to separate the overlapping clusters, and the falsely clustered outliers, which could be done with the help of interactive operations.

- The mapping is adjustable by α_i. α_i (i = 1,2,…k, $-1 \le \alpha_i \le 1$) can be regarded as the weight of the i-th dimension, which means how significant the i-th dimension is in the visualization. By changing α_i continuously, we can see the effect of the i-th dimension on the cluster distribution. In addition, when one α value or several α values are changed continuously at the same time, the k-D dataset is mapped to a series of smoothly changed projections, which provide important cluster clues.

The visual rendering operations

The VISTA system looks like Figure 4. The task of the VISTA cluster rendering system is to provide the interactive visualization techniques to help the users find and separate the overlapping clusters through continuously changed visualization. We have designed and implemented a set of interactive rendering operations in VISTA.

α-parameter adjustment

The most important operation in VISTA system is α-parameter adjustment (or simply, α-adjustment). This operation changes the α parameters defined in formula (1). Each change refreshes the visualization in real time (about several hundred milliseconds, depending on different hardware configurations and the size of dataset). α-parameter adjustment enables the user to find the dominating dimensions, to observe the dataset from different perspectives and to distinguish the real clusters from overlapping with a series of continuously changed visualizations.

Continuous α-parameter adjustment of one dimension looks at the effect of this dimension on the entire visualization. If we adjust the α value of the dimension i, the point movement can be represented by:

$$\begin{aligned}\Delta(i) &= \mathrm{A}(x_1,\dots,x_k,\alpha_1,\dots\alpha_i\dots,\alpha_k) - \mathrm{A}(x_1,\dots,x_k,\alpha_1,\dots\alpha_i'\dots,\alpha_k) \\ &= (c/k)\,(\alpha_i - \alpha_i')x_i\tilde{s}_i\end{aligned}$$

which means that the points having larger x_i will be moving faster along the direction of i-th coordinate in star coordinates, similar x_i moving in a similar way. This point movement reveals the characteristics of dimension i. In Euclidean datasets, changing α value of any dimension of the two points, which have similar values in each dimension and thus close to each other, does not change the distance of the two points a lot. This means close points tend to move together in any α-adjustment. This property makes α-adjustment very effective in revealing cluster overlapping. The corresponding dimension-by-dimension rendering rules [1] are proved very effective in practice to find cluster boundary.

Compared to the basic rendering techniques "axis scaling" and "axis rotation" in the original star coordinates system [9],

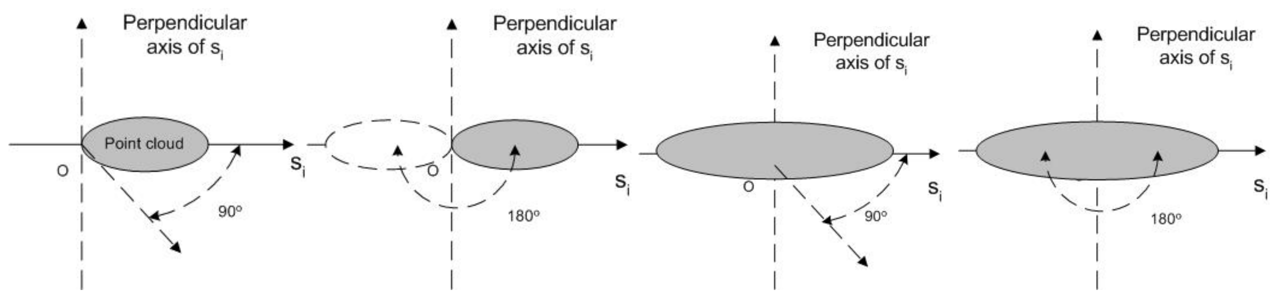

Perpendicular axis of s_i
Point cloud
O
s_i
90°
Perpendicular axis of s_i
O
s_i
180°
Perpendicular axis of s_i
O
s_i
90°
Perpendicular axis of s_i
s_i
180°

Figure 5-1: α [0,1], norm [0, 1]   5-2 α[-1,1], norm[0,1]   5-3 α[0,1], norm[-1,1]   5-4 α[-1,1], norm[-1,1]

VISTA α-adjustment, powered by the VISTA visualization model, can solely support the user to find satisfactory visualization. In fact, α-adjustment together with the zooming factor *c*, determines the range of possible projection planes that α-adjustment can cover. Since the zooming factor *c* does not change the structure of the content in visualization, only is the α-adjustment enough to provide the informative projection planes. The mapping described in the original star coordinates[9] can be actually rephrased in VISTA visualization model: the original *k*-dimensional point $P(x_0, x_1, \ldots x_i \ldots, x_k)$ is normalized to [0, 1] and α_i (i = 1,2,…k) is limited to [0, 1]. The positive-only α-adjustment shows the dynamic visual effect just as the "axis scaling" described by the author. Nevertheless, α-adjustment in [-1, 1] defines more than "axis scaling".

In general, α-adjustment in different range can be intuitively understood as the projection of "visual rotation[*] " of point cloud around the perpendicular axis of coordinate *i*. Different α-value range defines different rotation angle (Figure 5). For example, "axis scaling" can be regarded as the projection of rotation from 0º to 90º around the perpendicular axis as shown in Figure 5-1.

While the range of α-values determines the ability of interactive operations, the scope of the normalized original values influences the result of visualization. To make the demonstration clearer, without loss of generality we consider the situation that the main contributing dimension is *i* with a set of α-values, e.g. $(c/k)\sum_{1 \le j \le k} \alpha_j * x'_j * \vec{s_j} \approx (c/k)\, \alpha_i * x'_i * \vec{s_i}$. When the values in dimension *i* are normalized to [0, 1], in the original star coordinates the point cloud always distributes along the positive direction of S_i, which squeezes the effective visualization onto one side of the display area and shows less details inside the point cloud (Figure 5-1). Whereas, normalizing to [-1, 1] elongates the point cloud and allows the user to observe more details (Figure 5-3).

Combining the different scopes of normalized values and α values, we can observe different effects of α-parameter adjustment. We list four typical intuitive rendering effects as in Figure 5. α-mapping in VISTA model is the case in Figure 5-4, which efficiently utilizes the entire display area and enables "180º rotation" of data points along the centre perpendicular axis. The original mapping in star coordinates is the case in Figure 5-1, which tends to squeeze the point clouds and thus limit the observation angles. To sum up, VISTA visualization model enables the design of more powerful interactive operations.

[*] observed as rotation in continuous α-adjustment

Other operations

Since α-parameter adjustment is the most frequently used one, some operations, such as random rendering and automatic rendering, are used to increase the efficiency of α-parameter adjustment [1]. Another set of operations support point-set-oriented operations and are used to refine visual cluster definition after we get initial cluster visualization with α-parameter adjustment. These operations include subset selection, cluster marking, cluster splitting, cluster merging, and hierarchical structure defining. Domain knowledge in form of labelled items can be incorporated into visualization. Most of the following operations are designed for cluster refining and high-level structure defining yet not defined in [1].

- **Subset selection**

This operation defines a subset of points by freehand drawing an enclosed region on screen or selecting a range of one dimension. The selected subset can be used for further processing, such as cluster marking, merging and splitting.

Initially, we have one subset, which is the entire dataset. The clusters are defined as subsets. We name *i*-th subset as ss_i. After loading labels, which define *c* clusters, the subsets becomes (ss_1, ss_2, …, ss_c). Suppose before selection, we have had *m* subsets ordered as (ss_1, ss_2, …, ss_m) . The (*m*+1)-th subset is selected from one or more subsets. We define subset selection as following, where '-' is set difference operation.

$$SS(m): (ss_1, \ldots ss_i \ldots ss_m) \rightarrow (ss_1 - ss_{m+1}, \ldots, ss_i - ss_{m+1} \ldots ss_m - ss_{m+1}, ss_{m+1})$$

- **Merging & splitting clusters**

These two operations enable the user to refine the visualized algorithmic clustering result. If the user finds a part of a cluster should be semantically separated from the cluster, she/he can use selection operation to select this part and then excludes it from the cluster. If two nearby clusters should be regarded as one cluster from the domain knowledge, the user just selects them and merges them into one cluster. A Cluster boundary can be refined by merging and splitting operations, too. Splitting subset *i* to subset *i*1 and *i*2, and merging subset *i* to *j* are defined as following, where 'U' is set union operation.

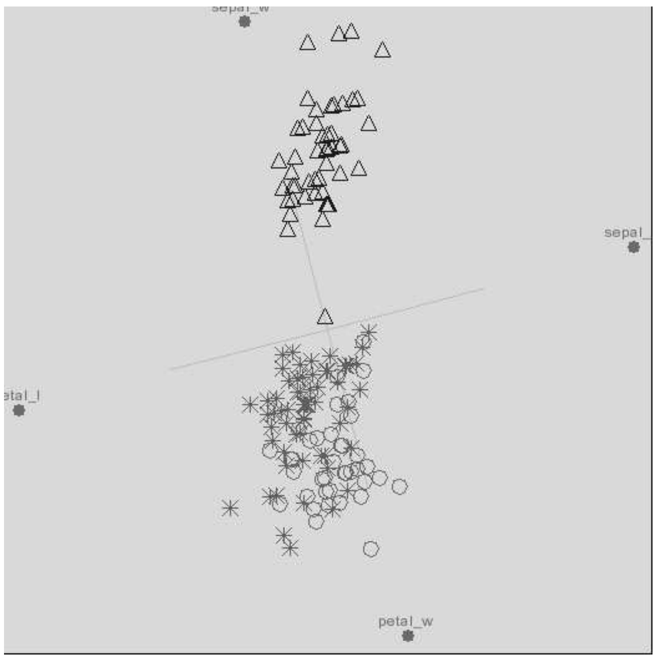


Figure 6-1: the initial visualization with k-means labels

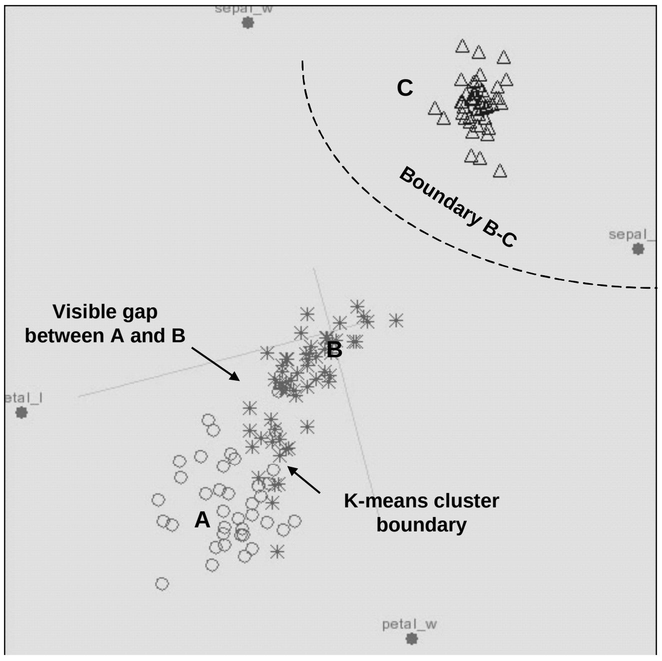


Figure 6-2: k-means result, RMSSTD =0.4108, RS = 0.8002, S_Dbw = 1.4158

$$Split\ \ (i, i1, i2, m): (ss_1, \ldots, ss_i \ldots, ss_m) \rightarrow (ss_1, \ldots, ss_i - ss_{i2} \ldots, ss_m, ss_{i2})$$
$$where\ \ ss_{i1} = ss_i - ss_{i2}$$

$$Merge\ \ (i, j, m):$$
$$(ss_1, \ldots ss_i \ldots ss_j \ldots, ss_m) \rightarrow (ss_1, \ldots ss_{i-1}, ss_{i+1} \ldots, ss_j \cup ss_i, \ldots, ss_m)$$

- **Defining hierarchical cluster structure**

With the operations of defining the cluster hierarchy, the user can group the clusters together to form a higher level cluster structure, or the user can zoom into one large cluster and find the fine cluster structure in the cluster iteratively. These operations together define a layered cluster structure. With the operations of zooming in or zooming out, the user can find the cluster details at different level. When we are rendering at some layer *j*, where it has *m* subsets, and want to group *k* selected clusters, the hierarchy defines:

$$H_j(m, ss_{i1}, ss_{i2} \ldots, ss_{ik}):$$
$$(ss_1, ss_2, \ldots, ss_m) \rightarrow (ss'_1, ss'_2, \ldots, ss'_{m-k}, (ss_{i1}, ss_{i2} \ldots, ss_{ik}))$$

where, ss_{i1} … ss_{ik} are the grouped subsets and $ss'_j \subset (ss_1, ss_2, \ldots ss_m)$.

- **Importing domain knowledge**

A set of domain knowledge is transferred to a set of k-D items with different group identities. These items are imported into the visual rendering system and rendered in different colours with different groups. These coloured items act as the guidance to re-define the cluster partition with domain knowledge.

If domain knowledge is represented by k groups of items, these items form *k* new subsets after they are loaded, which then used to direct further cluster splitting or merging. For example, if g1 covers ss_1 and ss_2, then we can consider merging them.

$$D(m, g_1, g_2 \ldots, g_k):$$
$$(ss_1, ss_2, \ldots, ss_m) \rightarrow (ss_1, ss_2, \ldots, ss_m, g_1, \ldots, g_k)$$

3. EMPIRICAL STUDY

In this section, we will introduce two examples of visual rendering. The first one demonstrates the ability of VISTA visual validating and interactive refining. The second one shows how to incorporate domain knowledge into VISTA visual cluster rendering. The datasets used in the examples can be found at [28].

3.1 Analyzing the "Iris" dataset

In this example, we will use the most popular clustering algorithm – k-means [12] to produce the clustering result on the dataset "iris", and then import the result into VISTA system. With VISTA system, we will validate the k-means result visually and then try to refine the clusters and improve the quality of the k-means clusters. The quality of clusters will be also evaluated by statistical indices RMSSTD, RS, and S_Dbw [18, 25] at the same time to see if the statistical indices are consistent with the visual improvement.

"Iris" dataset is a famous dataset widely used in pattern recognition and clustering. It is a 4-D dataset containing 150 instances, and there are three clusters, each has 50 instances.

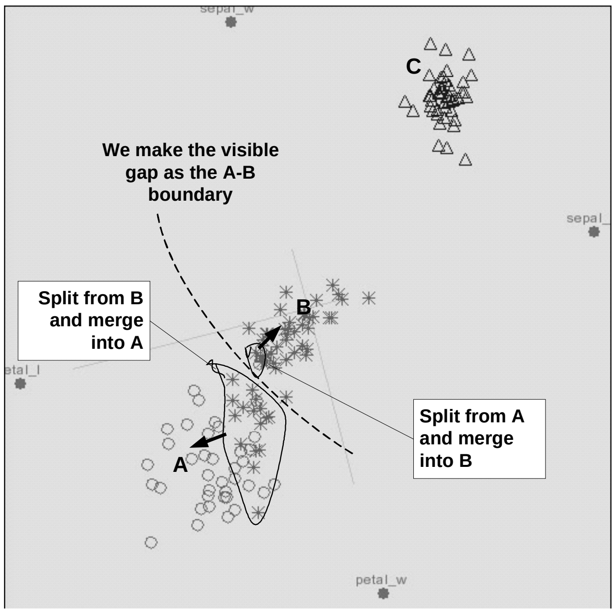


Figure 6-3: Editing the clusters

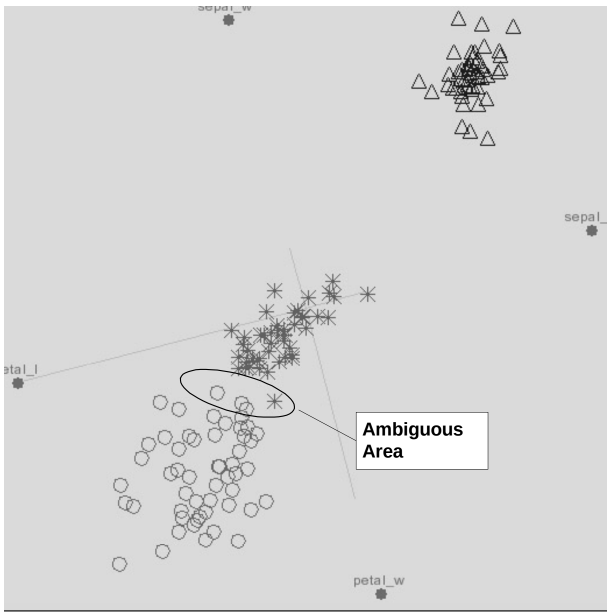


Figure 6-4: After editing, EMSSTD = 0.4396, RS=0.7712, S_Dbw =1.5115

One cluster is linearly separable from the other two; the latter two are not exactly linearly separable from each other.

Firstly, we load the dataset and import the k-means labels for "iris" dataset into the visualization. Different clusters are visualized in different colors. The initial visualization is like Figure 6-1, where we can find one cluster has been separated from the other two. After interactive cluster rendering, mainly the α-parameter adjustment, the visual boundaries become clearer (Figure 6-2). The boundary B-C clearly separates cluster C from the other two clusters. The gap between cluster A and B can be visually perceived but not so clear. The α-mapping model confirms that this gap does exist in the 4-D space since α-mapping does not break clusters. We make this gap as the visual boundary A-B. This visually perceived boundary A-B is not consistent with the k-means boundary, but we have more confidence with it since it has been intuitively confirmed. There is a principle in visual cluster rendering – we prefer visual perception rather than statistical information because we believe the visual ability is better than statistical methods in dealing with arbitrarily shapes.

Considering this visual boundary, we want to edit the k-means result with visual cluster editing operations. First, we split the points that belong to cluster A but visualized in cluster B, from cluster A. These points are then merged into cluster B. Do the same operation on the B points in cluster A as shown in Figure 6-3. After the editing operations, the points in the clusters are shown more homogeneously (Figure 6-4). The visual partition exactly reflects the real cluster distribution (compare Figure 6-4 and 6-5).

We check the validating results of the widely used cluster validity indices RMSSTD, RS and S_Dbw, to see if the statistical validation is consistent with the visual improvement. RMSSTD is used to estimate the homogeneity of the clusters. Smaller RMSSTD indicates that the clusters are more compact. RS is used to estimate the dissimilarity between clusters. Larger RS indicates higher dissimilarity between groups. S_Dbw is a compound evaluation of compactness and dissimilarity, e.g. the overall quality of clusters. The smaller S_Dbw implies the better quality. [18]

The statistical evaluation shows RMSSTD is increased from 0.4421 to 0.4614, RS is decreased from 0.8254 to 0.8098, and S_Dbw rises from 1.4158 to 1.5115 after visual rendering. This means the compactness of clusters and the dissimilarity between clusters are decreased at the same time – the quality of clustering after visual improvement is worse than the k-means result in statistics, which is not correct in practice! The irregular shapes of A and B, together with the closeness to each other, makes the statistical methods ineffective in this scenario.

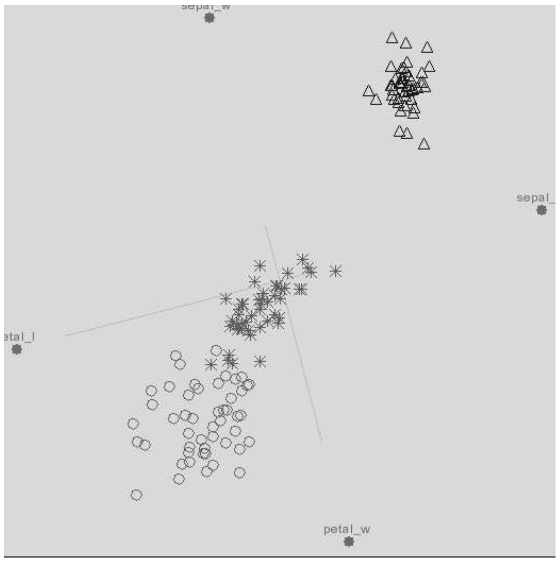


Figure 6-5: the real cluster distribution visualized with the labels from the original dataset.

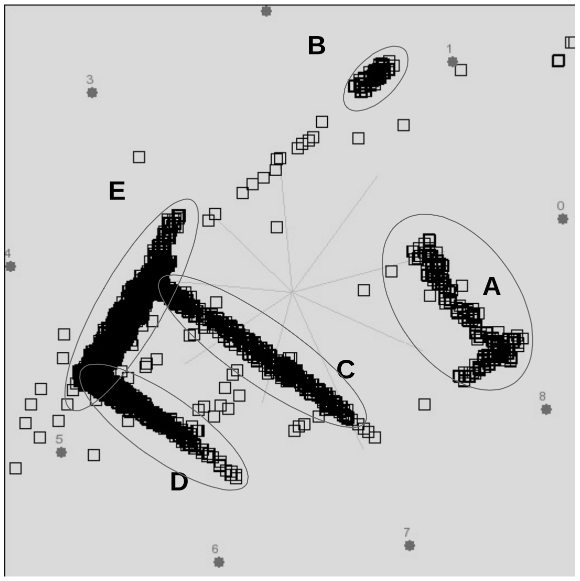


Figure 7-1: The visualization after initial rendering operations

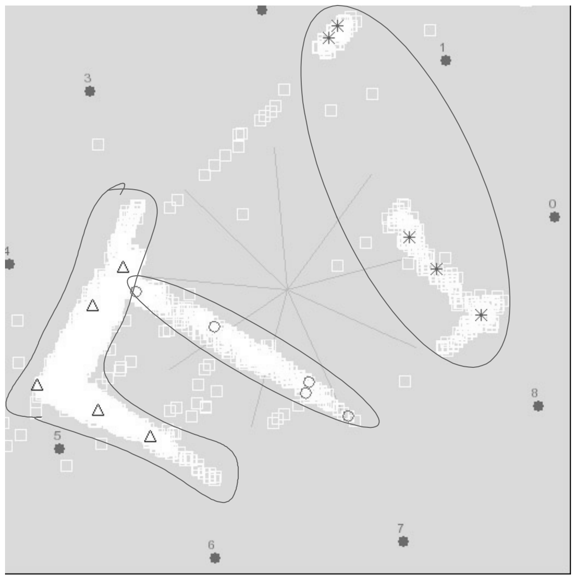


Figure 7-2: The landmarks and suggested cluster structure.

As the literature of the "iris" dataset mentioned, the clusters A and B are not linearly separable. To further refine the cluster definition, we can also informally define a small "ambiguous area" around the gap between A and B, the points in which have equal probability of belonging to A or B. In addition, in extended experiments with trained users, all users can find the visualization like Figure 6-4, which means visual validity could be very practical in exploring certain datasets. We will support this assertion with more experimental result in section 3.3.

In conclusion, we believe that the VISTA system is better than the statistical indices, in terms of validating arbitrarily shaped clusters. In this example, we have seen that sometimes the vague boundary between the two clusters is easily checked by human visual ability but it is not so easy for the automatic algorithms. In addition, this example also shows the power of online refining ability of the VISTA system – after validation, the user can improve the quality of clusters immediately by editing the clusters – which effectively combines the two steps "re-clustering" and "evaluation" together. Certainly, in cases where the clusters are not easily be visualized, e.g. clusters in very high-dimensional datasets, (e.g. >50 dims for VISTA), the statistical indices are still the only choice, even though it is not so effective.

3.2 Incorporating Domain knowledge

In this empirical example, we will demonstrate that the VISTA system can conveniently incorporate the domain knowledge into the clustering process and provide intuitive clues for the user to define the application-specific clusters. We first define the form of "domain knowledge" that can be utilized in VISTA system, and then show how to use the domain knowledge to distinguish the application-specific cluster distribution with the example of rendering "shuttle" dataset.

Domain knowledge plays a critical role in the clustering process [14]. It is the semantic explanation to the data, which is different from the structural clustering criteria, such as distance between points. Domain knowledge usually leads to a high-level cluster distribution, which may be different from the structural clustering results, for example, the original clusters may be grouped to form larger clusters or split to form finer cluster structure.

Domain knowledge can be represented in various forms in Artificial Intelligence [14]. In VISTA system, we require only one of the simplest forms to provide the domain-related clustering criteria. We define the domain knowledge as following:

> Suppose the dataset contains a set of instances $\{X_i\}$ and the user have some knowledge about the application. The form of the domain knowledge can be the specific properties, the experimental results, or any hypotheses the application holds. We need a small number of typical instances X_1, X_2, …, X_n (n << the number of items N in the dataset) to reflect the properties, or the experimental results. According to the domain knowledge, this set of instances should be partitioned into *m* groups. The *m* groups are represented by $g_1(X_{1,1}, X_{1,2},\ldots, X_{1,t1})$, $g_2(X_{2,1}, X_{2,2}, \ldots, X_{2,t2})$,…$g_m(X_{m,1}, X_{m,2}, \ldots, X_{m,tm})$. We give labels to the instances so that each instance is represented as (instance, label#). Therefore, we have the *n* instances labeled as
>
> $(X_{1,1}, 1)$ $(X_{1,2}, 1)$… $(X_{1,t1}, 1)$
>
> …
>
> $(X_{m,1}, m)$ $(X_{m,2}, m)$… $(X_{m,mt}, m)$
>
> They are regarded as additional points of the dataset, with domain categorical labels. We name them "landmarks" in VISTA system. The number of the instances is so small that they cannot work effectively as a training dataset for classification algorithms to classify the entire datasets.

When visualizing a dataset, the landmark points are loaded and visualized in different colors according to their categorical ID. This guiding information can direct the user to define the high-level cluster structure, or to refine the algorithmic clustering results. Automatic algorithms have no such abilities, or it is very inefficient or clumsy to incorporate this functionality into the automatic algorithms.

An alternative method is to visualize the dataset first and then sample some points from the "critical areas" on the visualization, such as the connective area of two point clouds. The sample points are classified with the domain knowledge and re-imported into the visualization as the "landmarks" to direct rendering.

We use the "shuttle" dataset and the second method to demonstrate how the VISTA system incorporates the domain knowledge into the clustering process. "Shuttle" dataset is a 9-D dataset. There are three large clusters and some tiny clusters in the dataset. Approximately 80% of the data belongs to one cluster. The other two large clusters have about 15% and 5% points, respectively. We use the testing dataset, which has 14500 items, for visualization.

After loading the dataset and adjusting the α parameters, we get the initial visualization, which shows the cluster distribution is highly irregular. There are five homogenous segments (Figure 7-1). We have no idea whether they are five individual clusters, or they should be grouped or split to form higher-level clusters.

We then pick several points from the visualization, which should ideally cover the connective areas. Suppose the "expert" with experiments or any domain knowledge (use the labels from the original datasets to mimic) tells us the sample points should be grouped into three clusters. Using them as the "landmarks", we find a possible cluster structure as in Figure 7-2. To observe the landmarks clearly, we visualized other data points in white color. The result shows the datasets probably should be partitioned in the suggested way. The real cluster distribution of the "shuttle dataset" is visualized in Figure 7-3 for comparison.

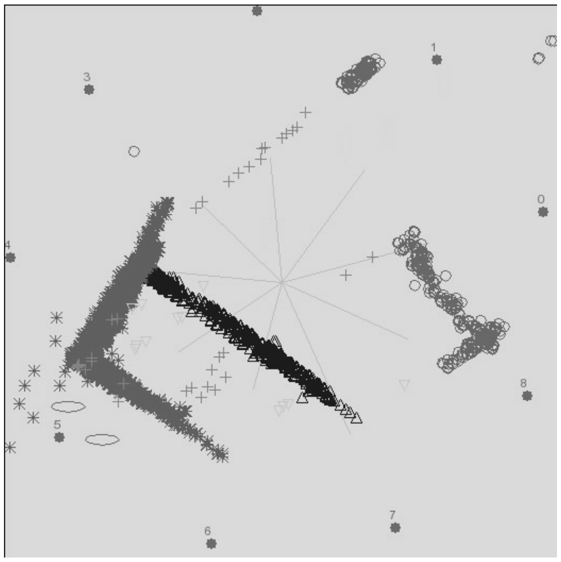


Figure 7-3: the clusters with original labels

To sum up, since the automatic algorithms exclude the human from the clustering process, the domain knowledge cannot be easily incorporated into the clustering process. With the help of VISTA system, the user is able to incorporate the domain knowledge into the clustering process to define application-specific cluster distribution online. This combination of human-based analysis/evaluation and clustering process breaks the gap between human and the machines, and thus improves the efficiency of the entire cluster analysis process.

3.3 More Experimental Results

The VISTA visual clustering system was implemented in Java. In this section we will introduce more experimental results to show the power of visual cluster rendering system in finding clusters individually or in combining any external information to provide better clustering results. These experiments were conducted on a number of well-known datasets that can be found in UCI machine learning database[28]. These datasets, although small or median in size, have irregular cluster distribution, which is an important factor for testing the effectiveness of the VISTA system.

Five well-trained users use the VISTA cluster rendering system to find satisfactory visualization for each of the ten datasets. After we use the interactive visual operations to find the satisfactory visualization, either solely by visual rendering or incorporated by algorithmic result, we mark the areas which are regarded as clusters and the items in each area are respectively labelled with the cluster ID. With the original labels in the datasets, we define the items that are wrongly clustered as the errors, the number of which divided by the size of the dataset is the error rate of visual cluster rendering on this dataset.

We firstly use unguided visual rendering (UGV) to find the visual partition. Unguided visual rendering does not rely on any external information and only depends on the visually observed dense-point areas and the gaps between the areas. Since there is visual bias on the visualization, the visual rendering sometimes may trap in local minima, where the user thinks the visualization is satisfactory enough for defining cluster boundaries. We want to avoid trapping in local minima by incorporating some external information, either from other clustering algorithms or domain knowledge. In our experiments, 10% of labelled items randomly selected from the original datasets are used as the "landmarks". We also compare the results of K-means and CURE algorithms on the experimental datasets. CURE clustering is recognized as one that can deal with irregular cluster shapes in some level, and K-means is the most popular algorithm commonly used in research or applications. The experiment shows that individually CURE or K-means cannot deal the arbitrarily shaped clusters very well and UGV may trap into some local minima, but by combining with the external information we can improve the UGV result more or less. The result also shows the visualization result, either UGV or combined rendering, is often better than algorithmic result for arbitrarily shaped clusters. Most of the interaction time is less than 5 mins, which means it is not difficult for a trained user to find a satisfactory visualization.

We list the experimental results in Table 1, where N is the number of rows in the given dataset, k is dimensionality of the dataset, and c is the number of clusters in the dataset. "UGV(%)" is error rates (%) of unguided visual rendering result. "Combo(%)" is the error rates(%) of the combining of UGV with the "landmark points". "Time(min)" is the interaction time used in the "combo" method. "CURE(%)" is error rates (%) of CURE clustering algorithm. "K-means(%)" is the error rates (%) of K-means clustering algorithm.

Dataset	*N*	*k*	*c*	UGV(%)	Combo(%)	Time (min)	CURE (%)	K-means(%)
Bre-canc-wisc	699	10	2	16.7	3.3 ± 0.4	1.6 ± 0.3	8.6	5.1
Crx	690	15	2	20.2	14.5 ± 0.3	2.3 ± 0.8	46.8	48.0
Iris	151	4	3	5.5	0.7 ± 1.2	1.9 ± 0.3	32.7	11.3
Page-blocks	5473	10	5	13.0	7.8 ± 0.2	2.6 ± 0.4	32.7	45.6
Hepatitis	155	19	2	21.9	21.7 ± 2.4	2.7 ± 0.2	35.7	42.6
Heart	270	12	2	24.0	18.6 ± 1.5	2.3 ± 0.5	47.4	21.1
Mushroom	8124	21	2	24.7	8.4 ± 0.3	5.5 ± 0.4	31.9	40.2
Australian	690	14	2	15.4	14.6 ± 0.7	2.6 ± 0.9	36.8	14.5
Wine	178	12	3	7.9	2.1 ± 0.5	3.1 ± 0.4	32.0	5.6
Shuttle.test	14500	9	7	10.2	4.0 ± 0.4	1.7 ± 0.2	20.5	23.2

Table 1: More experimental results on typical datasets having irregular cluster distribution

4. RELATED WORK

The common cluster analysis framework is described in the clustering review paper [14]. Recently, some algorithms have been developed to deal with arbitrarily shaped clusters. CURE [3] uses a set of representative points to describe the boundary of a cluster in its hierarchical algorithm. But the number of representative points increases dramatically with the increase of the complexity of cluster shapes in order to maintain the precision. CHAMELEON [23] employs a multilevel graph partitioning algorithm on the k-Nearest Neighbour graph, which may produce better results than CURE on complex cluster shapes for spatial datasets. But the high complexity of the algorithms prevents its application on higher dimensional datasets. DBSCAN [15] is a density-based algorithm but it is very sensitive to the parameter Eps and MinPts. The distribution-based algorithm DBCLASD [22] and the wavelet transformation based algorithm WaveCluster [20] were also reported as being efficient only in spatial datasets. In conclusion, the automatic algorithms can deal with the arbitrarily shaped clusters in some situations, but the results are not general enough to apply to any application which has dimensionality higher than 3D. The most difficult problem is, for high-dimensional (>3D) datasets, the arbitrarily shaped clusters produced by the automatic algorithms are hard to be validated, since the statistical indices [18] are not effective for such clusters.

Information visualization is commonly recognized as a useful method for understanding sophistication in datasets. Many efforts have been made to analyze the datasets in a visual way. We discuss the scatterplot-based techniques only because it is the most intuitive techniques for cluster visualization. The early research on general plot-based data visualization is Grand Tour and Projection Pursuit [7]. Since there are numerous projections from a multidimensional data space to a 2D space, the purpose of the Grand Tour and the Project Pursuit is to guide the user to find the interesting projections. L.Yang [8] utilizes the Grand Tour technique to show projections of datasets in an animation. They projected the dimensions to coordinates in a 3D space. However, when the 3D space is shown on a 2D screen, some axes may be overlapped by other axes, which make it hard to perform direct interactions on dimensions. Dhillon [5] provides a method for visualizing only 3 clusters while preserving the distances. When more than 3 clusters exist, his method needs the help of Grand Tour techniques. Other techniques, such as Scatterplot matrices, coplots, prosection [2] and FastMap based visualization [21, 19] only create static visualization, which inevitably distorts the cluster structure but have no effective methods to rectify it, thus do not provide enough information for correct clustering. In the KDD 2002 tutorial [13], more visualization methods were also discussed.

Star Coordinates [9] is a visualization system designed to visualize and analyze the clusters interactively. We utilize the form of Star Coordinates and build a normalized α-mapping model in our system. We have discussed that α-mapping model extends the ability of the original mapping in star coordinates paper and demonstrated the particular ability of VISTA system in visually validating and refining clusters.

HD-Eye [26] is another interactive visual clustering system. HD-Eye visualizes the density-plot of the interesting projection of two of the *k* dimensions. It uses icons to represent the possible clusters in each projection and the relationship between the clusters. However, it is hard for users to synthesize all of the interesting 2D projections to find the general pattern of the clusters. In fact, visually determining the basic cluster distribution solely through user interaction is not necessary. The semi-automatic 1D visualization based algorithm OPTICS [27] actually works well in finding the basic arbitrarily shaped clusters, the result of which can be utilized by some high-level visualization systems, such as VISTA. However, OPTICS itself cannot be easily applied to analyze the shape of clusters and the distance relationship between clusters, or to incorporate any domain knowledge in cluster analysis.

5. CONCLUSION

Most of researchers have focused on automatic clustering algorithms, but very few have addressed the human factor in the clustering process. Although the existing clustering algorithms and cluster validity methods are working well on spherical clusters with some statistical assumptions, they have encountered the difficulty in dealing with arbitrarily shaped clusters. In order to solve this problem, we have to check the human factor in the clustering process more carefully. The VISTA system demonstrates some possible ways to introduce the users into the clustering process.

In this paper, we proposed the VISTA visual framework to combine the algorithmic results with visual cluster rendering system. The power of VISTA visual cluster rendering system enables the users to visually validate and interactively refine the clusters. It also allows the users to incorporate domain knowledge into the clustering process in a convenient way. The empirical study shows that the VISTA framework/system works very well in visual validating and refining the algorithmic clustering results.

The current VISTA system can handle datasets with dimensionality less then 50. Dimensionality higher than or close to 50 will cause the difficulty in human visual understanding and operations. The system capability also restricts the number of data items that can be handled. In the experimental system (P4, 1.6G, 256M), the VISTA system can handle about 100K points and refresh the visualization in real-time (several hundreds of milliseconds). The future work will be focused on handling high-dimensional and larger datasets.

REFERENCE

[1] Keke Chen and Ling Liu: “Cluster Rendering of Skewed Datasets via Visualization”. ACM Symposium on Applied Computing 2003, Melborne, FL.

[2] W.S.Cleveland. “Visualizing Data”, AT&T Bell Laboratories, Hobart Press, NJ.1993.

[3] G.Guha, R.Rastogi, and K.Shim. “CURE: An efficient clustering algorithm for large databases”, in Proc. of the 1998 ACM SIGMOD

[4] J.Larkin and H.A.Simon. “Why a diagram is (sometimes) worth ten thousand words”, Cognitive Science, 11(1): 65-99, 1987

[5] I. S. Dhillon, D. S. Modha& W. S. Spangler. “Visualizing Class Structure of Multidimensional Data”, In Proc. of the 30th Symposium on the Interfaces, May 1998.

[6] D. Keim. “Visual Exploration of Large Data Sets”, Communications of the ACM. August 2001, V. 44. No. 8

[7] Cook, D.R, Buja, A., Cabrea, J., and Hurley, H. “Grand tour and projection pursuit”, Journal of Computational and Graphical Statistics, V23, pp. 225-250

[8] Li Yang. “Interactive Exploration of Very Large Relational Datasets through 3D Dynamic Projections”, in Proc. of SIGKDD2000

[9] E. Kandogan. “Visualizing Multi-dimensional Clusters, Trends, and Outliers using Star Coordinates”, in Proc. of SIGKDD2001.

[10] Liu, H. and Motoda, H. “Feature Extraction, Construction and Selection: A Data Mining Perspective”, Kluwer Academic Publishers, Boston, 1998.

[11] Joseph B. Kruskal and Myron Wish. “Multidimensional scaling”, SAGE publications, Beverly Hills,1978

[12] A. Jain and R.Dubes. “Algorithms for Clustering Data”, Prentice hall, Englewood Cliffs, NJ, 1988

[13] Grinstein G., Ankerst M., Keim D.A.: Visual Data Mining: Background, Applications, and Drug Discovery Applications”, Tutorial at ACM SIGKDD2002, Edmonton, Canada.

[14] Jain, A.K., Murty, M.N. and Flynn, P.J.: Data Clustering: A Review. ACM Computing Surveys, 31(3), P264-323

[15] Ester, M., Kriegel, H., Sander, J. and Xu, X. “A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”

[16] Hinneburg, A. and Keim, D. “ An Efficient Approach to Clustering in Large Multimedia Databases with Noise”, in Proc. of KDD-98, pp. 58-65

[17] Ben Shneiderman: Inventing Discovery Tools: Combinning Information Visualization With Data Mining”, Infromation Visualization 2002, 1, p5-12

[18] Maria Halkidi, Yannis Batistakis, Michalis Vazirgiannis: “Cluster Validity Methods: Part I&II”, SIGMOD Record, Vol31, No.2&3, 2002

[19] Zhexue Huang, David W.Cheung, Michael K. Ng: “An Empirical Study on the Visual Cluster Validation Method with FastMap”, Database Systems for Advanced Applications, DSFAA2001

[20]G.Sheikholeslami, S.Chatterjee, and A.Zhang. “Wavecluster: A multi-resolution clustering approach for very large spatial databases”, In Proc. VLDB98’, 1998

[21] C. Faloutsos, K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Tradditional and Multimedia Datasets," in Proc. of SIGMOD 1995

[22] Xiaowei Xu, Martin Ester, H. Kriegel, J.Sander: “A Distribution-based Clustering Algorithm for Mining in Large Spatial Databases”

[23] G. Karypis, E. Han, V. Kumar: “CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modelling”, IEEE Computer, Vol32, No8, pp68-75, 1999

[24] Jean Gallier: “Geometric methods and applications: for computer science and engineering”, Springer-Verlag, NY, c2001

[25] Sharma, S.C.: “Applied Multivariate Techniques”, John Willy&Sons, 1996.

[26] A. Hinneburg, D. Keim, and M. Wawryniuk**:** “Visual Mining of High-dimensional data”, IEEE Computer Graphics and Applications. V19, No 5, 1999

[27] M. Ankerst, M. Breunig, H. Kriegel and J. Sander: “OPTICS: Ordering Points To Identify the Clustering Structure”, in proc. of SIGMOD1999

[28] UCI machine learning database: http://www.ics.uci.edu/~mlearn/Machine-Learning.html