Cluster Rendering of Skewed Datasets via Visualization

Keke Chen Ling Liu
College of Computing, Georgia Institute of Technology
801 Atlantic Drive
Atlanta, GA 30332
Tel: 1-404-894-7008, 1-404-385-1139
{kekechen, lingliu}@cc.gatech.edu

ABSTRACT
Information Visualization is commonly recognized as a useful method for understanding sophistication in large datasets. In this paper, we introduce a flexible clustering approach with visualization techniques, aiming at the datasets that have skewed cluster distribution. This paper has three contributions. First, we propose a framework *Vista* that incorporates information visualization methods into the clustering process in order to enhance the understanding of the intermediate clustering results and allow user to revise the clustering results easily. Second, we develop a visualization model that maps multidimensional dataset to 2D visualizations while preserving or partially preserving clusters. Third, based on the visualization model, a set of operating rules are proposed to guide the user rendering clusters efficiently. Experiments show that the *Vista* system can yield lower error rates for real datasets than typical automated algorithms.

Keywords
Information visualization, clustering, interactive techniques

1. INTRODUCTION
Clustering is a common technique used in understanding and manipulating datasets, such as bioinformatic datasets and high-energy physical datasets. Many clustering algorithms have been proposed. Most of them automate the entire clustering process. What user needs to do is to set parameters at the beginning, such as the number of representative points and shrink factor in CURE [3]. Even though most of the algorithms allow people to use different input parameters, once the input arguments are set, the clustering result is produced with no interruption, leaving less flexibility to incorporate any application-specific revision into the clustering rules. Many of them are also based on the statistical evaluation methods, which assume the clusters in some normal distributions. But this assumption is not held for many real-world datasets featured by skewed cluster distribution – clusters in irregular shape and size.

SAC 2003, Melbourne, Florida, USA

A key challenge of the clustering algorithms is to reduce the clustering error rate for these skewed datasets. In order to get satisfactory clustering results, the algorithms should have some knowledge about the characteristics of cluster distribution or allow experts to incorporate the application-specific knowledge into the clustering process to yield more satisfactory results. However, it is really difficult to model different skewed cluster distributions in an efficient mathematical way.

With this problem in mind, we develop a flexible visual clustering approach. This approach has three unique features. First, it has a framework ***Vista*** that incorporates information visualization methods into the clustering process for large multidimensional datasets in order to enhance the understanding of the intermediate clustering results and increase the flexibility of the system. Second, it uses a visualization model that maps multidimensional dataset to visualizations while preserving clusters partially. This model also enables interactive operations on visualization. Third, based on the visualization model, a set of operating rules are proposed to guide the user to render the clusters more efficiently.

We organize this paper as follows: section 2 presents the related work, and a closer look at the problems of existing clustering methods and visualization methods; section 3 describes the visualization model; section 4 overviews the visual cluster rendering system *Vista*; section 5 concerns the concepts and rules for rendering clusters; section 6 shows some experimental measurement; finally, we discuss potential problems and the future work.

2. PROBLEM STATEMENT AND RELATED WORK
Data Clustering has been extensively studied over the past decade. Generally, clustering can be defined as follows: given n data points in a d-dimensional metric space, partition the data points into k clusters such that the data points within a cluster are more similar to each other than data points in different clusters. Clustering algorithms should concern about the following issues: 1) the definition of similarity of data items; 2) the characteristics of clusters, including size, shape and statistical properties; 3) the computation cost and error rate of the result.

Most of the past research on clustering has been focused on efficient and effective clustering on the datasets having regular cluster distribution, in which clusters have spherical shapes and can be represented by centroids and radiuses in an approximate way [11], but they do poorly (produce high error rate) on those

skewed datasets, which have non-spherical regular or totally irregular cluster distributions. CURE [3] realized the problem of dealing with the skewed datasets. Instead of using centriod and radius/diameter to represent a cluster, CURE uses several representative points to describe the boundary of a cluster approximately. As a result, CURE can reduce the error rate for some non-spherical regular clusters, such as clusters in the shape of ellipse or stick. However, CURE still generates high error rates over the datasets of irregular cluster distributions. In principle, the more irregular cluster distribution, the more representative points are needed to distinguish the clusters precisely. Since the user may not know how irregular the cluster distribution is, it is hard for her/him to know how many representative points are enough to precisely model the cluster boundary. In this case, it is very difficult to tune the parameters of the algorithm to find a satisfactory result.

It is well known that, given a dataset, it is possible to have more than one criterion to partition the dataset with respect to different application requirements. It is also recognized that the automated algorithms are lack of the flexibility to enable people to realize the cluster distribution and make any modification to the clustering result easily. For example, Figure 1 is a 2D projection of a 3-dimensional dataset. It shows 7 possible clusters that can also be found by some automated algorithms, such as CURE. However, if the application needs to consider B1, B2 and B3 in one cluster rather than 3 clusters, the automated algorithms are not flexible to incorporate the requirement by setting parameters only.

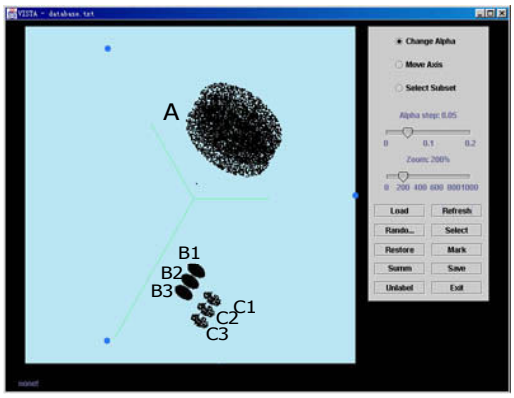

Figure 1. Ambiguity in clustering

Information visualization is commonly recognized as a useful method for understanding sophistication in large datasets. Many efforts have been made to analyze the datasets in a visual way. We mainly discuss some scatterplot based visualization techniques. 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 user to find the interesting projections. L.Yang [8] utilizes the Grand Tour technique to show projections of datasets in an animation. They project 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[4] 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, and prosection [2] create static visualization only, which distort the datasets more or less, thus do not provide enough information for correct clustering.

Star Coordinates [9] is a visualization system designed to visualize and analyze the clusters interactively. The mapping function is, to some extent, similar to that in our *Vista* cluster rendering system. But *Vista* system has some unique features. First of all, *Vista* system normalizes the data space so that the point arrangement utilizes the visual space more effectively. Second, *Vista* introduces the concept of α adjustment into the visual operation, and implements α widgets, which enable users to interact with visualization more efficiently. Last but not least, *Vista* explores the characteristics of dimension by dimension rendering and gives some practical guidelines to help users find satisfactory cluster visualization efficiently.

3. CREATING INTERACTIVE VISUALIZATION: THE *Vista* MODEL

Detecting clusters in visualization also brings up three particular problems. The most important one is how to visualize the clusters without introducing too much visual bias. This is known as the problem of cluster preserving. Usually, the visual bias can not be avoided because the mapping from a high dimensional space to a low dimensional space must introduce some errors into the visualization. Thus, the second problem is how to design the interactive operations, with which we can improve the quality of the clusters in visualization. The third problem is how to improve the efficiency of human-computer interactions. Human-computer interactions are usually slower than the automated algorithms. However, we can still try out best to improve the performance of human-computer interactions.

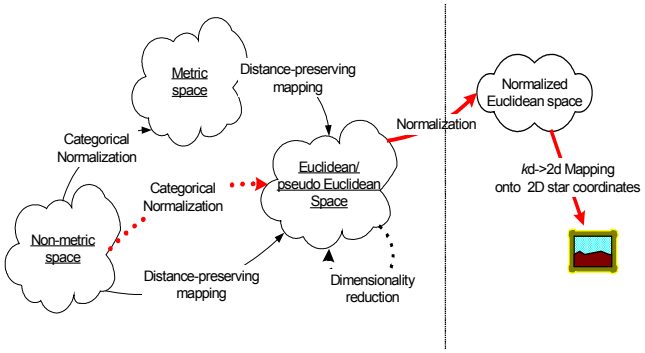

Figure 2: Transformations in *Vista* visualization model

We address the above visualization problems in following way. In the first version of *Vista*, we use the scatterplot like visualization because we believe it is the most intuitive way to visualize clusters. However, the advantage does not come for free. With different mapping model, scatterplot visualization may introduce two kinds of visual bias: it may break a cluster into two clusters or may overlap two clusters as seen in one. In *Vista*, a visualization model that maps raw dataset onto 2D star coordinate is used to avoid breaking clusters, and a set of interactive operations are used to improve the cluster quality or distinguish the overlapped clusters. We address the efficiency problem by introducing a set of operating rules that can help users either in unsupervised or in supervised cluster rendering.

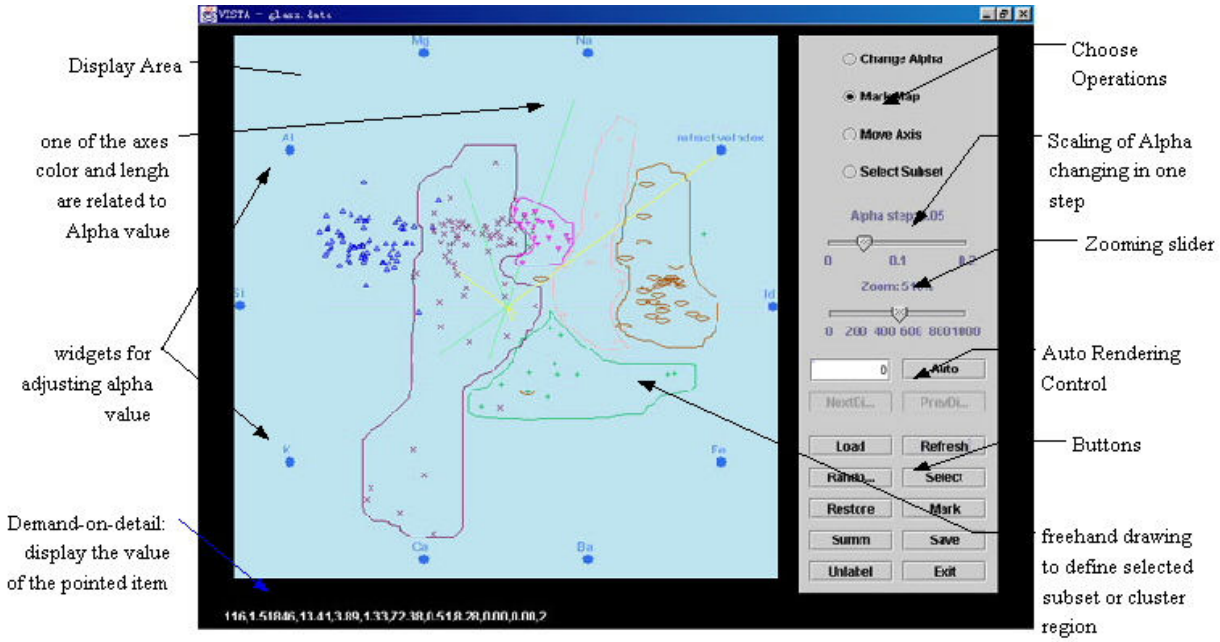

Figure 4: User interface of *Vista* system

The key idea in *Vista* visualization model is to transfer different distances into Euclidean distance and then map the Euclidean distance to visible 2D Euclidean distance while preserving distance relationship between points. However, it is inevitable to introduce visual bias when mapping *k*-d dataset to 2d dataset, such as overlapped clusters. We want to provide a highly interactive visualization that enables user to manipulate the visualization with visually adjustable parameter on each dimension to reveal all clusters.

We consider five possible data spaces involved in our framework: non-metric, metric, Euclidean, normalized Euclidean, and 2D visual Euclidean space. Datasets that contains non-numeric data, such as categorical or Boolean data, are in non-metric space. Non-metric space can be normalized to metric space. Many practical datasets are defined in Euclidean space. However, non-Euclidean datasets can also be transformed to Euclidean datasets if distances between points are given. Normalized Euclidean space contains only numeric data located in [0, 1] that is used to create visualization in scalable space further. In the first version of *Vista* system, we enhance the mapping used in Star Coordinate [9] with the α adjustment and max-min normalization, and name it α-mapping. A Fastmap [6] based dimensionality reduction is integrated into this version. A simple hashing based categorical normalization is also effective for some datasets but the meaning of distance becomes imprecise when it is applied. We mainly concern the Euclidean space in the sketch of this model, thus, the categorical normalization is excluded form the main transformation chain in Figure 2, where the main transformation chain is **Fastmap ⊕ max-min normalization ⊕ α-mapping,** which has an important **property**: The main transformation chain, Fastmap ⊕ (max-min normalization ⊕ α-mapping), can preserve cluster structure partially.

Given the facts that, 1) Fastmap can preserve the distances approximately [6]. 2) We have shown that the α-mapping is a linear mapping in [1], given constants α_i , i =1, 2, …k. 3) The max-min normalization is a linear transformation. 4) The composite of linear mapping (max-min normalization ⊕ α-mapping) is still linear mapping, which can be found in any linear algebra textbook. 5) the linear mapping has an important property that it does not break clusters but may cause cluster overlapping [5], which means it can preserve the cluster structure partially. Therefore, the main transformation chain preserves the cluster structure partially. Due to the space limitation, more discussion about the visualization model can be found in [1].

4. *Vista*: AN OVERVIEW

4.1 The system Framework

Based on the mapping model, we build an interactive visual clustering system - *Vista*. Our system is designed for Euclidean datasets in current version, so *FastMap* is used to reduce dimensionality only. This system can also be used to import clustering results produced by existing algorithms to understand the relationship between clusters and then label the whole large dataset in a fast and reliable way. Due to the limitation of system capacity, *Vista* can visualize a limit number of points in real time (refreshed in less then 10ms), for example in a PIII700MH 256M PC, the number is 50000 rows. The number of dimensions is actually limited by capability of human visual understanding. We recommend visualizing the datasets having less than 50 dimensions with *Vista* system. Larger datasets can take advantage of sampling techniques to reduce the size and dimensionality reduction techniques to reduce the dimensionality.

4.2 GUI and Interactive Operations

We designed a set of interactive operations for the *Vista* cluster rendering system. The goal of interactive operations is to observe the dataset from different projection planes and help users to recognize those overlapped clusters or falsely clustered outliers. A set of interactive operations is provided in the *Vista* cluster rendering system. Due to the space limitation, we only discuss

four operations, which are α parameter adjustment, subset selection by region, subset selection by range selection and axis rotation. The complete introduction can be found at http://disl.cc.gatech.edu/*VISTA*.

α Adjustment

α adjustment allows user to change the values of α parameters in Γ mapping[1], which are represented by axis length(the scale of the absolute α value |α|), and axis color (green for positive values and yellow for negative values). Changing α value of one attribute increases or decreases the contribution of this particular attribute on the resultant visualization. In the first version, we also allow users to visualize the datasets that contain few categorical data attributes, by hashing the categorical data to a number in 0..*n*, where *n* is the number of all categories in the attribute. Then the numeric representation of categories is normalized to [0, 1] as same as the other numeric attributes. Groups classified by a particular attribute, regardless of numeric or numeric representation of categories, can be obviously observed when adjusting the corresponding α value. We will discuss this particular property in next section. A visualization example of web log dataset provided by Jim Gray [12] shown as in the Figure 5, the user scales up the "ClientIP" attribute from the initial layout to examine the distribution of the data points belonging to different ClientIP. With α adjustment only, we can see there are 2 groups.

α adjustment is also a natural way to interact with hierarchically expanding and collapsing clusters. In Figure 5b, after scaling up the "ClientIP" attribute, a subsequent scaling of "Browser" attribute shows there are 3 sub-clusters (MSIE, Mozilla and Mozilla for NT) in the group Client2, but there is no more sub-cluster in the group Client1 (MSIE only).

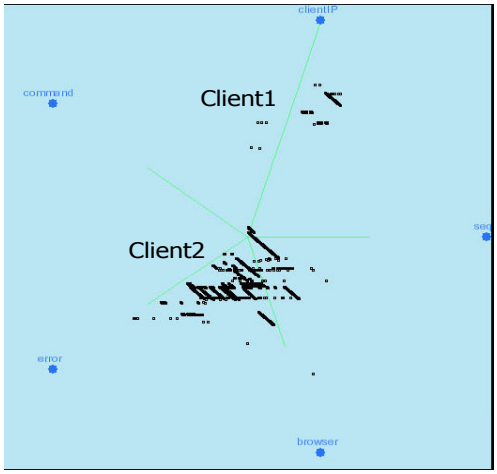

Figure 5a: Maximize the α value of ClientIP after loading the web log dataset

To apply α adjustment, users operate on the α widgets, which are blue and at the end of each α axis. After loading a dataset, the axes (i = 1, 2, … , k), which correspond to the dimensions, are created, and the initial (i = 1, 2, … , k) (one per dimension) are set to 0.5. By pressing and holding the mouse button on α widget, the corresponding α value is increased, and by pressing mouse button plus "Ctrl" key the α value will be decreased.

Axis Rotation

Axis rotation changes the direction of an axis, thus making a particular data attribute more or less correlated with other attributes. When several axes are rotated to the same direction, their contribution to the resultant visualization is aggregated. To contrast the effects of two sets of interesting attributes, for example, one is regarded as a positive set and the other as a negative set, the user can place the two sets in two reverse directions and deploy the other undesirable attributes in the perpendicular directions. For example, in year-1984 house voting dataset we want to observe the trade-off between the voting to the education expense (the attribute: education-spending) and the military/communication budgets (attributes: anti-satellite-test-ban and mx-missile). We rotate the axis (#13 education-spending) to the up-right corner and the other two (#8 anti-satellite-test-ban and #10 mx-missile) to the bottom-left corner. Also we put other axes to the perpendicular positions. The visualization shows that, in most congressmen's mind, the military/communication budgets overwhelm the education spending.

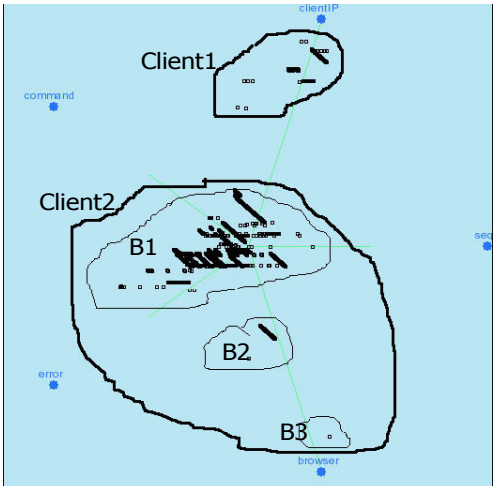

Figure 5b: Scaling up the "browser" attribute

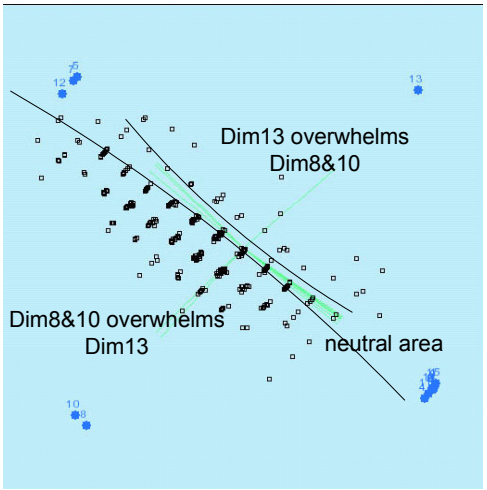

Figure 6: Axis rotation of the "voting" dataset

Axis rotation can also be used to observe the relationship between any two attributes in a precise way. The user can rotate the target

axes to two perpendicular directions and then set the α values of other attributes to 0, which produces a standard 2D plot. For example, we want to observe the relationship between the attributes "Command" and "Error Code" in the web log dataset, where the user can find which commands cause a certain error code, such as which commands causes the "500" error code, which implies that the web user tries to hack the web server system.

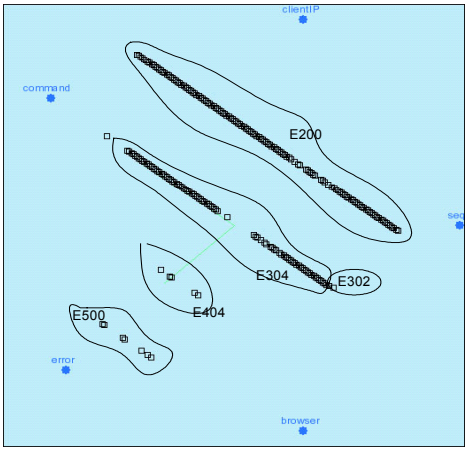

Figure 7: Axis rotation of the web log dataset

Subset Selection by Region

Vista system enables subset selection by free-drawing the enclosed region you want to observe. To begin subset selection, firstly the user needs to shift the operation mode to "subset selection". To draw the selection boundary, click the mouse on any beginning point on the boundary, then drag mouse to enclose the boundary. The ending point does not need to be exactly on the beginning point – subset selection will automatically enclose the area by connecting the ending point and the beginning point with a line segment. Subset selection can also be done on the selected subset, which enables the user to observe the dataset in a hierarchical way to focus on in-depth details of part of a dataset. Subset selection combining zooming is especially useful when exploring large amount of data points in a dense area. The example is the cluster rendering of dataset DS1-3D. The dataset has five clusters – one big spherical, two small spherical and two connected elliptical clusters. In the initial visualization (Figure 10a), we can see three clusters. The other two clusters may be hidden in the largest cluster. We select and zoom in the large cluster to see if there are other hidden clusters (Figure 10b). By using α parameter adjustment and zooming, together with subset selection, we can easily observe the detail of any section. Figure 10c shows the detail of the select largest cluster in Figure 10b. In Figure 10c, we can see there are two overlapped clusters hidden in the large cluster. By adjusting the α values of different dimensions, we can move the overlapped clusters out of the large cluster. Finally, zooming out and restoring the shaded data points, we find the five clusters in visualization.

Subset Selection by Range

Range selection is a kind of subset selection, which selects a subset by specifying a range of any one attribute. In current version, we only implement range selection of one attribute. In the future version, the *Vista* cluster rendering system can combine ranges of different attributes with AND or OR to show more complicated situations. To activate the range selection, click right mouse button on an alpha widget. The range selection dialog box will appear at the top-right corner of the screen. In the extended functions, since we allow the user to explore datasets that contains categorical data attributes, we provide two kinds of range selection interface for numerical and categorical data, respectively. When the user changes the numerical range or category selections, the corresponding selected data points are shown in red color. The user, then, can do all of the operations that require doing subset selection in priori.

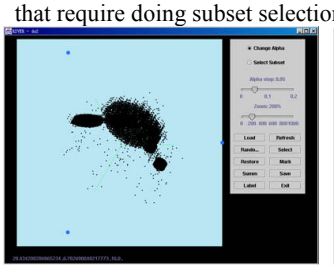

Fig8a.The initial visualization

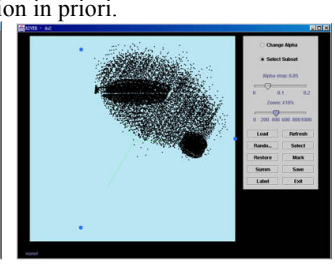

Fig8b. Select subset and zoom in Two clusters inside

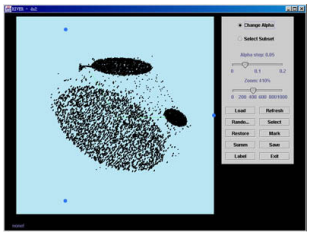

Fig8c. Adjust parameters

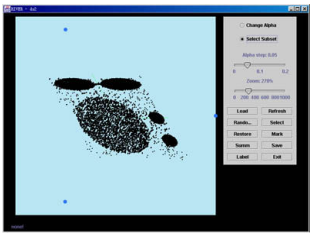

Fig 8d. Restore and zoom out

5. RENDERING CLUSTERS: GUIDELINES FOR USING *Vista* EFFICIENTLY

Interactive visualization systems usually have a problem of training users to use it. If the exploration process is too complicated or too tedious to use, I believe no user will like to use it. The main advantage of the *Vista* cluster rendering system is that the user can follow some guidelines we derived from the *Vista* visualization model to render the clusters efficiently and systematically. In this section, we will briefly discuss these operation guidelines and related concepts. In the progress of understanding of the *Vista* visualization model and interactive operations, the user can actually discover or design more rules by themselves.

There are two kinds of rendering used in *Vista* system, unsupervised and supervised. Unsupervised cluster rendering does not rely on any cluster labels or other knowledge of the dataset. The user tries to find the satisfactory cluster visualization by observing the clusters emerging when applying a series of interactive operations. The shape, density and boundary of a cluster become the main factors for the user to make decision. Supervised rendering uses the existing cluster labels as clue to find a satisfactory visualization that can separate the labelled clusters approximately. It is much easier and more reliable than unsupervised rendering. The first typical operation for supervised rendering is loading the cluster label set. The user then can see the points in different clusters labelled in different colors and shapes. With this label information, user can use more flexible strategies

in the various situations. Due to the space limitation, we discuss guidelines for unsupervised rendering only. The additional supervised rendering guidelines can be found in [1].

5.1 Concepts and Guidelines for Unsupervised Cluster Rendering

In the previous discussion about the *Vista* visualization model, we know that dense point clouds in a static visualization partially imply the real clusters in a dataset because there are also false clusters formed by overlapping. We want to confirm the "clusters" or separate the possible "clusters" to find the satisfactory cluster visualization by applying the interactive operations, which have been introduced in the interactive techniques section. However, it is still kind of difficult for a new user, who does not know the whole visualization model thoroughly, to apply a sequence of interactive operations to find some satisfactory visualization. In this section, we describe relations between the "cluster behaviours" in visualization and the visualization model first, and then propose some rules that suggest when and what operation can be used towards the goal of finding well-separated cluster visualization.

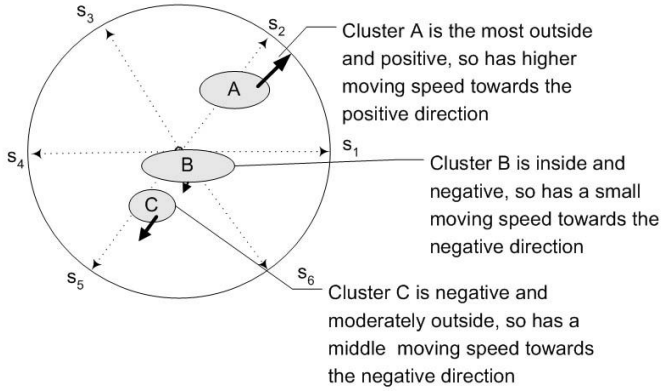

Figure 9: The "cluster behaviours" when the α parameter of dimension S_2 is positive and increasing.

We mainly consider the datasets that use Euclidean distance as similarity function, which is widely used for many applications. Most of such datasets have very specific meanings for each dimension, and thus two similar data items in a dataset not only have very small Euclidean distance, but also imply little difference over each main dimension that dominates the distance. Our following discussion will be based on this assumption or observation, which makes it possible to explore such datasets in a dimension-by-dimension way.

We define the term "cluster behaviour" first. A continuous change of visualization parameters, mainly α parameters, produces a continuous animation in visualization. In such an animation, the user may find a group of data points moving together. This movement, we called "cluster behaviour", provides the important information to verify the existing "clusters" in visualization. "Cluster behaviour" of a cluster can be defined as ($\vec{m}$, *c, d*, *s*). $\vec{m}$ defines the movement of the cluster, which represents the speed and the direction of the movement. *c* means the clarity of the cluster, e.g. the lower the percentage of overlapped area with other clusters, the higher the clarity of the cluster. *d* is the density of the cluster and *s* is the area the cluster covered in visualization.

We prefer clusters having high clarity and high density. When a α parameter is changed, more than one "cluster" probably emerge. The contrast of the "cluster behaviours" of the set of "clusters" in the animation is very useful to determine the effect of the one dimension to the entire visualization. We expect the local "cluster behaviours" can help to find a global satisfactory visualization.

There are some typical "cluster behaviours". When a α parameter is changed, the possible clusters along this changed dimension may move in three different ways. A part of the points does not move or move a little, which probably distribute around the centre of visualization. The second part of the points moves to the positive direction and the third part of the points moves to the negative direction. We name the stable clusters around the visualization centre as the inside clusters, and the clusters far away from the centre as the positive/negative outside clusters as in Figure 11. In terms of one dimension s_i, from the Γ mapping function: $\overrightarrow{cq} = (1/k)\sum_{1 \le i \le k} \alpha_i x_i(p) \overrightarrow{cs_i}$ [1] the points having value in dimension *i* that are close to 0.5 – the middle of the value range of this dimension, belong to the stable cluster. Respectively, the points having $x_i(p)$ greater than 0.5 are in positive outside clusters, and the points having $x_i(p)$ less than 0.5 are in negative outside clusters. There is an observation:

Observation: *When the α parameter is positive and increasing, the inside clusters stay stable, which have very low moving speed* $|\vec{m}|$*, while the outside clusters move along the two reverse directions. The farther away from the visualization centre the clusters, the faster they move.*

Some dimensions/attributes can form groups by itself only. As we discussed, by adjusting α parameter only, some inside and outside clusters emerge. But other attributes do not contribute much to the cluster visualization, such as sequence number and time stamp. We call the dimensions that produce clear "cluster behaviours", the "main contributing attributes". We name those that can distinguish a part of points, the "minor contributing attributes", and those that do not affect the visualization, the "non-contributing attributes". The main contributing attributes are used to produce the skeleton of cluster distribution. The minor contributing attributes are applied to polish the skeleton. Non-contributing attributes do nothing to the visualization – when the α parameter of a non contributing attribute is changed, all of the points move together, so we just omit these attributes in the rendering process. From the properties of visualization model and our experience of using the *Vista* tool, we derived some guidelines for unsupervised cluster rendering (GUR).

GUR1: *Render the visualization in an order of dimension, such as in counter-clockwise direction, beginning at the dimension 0. You can also use the automated rendering function to make the process easier.*

GUR2:*When rendering the visualization in an order of dimension, find the main contributing dimensions and maximize α parameters of some, either to 1 or -1, to separate all clusters as possible.*

GUR3:*Use the minor contributing dimensions to polish the clusters got by applying rule3. They are used to increase the cohesion of clusters and make the boundary of clusters more clear.*

GUR4:*When we capture a cluster in visualization, we consider the density of the cluster first, the clarity, and then the size of the*

cluster. We select the clusters having the densest point cloud and the clearest boundary first.

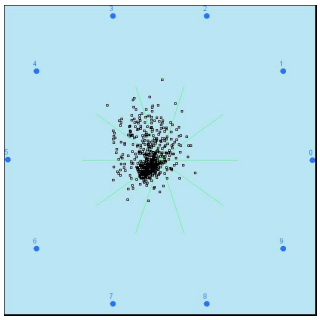

Figure 10a: load the dataset

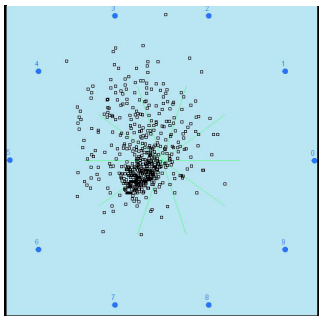

Figure 10b: Zoom in

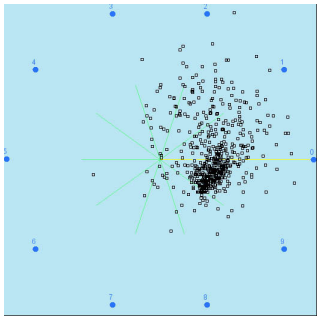

Figure 10c: adjust α value of dimension 0, the distribution is not changed

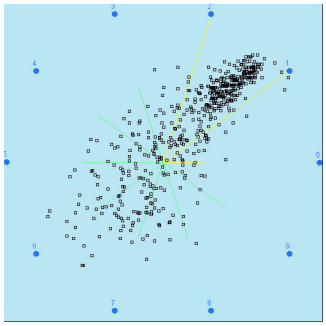

Figure 10d: adjust main contributing dimensions 1 and 2 to separate the potential clusters.

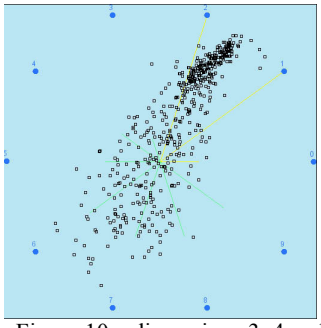

Figure 10e: dimensions 3, 4 and 5 are minor contributing attributes

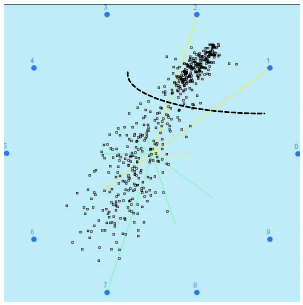

Figure 10f: adjust main contributing 6 and 7. A clear boundary appears

Example: unsupervised rendering for the "Wisconsin breast cancer" dataset. This dataset has 10 dimensions (all are numeric) and 699 instances, which can be found in UCI machine learning database (www.ics.uci.edu). This dataset has 2 clusters inside. We try to find the 2 clusters by applying the guidelines we introduced above. Finally, we contrast the result by loading the original labels to see how precise the result is (Figure 10h).

Step1: Load the dataset and zoom in to get a clear visualization. A possible cluster is in the center of the visualization. (Figure 10a,b)

Step2: adjust dimension 0, we found it is a non-contributing attribute (Figure 10c)

Step3: adjust dimension 1 and 2. They are main contributing attributes. We adjust the α values of them to -1 so that all possible clusters are separated. One cluster emerges (Figure 10d)

Step4: dimension 3, 4 and 5 are minor contributing attributes. We use them to polish the boundary later (Figure 10e).

Step5: dimension 6 and 7 are main contributing attributes again. We use them to make the boundary clearer (Figure 10f).

Step6: dimension 8 and 9 are minor contributing attributes again. They are used to polish the density and boundary of the two clusters (Figure 10g).

Step7: adjust the dimensions in another iteration to determine the satisfactory visualization, if necessary.

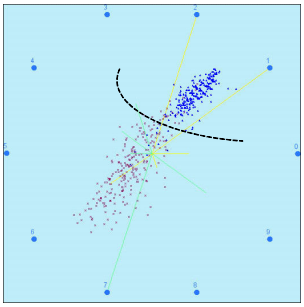

Figure 10h: load the labels. We see the boundary separate the two clusters well

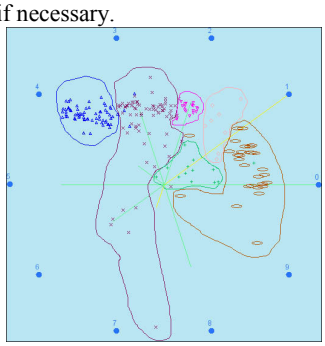

Figure 11: final visualization of "glass" dataset with guidelines for supervised rendering

6. EXPERIMENTAL RESULTS

The *Vista* visual clustering system was implemented in JDK1.2. We have done bunch of experiments on the system capacity and the system usability. This system running in the machine of PIII700 and 256M memory can deal with the datasets having up to 50000 rows and 50 dimensions in real time (less than 20ms), which means that human can see the smooth change of the visualization when the interactive operations are performed. The limitation of dimensionality comes from the deployment of the axes and the limitation of human understanding and operating. Too many dimensions not only congest the visualization but also leave too much visual burden to the users. In terms of system usability, two aspects are investigated – how easy a new user can learn to use and how precise the result can be. With the guidelines discussed in section 5, a new user trained in 1hour can find satisfactory visualizations for most experimental datasets. Due to the space limitation, we will mainly discuss the usability experiments that show the *Vista* cluster rendering system can yield lower error rate clustering results than the typical automated algorithm CURE. Our experiments were conducted on a number of well-known datasets that can be found in UCI machine learning database (www.ics.uci.edu). These datasets, although small in size, have irregular cluster distribution, which is an important factor for testing the usability of the *Vista* system.

It is well known that CURE can give better results than other existing automated algorithms by recognizing clusters in non-spherical regular shapes. To show how effective the *Vista* visual clustering is in terms of error rate reduction, we choose to compare the error rates of the *Vista* visual clustering with the CURE clustering algorithm. Two kinds of visual clustering methods are used in our experiments, one is unsupervised process and the other is supervised, which are discussed in previous section. In unsupervised visual clustering process, the visualization may trap into local minima that does not reflect actual cluster distribution very well, thus may lead to a higher error rate than the supervised clustering. In the supervised visual

clustering process, the training datasets are labelled and the items in different clusters are visualized in different colors. The user's responsibility is to find a good visualization that distinguishes the labelled clusters in different regions. The labelled data may provide enough hints to avoid false partitions and direct the user to find a satisfactory visualization much quicker. Because of these advantages the supervised visual clustering usually gives better results than the unsupervised visual clustering.

Table 1 shows the experiments on usability of *Vista* over several well-known datasets. Interesting to note is that it takes a trained user 5 mins to 20 mins to find satisfactory visualizations for these datasets. The supervised clustering also costs less than the unsupervised clustering. The experiments show that *Vista* usually offers lower error rates than CURE on these real datasets that have irregular cluster distribution. Some of the datasets, such as "mushroom", also contain categorical attributes. We use the simple hashing based categorical normalization (section 4.2), with which the results are satisfactory.

Dataset	*N*	*k*	*Vista*(%)		CURE(%)
			U.Su	Su	
breast-cancer-wisc	699	10	16.7	4.3	36.6
Crx	690	15	20.2	14.5	31.7
Hepatitis	155	19	21.9	20.6	41.3
Votes	435	16	25.5	5.5	38.2
Iris	151	4	5.5	3.3	35.7
page-blocks	5473	10	13.0	7.0	53.4
Heart	270	12	24.0	16.7	49.6
Mushroom	8124	21	24.7	2.5	36.8
Australian	690	14	15.4	14.4	35.7
Wine	178	12	7.9	6.2	34.3
Shuttle	14500	9	10.2	4.2	17.5

Table 1: Experiments on usability of *Vista* comparing error rates of *Vista* with CURE, *N* is the number of rows in dataset, *k* is dimensionality, U.Su. is the abbreviation of unsupervised, and Su is supervised.

7. CONCLUSION

We present the *Vista* approach for efficient and flexible clustering with visualization techniques. Our approach has several significant features: 1) the visualization model that is proved to preserve the cluster distribution partially; 2) a bunch of interactive techniques are designed to distinguish the false clusters or overlapped clusters and to improve the quality of clusters, too; 3) based on the analysis of the *Vista* visualization model, a set of operating rules are proposed to help a user find the satisfactory visualization quickly in supervised or unsupervised cluster rendering. Some problems related to our *Vista* system are also discussed. The first prototype of *Vista* is available at disl.cc.gatech.edu/*VISTA*.

Our research on *Vista* continues along two directions. On the theoretical side, we are interested in investigating issues how to deal with large high-dimensional datasets, and the enhancement about visualization model, such as dimensionality reduction and visualization of categorical/Boolean datasets. On the practical side, we are interested in tailoring the capabilities of *Vista* to some application-specific datasets, such as, large-scale web log datasets and datasets in intrusion detection systems.

8. REFERENCE

[1] Keke Chen and Ling Liu. "*Vista*: a fast and flexible clustering approach for very large multidimensional datasets". Technical Report TR-02-03-2002, http://disl.cc.gatech.edu/*VISTA*

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

[5] I. S. Dhillon, D. S. Modha& W. S. Spangler. "Visualizing Class Structure of Multidimensional Data", In Proc. of the 30th Symposium on the Interface: Computing Science and Statistics, May 1998.

[6] C. Faloutsos, K. Lin, "FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Tradditional and Multimedia Datasets," in Proceedings of 1995 ACM SIGMOD, vol.24, no.2, p 163-174.

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

[11] A. Jain and R.Dubes. "Algorithms for Clustering Data", Prentice hall, Englewood Cliffs, NJ, 1988

[12] Jim Gray, http://skyserver.sdss.org/en/tools/search/sql.asp

AUTHOR BIOGRAPHY

Keke Chen is a PhD student at College of Computing, Georgia Tech. He received his B.S. degree in Computer Science from Tongji University, China. in 1996 and M.S. degree in Computer Science from Zhejiang, China in 1999. His research interests include databases, information visualization, and distributed/mobile computing.

Ling Liu is an associate professor at College of Computing, Georgia Tech. She received her PhD in Computer Science from Tilburg University, Netherlands. Before joining GT, she was on the faculty at the Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology from 1997-1999, and on the faculty at the Department of Computer Science, University of Alberta from 1994 - 1998. She is currently a member of ACM and IEEE Computer Society. Her research interests are in the areas of distributed data intensive systems and in particular distributed middleware systems, advanced Internet application developments, and Internet data management