DENSITY BASED CLUSTERING USING MATHEMATICAL MORPHOLOGY

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

COŞKU ERDEM

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF INFORMATION SYSTEMS

DECEMBER 2006

ABSTRACT

DENSITY BASED CLUSTERING USING MATHEMATICAL MORPHOLOGY

Erdem, Coşku
M.Sc., Department of Information Systems
Supervisor: Assoc. Prof. Dr. Yasemin YARDIMCI

December 2006, 79 pages

Improvements in technology, enables us to store large amounts of data in warehouses. In parallel, the need for processing this vast amount of raw data and translating it into interpretable information also increases. A commonly used solution method for the described problem in data mining is clustering. We propose "Density Based Clustering Using Mathematical Morphology" (DBCM) algorithm as an effective clustering method for extracting arbitrary shaped clusters of noisy numerical data in a reasonable time. This algorithm is predicated on the analogy between images and data warehouses. It applies grayscale morphology which is an image processing technique on multidimensional data. In this study we evaluated the performance of the proposed algorithm on both synthetic and real data and observed that the algorithm produces successful and interpretable results with appropriate parameters. In addition, we computed the computational complexity to be linear on number of data points for low dimensional data and exponential on number of dimensions for high dimensional data mainly due to the morphology operations.

Keywords: Data mining, Clustering, Mathematical Morphology

ÖZ

MATEMATİKSEL MORFOLOJİ KULLANARAK YOĞUNLUK BAZLI KÜMELEME

Erdem, Coşku
Yüksek Lisans, Enformatik Enstitüsü
Tez Yöneticisi: Doç. Dr. Yasemin YARDIMCI

Aralık 2006, 79 sayfa

İlerleyen teknoloji hızlanarak artan miktarda veriyi veri depolarında saklayabilmemize olanak sağlamaktatır. Beraberinde bu çok büyük miktardaki ham verinin işlenerek yorumlanabilir bilgiye dönüştürülme ihtiyacı da büyümektedir. Veri madenciliğinde tariflenen problemin sıkça başvurulan çözüm metodlarından biri de kümelemedir. Gürültülü numerik bir verinin içindeki farklı şekillere sahip kümelerin makul süreler içerisinde belirlenebilmesi için etkin bir kümeleme metodu olarak "Matematiksel Morfoloji Kullanarak Yoğunluk Bazlı Kümeleme" algoritmasını teklif ediyoruz. Bu algoritma veri depolarının imgelere benzerliğinden yola çıkarak bir imge işleme tekniği olan gri tonlu morfolojinin çok boyutlu veri üzerine uygulanması temeline dayanmaktadır. Bu çalışmada, önerilen algoritmanın gerek sentetik gerekse doğal veri üzerindeki başarımını değerlendirdik ve uygun parametrelerle çalıştırıldığında başarılı ve yorumlanabilir sonuçlar üretebildiğini gördük. Ek olarak, algoritmamızın işlemsel karmaşıklığının düşük boyutlu veri için veri noktası sayısı ile doğrusal, yüksek boyutlu veri içinse temelde morfoloji işlemlerine bağlı olarak boyut sayısı ile üstel olarak artığını hesapladık.

Anahtar Kelimeler: Veri Madenciliği, Kümeleme, Matematiksel Morfoloji

ACKNOWLEGMENTS

It is a pleasure for me to express my sincere gratitude to my supervisor Assoc. Prof. Dr. Yasemin YARDIMCI for her patience, encouragement and guidance throughout the study. I greatly appreciate her share in every step taken in the development of the thesis. I also thank Assoc. Prof. Dr. Şebnem DÜZGÜN and Assist. Prof. Dr. Tuğba TEMİZEL for their positive criticisms.

Moral assistance of my sister Ece and my parents is also gratefully acknowledged. Thanks to my managers Dilek ÇAMLI and Ahmet Onur MİRASOĞLU for their faith in me and supports during my studies.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

I.1. Motivation and Problem Definition

In the information age enormous amount of data is accumulated in large data warehouses. As the amount of data collected grows exponentially, mining of raw data for knowledge discovery is becoming more important.

Although some of presently existing clustering algorithms claim they can process large datasets in a reasonable duration and produce useful results, there exists massive data with different characteristics where such techniques are still redeemed inadequate for several applications and cluster analysis is still a quite challenging activity. At this point, we think that it is possible to utilize some low complexity algorithms used in image processing for cluster analysis under the assumption that large data warehouses may be considered as multidimensional images. In particular, assuming every column in a data table as a dimension of a virtual space, we basically get a multidimensional binary image when we represent each data point in this virtual space. This analogy between images and data warehouses may enable us to use image processing algorithms on data warehouses. Our objective in this study is to evaluate the potential of mathematical morphology for clustering and to see whether we can process large amount of multidimensional data efficiently using this method.

Mathematical Morphology is a frequently used technique in image processing. Some of its main uses are extracting edges of the objects, figuring out skeleton of an object, determining convex hull for an object, etc. In this study, we propose using mathematical morphology for preprocessing and extracting arbitrary shaped clusters in large data warehouses. We would also like to demonstrate the strength of this procedure on a large, multidimensional, noisy dataset.

For the specific clustering technique, we make suggestions for selecting parameters based on the nature and characteristics of the data and the application.

We implemented the proposed method using Java. Open source VisAD library is used for 3D demonstration of results. Finally, we tried to evaluate the strengths and weaknesses of the procedure developed and its applicability for different date sets using the outcomes of experiments.

I.2. Mathematical Morphology

Mathematical morphology is a branch of digital image processing and analysis, which uses concepts from algebra (set theory, complete lattices) and geometry (translation, distance, convexity). It originates from the work of Matheron [1] and Serra [2], both researchers at the Paris School of Mines in Fontainebleau, who worked on problems in petrography and mineralogy. Their objective was to characterize physical or mechanical properties of certain materials (sections of rocks, polycrystalline ceramics), such as the permeability of porous media, by examining the geometrical structure. Due to their pioneering work, mathematical morphology has achieved the status of a powerful tool in image processing with applications in materials science, microscopic imaging, pattern recognition, medical imaging, and even computer vision. Also, its theoretical foundations have been well established during the last ten years.[1]

The basics of morphology are covered in books by Serra [2, 3], Giardina and Dougherty and [4], the tutorial paper by Maragos [5], by Haralick et al [6], and the book by Heijmans [7].[2]

I.2.1. Dilation and Erosion

Mathematical morphology is based on set theory. Sets in mathematical morphology corresponds to regions in an image. There are two basic morphological operations, which constitute a base for other morphological operations: dilation and erosion. These basic operations are defined using translation and reflection of a set.

Assume, A is an ordinary set of vectors, x is a vector and '+' sign represents the vectorial sum. Then the translation of set A by vector $x = (x1, x2)$ is given by,

$$(A)_x = \{ b \mid b = a + x,\ for\ a \in A \} \qquad (I.1)$$

1 Mathematical Morphology: Basic Principles. Heijmans [8]

2 Digital Image Processing. Gonzales & Woods [9]

Reflection of set *S* is defined as,

$$S' = \{ x \mid x = -s, \ for\ s \in S \} \qquad (I.2)$$

Basic morphological operations, dilation and erosion, use two operands: The set *A* and the "Structuring Element" *S*, which is also a set of vectors and determines the precise details of the effect of the operator on the input image.

➢ ***Binary Dilation***

Dilation of set *A* by structuring element *S* is defined as,

$$A \oplus S = \{ x \mid (S')_x \cap A \neq \emptyset \} \qquad (I.3)$$

➢ ***Binary Erosion***

Erosion of *A* by *S* is defined as,

$$A \ominus S = \{ x \mid (S')_x \subseteq A \} \qquad (I.4)$$

The dilation and erosion operations expand and shrink an image, respectively. This is demonstrated in Figure I.1.

I.2.2. Opening and Closing

There are two other important operations derived from dilation and erosion: "Opening" and "Closing". "Opening generally smoothens the contour of an image, breaks narrow isthmuses, and eliminates thin protrusions. Closing also tends to smooth sections of contours but, as opposed to opening, it generally fuses narrow breaks and long thin gulfs, eliminates small holes, and fills gaps in the contour."[3]

➢ ***Opening***

The opening of set *A* by structuring element *S*, denoted $A \circ S$, is defined as,

$$A \circ S = (A \ominus S) \oplus S \qquad (I.5)$$

which implies that the opening of *A* by *S* is simply the erosion of *A* by *S*, followed by a dilation of the result by *S*.

3 Digital Image Processing : Gonzales & Woods [9]

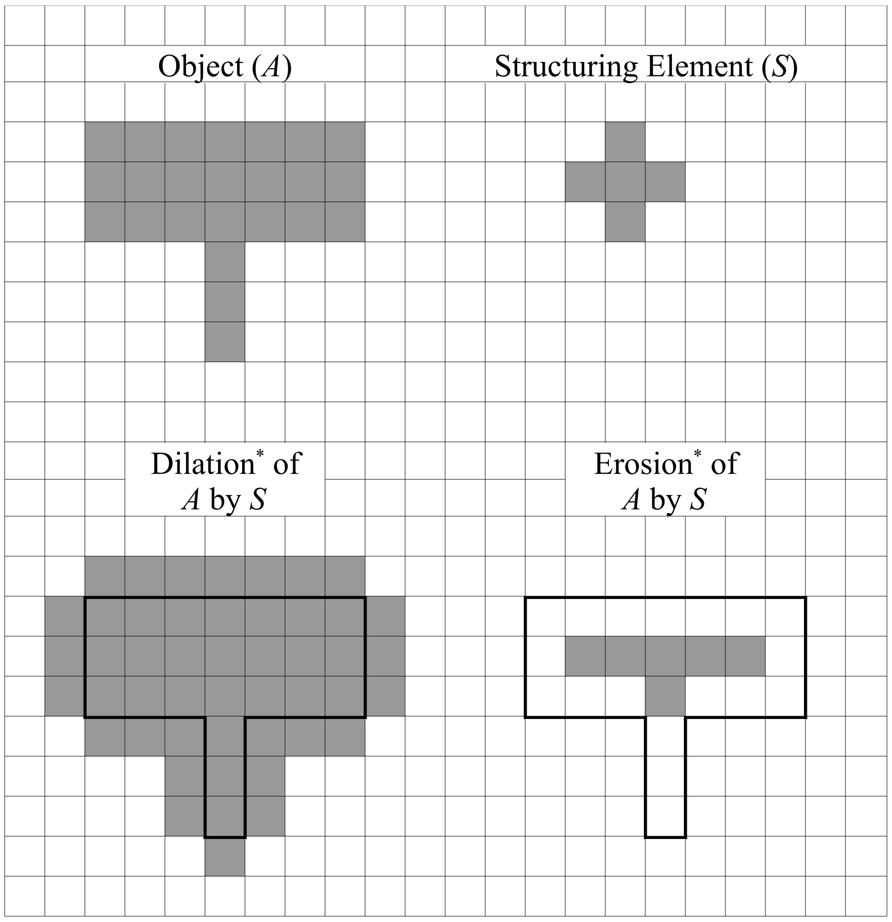


Figure I.1: Graphical Interpretation of Binary Dilation and Erosion for a two dimensional object.

➢ ***Closing***

The closing of set A by structuring element S, denoted $A \bullet S$, is defined as,

$$A \cdot S = (A \oplus S) \ominus S \qquad (I.6)$$

which says that the closing of A by S is the dilation of A by S, followed by the erosion of the result by S.

I.2.3. Grayscale Morphology

Since all other morphological operations including opening and closing are derived from dilation and erosion, extending only these two basic operations is sufficient for us to utilize all morphological operations on grayscale images. Digital input image and structuring element are denoted by *f(x,y)* and *s(x,y)*, respectively. In

* Original object boundary is depicted by thick lines.

this notation *f* and *s* are functions that return intensity value for each distinct pair of coordinates *(x,y)*.

- ***Grayscale Dilation***

Grayscale Dilation of input image *f(x, y)* by structuring element *s(x, y)* is defined as,

$$(f \oplus s)(p,q) = \max\{ f(p-x, q-y) + s(x,y) \mid (p-x),(q-y) \in D_f ; (x,y) \in D_s \} \quad (I.7)$$

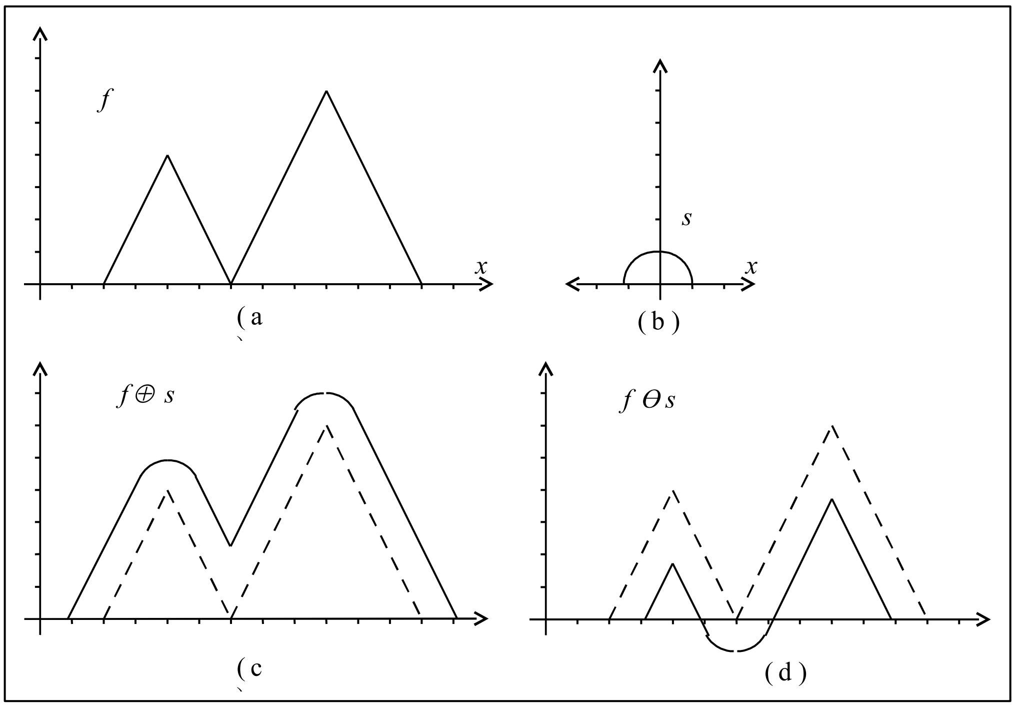
where D_f and D_s are the domains of *f* and *s* respectively.

- ***Grayscale Erosion***

Grayscale Erosion of input image *f(x, y)* by structuring element *s(x, y)* is defined as,

$$(f \ominus s)(p,q) = \min\{ f(p+x, q+y) - s(x,y) \mid (p+x),(q+y) \in D_f ; (x,y) \in D_s \} \quad (I.8)$$

where D_f and D_s are the domains of *f* and *s* respectively. Graphical interpretation of grayscale dilation and erosion is shown in Figure I.2[4].



(a) One-dimensional signal *f* (b) Structuring element *s*
(c) Dilation of *f* by *s* (d) Erosion of *f* by *s*

Figure I.2: Grayscale Dilation and Erosion on a one dimensional signal

4 Adopted from Digital Image Processing : Gonzales & Woods [9]

I.3. Clustering

Cluster analysis is a basic human activity which starts from the early childhood days and develops in time: an infant would subconsciously learn to differentiate animals from plants or tables from chairs. Aggregation of data objects into groups so that members of the same group are similar to each other and relatively dissimilar to the ones that are of different groups, is defined as a clustering. In real world cluster analysis has a wide range of applications, including pattern recognition, data analysis, image processing, and market research. Using clustering, dense and sparse regions in a space can be identified and general distribution patterns and unpredictable correlations among dimensions of data can be discovered.

> Representing the data by fewer clusters inevitably loses certain fine details, but achieves simplification. Data modeling puts clustering in a historical perspective rooted in mathematics, statistics, and numerical analysis. From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning, and the resulting system represents a data concept. From a practical perspective clustering plays an outstanding role in data mining applications, such as scientific data exploration, information retrieval and text mining, spatial database applications, web analysis, CRM, marketing, medical diagnostics, computational biology, and many others.[5]

The textbook Han & Kamber [11] represents a good introduction to modern data mining and clustering techniques. In addition, papers Xu & Wunsch [16], Kotsiantis & Pintelas [17] are comprehensive and detailed surveys for many clustering methods.

I.3.1. Classification of Clustering Methods

There is a large number of clustering methods in the literature. Selection of clustering algorithm should be made according to both characteristics of the data available and on the particular application. When using cluster analysis as a knowledge discovery tool, we suggest trying several algorithms on the same data to see what the data may reveal.

In general, major clustering algorithms can be classified as:

5 Survey of clustering data mining techniques. Berkhin [10]

- Partitioning Methods:

A partitioning method starts with an initially constructed partition of the data. It then uses an iteration for all data objects to improve the partitioning by moving objects from one cluster to another.

- Hierarchical Methods:

A hierarchical method creates a hierarchical decomposition of the target set of data points. There are two types of hierarchical methods, agglomerative or divisive, based on how the hierarchical decomposition performed.

- Density-Based Algorithms:

Density based algorithms try to find out dense regions on the given data using several techniques. In general if number of data points in some “neighborhood” of a given radius exceeds a particular threshold, a cluster is formed.

- Grid-Based Methods:

Grid-based methods simply quantize data domain into a finite number of bins that form a grid. All of the clustering operations are performed on the grid structure.

- Model Based Methods:

Model-based methods assume a model for each of the clusters and find the best fit of the data to the given model.

Proposed method in this study is classified as a hybrid of density based and grid based methods.

I.3.2. Criteria for a Good Clustering Algorithm

Clustering algorithms in the literature can be compared to each other using some criteria. As expected, it is very hard for a single clustering algorithm to sufficiently fulfill all requirements. Most of the algorithms are only strong in a certain number of these criteria. In fact, it is not really necessary to fulfill all of them because only some of these criteria are crucial for a specific case depending on the data and the application. Most common criteria for a clustering algorithm can be listed as:

- its ability to extract arbitrary shaped clusters (non-convex),
- its ability to work with noisy data and outliers,

- its independence from the order of the input records,
- its ability to work with very large amount of data,
- its ability to work with high dimensional data,
- whether it produces interpretable and useful output,
- if it requires less domain information when determining input parameters,
- its time complexity,
- its ability to extract clusters under constraints, and
- its ability to process different types of properties (e.g. categorical attributes such as colors).

I.3.3. Common Clustering Methods

Categorization of well known clustering algorithms are shown in Figure I.3[6].

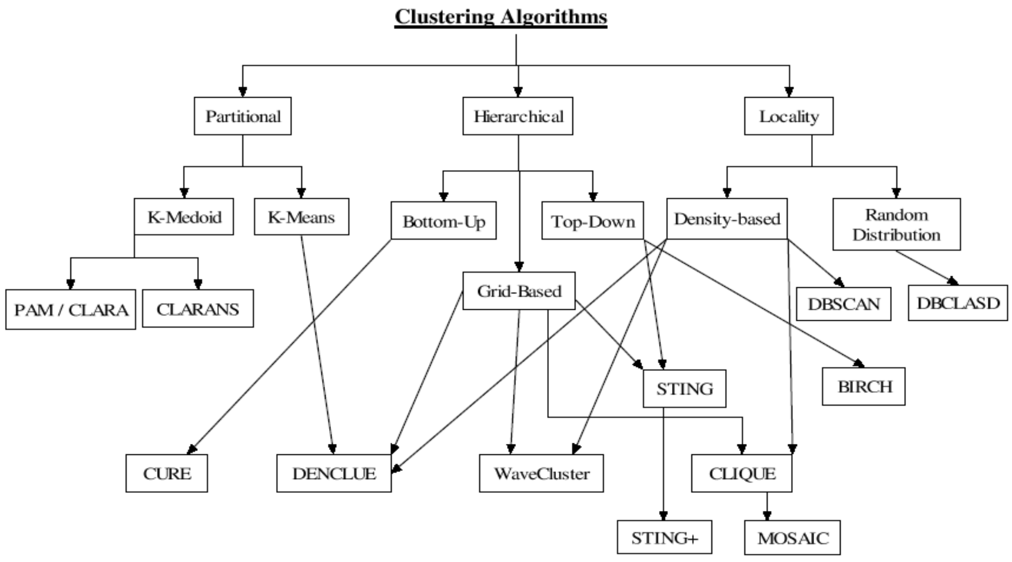


Figure I.3: Classification of common clustering algorithms

PAM, (Partitioning around Medoids) [Kaufmann and Rousseeuw, 1990] uses kclustering on medoids to identify clusters. It works efficiently on small data sets, but it is extremely costly for larger ones. This led to the development of CLARA. CLARA (Clustering Large Applications) [KR90] creates multiple samples of the data set, and then applies PAM to the sample. CLARA chooses the best clustering as the output, basing quality on the similarity and dissimilarity of objects in the entire set, not just the samples. One of the first clustering algorithms

6 Clustering Algorithms for Spatial Databases. Kolatch [12]

> specifically designed for spatial database was CLARANS [NH94] which uses k-medoid method of clustering. CLARANS was followed by DBSCAN [EKSX96] a locality based algorithm relying on the density of objects for clustering. DBCLASD [XEKS98] is also a locality-based algorithm, but it allows for random distribution of the points. Other density or locality-based algorithms include STING [WYM97], an enhancement of DBSCAN, WaveCluster [SCZ98], a method based on wavelets, and DENCLUE [HK98], which is a generalization of several locality-based algorithms. Three other algorithms, BIRCH [ZRL96],CURE [GRS98], and CLIQUE [AGGR98], are hybrid algorithms, making use of both hierarchical techniques and grouping of related items.[7]

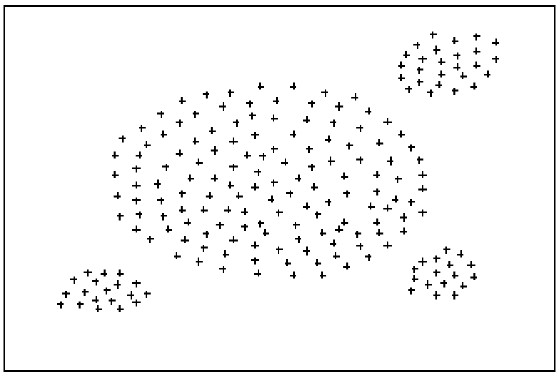
A similar approach to our method is found in Postairea at al. [18]. This paper studies utilization of binary morphology in clustering after forming a binary signal from the raw data. An extension of this work in the aspects of quantization of raw data and proper structuring element selection is presented in Lou at al. [19] which also studies binary morphology. We use grayscale morphology which makes our technique a density based clustering method.

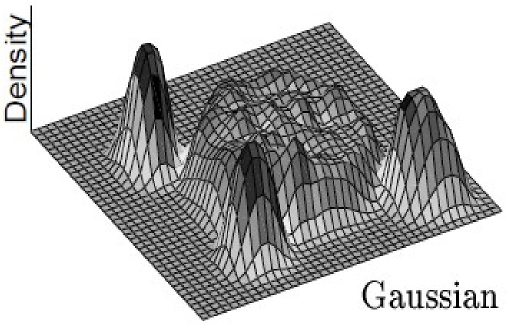
Since our algorithm is also a hybrid of grid-based and density based methods, DENCLUE, WaveCluster and CLIQUE can be alternatives to the proposed method. We will summarize these techniques.

- ***DENCLUE (DENsity based CLUstEring)***

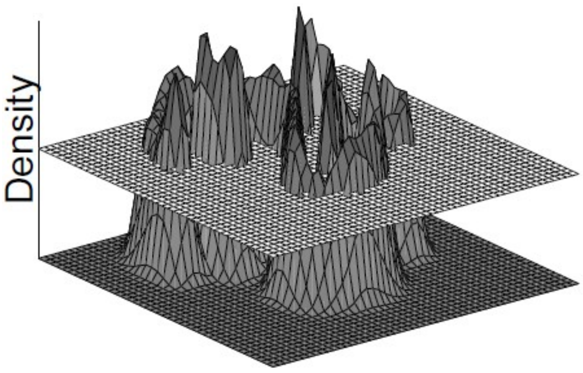
DENCLUE is based on the idea that the influence of each data point can be modeled formally using a mathematical function which is called the influence function. It describes the impact of a data point within its neighborhood. Examples for influence functions are parabolic, rectangular or the Gaussian functions. Every data point contributes to the final density using its influence function. Overall density is computed as the sum of the influences of individual points. The density attractors are determined by hill climbing techniques that search for local maxima of the overall density. Data points with the same density attractors are marked as cluster members provided that their attractor is strong enough. The graphical demonstration of DENCLUE algorithm is presented in Figure I.4[8]. The roughness of the final density is mainly determined by the variance of the Gaussian influence function .

7 Clustering Algorithms for Spatial Databases. Kolatch [12]
8 An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Hinnenburg & Keim [13]


(a) Raw Data



(b) Density Function



(c) Extraction of clusters

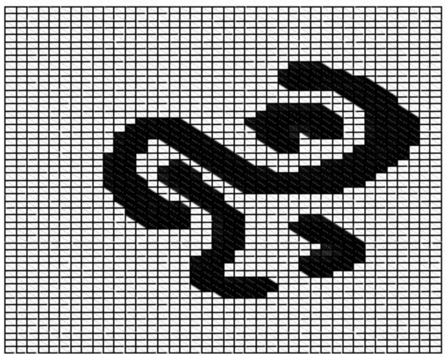


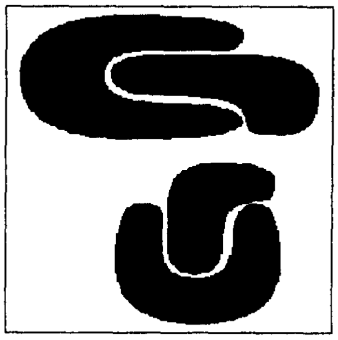
Figure I.4: Graphical Illustration of DENCLUE

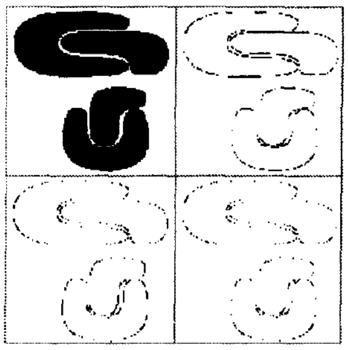
Although, DENCLUE has a firm mathematical basis, its runtime changes according to the distribution of the data processed. If number of highly populated cubes approaches number of populated cubes, runtime of the algorithm could approximate DBSCAN. Furthermore loss of consistency with increased number of dimensions and noise are negative points for DENCLUE. A theoretical comparison of these algorithms with proposed method is given in Table II.7 in Chapter II.

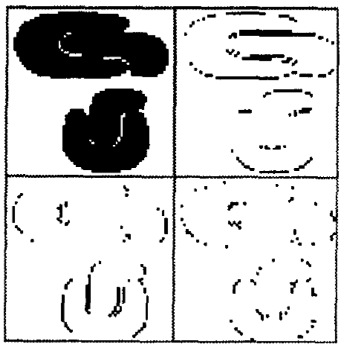
- ***WaveCluster***

WaveCluster uses wavelet transformation to transform the multiresolution feature space, formed by summarizing data by imposing a multidimensional grid structure on to the data space. A sample feature space, and its multiresolution representation is shown in Figure I.5[7].

WaveCluster considers the multidimensional data as a multidimensional signal just like we do and applies a signal processing technique - wavelet transforms - to convert the data into the frequency domain. "In wavelet transform, convolution with an appropriate kernel function results in a transformed space where the natural clusters in the data become more distinguishable.”[9] Then, the clusters are identified by finding the dense regions in the transformed domain.


(a) A sample of two dimensional feature space.


(b)High resolution


(c) Medium Resolution

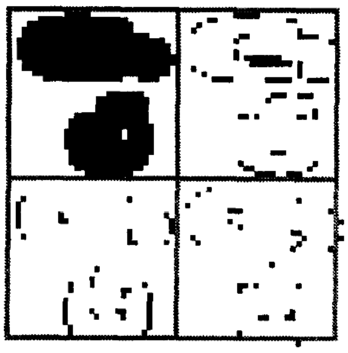

(d) Low Resolution

Figure I.5: A sample feature space, and its multiresolution.

WaveCluster considers the multidimensional data as a multidimensional signal just like we do and applies a signal processing technique - wavelet transforms - to convert the data into the frequency domain. “In wavelet transform, convolution with an appropriate kernel function results in a transformed space where the natural clusters in the data become more distinguishable.”[9] Then, the clusters are identified by finding the dense regions in the transformed domain.

WaveCluster conforms with most of the criteria for a good clustering algorithm listed in Section I.3.2. The algorithms complexity is *O(N)* for low dimensional data, but exponentially grows with the number of dimensions.

➢ ***CLIQUE***

CLIQUE partitions the n-dimensional space into non overlapping units at lower dimensional subspaces and identifies the dense regions in the lower

9 WaveCluster: A Multiresolution Clustering Approach for Very Large Spatial Database. Sheikholeslami et al [14]

dimensional space. The dense regions corresponding to clusters found in the projections are then back projected to the original space reconstructing the clusters there.

CLIQUE, consists of the following steps[10]:

1. Identification of subspaces that contain clusters.
2. Identification of clusters.
3. Generation of minimal description for the clusters

CLIQUE scales well as the number of dimensions of the data set and scales linearly with the size of data set. Although is it scalable, CLIQUE's ability to accurately extract arbitrary shaped clusters is low.

Subspace clustering has the added advantage of being tolerant to missing values in input data. From this perspective CLIQUE is similar to our morphology based technique.

10 Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. Agrawal et al [15]

CHAPTER II

DENSITY BASED CLUSTERING USING MATHEMETICAL MORPHOLOGY

II.1. Introduction

Many clustering algorithms are based on distances between data points. Typically distances between neighboring data points are calculated for each data point and if any of these distance values is small enough then two points are put in the same cluster. However this type of algorithms might not be completed in a reasonable time for very large data sets, because the number of pairwise comparisons needed increases faster than the number of data points.

Our algorithm first obtains a lower resolution n-dimensional data by combining adjacent data points in 'bins'. Then this lower resolution data is passed through morphological operators to remove unnecessary details. The resultant data is passed through a threshold to remove the background noise and then divided into clusters using segmentation techniques (Figure II.1).

Our approach to clustering is mainly based on a multi dimensional histogram calculation. Our choice of "Bin Size" for this histogram together with the threshold value essentially determines how dense the clusters should be. We assume that data points in the same bin are close enough to each other, so we may aggregate those data points into one big data point which has the weight of total number of data points in its bin.

Thinking this way, if we calculate the histogram for all of our data, we simply come up with a lower resolution n-dimensional signal. This signal then passed through a simple threshold operation. This preprocessing cleans out the background noise from our signal to a large extend. However, unexpected imperfections in the foreground data appear as gaps or sharp edges in our signal preventing us from extracting right clusters from the raw signal. Filling such unwanted gaps and flattening sharp edges will enable us to better extract clusters.

We use simple mathematical morphological operations, opening & closing, to make these corrections. We choose appropriate structuring elements for our morphological operations with suitable diameters to fill gaps and flatten sharp edges when needed. After applying grayscale morphological operations on our raw signal using our chosen structuring elements, we obtain a processed signal, which has fewer imperfections.

In order to extract clusters from our newly processed signal, we simply use a threshold and choose bins that are above the threshold. This threshold is the minimum value for a bin that belongs to a cluster. Chosen bins form regions on the cross-section of our 'Threshold Plane' and 'Signal'. These regions are differentiated using segmentation techniques and are assumed to mark clusters that we are looking for. Calculating where each data point corresponds on the cross-section, we are able to find if it belongs to any cluster or not and if so which cluster it is.

A simple flow of our algorithm is shown in Figure II.1. The input parameters in this figure will be explained in this chapter.

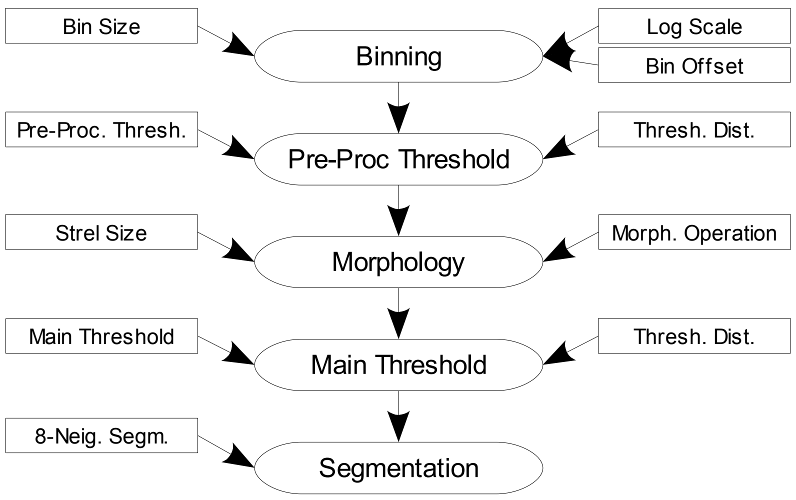


Figure II.1: Algorithm Flow

II.2. Description of the Algorithm

Our algorithm consists of five stages. These are Binning, Preprocessing Threshold, Multi Dimensional Mathematical Morphology, Main Threshold and Segmentation of Classes. Each stage is described in detail below:

II.2.1. Binning

We use binning to aggregate raw data in a low-resolution multi-dimensional signal. This procedure is dividing the coordinate axes into regular intervals and assigning every data point to its corresponding interval. This type of division is also called linear binning. Linear binning is essentially a low pass filtering operation on a grayscale image. Irregular division of the coordinate axes results in non-linear binning. Generally linear binning is satisfactory. However, if it is the case that condensed areas in data cannot be observed using linear binning, e.g. if majority of the data fall in only one of the bins but data in other bins cannot be ignored as well, non-linear binning is required. This work will concentrate on data which can be discretized using linear or logarithmic scales.

In our work, input for binning stage is raw data, since it is the first stage, and output of binning is an n-dimensional low-resolution signal or discretized data signal. In addition, we have to specify two parameters for each dimension for binning: Number of bins (`int nob`) and a flag indicating if the dimension calculated as logarithmic scale (`boolean ls`).

- ***Linear Scale***

When working with linear scale, bin size is directly and linearly affected by "number of bins" parameter. This parameter should be chosen such that the bin size is meaningful for the particular problem and data. Also bin size choice together with the threshold values determine how dense regions in data should belong to a cluster.

Multidimensional linear scale binning is done using the algorithm written as Table II.1.

Table II.1: Linear Scale Binning Pseudo Code

```
DECLARE rawData AS Input Array of Raw Data Points
DECLARE dimensions AS Array of All Dimensions
DECLARE binnedData AS Quantized Output Signal (Multidimensional Image)
FOR each dataPoint in rawData
  SET binIndex to 0
    FOR each dimension in dimensions
      COMPUTE indexOf[dimension] as minimum of
              (( dataPoint minus dataMinimumOf[dimension] )
               divided by dataBinSize)
               and ( numberOfBinsOf[dimension] minus 1)
      SET binIndex to binIndex plus indexOf[dimension]
      IF dimension is not the last one
        SET binIndex to binIndex multiplied by
                                 numberOfBinsOf[nextDimension]
      END IF
    END FOR
    SET binnedData[binIndex] to binnedData[binIndex] plus 1
END FOR
```

➢ ***Logarithmic Scale***

Using logarithmic scale, bin size increases exponentially for a particular dimension. Base of the logarithmic scale is determined according to the number of bins and the data range.(II.1)

$$base_i = e^{\frac{\ln dataRange_i}{numberOfBins_i}} \qquad (II.1)$$

Implementation of Multidimensional logarithmic scale binning is written in Table II.2.

➢ ***Nature of Data***

Bin size choice is one of the critical decisions for proposed algorithm. This parameter effects both completion time of the algorithm and quality of the output. Larger bin size leads to fewer bins and linearly less completion time, but unsuitable bin size may lead to low quality output.

Table II.2: Logarithmic Scale Binning Pseudo Code

```
DECLARE rawData AS Array of Raw Data Points
DECLARE dimensions AS Array of All Dimensions
DECLARE binnedData AS Quantized Output Signal (Multidimensional Image)
FOR each dataPoint in rawData
  SET binIndex to 0
    FOR each dimension in dimensions
      COMPUTE indexOf[dimension] as minimum of
                { maximum of
                  log ( dataPoint minus dataMinimumOf[dimension] ) base
                           baseOf[dimension] and
                  zero
                } and
                numberOfBinsOf[dimension] minus 1
      SET binIndex to binIndex plus indexOf[dimension]
      IF dimension is not the last one
       SET binIndex to binIndex multiplied by numberOfBinsOf[nextDimension]
      END IF
    END FOR
    SET binnedData[binIndex] to binnedData[binIndex] plus 1
END FOR
```

When choosing bin size for each dimension nature of data should be taken into consideration. It is the most important clue for us to decide on the most appropriate bin size. This is to say, bin size choice determines maximum distance between data points that can be said to be sufficiently close to each other for each dimension. These points will probably show similar behavior and can be aggregated into one.

- ***Offsetting Bins***

It is obvious that especially when using linear binning, two data points that are close to each other may fall into two adjacent bins according to our number of bins parameter. This may decrease quality of the output of our algorithm. In order to observe this affect, bins should be offset by half bin size, while keeping all other parameters unchanged.

II.2.2. Preprocessing Threshold

In this stage, raw signal is passed through a threshold before any morphological operations being done on the signal. Our purpose is to clean out data that can be declared as noise with the potential of affecting the overall output negatively, before doing any processing on the signal. Feeding this cleaned signal as input to the next stage will produce more reliable output.

We use one master parameter in this stage, the threshold value. This parameter increases robustness of the algorithm in the sense that it prevents us from making a dramatic error by choosing any of the other parameters inappropriate. In particular, effective removal of noise using a good preprocessing threshold will enable the following stages achieve their goals. Nevertheless, the threshold parameter in this stage should be chosen relatively small. Although this parameter is not a critical one, selecting threshold values significantly higher than the optimal has more detrimental effects than selecting threshold values that are smaller. If it is chosen larger than optimal important data will be evaluated as noise and will be thrown out, which will considerably affect quality of the output of the algorithm. Choosing a small value, even zero, only limits the effect of this parameter and increases the effect of the parameters those will be chosen for the following stages.

If we have no information about data, choosing a good preprocessing threshold value can be experimentally determined. In such a case the mean of the distribution of values of all the bins could be used as the starting point.

- ***Using Variable Threshold***

When it is expected that different threshold values should be used in order to filter noise from particular regions of data, utilizing some kind of variable threshold would produce more effective results. In such cases our implementation offers two options for threshold distribution, staircase threshold and linear threshold. Both functions take position of a specific bin between minimum and maximum values in data for a particular dimension and calculates the threshold value according to the distribution function we defined for each bin. Thus definition of the variable threshold distribution is another parameter for our algorithm.

II.2.3. Multi Dimensional Mathematical Morphology

This is the stage where we enhance our signal. Although we had cleaned out background noise from our raw signal, there is still some foreground noise on it. We will clear this foreground noise using morphological operators, and output will be a relatively smooth signal.

➢ *Geometric Interpretation of Mathematical Morphology*

Geometric interpretation of morphological operations that we will use is essentially like rolling a ball on a surface. Here, the ball is our structuring element and the surface is our signal. If we apply grayscale opening operation on our signal, this means we are rolling our ball on the inner surface of our signal and marking the highest point on the ball as the new value of the signal for all pixels. Doing so, high and sharp hills on the surface where our ball could not fit into will no longer exist in our signal. Similarly, grayscale closing is rolling the ball on the outer surface of the signal and marking the lowest point on the ball. Closing operation removes deep and sharp holes on the surface. Graphical demonstration is shown in Figure II.2.

Neither deep and sharp holes nor high and sharp hills will remain on our signal after we pass it through both opening and closing operations. However, the order of these operations will affect the output. Generally, if there exists a compact sequence of high hills, closing followed by an opening operation unites these hills whereas opening followed by a closing operation flattens the hills. An opposite effect is obtained for a sequence of valleys. In our work, closing followed by opening operation is the default order of operations. As we are trying to extract dense regions from a set of data points, we likely want to combine a series of dense regions instead of removing them. Effect of changing order of morphological operations on the signal drawn in Figure II.2 (b) is shown in Figure II.3.

Using different sized structuring elements for opening and closing may lead to better output in some cases. If we want to fill large gaps but do not want to flatten thin hills, then we should choose a smaller structuring element for opening than the one we choose for closing. These two structuring elements can be selected separately in our application but by default we use the same structuring element for both operations.

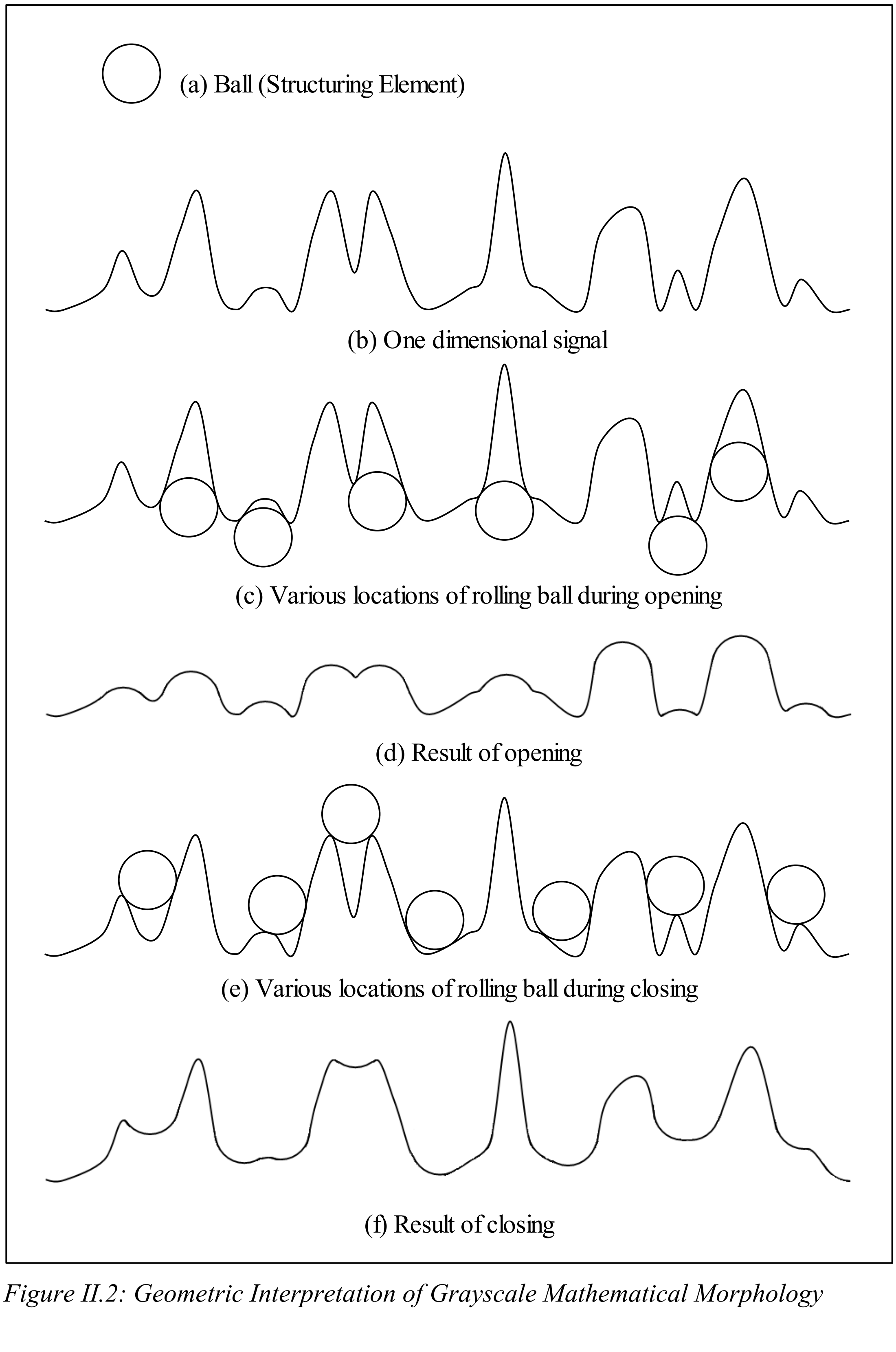

(a) Ball (Structuring Element)
(b) One dimensional signal
(c) Various locations of rolling ball during opening
(d) Result of opening
(e) Various locations of rolling ball during closing
(f) Result of closing

Figure II.2: Geometric Interpretation of Grayscale Mathematical Morphology

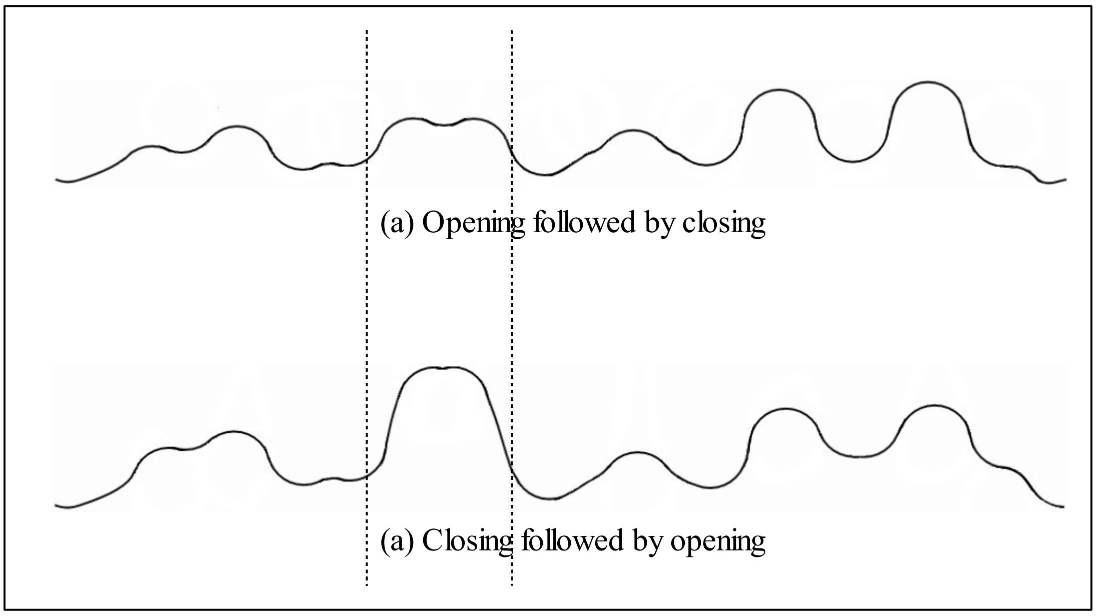

(a) Opening followed by closing
(a) Closing followed by opening

Figure II.3: Effect of Changing Order of Morphological Operations

➢ ***Effect of Structuring Element Size on Clustering***

Consequence of opening and closing operations applied on our signal is that neighboring bins influence each other. That is, any bin with a relatively large (small) value make its neighbors larger (smaller) than their original. Degree of this influence is determined by the shape and size of the structuring element. Choosing a larger structuring element makes more distant bins affect each other.

➢ ***Implementation of Multidimensional Mathematical Morphology***

We use opening and closing operations to remove foreground noise from input signal. Both of these operations consist of dilation and erosion operations in different order. Our implementation of these operations is shown in Table II.3.

Table II.3: Basic Morphological Operations Pseudo Code

```
DECLARE data AS Input Signal (Multidimensional Image)
DECLARE structuringElement AS Structuring Element
DECLARE dimensions AS Array of All Dimensions
FUNCTION open ( data, strucrutingElement )
  RETURN morphologicalOperation (
           morphologicalOperation ( data, strucrutingElement, ERODE ),
           strucrutingElement, DILATE
         )
END FUNCTION
FUNCTION close ( data, strucrutingElement )
  RETURN morphologicalOperation (
           morphologicalOperation ( data, strucrutingElement, DILATE ),
           strucrutingElement, ERODE
         )
END FUNCTION
FUNCTION morphologicalOperation ( data, strucrutingElement, operationType )
  FOR each pixel in data
    FOR each strelPoint in structuringElement
      IF NOT strelPoint less than 0
        FOR each dimension in dimensions
          CALCULATE offsetPixel
        END FOR
        COMPUTE maxValueForPixel as maximum of
                  maxValueForPixel and
                  ( data[offsetPixel] plus structuringElement[strelPoint] )
        COMPUTE minValueForPixel as maximum of
                 ( minimum of minValueForPixel and
                  ( data[offsetPixel] minusstructuringElement[strelPoint] )
                 ) and 0
      END IF
    END FOR
    SET dilatedDataOf[pixel] to maxValuesForPixel
    SET erodedDataOf[pixel] to minValuesForPixel
  END FOR
  IF operationType equals Dilate
    RETURN dilatedData
  ELSE
    RETURN erodedData
END FUNCTION
```

➢ ***Multi Dimensional Structuring Element***

In this work, we use ellipsoid structuring elements. A multidimensional ellipsoid structuring element is calculated using Equation (II.2). In the equation y is the value of the structuring element at the point $(x_1, x_2, \ldots, x_n)$, h is the height of the structuring element and r_i is the radius of the structuring element for dimension i.

$$\frac{x_1^2}{r_1^2}+\frac{x_2^2}{r_2^2}+\ldots+\frac{x_n^2}{r_n^2}+\frac{y^2}{h^2}=1 \qquad (II.2)$$

Implementation of this formula as an algorithm is shown in Table II.4

Table II.4: Multidimensional Structuring Element Construction Pseudo Code

```
DECLARE dimensions AS Array of All Dimensions
FOR each dimension in dimensions
  COMPUTE centerFor[dimension] as sizeFor[dimension] dividev by 2
END FOR
FOR each pixel in structuringElement
  FOR each dimension in dimensions
    CALCULATE pixelIndexFor[dimension]
  END FOR
  SET total to 1
  FOR each dimension in dimensions
    COMPUTE radius as
       distance( pixelIndexFor[dimension] plus 1, sizeFor[dimension] )
    COMPUTE total as total minus radius square
  END FOR
  IF total is greater than 0
    COMPUTE valueOf[pixel] as height times squareroot of total
  ELSE
    SET valueOf[pixel] to minus 1
  END IF
END FOR
FUNCTION distance ( pixelIndex, size )
  IF size mod 2 equlas 0
    RETURN ( 2 times pixelIndex minus size minus 1 )
             divided by ( size plus 1 )
  ELSE
    RETURN ( 2 times pixelIndex minus size minus 1 )
             divided by ( size minus 1 )
END FUNCTION
```

Required parameter for structuring element construction are diameter values for each dimension, and height of structuring element.

II.2.4. Main Threshold

This is the fundamental threshold operation, which we use to extract classes from the enhanced signal. We mark the bins with values larger than or equal to the threshold value and extract classes from the signal by simply extracting regions that are formed on the cross-section of the cut plane and the signal data. In this way we obtain an n-dimensional binary signal indicating if any bin belongs to any of the clusters. This threshold value determines how dense bins should be so that they are marked as a cluster member. For a larger bin size, a larger threshold value should be chosen to extract classes that have the same density.

Using variable threshold described in section II.2.2 is also an option for this stage.

II.2.5. Segmentation of classes

This stage is simple segmentation of clusters that are extracted and designated as the foreground in the n-dimensional binary input signal. Input to this stage is the n-dimensional binary signal and the raw data, and output is segmented signal and marked data points. In our work, segmentation based on four-neighborhood is the default for clustering. In addition to this, eight-neighborhood segmentation can be used verify the results. With eight-neighborhood segmentation diagonally connected bins are put in the same cluster whereas with four-neighborhood they are not. Graphical description of neighbors is shown in Figure II.4.

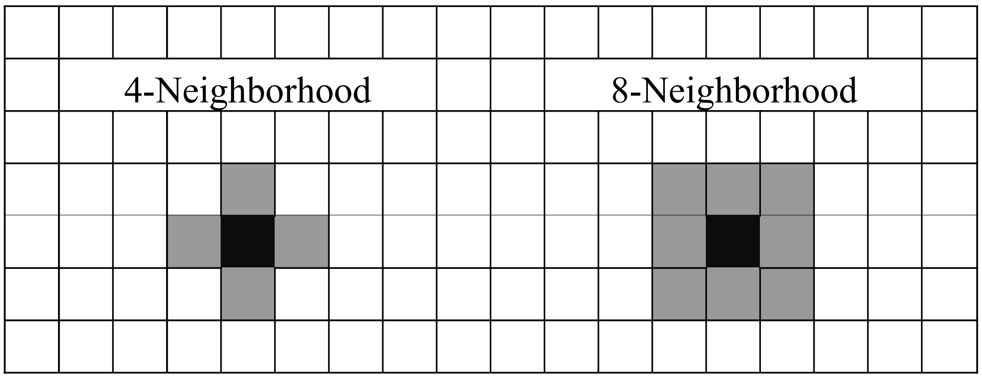


Figure II.4: Description of Neighbors

Segmentation is done by a simple pass of all bins and marking adjacent bins to be in the same segment according to the neighborhood criteria. Any subsegments of the same cluster are united with each other on the fly. Secondly all data points in raw data is cycled and they are associated with individual segments.

Our implementation of multidimensional segmentation is shown in Table II.5.

Table II.5: Multidimensional Segmentation Pseudo Code

```
OBJECT subSegment
  VARIABLE integer segmentIndex
  VARIABLE subSegment connectedSubSegment
  FUNCTION getSegmentIndex
    IF connectedSubSegment if null
      RETURN segmentIndex
    ELSE
      RETURN connectedSubSegment.getSegmentIndex
    END IF
  END FUNCTION
  FUNCTION setConnectedSubSegment ( SubSegment css )
    IF connectedSubSegment if null
      SET connectedSubSegment as css
    ELSE
      CALL connectedSubSegment.setConnectedSubSegment ( css )
    END IF
  END FUNCTION
END OBJECT
IF eightNeigboured
  COMPUTE numberOfNeigbours ( 3 to the power dimensionCount minus 1 )
divided  by 2
else
  SET numberOfNeigbours as dimensionCount
FOR each pixel in signal
  IF pixelValue greater than 0
    LABEL segmentation:
    FOR each neighbour in neigbours
      FOR each dimension in dimensions
        CALCULATE neighbourIndex
        IF neighbourIndex less than zero
           or greater then signalSizeFor[dimension]
          CONTINUE segmentation
        END IF
      END FOR
```

Table II.5: cont'd: Multidimensional Segmentation Pseudo Code

```
      IF subSegmentFor[neighbour] is not null
        IF subSegmentFor[pixel] is not null and
             subSegmentFor[pixel].getSegmentIndex is not equal to
             subSegmentFor[neighbour].getSegmentIndex
          CALL
            subSegmentFor[neighbour]
             .setConnectedSubSegment ( subSegmentFor[pixel] )
        ELSE
          SET subSegmentFor[pixel] as subSegmentFor[neighbour]
        END IF
      END IF
    END FOR
    IF subSegmentFor[pixel] is null
      SET subSegmentFor[pixel] as new subSegment
    END IF
  END IF
END FOR
FOR each pixel in signal
  SET segmentFor[pixel] as subSegmentFor[pixel].getSegmentIndex
END FOR
```

II.3. Discussion of Parameters

Our algorithm requires three basic information:

- How to quantize the data?
- What size of structuring element should be used for morphology?
- What threshold value should be used in order to find out dense bins?

When it comes to the implementation, on the other hand, these three questions are answered via a combination of *(6d + 5)* parameters, where *d* is the number of dimensions. Namely:

- Number of bins for each dimension
- Opening structuring element diameter for each dimension
- Closing structuring element diameter for each dimension
- Logarithmic scale flag for each dimension
- Bin offsetting flag for each dimension
- Threshold function for each dimension

- Preprocessing threshold
- Main threshold
- Opening structuring element intensity
- Closing structuring element intensity
- Order of morphological operations

So many parameters would not be a desirable if no guidelines could be set for each. Fortunately, we may extract some clues from data about these parameters. In the end, only three of them can be said to be critical: number of bins, structuring element size and main threshold. If these three values are chosen intelligently then sub-optimal selections of the others will not have that much detrimental effect. Trial and error will help us to find appropriate parameters, this issue will be discussed in Chapter III.

➢ Number of Bins

For a given data range choosing the number of bins directly determines the bin size for each dimension. Unfortunately, not many clues exist when we try to determine the bin size. Guidelines we suggest:

- Most effective clue we have is the nature of data. Bin size should be chosen according to the problem and data. Since data points in the same bin will be aggregated, bin size should be small enough so that data points in the same bin would show similar behavior for the particular problem.
- Number or bins for each dimension should be large enough, so that opening and closing operations work properly. For instance, for a minimum structuring element diameter of three pixels, only four or five bins on a dimension would not be appropriate.

➢ Logarithmic Scale Flag

Whether the scale of an axis should be linear or logarithmic is a critical decision, fortunately, it is rather easy to decide on. If data range for a particular dimension is very large and after linear binning almost all data points crowd together in the first few bins, using logarithmic scale may alleviate this problem and logarithmic scale flag should be set to true for that particular dimension.

➢ Bin Offsetting

Bin offsetting only applies to linear scale binning and would be critical if there occurs a big difference in the result when used. Thus it would be a good practice to always check what happens when bins are offset.

For the logarithmic binning a similar effect can be achieved by using a different log base.

➢ Preprocessing Threshold

Since this parameter will be used to clean out noise from discretized signal it should be chosen so that most of the noise is removed while the main signal remains.

➢ Threshold Distribution

When nature of data requires a variable threshold, desired threshold distribution is also entered.

➢ Structuring Element Size

When deciding on structuring element size we should consider:

- To what extend large holes should be filled and wide hills should be flattened on our signal
- If number of bins parameter is rather small structuring element size should also be small so that morphological operations give the desired effect.
- How distant bins should affect each other.
- Odd numbers (3, 5, 7, …) are the most desirable to avoid symmetry problems.

- Choosing a large value for the structuring element size lengthens completion time of morphology stage.

➢ Main Threshold

Main threshold parameter should be calculated according to our expected density of a dense region. Similar parameters used in other algorithms are, r in DBSCAN and ξ in DENCLUE. If the user does not have any idea of the expected density of a cluster then she/he could check the threshold value for which a reasonable number of clusters is obtained. Most probably there will be a value for the number of clusters that stays consistent over different threshold values. Minimum threshold value for which number of clusters stay consistent could be a heuristically good main threshold value. Demonstration example is shown in Figure II.5. This approach is also used, in order to determine number of density attractors, by DENCLUE [13].

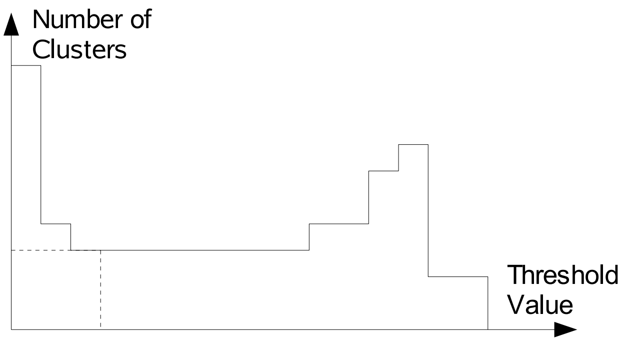


Figure II.5: Main Threshold Selection Heuristic

➢ Order of Morphological Operations

Closing followed by opening will succeed in many cases. Since we are looking for dense areas in data, aggregating a sequence of hills in our signal would be desirable.

II.4. Determining Quality of Output

Although we have many parameters to set for the whole process, we have a second chance after all calculation is over which indicates whether our parameter selections were appropriate for extracting right clusters. In this section there are some heuristic indicators of quality of the output used in the present study.

II.4.1. Number of Clusters Extracted

Number of clusters extracted is a good indicator of quality. This value should be reasonable in the sense that it should reflect the possible clusters in the problem at hand. Moreover excessive number of clusters for a given data size can be interpreted as inadequate removal of noise.

II.4.2. Density of Clusters and Background

Density of any region in data is calculated as the number of data points in the region divided by number of bins that constitutes the region. Defining D_i as the density of cluster i, D_b as the density of background and D_a as the average density of all data, quality of our output will be said to be low if the following inequalities are not valid:

- $D_b < D_a$
- $D_b << D_i$ for all i

II.4.3. Size of Clusters

We can estimate size of a cluster using data range for that cluster for each axis. We expect the cluster to cover multiple pixels on each dimension and should not cover the whole data range. For example, if we have extracted any cluster with a thickness of only one bin for any dimension, it is an obvious indicator for us that either we have chosen a wrong “Number of bins” parameter for that dimension or that one is not a real cluster.

II.4.4. Number of Clusters for Eight Neighborhood Segmentation

If the number of clusters extracted decreases when eight-neighborhood segmentation is used and keeping every other parameter unchanged that means there occurs diagonally touching clusters when we use four-neighborhood segmentation. We should not expect both to have a distance less than single bin length between two clusters and to have clusters with single bin thickness. Consequently cluster analysis should be repeated after changing bin size parameters. Results of bin offsetting should also be tested in such a case.

II.5. Advantages and Disadvantages of Proposed Algorithm

II.5.1. Degree of Complexity

In order to calculate degree of complexity for our algorithm we consider each stage separately. Expressions used in calculations are defined in Table II.6.

Table II.6: Definition of Expressions used in Complexity Calculations

Definitions:

N	: Number of data points in the data set
d	: Number of dimensions
NOB_i	: Number of bins for dim i
B	: Total number of all bins, i.e $B=\prod_{i=1}^{d} NOB_i$
DSO_i	: Diameter of opening structuring element for dim i
DSC_i	: Diameter of closing structuring element for dim i
SP	: Number of potential segments

Stage 1.Binning

This stage is a one pass linear scan of all data points in the data set and time complexity is,

$$O(N) \tag{II.3}$$

Stage 2.Preprocessing threshold

This stage is a one pass linear scan of all bins and complexity is,

$$O(B) \tag{II.4}$$

Stage 3.Morphology

For our implementation of grayscale morphology, time complexity is,

$$O\left(B.\left(\prod_{i=1}^{d} DSOi+\prod_{i=1}^{d} DSCi\right).d\right) \tag{II.5}$$

Assuming c as the largest diameter in all dimensions for both of the structuring elements, it can be said to be,

$$O\left(\prod_{i=1}^{d} DSOi+\prod_{i=1}^{d} DSCi\right)\leq O\left(c^d\right) \tag{II.6}$$

and assuming k as the largest number of bins in all dimensions, B can be approximated as,

$$B \leq k^d \tag{II.7}$$

Derived time complexity for this stage is at most,

$$O\left(k^d \, . \, c^d \, . \, d\right) \tag{II.8}$$

Stage 4.Main Threshold

This stage is also a one pass linear scan of all bins and complexity is,

$$O\left(B\right) \tag{II.9}$$

Stage 5.Segmentation

Time complexity of this stage is,

$$O\left(B \, . d \, . \left(d+SP\right)\right) \tag{II.10}$$

for four-neighborhood segmentation and,

$$O\left(B \, . \, 3^d . \, \left(d+SP\right)\right) \tag{II.11}$$

for eight-neighborhood segmentation. *SP* is determined at runtime by a recursive algorithm and increases with the number of edges of non convex clusters extracted. In these calculations *SP* is assumed to be a moderate constant value for any reasonable arbitrary shaped cluster.

Since eight-neighborhood segmentation is used only as a verification stage, we basically do not concerned with its complexity.

Stage 6.Final Output

This stage is also a one pass linear scan of all data points in set *D* and complexity is,

$$O\left(N\right) \tag{II.12}$$

We conclude that, for low dimensional data our complexity equals *O(N)* since all other multipliers would be approximated as small constants. For high dimensional data, on the other hand, the stage that dominates our time complexity is the morphology. Thus our resulting complexity can be calculated using Formula *II.8* for higher dimensions.

II.5.2. Interpretation of Time Complexity

As it is calculated above, time complexity of developed cluster analysis method is linearly dependent on number of data points to be analyzed. For low-dimensional data completion time of the algorithm linearly increases as the data set becomes larger. In other words time complexity of the algorithm is *O(N)* for low dimensional cases. However, as the number dimensions in data to be analyzed increases, curse of dimensionality effects our algorithm also. Completion time of the algorithm exponentially increases as the number of dimensions increase, which is due to the nature of the morphology algorithm. Under these circumstances it may not be practical to use this method to analyze high-dimensional data.

II.5.3. Comparison to Similar Clustering Algorithms

A table of similar algorithms is given as Table II.7. We see that our algorithm has desirable characteristics like ability to handle non-convexity and/or higher dimensionality and robustness to noise in addition to its relatively low computational complexity. Its main disadvantage is requirement for a large number of parameters.

Table II.7: Comparison of several clustering algorithms

ALGORITHM	**Efficient / Scalable**	**Handles higher dimensionality**	**Handles Irregularly Shaped Clusters**	**Insensitive to Noise**	**Independent of data input order**	**No a-priori knowledge or inputs required**
DBSCAN	O(NlogN) where N = size of dataset	No	Not completely	Yes	Yes	2 parameters required
WaveCluster	O(N) for low dimensions only, N = size of dataset	Not well	Yes	Yes	Yes	Yes
DENCLUE	O(DlogD), where D = # of active data sets	Somewhat	Yes	Yes	Yes	2 parameters required
CLIQUE	Quadratic on # of dimensions	Yes	Minimal	Partially	Yes	2 parameters required
DBCM	**O(N) for low dimensions only, N = size of dataset**	**Yes**	**Yes**	**Yes**	**Yes**	**3 Main Parameters Required**

CHAPTER III

DBCM SOFTWARE

III.1. Introduction

Proposed method consists of five serial main stages and the output of each stage is the input to the next one.

III.1.1. Prototype

In early phases of this work, to see whether the proposed method can be used to produce meaningful results, all of its stages are implemented using MATLAB 6.5 [16]. This prototype MATLAB implementation was designed to work for only for two dimensional data but it also had both linear and logarithmic scale options like the real implementation.

In the prototype, built in MATLAB functions were utilized for structuring element formation and two dimensional grayscale morphology operations. Also a simple user interface was created in order to get input parameters and demonstrate output. A capture of the MATLAB implementation interface is shown in Figure III.1.

III.1.2. Proposed Algorithm Implementation

After prototype work in MATLAB gave promising results, multi-dimensional binning, structuring element formation, multi-dimensional grayscale erosion and dilation and segmentation procedures are developed using Java. In addition code written on MATLAB is ported to Java and is extended to process multidimensional data.

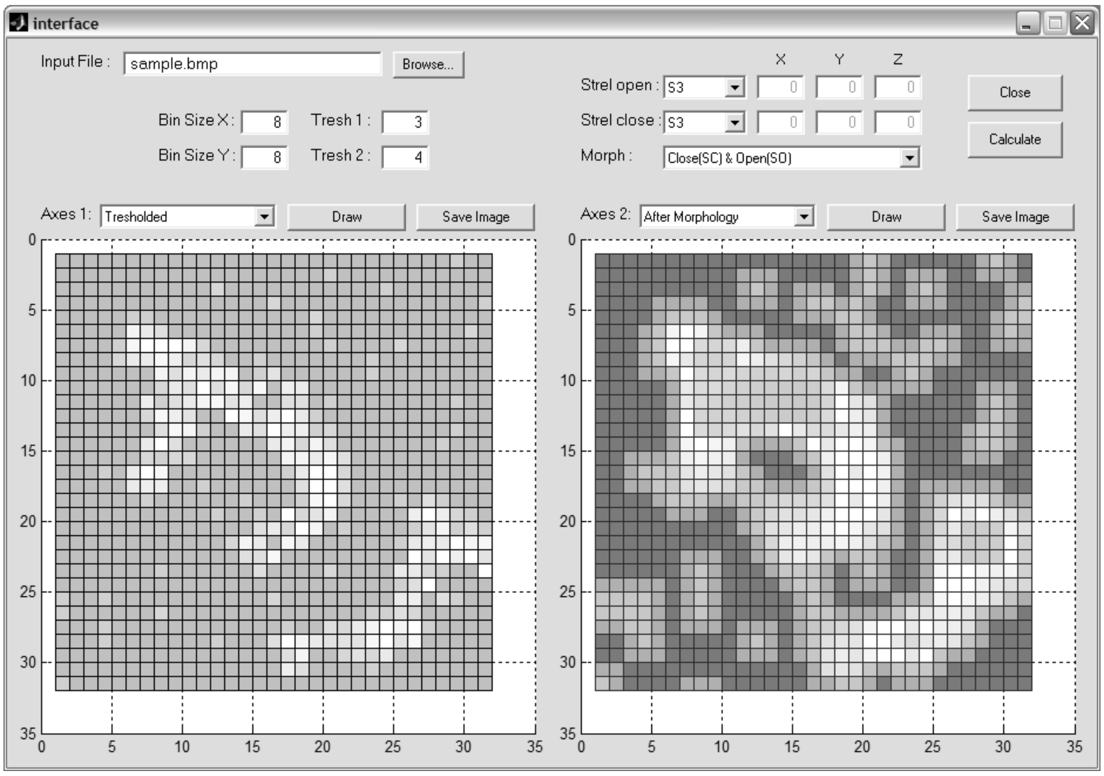


Figure III.1: MATLAB Prototype Implementation User Interface

In Java implementation, first of all, data is loaded into memory. In the first stage it is binned according to the parameters given. After binned data is passed through preprocessing threshold, it is processed using morphological operations. In the last stage, extraction of classes is implemented in two phases, first processed data is passed through the main threshold then classes that occur on the remaining signal are differentiated using segmentation.

We used VisAD [17] library to demonstrate three dimensional outputs in a comprehensible way. In addition, an intermediate class is developed to enable the output to be shown using 3D display panels of VisAD. This intermediate class is also designed to capture the output view and export as an external image using PNG (Portable Network Graphics) format via VisAD functions.

The graphical user interface was designed so that the user had control over all parameters and options. Also, in order to obtain good quality output, repeated trials was often necessary and the user interface had to enable to user to make these trials easily. At this point the software is designed to easily repeat any stage(s) separately, independent of the others after changing related parameters.

Input data format independence is achieved by reading input data from a semicolon separated text file. In addition, although dimensions higher than three cannot be visualized, all of the classes and methods in the software is appropriately coded so that there is no theoretical boundary on the number of dimensions of data to be processed. There is also an option to disable irrelevant and/or undesired columns in input data, which results in not including these dimensions in calculations.

III.1.3. Synthetic Data Production

Synthetic data is needed in our work for two main purposes: the first one is debugging implementation code in development phase of the algorithm and the second objective is observing performance of the proposed algorithm in extracting classes for a given data. In order to supply synthetic data, a separate synthetic multi-dimensional data producer is implemented. Synthetic data producer, simply produces randomized data points in a multidimensional space of a given size by marking pixels in the synthetic space as filled or not. Effective probability of pixels that fall inside or near an object in this space is calculated using the density of the object, instead of background. Although objects in this synthetic space can be defined only as ellipsoids, two or more ellipsoids touching each other may be used to produce non-convex classes..

III.2. Development Platform and External Libraries Used

In order to make the software to be platform independent and run on any environment without difficulty Java programming language is selected for the implementation.

Eclipse v3.2 [18] is used as development environment and the Java Swing library is chosen for the objects that constitutes the graphical user interface of the software. In addition to run software Sun Java Runtime Environment version 1.5.0 or later should be used as Java Virtual Machine.

III.2.1. VisAD

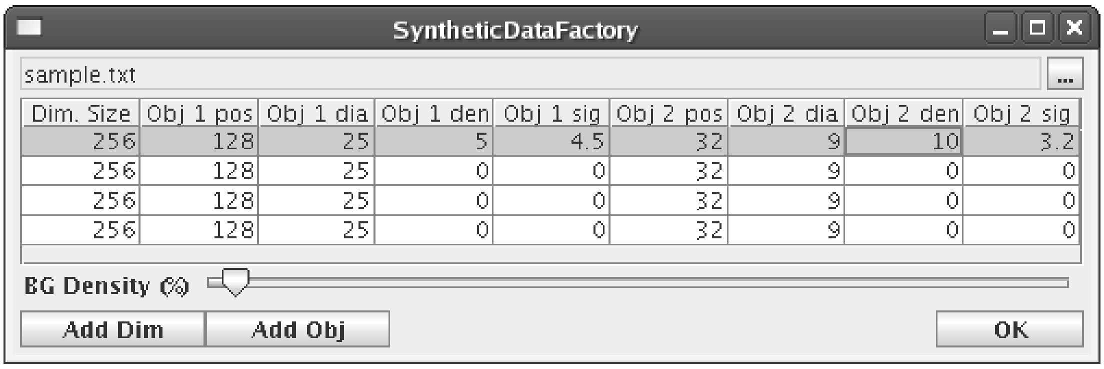
VisAD is an open source Java component library for interactive and collaborative visualization and analysis of numerical data. The name VisAD is an acronym for "Visualization for Algorithm Development". Using VisAD we have the ability to visualize any two or three dimensional numerical data. Java3D 1.2.1[19] or later is required for VisAD in order to have proper support for 3-D displays. In our work, we use VisAD library to demonstrate two or three dimensional outputs of any stage of the developed method. In addition, it is used to visualize the structuring elements that are used in the morphology stage of our clustering method.

III.3. Synthetic Data Producer Software

III.3.1. Structure, GUI & Logic

To produce the desired synthetic data, virtual multi-dimensional space and several ellipsoid objects in this space need to be defined. Namely we need, number of dimensions, size of space in each dimension, number of ellipsoid objects, position of each object in space, diameter of each object in each dimension, internal density of each object, standard deviation of the normal distribution which is used to diffuse borders of each object in space and background density of space. Background density will be termed as noise in our algorithm.

In our "Synthetic Data Factory" software all input needed to produce synthetic data is collected through an input table. While each row appended to the table corresponds to a new dimension, for each object defined in the space four columns are added to the input table. A screen capture of "Synthetic Data Factory" software for a four dimensional space containing two spheroid objects is shown in Figure III.2.



Dim. Size	Obj 1 pos	Obj 1 dia	Obj 1 den	Obj 1 sig	Obj 2 pos	Obj 2 dia	Obj 2 den	Obj 2 sig
256	128	25	5	4.5	32	9	10	3.2
256	128	25	0	0	32	9	0	0
256	128	25	0	0	32	9	0	0
256	128	25	0	0	32	9	0	0

Figure III.2: Synthetic Data Producer User Interface

Data production principle of "Synthetic Data Factory" software is based on the probability of each pixel to contain a data point. Every object in the space affects the probability of every pixel. If a pixel is inside of an object then the effect of that object on particular pixel is directly calculated as density of object, otherwise effect of the object on the pixel is calculated using a Gaussian Function according to the distance of the pixel to the object and sigma given for the object. For each pixel a total probability is calculated by multiplying effects of each object on that pixel and background density. A pixel is marked as filled if a random number between 0 and 1 is uniformly greater than total probability of the pixel.

Gaussian calculation prevents having sharp edged objects in our space, which would be rather unnatural. Pixels that are marked as filled are then written into the output file as data points.

III.3.2. Usage

In order to run "Synthetic Data Factory" application, class "`SyntheticDataFactory`" is called from the package "`cosku.dbcm.syndata`".

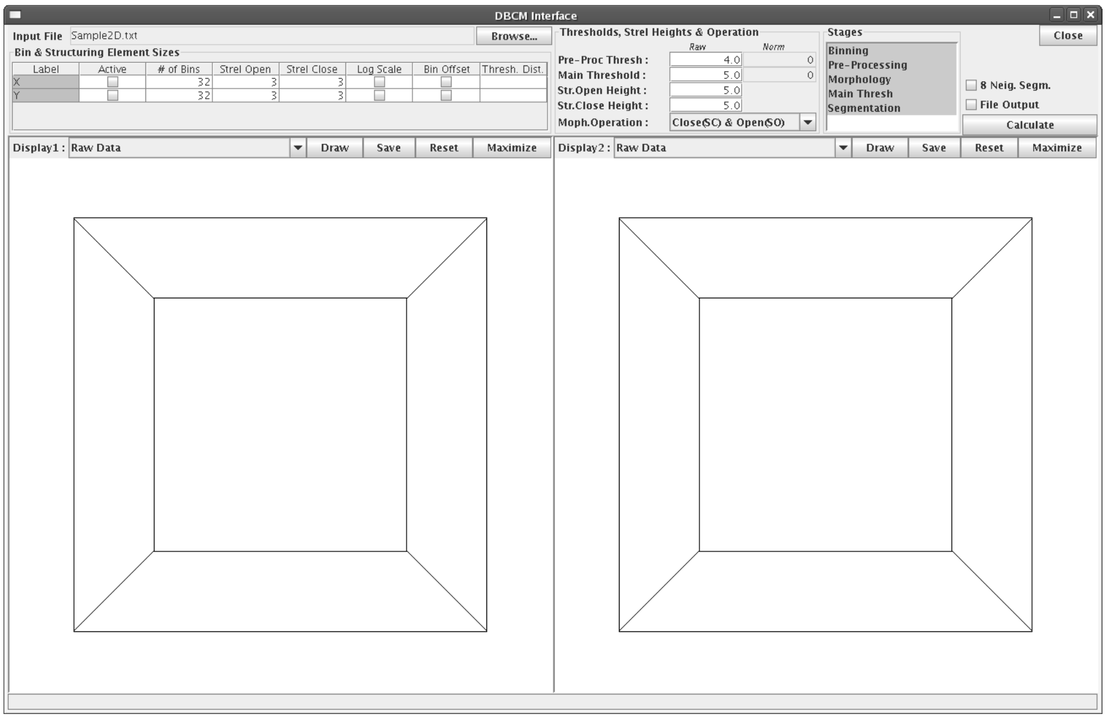
Usage of "Synthetic Data Factory" software is fairly simple. First output text file to be created should be selected using "*Browse...*" button. By default this file is "sample.txt" in the runtime directory. Number of dimensions is determined by adding rows, starting from three, to the table using "Add Dim" button. Number of objects is determined by adding objects using "Add Obj" button. As stated before for each object, four columns are added to the table.

After size of input table is determined we should fill its cells. Size of each dimension should be entered to the first column of input table. Position and diameter parameters are filled for each object in each dimension respectively into "Obj *X* pos" and "Obj *X* dia" columns where *X* stands for the object number. Density and sigma parameters should be entered into the first row of the corresponding columns, "Obj *X* den" and "Obj *X* sig" respectively, for each object. Rows other that the first are not taken into account for the density and sigma columns for each object and density values are read as percentages. At last background density of our space should be determined using the "BG Density" slider under input table.

After all necessary parameters are entered, pressing "OK" button will produce randomized synthetic data and write these data into the output file in semi-colon separated format.

III.4. DBCM Implementation and GUI

Main window of DBCM software graphical user interface can be seen on the screen capture (Figure III.3). In order to start “DBCM”Application, “`DbcmMainFrame`” class should be run from the package “`cosku.dbcm`”.


DBCM Interface

Input File: Sample2D.txt

Bin & Structuring Element Sizes

Label	Active	# of Bins	Strel Open	Strel Close	Log Scale	Bin Offset	Thresh. Dist.
X	☐	32	3	3	☐	☐	
Y	☐	32	3	3	☐	☐	

Thresholds, Strel Heights & Operation

	Raw	Norm
Pre-Proc Thresh :	4.0	0
Main Threshold :	5.0	0
Str.Open Height :	5.0	
Str.Close Height :	5.0	
Moph.Operation :	Close(SC) & Open(SO)	

Stages: Binning, Pre-Processing, Morphology, Main Thresh, Segmentation

☐ 8 Neig. Segm.
☐ File Output

Display1 : Raw Data
Display2 : Raw Data

Figure III.3: DBCM Main Screen

DBCM software includes a graphical human interface and implementations of algorithms which construct the basis of this study. The user interface collects input parameters and calculation options from the user and, for two and three dimensional data, demonstrates output of calculations in a comprehensive way.

Layout of input parameters and calculation options on user interface is shown in Figure III.3. All of these input parameters and options are taken in to calculation in different stages of the implemented algorithm. This flow is shown in Figure II.1

III.4.1. Input Parameters

- ***Input Data File***

First of all input data file is chosen via a file chooser. Required file format is semi-colon separated text file where each column corresponds to a dimension and each row corresponds to a single data point. It is assumed that first row of the input file consists of labels of corresponding dimensions and each cell of the file is filled with a number even if the value of cell is zero. Improper file format will result in a software exception. A sample input file is written in Table III.1.

Table III.1: Sample Input Data File

```
Temperature;Weight;Diameter
132;22.4;21
144;42;23
107;25;54.3
140;65;56
156.3;53;0
117;26;25
128;63;62.4
150;0;75
57;17.3;23
104.3;73;23
93;18;3
```

➢ *Dimension Specific Parameters*

After the input file is chosen from the file system, software reads the first line of the data file to find out the number of dimensions and labels of these dimensions. According to this information a table is automatically constructed in the left upper pane of the main window labeled named “Bin & Structuring Element Sizes”. This table has a row for each dimension and is used o collect parameters that are specific to each dimension. Every column in the table reads a parameter:

- First column of this table, named “Active”, is used to determine if a dimension is taken into calculation, according whether the check box on the cell for a specific dimension is checked or not.
- Column named “# of bins”, is used to read “Number of Bins” parameter for a dimension, value entered in a cell on this column is assumed to be an integer.
- Columns named “Strel Open” and “Strel Close”, are used to read diameters of structuring elements on each dimension which are used in “Opening” and “Closing” morphological operations respectively. Values are assumed to be decimal numbers.
- Column named “Log Scale”, is used to determine whether that dimension should be calculated as logarithmic scale or using linear scale.
- Column named “Bin Offset”, is used to determine whether linear bin offsetting should be used for that dimension.

- Column named "Thresh Dist.", is used to determine the distribution of thresholds for each dimension. There are two options available in our implementation for threshold function. First option is a staircase and other is linear. Each step of a staircase function is described using a "Range Percent, Threshold Percent" tuple. Each tuple means: *For the range starting from the ending of the previous step to the "Range Percent" of the data range for related dimension, use "Threshold Percent" of the given threshold value*. Steps are separated with semicolons and values in a tuple are separated with a comma. An example illustration of threshold distribution is shown in Figure III.4. If the "Range Percent" value is "0" or "100" for the first tuple, then a linear function ascending from or descending to "Threshold Percent" value is used respectively. Distribution described here is applied to both preprocessing and main thresholds. If threshold distribution is given for more than one dimension then product of percentages is calculated for each point in space.

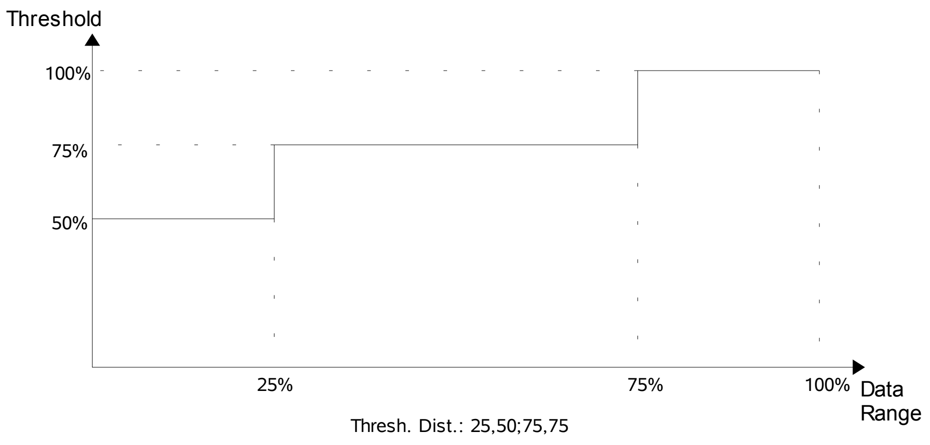


Figure III.4: Threshold Distribution Example

➤ *General Parameters*

Using the pane labeled "Thresholds, Strel Heights & Op.", threshold values, heights of structuring elements used in "Opening" and "Closing" operations and desired order of morphological operations to be used in "Morphology" stage are entered.

➤ *Normalized Thresholds*

Raw threshold values are read as "Number of Data Points". Increasing and decreasing bin size is another input parameter that should be selected in parallel to threshold values. To automatically revise the threshold values after changing bin size normalized threshold option may be used. Related text box on GUI is activated by clicking on. When activated, current level of corresponding raw threshold is calculated and written in the text box, at the same time corresponding raw threshold text box is inactivated. After normalized threshold is activated, raw threshold value is calculated for according to the volume of bins and threshold level entered. There is no unit for this threshold level and it is only comparable with its previous values. In order to inactivate normalized threshold option, raw threshold text box should be clicked.

III.4.2. Calculation Options

Needed stage(s) are chosen from the "Stages" pane. Stage selection is useful when quick retrials are desired as described in section 3.4.4.3. Eight neighborhood (as opposed to four neighborhood) segmentation and producing of output files options may be chosen by clicking the relevant check boxes.

Eight neighborhood segmentation option is used as a quality evaluation criteria, which is described in Chapter 2.

File Output option should be used if the output of the calculations are desired to be written on several text files. If this option is selected, output files are produced under the "*out*" directory of the directory containing the input file on the file system. File names of produced files have the pattern "*<Original File Name>_<Date>-<Time>_<File Suffix>.txt*". Contents of these output files are described in Section III.4.4.

III.4.3. Calculation

After all of the parameters and options are chosen, calculation can be done for the stages selected. If the calculation ends successfully, total time passed during the calculation is displayed on the status bar and "Calculate" button is inactivated. It remains inactive until any parameter or option is changed. On the other hand if the calculate button is not inactivated after the calculation that means an unexpected result has occurred during calculation process for some reason, for example improper input data format. If this case, it will be written to standard error stream and will be displayed on the status bar located at the bottom of the main window.

III.4.4. Output

- ***Demonstration panes***

There are two VisAD panes in the middle of the GUI window to demonstrate the output of the algorithms. If all calculations are completed successfully, just after a stage is selected from the select box above the pane and "Draw" button is clicked, visualization of output of regarding stage is drawn in the VisAD display. Image on the VisAD display can be rotated, zoomed and shifted using drag and drops on the display. "Save" button is used the capture the image shown in the display, "Reset" button resets the image position in the pane and "Maximize" button is used to make VisAD display to fill all of the main window.

- ***Output Files***

If "File Output" option is checked, seven output files are produced during calculations. Files other than the report file are semicolon separated text files. Contents of these output files are described below.

- ***Number of Bins File:*** File suffix: "00-nob". This file has a row for each dimension where first column is dimension index and second is the total number of bins in that dimension.
- ***Binned Data File:*** File suffix: "01-binned". This file has a row for each bin where first column is the bin index and second is the number data points in that bin.
- ***Preprocessed Bins File:*** File suffix: "02-preproc". This file has a row for each bin where first column is the bin index and second is the value of that bin after "Preprocessing Threshold" stage.
- ***Morphology Output File:*** File suffix: "03-morph". This file has a row for each bin where first column is the bin index and second is the value of that bin after "Morphology" stage.
- ***Segmentation File:*** File suffix: "04-segments". This file has a row for each bin where first column is the bin index and second is the segment index of that bin.

- ***Final Output File:*** File suffix: “05-final”. This file has a row for each data point where every data point is written just like in the input data file and as the last column the cluster index for each data point is indicated.

- ***Report File:*** File suffix: “06-report”. This file is calculation report which shows all parameters used in that run and the output quality indicators described in Section 2.4.

➢ ***Evaluation of quality***

After calculation is over following output quality indicators are printed to report file. An example is shown in Table III.2.

- Number of Clusters Extracted
- Density of Clusters and Background
- Range Coverage of Clusters

Table III.2: Example Report File

```
Parameters    | Dims   :         X               Y Heig
Pre :        4| #ofBins:        32              32
Main:        5| BinSize:     7.969           7.969
Oper: Cl & Op | StrelOp:         3               3            1
Segm: 4-Nhood | StrelCl:         3               3            1
                BinOffs:     false           false
                Th.Dist:

       | Member Bin    | Member Data Pt. |         | X             | Y
       | Count/Percent | Count/Percent   | Density | Range/Percent | Range/Percent
------------------------------------------------------------------------------------
Overall|          1024 |            2783 |    2.72 |
Backgr.|      839/81.9 |       1259/45.2 |    1.50 |      32/100.0 |       32/100.0
Cl. 1  |      122/11.9 |        995/35.8 |    8.16 |       18/56.2 |        16/50.0
Cl. 2  |        63/6.2 |        529/19.0 |    8.40 |       13/40.6 |        16/50.0
Cl. All|               |       1524/54.8 |    8.24 |
```

The report consists of two sections, the first half of the report summarizes input parameters for the specific run and the second half prints a table containing output statistics and quality indicators.

On the output statistics table, the first row shows the total number of samples in the data and also the corresponding number of bins. This is denoted by 'overall' in the first column. The second row corresponds to the set of data points in the background, i.e. the points that are not included in any cluster. Afterwards, one row is added to the table for each cluster found and the last row corresponds to the set of data points that are a member of any cluster. The sum of the 'Background' and 'Cl. All' should equal the value in the 'Overall' field. The columns of the table are organized in the following manner:

- The second column shows “Member Bin Count” which is the total number of bins in the corresponding set and its percentage to overall. For the first row, percentage is not given since it will always be 100%.
- The third column shows “Member Data Point Count” which is the total number of data points in the corresponding set and its percentage to overall. Again for the first row, percentage is not given.
- The fourth column shows the average “Density” of the bins in the related set. This value equals member data point count divided by member bin count.
- The remaining columns show “Range Coverages for each Dimension” and their percentages computed as the ratio of data range covered by the member bins of the related set and whole data rage on the corresponding dimension. For the first and last rows these columns are empty, since it would not be meaningful.

In this example number of clusters extracted is 2, density of both clusters are larger than 8 where density of the background is 1.5, size of both clusters are nearly half of the whole data range on each dimension.

In order to get number of clusters for eight-neighborhood segmentation, we may rerun the calculation with same parameters with eight-neighborhood segmentation option is checked.

➢ ***Retrials***

Appropriate parameters for a particular data set can be found out using trial and error. When making retrials only the stages that are affected from the change should be recalculated for the sake of simplicity. Stages effected from particular parameters can be seen from the algorithm flow (Figure II.1).

If one or more intermediate stages are selected for recalculation, output of the previous stage from the previous run is taken as input. In addition, outputs of the following stages from the previous run are deleted from the memory to prevent any confusion.

CHAPTER IV

APPLICATIONS OF DBCM

IV.1. Introduction

This chapter represents the results disclosed after DBCM algorithm is applied on several synthetic and real data sets. Java implementation of DBCM introduced in Chapter 3 is used to obtain the results.

IV.2. Data Selection

IV.2.1. Synthetic Data

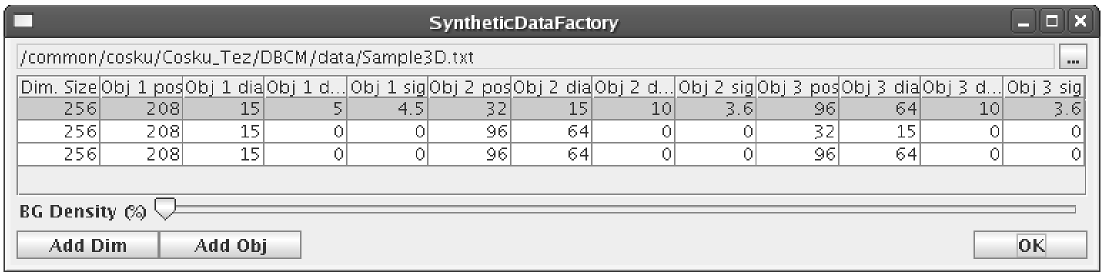
We generated three sets of synthetic data in this study which were two, three and four dimensional. General structures and usages of these data sets are described below.

➢ ***Two Dimensional Synthetic Data***

This set is produced at the stage of prototype implementation of DBCM. It is mostly used in development phase of the algorithm. It consists of 2,783 data points which form two visually differentiable non convex clusters and abundant amount of noise in the background. It is produced as a binary bitmap image for prototype work and then converted into a list of data points to be used with with final implementation. Bitmap projection of the data set is shown in Figure IV.16 (a).

➢ ***Three Dimensional Synthetic Data***

Three dimensional data set is produced using "Synthetic Data Factory" described in Section 3.3. It is used to verify the algorithm to work on dimensions more than two and produce reasonable results. The set consists of 87,248 data points and contains two clusters, one of them is a non convex cluster consisting of two ellipsoid touching each other. The second is a spherical cluster located at the opposite edge, having rather less density than the other one. Parameters used to produce the data set are shown as a screen capture of “Synthetic Data Factory” in Figure IV.1.



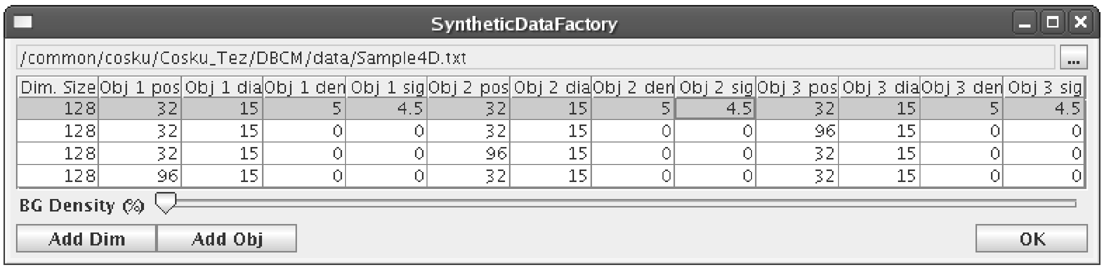
/common/cosku/Cosku_Tez/DBCM/data/Sample3D.txt

Dim. Size	Obj 1 pos	Obj 1 dia	Obj 1 d...	Obj 1 sig	Obj 2 pos	Obj 2 dia	Obj 2 d...	Obj 2 sig	Obj 3 pos	Obj 3 dia	Obj 3 d...	Obj 3 sig
256	208	15	5	4.5	32	15	10	3.6	96	64	10	3.6
256	208	15	0	0	96	64	0	0	32	15	0	0
256	208	15	0	0	96	64	0	0	96	64	0	0

Figure IV.1: Producing 3D Synthetic Data

➤ ***Four Dimensional Synthetic Data***

Four dimensional data set is also produced using “Synthetic Data Factory”. It is especially used to observe effect of increasing dimensions on time complexity. The set contains 687,693 data points and three spherical clusters at different edges. 3D projection of data on any axis can be visualized by inactivating relevant dimension during calculation. Parameters used to produce the data set are shown as a screen capture of “Synthetic Data Factory” in Figure IV.2.



/common/cosku/Cosku_Tez/DBCM/data/Sample4D.txt

Dim. Size	Obj 1 pos	Obj 1 dia	Obj 1 den	Obj 1 sig	Obj 2 pos	Obj 2 dia	Obj 2 den	Obj 2 sig	Obj 3 pos	Obj 3 dia	Obj 3 den	Obj 3 sig
128	32	15	5	4.5	32	15	5	4.5	32	15	5	4.5
128	32	15	0	0	32	15	0	0	96	15	0	0
128	32	15	0	0	96	15	0	0	32	15	0	0
128	96	15	0	0	32	15	0	0	32	15	0	0

Figure IV.2: Producing 4D Synthetic Data

IV.2.2. Real Data : Credit Card Usage Records

In this study credit card usage records of a private bank are used as a real data. The data does not hold any personal information about the customers of mentioned bank. Two sets of the same data is used which are collected at different intervals of time. These similar sets with the same characteristics are selected on purpose and are expected to produce similar results to verify each other and the proposed algorithm itself. Both data sets consists of three dimensions, which are: age and consumption categories of the customer and amount of transaction for a given month. “Age Group” and “Consumption Category” of the customer are preprocessed dimensions which are previously quantized. We used the same quantization in this study. On the “Amount of Transaction” dimension, we utilized logarithmic scale since data points are distributed nonlinearly on this dimension.

IV.3. Sample Runs

IV.3.1. Runs on Two Dimensional Synthetic Data

We use four different parameter families for two dimensional data as shown in Table IV.1. Changing values between these families are marked using bold characters on the table.

Table IV.1: Parameter Families for 2D Synthetic Data

	Run 2D.I		***Run 2D.II***		***Run 2D.III***		***Run 2D.IV***	
Dimension	*X*	*Y*	*X*	*Y*	*X*	*Y*	*X*	*Y*
# of Bins	32	32	32	32	32	32	**48**	**48**
Opening Strel. Dia.	3	3	3	3	3	3	3	3
Closing Strel. Dia.	3	3	3	3	3	3	**5**	**5**
Use Log. Scale	No	No	No	No	No	No	No	No
Use Bin Offsetting	No	No	No	No	**Yes**	**Yes**	Yes	Yes
Threshold Dist.	-	-	-	-	-	-	-	-
Pre. Thresh.	4		4		4		**3**	
Main Threshold	5		**6**		5		**3**	
Opening Strel. Height	1		1		1		1	
Closing Strel. Height	1		1		1		1	
Morphological Operation	Closing + Opening		Closing + Opening		Closing + Opening		Closing + Opening	

➢ ***Reference Run (2D.I)***

For the first run we quantized both dimensions into 32 bins which constitutes 1024 bins in total. Bins containing more than 4 data points are filtered and used as input for a Closing followed by an Opening operation, both using the same 3x3 structuring element with height 1. A main threshold with value 5 is applied on resulting signal and clusters are segmented and data points are marked. All phases of Run I is shown in Figure IV.16, and calculation report for Run I is on Table IV.2.

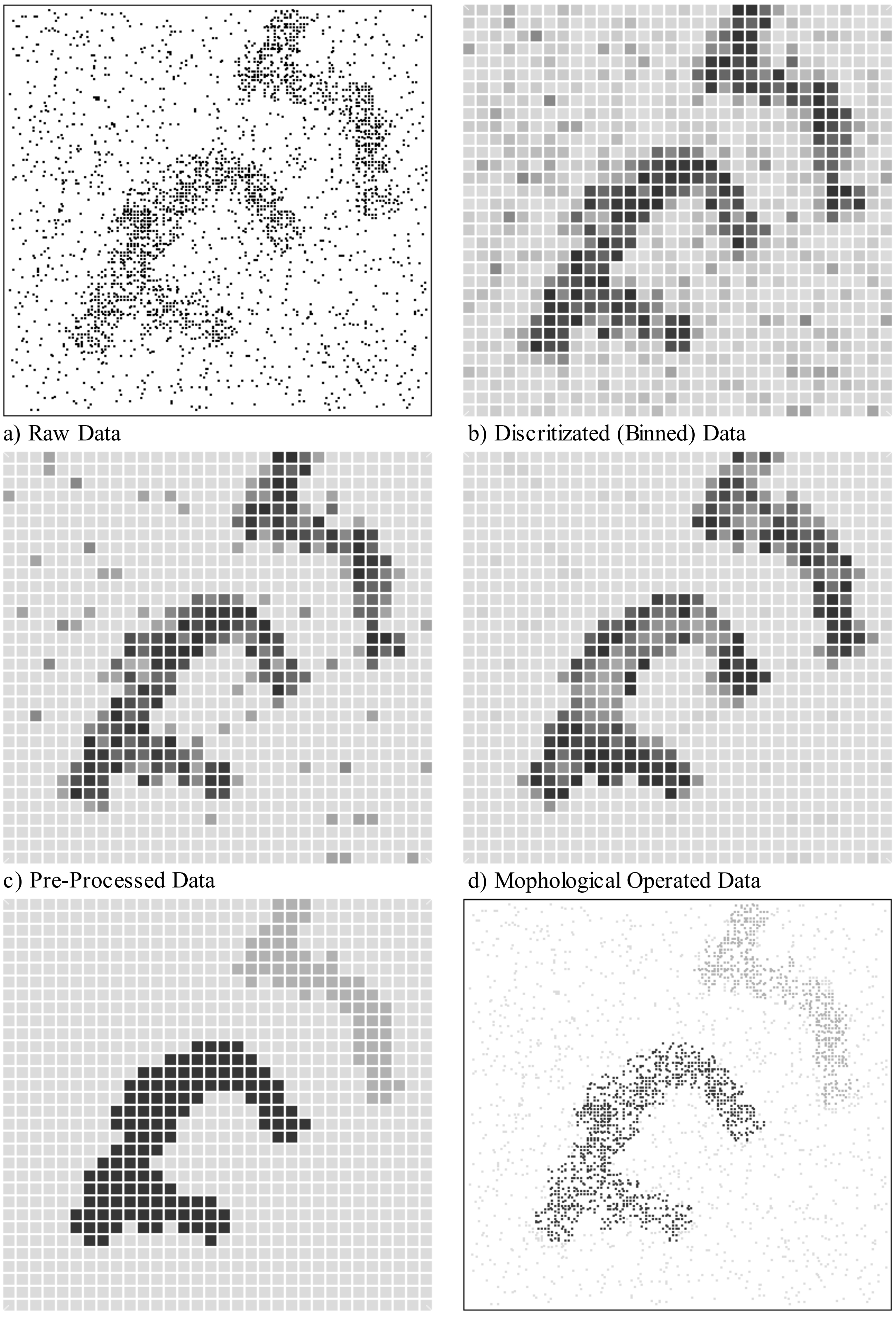

a) Raw Data b) Discritizated (Binned) Data
c) Pre-Processed Data d) Mophological Operated Data
e) Segmented Data (Extracted Clusters) f) Final Output

Figure IV.3: Stages of Run I for 2D synthetic data

Table IV.2: Calculation Report for Run 2D.I

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***X Range/Percent***	***Y Range/Percent***
Overall	1024	2783	2.72		
Backgr.	839/81.9	1259/45.2	**1.50**	32/100.0	32/100.0
Cl. 1	122/11.9	995/35.8	**8.16**	**18/56.2**	**16/50.0**
Cl. 2	63/6.2	529/19.0	**8.40**	**13/40.6**	**16/50.0**
Cl. All		1524/**54.8**	8.24		

This run is the reference run for our two dimensional synthetic data., Two non-convex clusters are successfully extracted as shown in (Figure IV.16). Quality criteria printed in report also gives us a quite positive impression, which is consistent with the output images:

- Two clusters are found. This result is consistent with our expectations.
- Density of background is 1.50 and density of the clusters are larger than 8. Extracted clusters are more than 5 times denser than the background.
- Range coverage of our clusters are approximately 50% for each dimension. It is acceptable for our 2D synthetic data.

◇ ***Effect of Morphology Stage***

As morphological processing is the basis of this work, we checked to see what output would be produced if we bypass the morphological operation stage. The result, was not surprising: without morphology, many small clusters occurred in the background and a few small holes opened in our main clusters. These effects are shown in Figure IV.4.

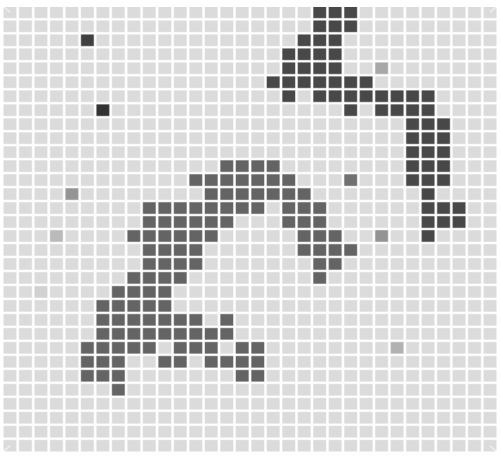


Figure IV.4: Segmented Data for Run 2D.I without using Morphology

◇ ***Output of DENCLUE***

We also used DENCLUE implementation of MIPAV [24] to verify the results of our method on the same two dimensional synthetic data. The output of DENCLUE is shown in Figure IV.5 and its similarity to the output of DBCM (Figure IV.3) is clear. If more detail is desirable when using DBCM the number of bins parameter should be increased.

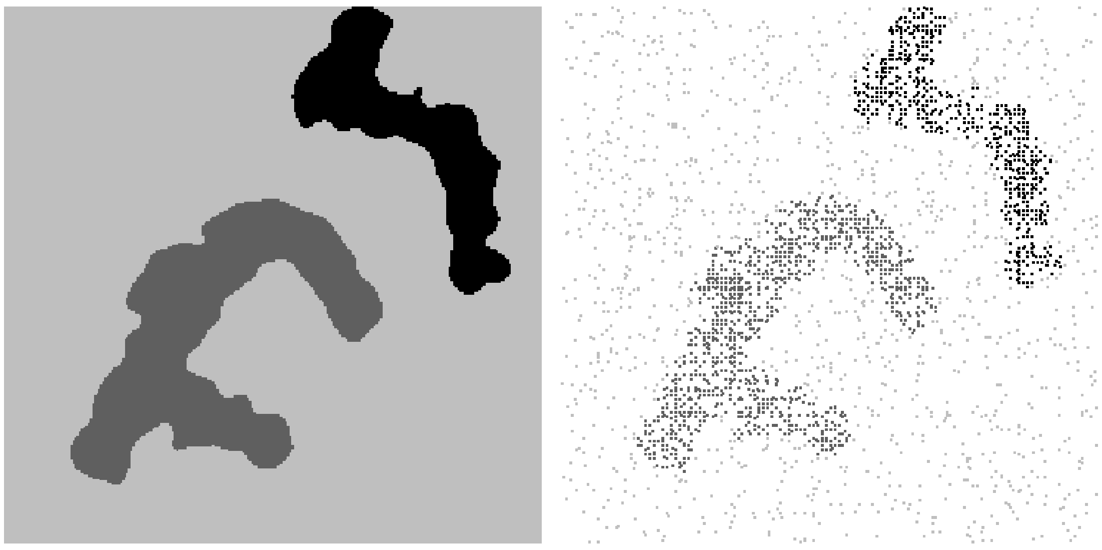
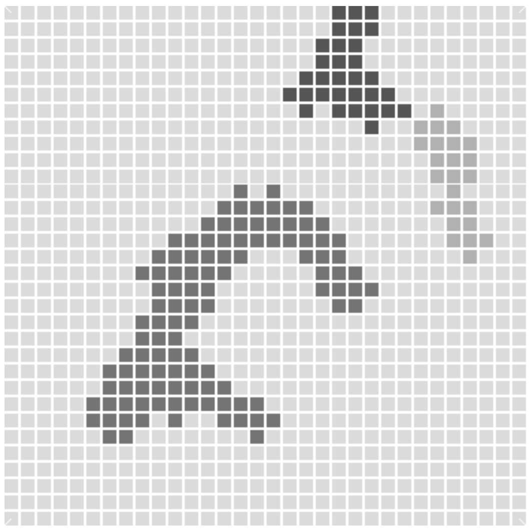

(a) Segmented Data (b) Final Output

Figure IV.5: Output of DENCLUE implementation on 2D synthetic data

➢ ***High Threshold Run (2D.II)***

Keeping all other parameters same with previous run, we just increased our main threshold to 6. Only extracted classes are effected from this change as expected. Segmented data for this run is shown in Figure IV.6 using both 4-Neighborhood and 8-Neighborhood segmentation options. Number of clusters extracted were 3 and 2, using 4 and 8 neighborhood segmentation respectively. Since number of clusters extracted changes with segmentation option we could conclude that 6 is not a appropriate main threshold value for our “Two Dimensional Synthetic Data” set, after being quantized to 32 bins for each dimension. Calculation reports printed using 4 and 8 neighborhood segmentation are given in Table IV.3 and Table IV.4 respectively. Although there are no negative indicators in both reports, this run is marked as an unsuccessful one due to changing number of clusters with neighborhood selection.


(a) 4-Neighborhood

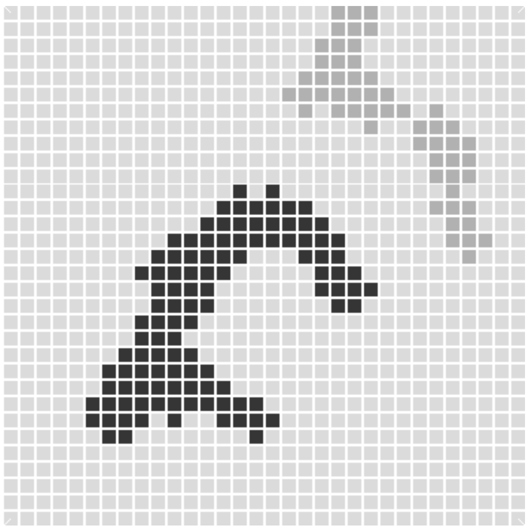

(b) 8-Neighborhood

Figure IV.6: Segmented Data for Run 2D.II

Table IV.3: Calculation Report for Run 2D.II using 4-Neigh. Segmentation

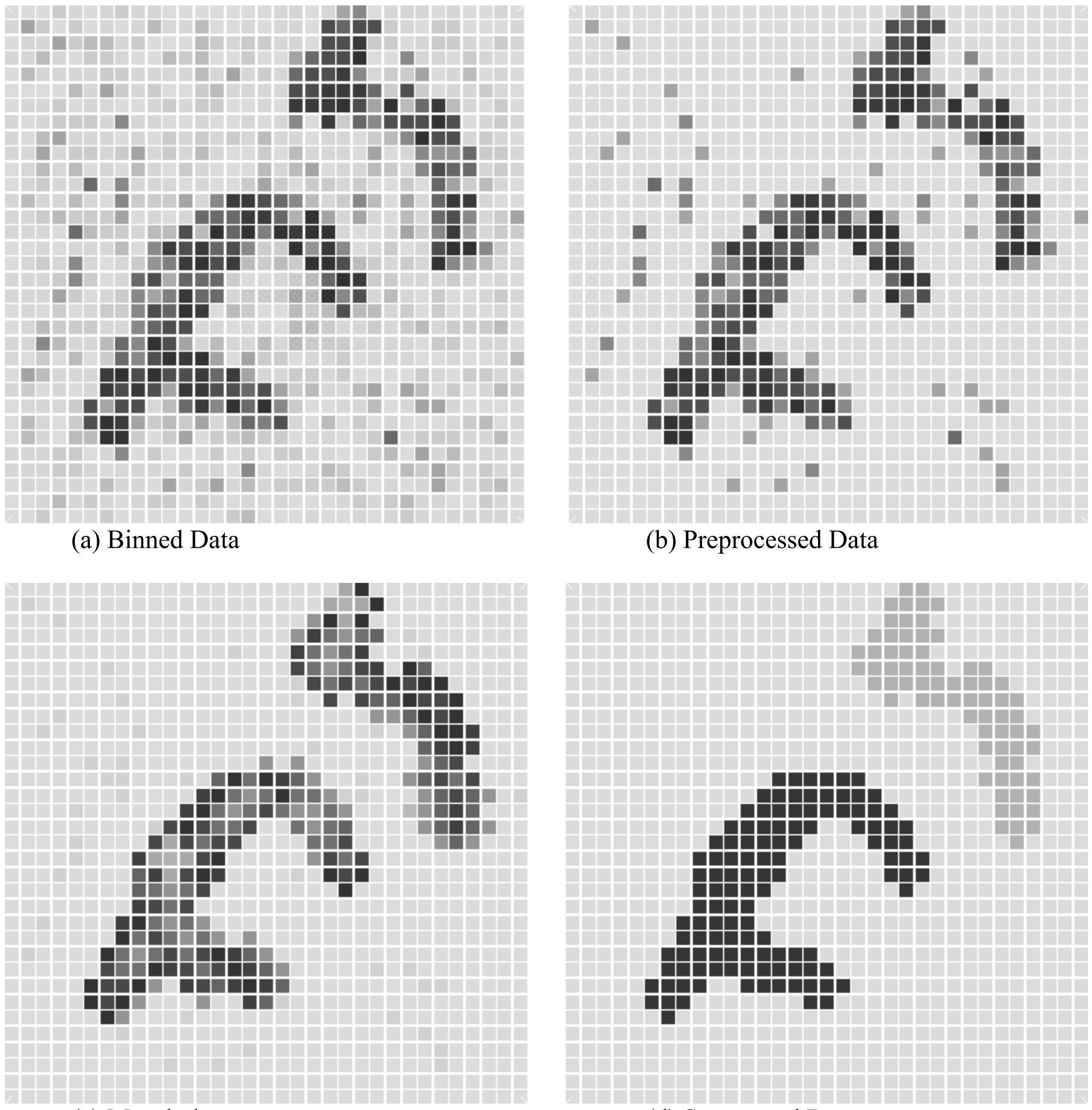
	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***X Range/Percent***	***Y Range/Percent***
Overall	1024	2783	2.72		
Backgr.	860/84.0	1368/49.2	**1.59**	32/100.0	32/100.0
Cl. 1	109/10.6	932/33.5	**8.55**	**18/56.2**	**16/50.0**
Cl. 2	31/3.0	268/9.6	**8.65**	**8/25.0**	**8/25.0**
Cl. 3	24/2.3	215/7.7	**8.96**	**5/15.6**	**10/31.2**
Cl. All		**1415/50.8**	8.63		

Table IV.4: Calculation Report for Run 2D.II using 8-Neigh. Segmentation

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***X Range/Percent***	***Y Range/Percent***
Overall	1024	2783	2.72		
Backgr.	860/84.0	1368/49.2	**1.59**	32/100.0	32/100.0
Cl. 1	109/10.6	932/33.5	**8.55**	**18/56.2**	**16/50.0**
Cl. 2	55/5.4	483/17.4	**8.78**	**13/40.6**	**16/50.0**
Cl. All		**1415/50.8**	8.63		

➢ ***Bin Offsetting Run (2D.III)***

In this run, we used same parameters with our reference run "Run 2D.I", except bin offsetting flag. Bin offsetting flag is set to true for both dimensions.


(a) Binned Data (b) Preprocessed Data

(c) Morphology (d) Segmented Data

Figure IV.7: Stages for Run 2D.III

Although we entered 32 as "Number of Bins" for each dimension, due to offsetting calculation "Number of Bins" parameter is set as 33 for both, at runtime.

Visually, extracted classes are very similar to "Run 2D.I" as shown in Figure IV.7. This verifies that our quantization parameters were appropriate and has no negative effects on clustering results.

Directly comparing final results, we see that a total number of 148 data points are marked different in runs "Run 2D.I" and "Run 2D.III", which corresponds to 9.71% of all data points that are marked as a member of a cluster. Calculation report for "Run 2D.III" is shown in Table IV.5.

Table IV.5: Calculation report for Run 2D.III

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***X Range/Percent***	***Y Range/Percent***
Overall	1089	2783	2.56		
Backgr.	897/82.4	1222/43.9	**1.36**	33/100.0	33/100.0
Cl. 1	120/11.0	992/35.6	**8.27**	**18/54.5**	**16/48.5**
Cl. 2	72/6.6	569/20.4	**7.90**	**12/36.4**	**17/51.5**
Cl. All		**1561/56.1**	8.13		

- ***High Resolution Run (2D.IV)***

Parameters for this run are selected relatively different with respect to the reference. “Number of Bins” for both dimensions are increased to 48, both threshold values are decreased to 3 and closing structuring element diameter is enlarged to 5 for both dimensions.

Stages of run 2D.IV is shown in , and calculation report is shown in Table IV.6. Results appeared very similar to the reference run. Direct comparison of outputs point out that, there exists 151 shifted data points which corresponds to the 9.91% of all data points that are marked to be a cluster member. These results reveals that the algorithm is quite robust to parameter selections. For visual comparison purposes final results for “Run 2D.I” and “Run 2D.IV” are shown together in Figure IV.9.

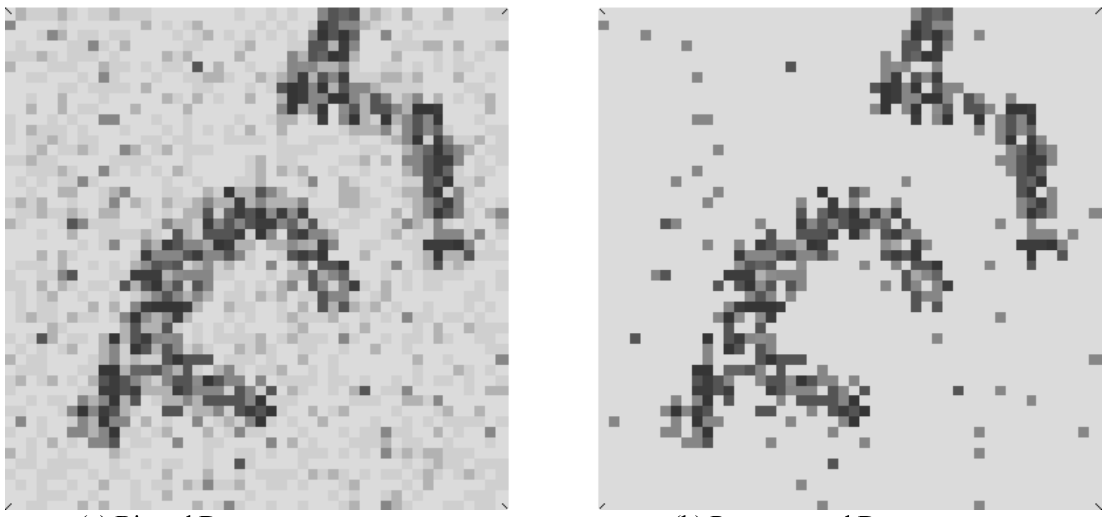

(a) Binned Data (b) Preprocessed Data

Figure IV.8: Stages for Run 2D.IV

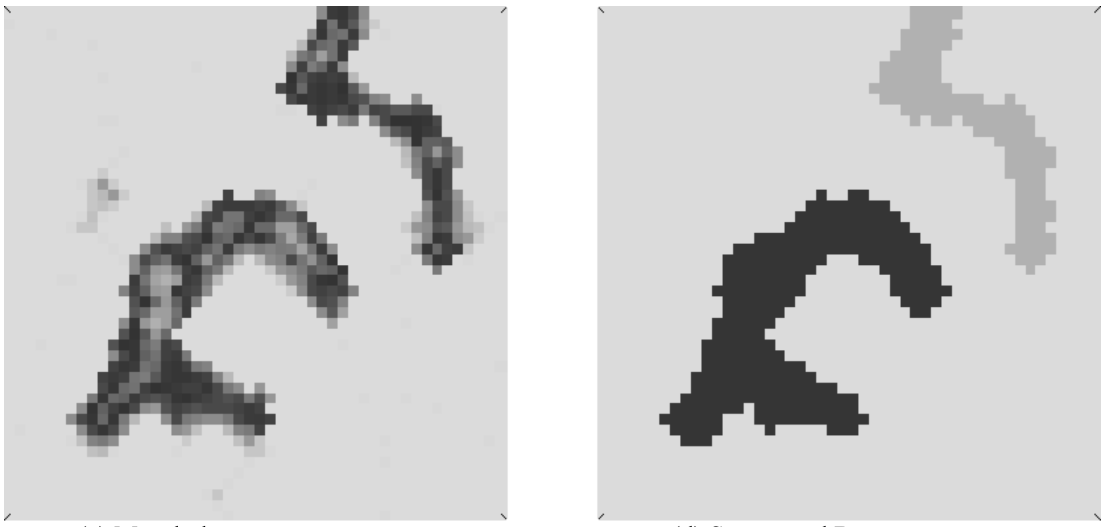

(c) Morphology (d) Segmented Data

Figure IV.8 cont'd: Stages for Run 2D.IV

Table IV.6: Calculation Report for Run 2D.IV

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***X Range/Percent***	***Y Range/Percent***
Overall	2304	2783	1.21		
Backgr.	1899/82.4	1273/45.7	**0.67**	48/100.0	48/100.0
Cl. 1	272/11.8	989/35.5	**3.64**	**28/58.3**	**24/50.0**
Cl. 2	133/5.8	521/18.7	**3.92**	**18/37.5**	**25/52.1**
Cl. All		**1510/54.3**	3.73		

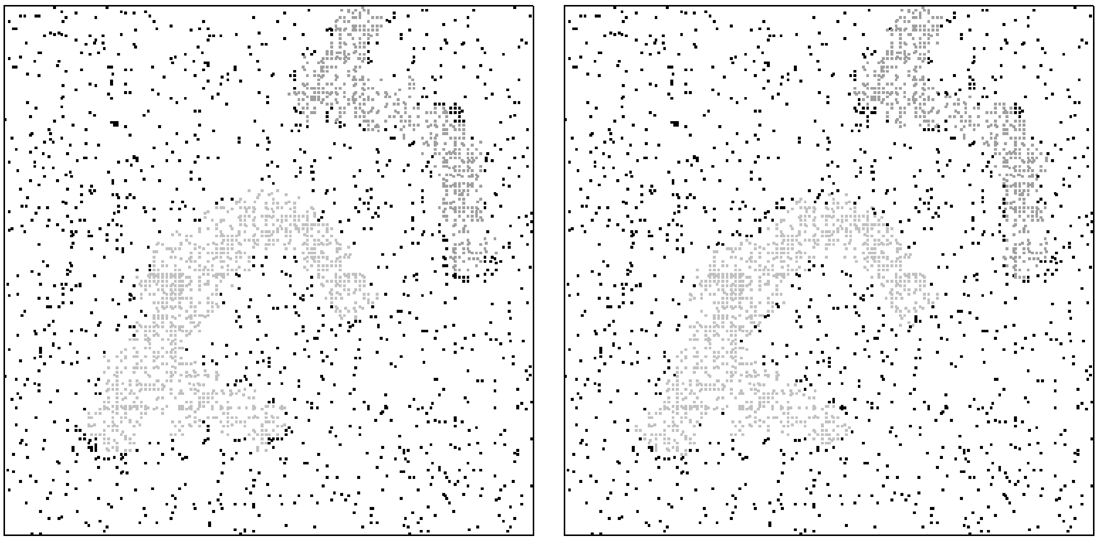

(a) Run 2D.I (b) Run 2D.IV

Figure IV.9: Comparison of final results for "Run 2D.I" and "Run 2D.IV"

IV.3.2. Runs On Three Dimensional Synthetic Data

We use three different parameter families to process our three dimensional synthetic data, which are shown in Table IV.7 in detail. In Run 3D.II the effect of using variable threshold is observed. Run 3D.III, on the other hand, shows the effect of using a structuring element that emphasizes certain directions.

Table IV.7: Parameter Families for 3D Synthetic Data

	Run 3D.I			***Run 3D.II***			***Run 3D.III***		
Dimension	*Dim1*	*Dim2*	*Dim3*	*Dim1*	*Dim2*	*Dim3*	*Dim1*	*Dim2*	*Dim3*
# of Bins	52	52	52	52	52	52	52	52	52
Opening Strel. Diameter	3	3	3	3	3	3	3	**13**	3
Closing Strel. Diameter	3	3	3	3	3	3	3	3	3
Use Logarithmic Scale	No	No	No	No	No	No	No	No	No
Use Bin Offsetting	No	No	No	No	No	No	No	No	No
Threshold Distribution	-	-	-	**70,100; 100,55**	-	-	-	-	-
Preprocessing Threshold	4			4			4		
Main Threshold	4			4			4		
Opening Strel. Height	1			1			1		
Closing Strel. Height	1			1			1		
Morphological Operation	Closing + Opening			Closing + Opening			Closing + Opening		

➢ ***Reference Run (3D.I)***

First run on our three dimensional synthetic data is the reference run for this data set. Two clusters described in Section IV.2.1 are extracted successfully. Visual demonstration of stages and result are shown in Figure IV.10 and calculation report is shown in Table IV.8.

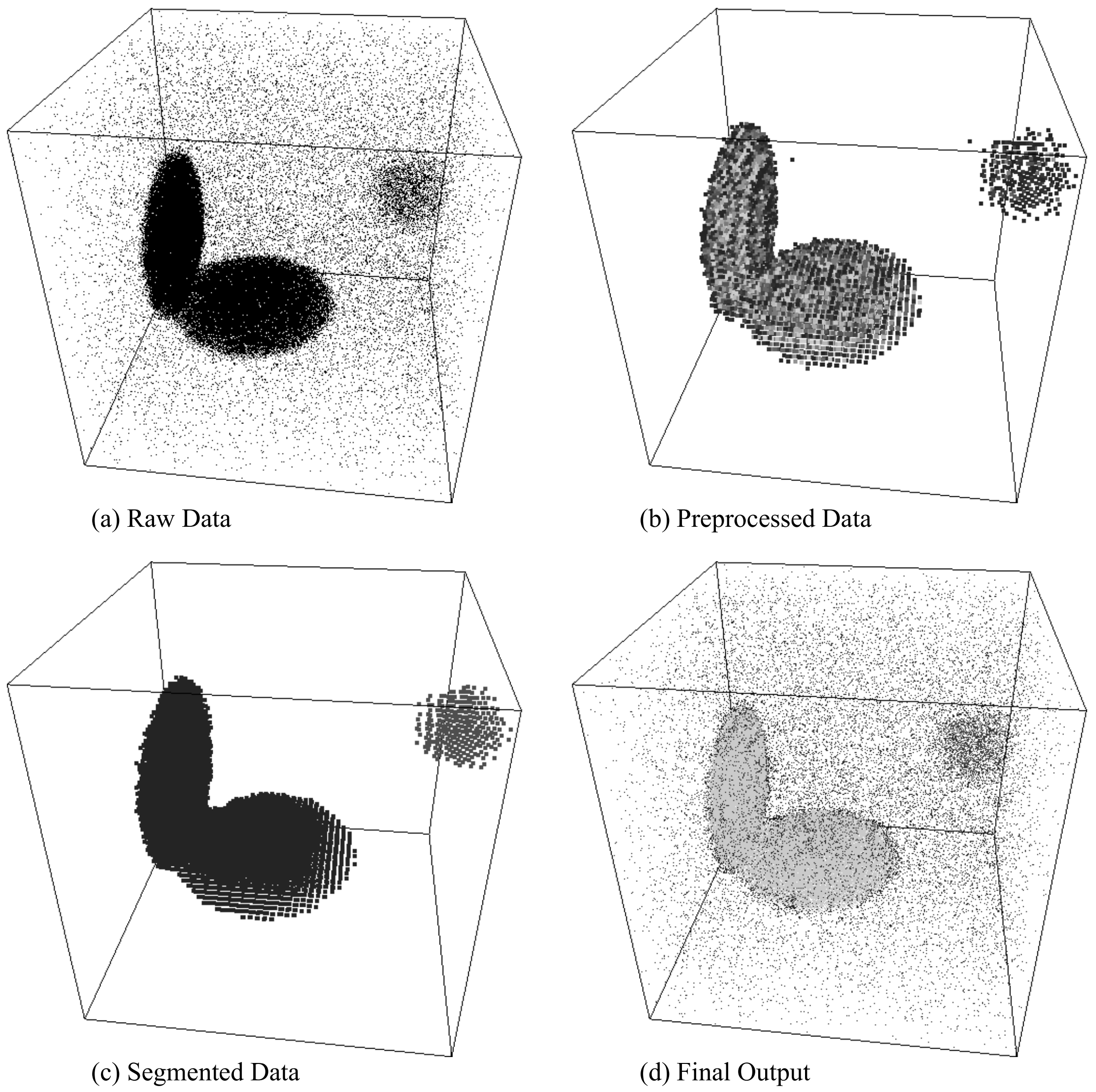


Figure IV.10: Output of Stages for Run 3D.I

Table IV.8: Calculation Report for Run 3D.I

	Member Bin Count/ Percent	***Member Data Pt Count/ Percent***	***Density***	***Dim_1 Range/ Percent***	***Dim_2 Range/ Percent***	***Dim_3 Range/ Percent***
Overall	140608	87248	0.62			
Backgr.	134231/95.5	19803/22.7	**0.15**	52/100.0	52/100.0	52/100.0
Cl. 1	6108/4.3	65984/75.6	**10.80**	**32/61.5**	**32/61.5**	**29/55.8**
Cl. 2	269/0.2	1461/1.7	**5.43**	**9/17.3**	**9/17.3**	**9/17.3**
Cl. All		**67445/77.3**	10.58			

➢ ***Threshold Distribution Run (3D.II)***

Although we are aware of that one of the clusters had less density, in the reference run we used constant threshold values for preprocessing and main threshold stages. In this run we use a staircase threshold function on related dimension in order to mark members of the loose cluster more accurately. We set to use 55% of the real threshold value after the 70 % of the data range on axis “Dim_2”.

Results are demonstrated in Figure IV.11 and report is shown in Table IV.9. Comparing to the reference run, second cluster is enlarged 26% in volume and members of the second cluster are increased 16.5% in number.

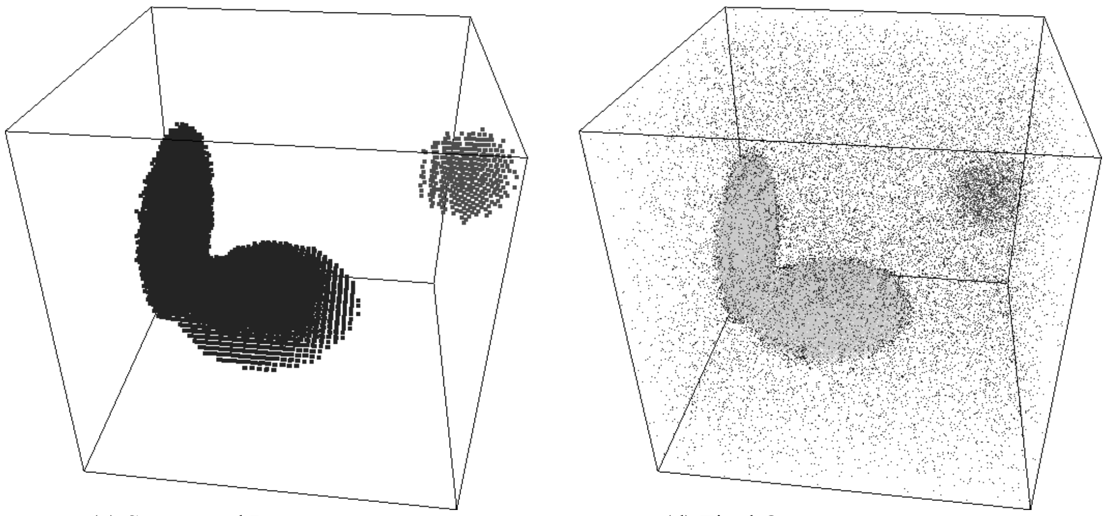

(c) Segmented Data (d) Final Output

Figure IV.11: Segmented Data and Final Output for Run 3D.II

Table IV.9: Calculation Report for Run 3D.II

	Member Bin Count/ Percent	***Member Data Pt Count/ Percent***	***Density***	***Dim_1 Range/ Percent***	***Dim_2 Range/ Percent***	***Dim_3 Range/ Percent***
Overall	140608	87248	0.62			
Backgr.	134231/95.5	19803/22.7	**0.15**	52/100.0	52/100.0	52/100.0
Cl. 1	6108/4.3	65984/75.6	**10.80**	**32/61.5**	**32/61.5**	**29/55.8**
Cl. 2	339/0.2	1702/2.0	**5.02**	**9/17.3**	**9/17.3**	**9/17.3**
Cl. All		**66644/76.4**	5.95			

➢ ***Non Spherical Structuring Element Run (3D.III)***

This time we assume that our problem is finding vertical or horizontal clusters in the three dimensional space. In order to extract vertical clusters we just need to use a vertical structuring element for opening.

Results, demonstrated in Figure IV.12 and reported in Table IV.10, are as expected. A single vertical cluster is extracted, which is the vertical side of the

"Cluster 1" extracted in the reference run. Although calculation report shows that background density is very close to the overall density for this run, this is because dense regions that are thin on vertical axis are not marked as clusters.

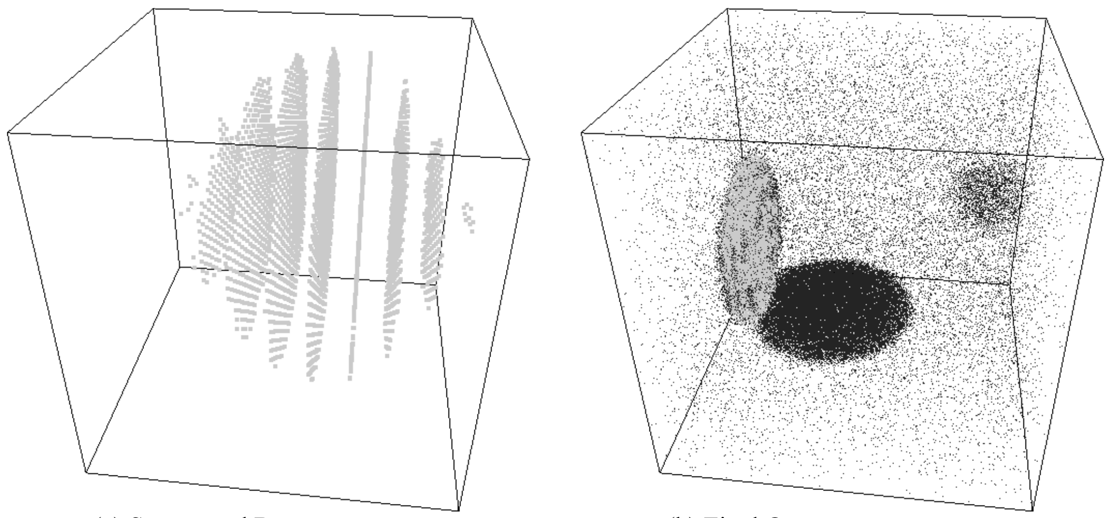

(a) Segmented Data (b) Final Output

Figure IV.12: Segmented Data and Final Output for Run 3D.III

Table IV.10: Calculation Report for Run 3D.III

	Member Bin Count/ Percent	***Member Data Pt Count/ Percent***	***Density***	***Dim_1 Range/ Percent***	***Dim_2 Range/ Percent***	***Dim_3 Range/ Percent***
Overall	140608	87248	0.62			
Backgr.	137702/97.9	54836/62.9	**0.40**	52/100.0	52/100.0	52/100.0
Cl. 1	2906/2.1	32412/37.1	**11.15**	**9/17.3**	**30/57.7**	**28/53.8**
Cl. All		**32412/37.1**	11.15			

IV.3.3. Runs On Four Dimensional Synthetic Data

Our aim in this run is showing our implementation to handle dimensions more than three. A four dimensional data set is selected in order to visually demonstrate the output as 3D projection on one of the dimensions. We made two runs on four dimensional data set. The first is the reference run and the second is the projection run. Parameter families user in these runs are detailed in Table IV.11.

Table IV.11: Parameter Families for 4D Synthetic Data

	Run 3D.I				**Run 3D.II**		
Dimension	*Dim1*	*Dim2*	*Dim3*	*Dim4*	*Dim2*	*Dim3*	*Dim4*
# of Bins	32	32	32	32	32	32	32
Opening Strel. Diameter	3	3	3	3	3	3	3
Closing Strel. Diameter	3	3	3	3	3	3	3
Use Logarithmic Scale	No	No	No	No	No	No	No
Use Bin Offsetting	No	No	No	No	No	No	No
Threshold Distribution	-	-	-	-	-	-	-
Preprocessing Threshold	4				36		
Main Threshold	5				36		
Opening Strel. Height	1				1		
Closing Strel. Height	1				1		
Morphological Operation	Closing + Opening				Closing + Opening		

➢ ***Reference Run (4D.I)***

In our reference run, we cannot visually demonstrate the output since it is four dimensional. The only output we have is the calculation report, shown in Table IV.12. We extracted three clusters as expected. We marked only 18.7% of the data points as member of one of the clusters in this run, but this is due to relatively large space consisting of 1,048,576 pixels and total volume of our clusters is 13,155 pixels.

Table IV.12: Calculation Report for Run 4D.I

	Member Bin Count/ Percent	***Member Data Pt Count/ Percent***	***Density***	***Dim_1 Range/ Percent***	***Dim_2 Range/ Percent***	***Dim_3 Range/ Percent***	***Dim_4 Range/ Percent***
Overall	1048576	687693	0.66				
Backgr.	1035421/98.7	559348/81.3	**0.54**	32/100	32/100	32/100	32/100
Cl. 1	4434/0.4	43305/6.3	**9.77**	**12/37.5**	**12/37.5**	**12/37.5**	**12/37.5**
Cl. 2	4378/0.4	42818/6.2	**9.78**	**12/37.5**	**12/37.5**	**12/37.5**	**12/37.5**
Cl. 3	4343/0.4	42222/6.1	**9.72**	**12/37.5**	**12/37.5**	**12/37.5**	**12/37.5**
Cl. All		**128345/18.7**	9.76				

➢ ***Projection Run (4D.II)***

This run is projection of the previous on dimension “Dim_1”, visual output is shown in Figure IV.13 and calculation report is shown in Table IV.13.

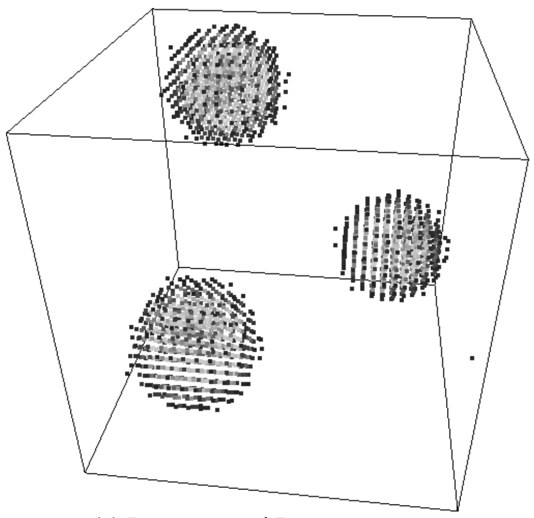
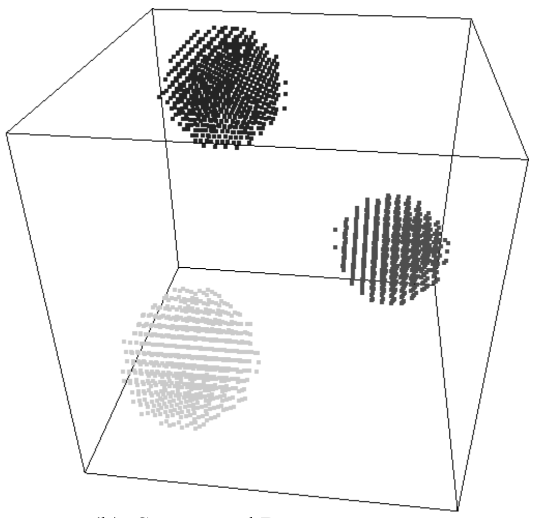


(a) Preprocessed Data (b) Segmented Data

Figure IV.13: Preprocessed Data and Segmented Data for Run 4D.II

Cluster members are increased in number with respect to the reference run because while projecting from four dimensions to three, background noise that fall under any object on the projection axis is marked as a member of that object as well.

Table IV.13: Calculation Report for Run 4D.II

	Member Bin Count/ Percent	***Member Data Pt Count/ Percent***	***Density***	***Dim_2 Range/ Percent***	***Dim_3 Range/ Percent***	***Dim_4 Range/ Percent***
Overall	32768	687693	20.99			
Backgr.	30752/93.8	518782/75.4	**16.87**	32/100.0	32/100.0	32/100.0
Cl. 1	666/2.0	56293/8.2	**84.52**	**12/37.5**	**11/34.4**	**11/34.4**
Cl. 2	676/2.1	56742/8.3	**83.94**	**11/34.4**	**12/37.5**	**12/37.5**
Cl. 3	674/2.1	55876/8.1	**82.90**	**12/37.5**	**11/34.4**	**11/34.4**
Cl. All		**168911/24.6**	83.79			

IV.3.4. Runs on Real Data

As it is stated in Section 4.2.2, we have two three dimensional data sets with same characteristics. Data in these sets are collected in different months, thus we will label them as “*Month A Data*” and “*Month B Data*”. “Month A Data” consists of 1,432,780 data points and “Month B Data” consists of 1,329,481 data points.

Both sets are previously preprocessed and non-linearly quantized on two dimensions, “Age Group” and “Consumption Interval”. Quantization rules for these dimension are shown in tables Table IV.14 and Table IV.15 respectively.

Table IV.14: Quantization on Dimension "Age Group"

Age Group Id	***Age Group***
1	0 - 17
2	18 - 22
3	23 - 29
4	30 - 39
5	40 - 49
6	50 - 59
7	60 and more

Table IV.15: Quantization on Dimension "Consumption Interval"

Consumption Interval Id	***Consumption Interval***
1	0 YTL - 49 YTL
2	50 YTL - 99 YTL
3	100 YTL - 199 YTL
4	200 YTL - 299 YTL
5	300 YTL - 399 YTL
6	400 YTL - 499 YTL
7	500 YTL - 599 YTL
8	600 YTL - 699 YTL
9	700 YTL - 799 YTL
10	800 YTL - 899 YTL
11	900 YTL - 999 YTL
12	1.000 YTL - 1.499 YTL
13	1.500 YTL - 1.999 YTL
14	2.000 YTL - 4.999 YTL
15	5.000 YTL and more

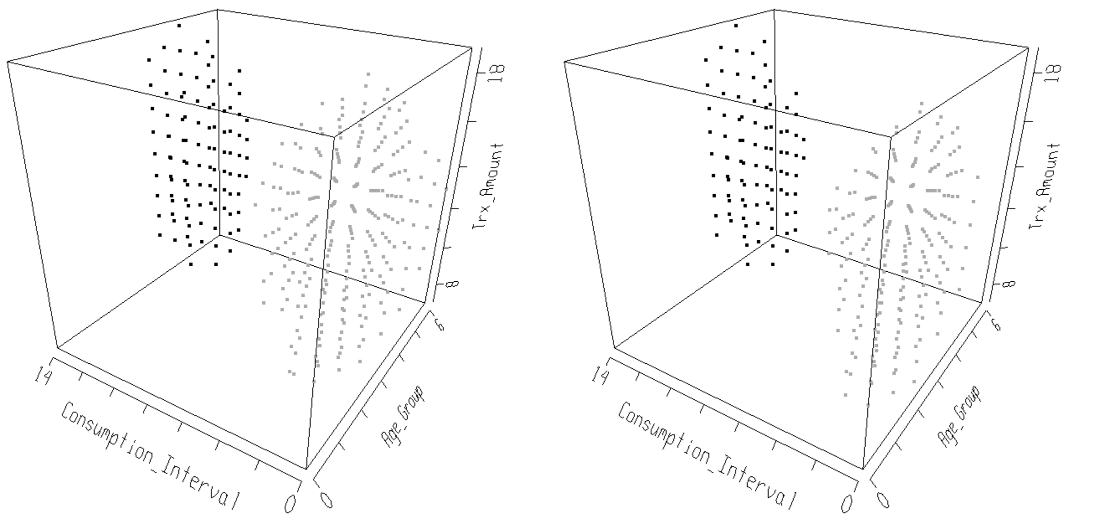
Parameter families used for runs on real data is shown in Table IV.16. First column in the table depicts the reference run, and changes from the reference for the other runs are indicated using bold characters and gray cells. Runs with changing the structuring element heights are not included, since no significant effects are observed for our particular data.

Table IV.16: Parameter Families for Runs on Real Data

	Dimension	N.I	N.II	N.III	N.IV	N.V	N.VI	N.VII	N.VIII	N.IX	N.X	N.XI
Age Group	*Number of Bins*	6	6	6	6	6	6	**4**	6	6	6	6
	Opening Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Closing Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Logarithmic Scale	No	No	No	No	No	No	No	No	No	No	No
	Bin Offsetting	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**No**	Yes	Yes
	Threshold Dist.	-	-	-	-	-	-	-	-	-	-	-
Consumption Interval	*Number of Bins*	14	14	14	14	14	14	**9**	14	14	14	14
	Opening Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Closing Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Logarithmic Scale	No	No	No	No	No	No	No	No	No	No	No
	Bin Offsetting	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**No**	Yes	Yes
	Threshold Dist.	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	*65,100; 100,75*	**-**
Trx Amount	*Number of Bins*	32	32	32	32	32	32	**10**	32	32	32	32
	Opening Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Closing Strel.	3	**5**	3	3	3	3	3	3	3	3	3
	Logarithmic Scale	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	**No**	Yes
	Bin Offsetting	No	No	No	No	No	No	No	No	No	No	No
	Threshold Dist.	-	-	-	-	-	-	-	-	-	-	-
	Pre. Thresh.	750	750	750	750	750	**1250**	**1500**	**250**	750	750	750
	Main Threshold	1000	1000	1000	1000	1000	**2500**	**3000**	**500**	1000	1000	1000
	Opening Strel. Hg.	5	5	5	5	5	5	5	5	5	5	5
	Closing Strel. Hg.	5	5	5	5	5	5	5	5	5	5	5
	Morphological Op.	Close Open	Close Open	**Close**	**Open**	**Open Close**	Close Open	Close Open	Close Open	Close Open	Close Open	Close Open

➢ ***Reference Run (N.I)***

A series of trials convinced us that "Run N.I" produces the best result according to the visual outputs and calculation reports, which are shown in Figure IV.14 and Tables IV.17, IV.18 respectively. On Figure IV.14, horizontal axis corresponds to "Consumption Interval", vertical axis corresponds to "Transaction Amount" and depth axis corresponds to "Age Group" dimensions.



(a) Month A Data (b) Month B Data

Figure IV.14: Segmentation Output of "Run N.I"

Interpreting output of “Run N.I”, for both data sets we observe that the credit card customers of this bank spans two clear distinct clusters. The first cluster contains people that have a relatively high spending habits. In this group, middle aged customers are dominant and there are no young customers. These people use their credit cards for a wide range of transaction amounts: they use their credit cards for all types of activities. The second cluster is larger and contains people that do not have a high consumption habits. In this group there are customers from every age group, but we observe that young and above middle-aged customers make moderate transactions in general.

In these results, there are two striking points.

1. The two clusters are distinctly separate from each other.
2. There is a large number of customers who makes high amount transactions but has low consumption habits.

As a result of this analysis, the bank may take different actions, such as:

- Mounting different promotion campaigns targeting two different clusters of customers.
- Improving relationships with customers that make a small number of high amount transactions but have low consumption habits. These customers may be working with other banks.

Table IV.17: Calculation Report of "Run N.I" for "Month A Data"

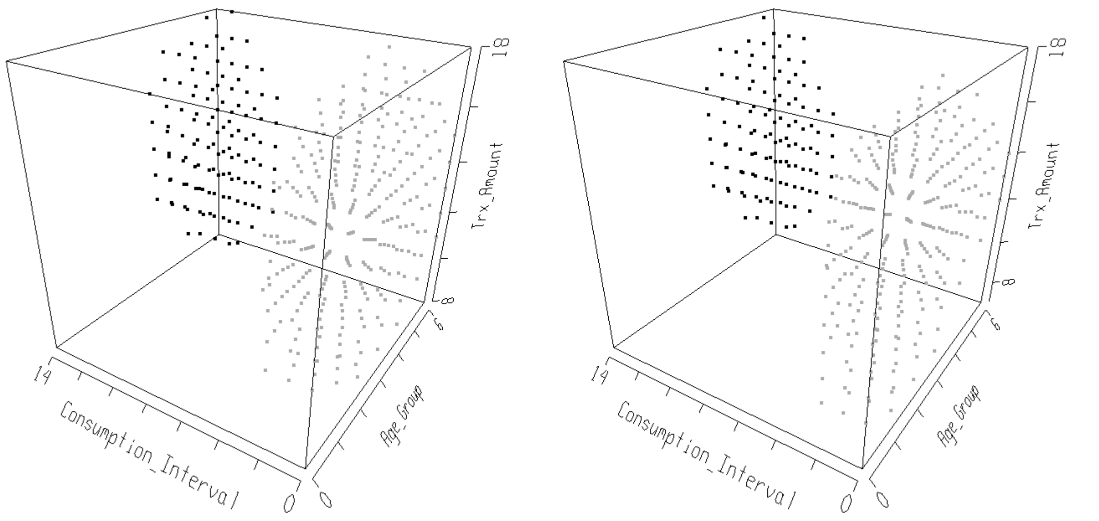
	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr.	2989/89.0	688215/48.0	**230.25**	7/100.0	15/100.0	32/100.0
Cl. 1	269/8.0	571297/39.9	**2123.78**	**7/100.0**	**9/60.0**	**13/40.6**
Cl. 2	102/3.0	173268/12.1	**1698.71**	**4/57.1**	**5/33.3**	**12/37.5**
Cl. All		**744565/52.0**	2006.91			

Table IV.18: Calculation Report of "Run N.I" for "Month B Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr.	3043/90.6	698009/52.5	**229.38**	7/100.0	15/100.0	32/100.0
Cl. 1	225/6.7	486395/36.6	**2161.76**	**7/100.0**	**8/53.3**	**12/37.5**
Cl. 2	92/2.7	145077/10.9	**1576.92**	**4/57.1**	**5/33.3**	**12/37.5**
Cl. All		**631472/47.5**	1992.03			

➢ ***Big Structuring Element Run (N.II)***

In this run we used larger structuring elements, each has a diameter of 5 in all dimensions. Similar convex clusters are formed as expected. Visual outputs and calculation reports are shown in Figure IV.15 and Tables IV.19, IV.20 respectively.



(a) Month A Data (b) Month B Data

Figure IV.15: Segmentation Output of "Run N.II"

Table IV.19: Calculation Report of "Run N.II" for "Month A Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr.	2936/87.4	706259/49.3	**240.55**	7/100.0	15/100.0	32/100.0
Cl. 1	300/8.9	566385/39.5	**1887.95**	**7/100.0**	**8/53.3**	**11/34.4**
Cl. 2	124/3.7	160136/11.2	**1291.42**	**4/57.1**	**5/33.3**	**10/31.2**
Cl. All		**726521/50.7**	1713.49			

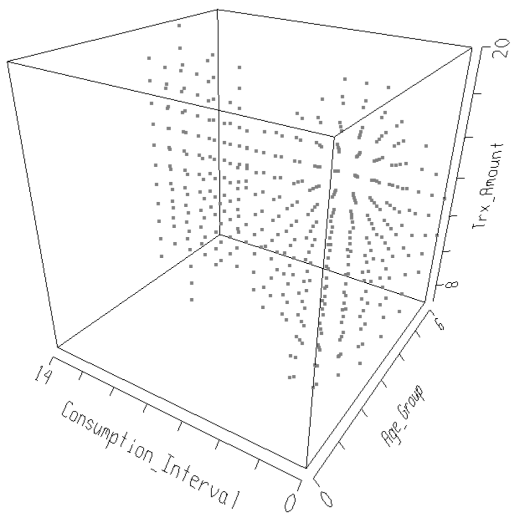
Table IV.20: Calculation Report of "Run N.II" for "Month B Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr.	2952/87.9	664203/50.0	**225.00**	7/100.0	15/100.0	32/100.0
Cl. 1	293/8.7	531786/40.0	**1814.97**	**7/100.0**	**9/60.0**	**11/34.4**
Cl. 2	115/3.4	133492/10.0	**1160.80**	**4/57.1**	**5/33.3**	**10/31.2**
Cl. All		**665278/50.0**	1630.58			

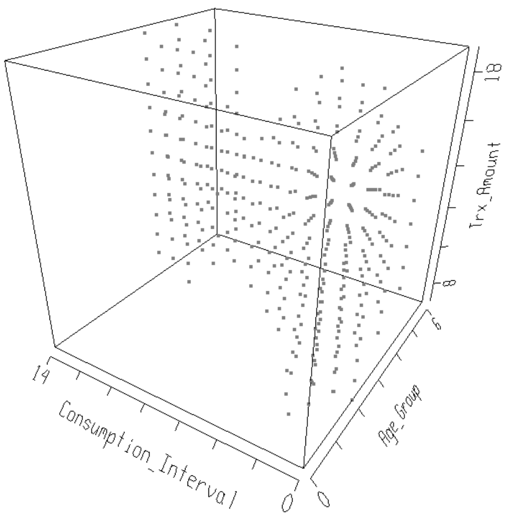
In this run we cleaned out more noise from the same clusters that are found in "Run N.I". As a result, although clusters found in this run are not as dense as those of the reference clusters, it will be inappropriate to mark this run as low quality.

- ***Closing Only Run (N.III)***

In this run we used only closing as morphological operation, clusters extracted in reference run are joined in this run. Visual outputs and calculation reports are shown in Figure IV.3 and Tables IV.21, IV.22 respectively.



(a) Month A Data



(b) Month B Data

Figure IV.16: Segmentation Output of "Run N.III"

Table IV.21: Calculation Report of "Run N.III" for "Month A Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr.	2920/86.9	574248/40.1	**196.66**	7/100.0	15/100.0	32/100.0
Cl. 1	440/13.1	858532/59.9	**1951.21**	**7/100.0**	**15/100.0**	**14/43.8**
Cl. All		**858532/59.9**	1951.21			

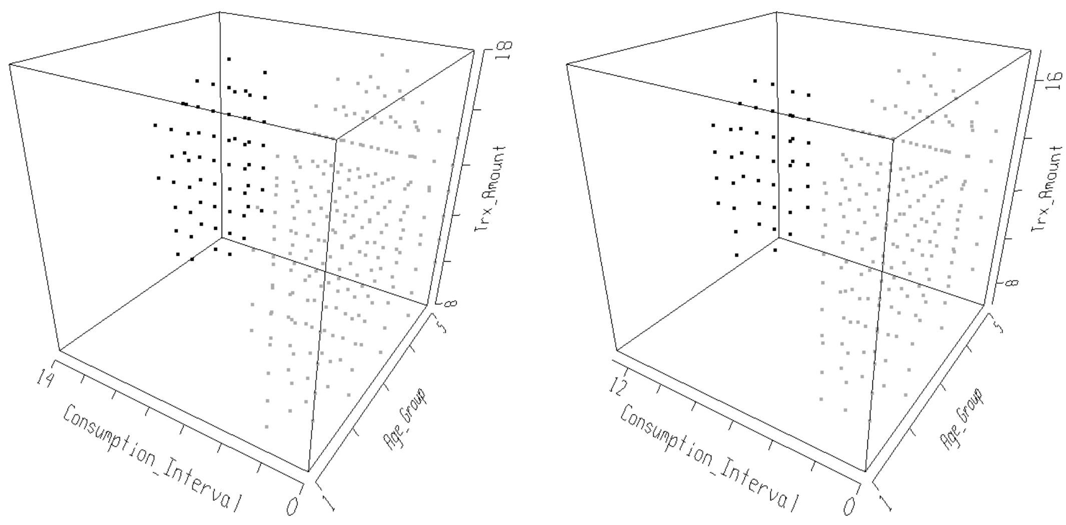
Table IV.22: Calculation Report of "Run N.III" for "Month B Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr.	2959/88.1	557133/41.9	**188.28**	7/100.0	15/100.0	32/100.0
Cl. 1	401/11.9	772348/58.1	**1926.05**	**7/100.0**	**15/100.0**	**13/40.6**
Cl. All		**772348/58.1**	1926.05			

Utilizing only closing operation resulted putting almost all data in a dense single cluster. We can infer this from the volume of the cluster extracted.

- ***Opening Only Run (N.IV)***

In this run we used only opening as morphological operation, clusters extracted in reference run are shrunk in this run. Visual outputs and calculation reports are shown in Figure IV.17 and Tables IV.23, IV.24 respectively.



(a) Month A Data (b) Month B Data

Figure IV.17: Segmentation Output of "Run N.IV"

Table IV.23: Calculation Report of "Run N.IV" for "Month A Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr.	3066/91.2	781265/54.5	**254.82**	7/100.0	15/100.0	32/100.0
Cl. 1	231/6.9	532150/37.1	**2303.68**	**5/71.4**	**9/60.0**	**11/34.4**
Cl. 2	63/1.9	119365/8.3	**1894.68**	**3/42.9**	**5/33.3**	**9/28.1**
Cl. All		**651515/45.5**	2216.04			

Table IV.24: Calculation Report of "Run N.IV" for "Month B Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr.	3118/92.8	789962/59.4	**253.36**	7/100.0	15/100.0	32/100.0
Cl. 1	197/5.9	460245/34.6	**2336.27**	**5/71.4**	**8/53.3**	**11/34.4**
Cl. 2	45/1.3	79274/6.0	**1761.64**	**3/42.9**	**4/26.7**	**8/25.0**
Cl. All		**539519/40.6**	2229.42			

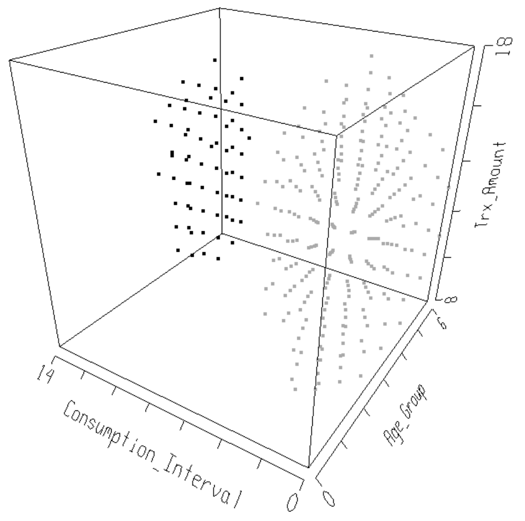
Using opening operation alone results in pushing some cluster members in dense regions out of the clusters, which degrades output quality.

- ***Opening Followed by Closing Run (N.V)***

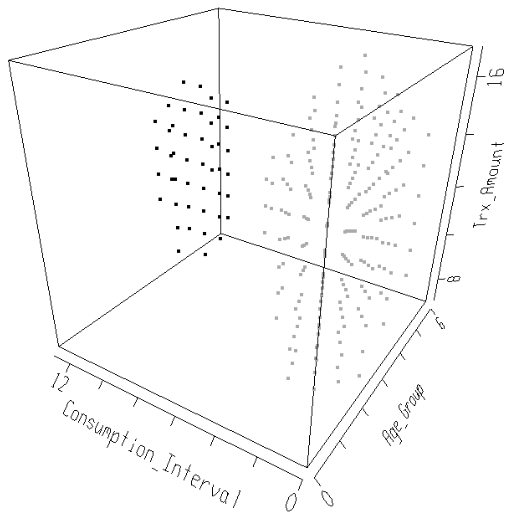
In this run we used opening followed by closing as morphological operation, Convex clusters are formed similar to the previous run. Visual outputs and calculation reports are shown in Figure IV.18 and Tables IV.25, IV.26 respectively.

Table IV.25: Calculation Report of "Run N.V" for "Month A Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr.	3039/90.4	760965/53.1	**250.40**	7/100.0	15/100.0	32/100.0
Cl. 1	257/7.6	551861/38.5	**2147.32**	**7/100.0**	**9/60.0**	**11/34.4**
Cl. 2	64/1.9	119954/8.4	**1874.28**	**4/57.1**	**5/33.3**	**9/28.1**
Cl. All		**671815/46.9**	2092.88			



(a) Month A Data



(b) Month B Data

Figure IV.18: Segmentation Output of "Run N.V"

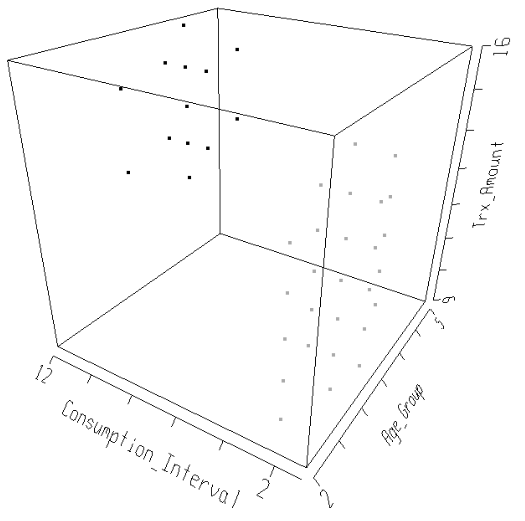
Table IV.26: Calculation Report of "Run N.V" for "Month B Data"

	Member Bin Count/Percent	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr.	3098/92.2	777332/58.5	**250.91**	7/100.0	15/100.0	32/100.0
Cl. 1	217/6.5	472875/35.6	**2179.15**	**7/100.0**	**8/53.3**	**11/34.4**
Cl. 2	45/1.3	79274/6.0	**1761.64**	**3/42.9**	**4/26.7**	**8/25.0**
Cl. All		**552149/41.5**	2107.44			

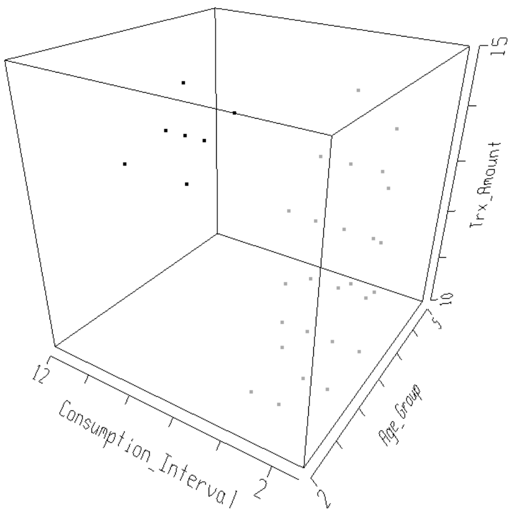
Since closing cannot fill all the bins that are pruned off by former opening operation, a very similar output to the previous run is obtained.

➢ ***High Thresholds Run (N.VI)***

In this run we used larger threshold values, 1250 and 2500 for preprocessing and main thresholds respectively. Rather smaller and denser clusters are formed than to the reference run. Visual outputs and calculation reports are shown in Figure IV.19 and Tables IV.27, IV.28 respectively.



(a) Month A Data



(b) Month B Data

Figure IV.19: Segmentation Output of "Run N.VI"

Selecting relatively large thresholds enables us to extract dense kernels of the real clusters. Although this is not high quality clustering operation, this output can be used as input for another clustering algorithm since it indicates the location of dense clusters.

Table IV.27: Calculation Report of "Run N.VI" for "Month A Data"

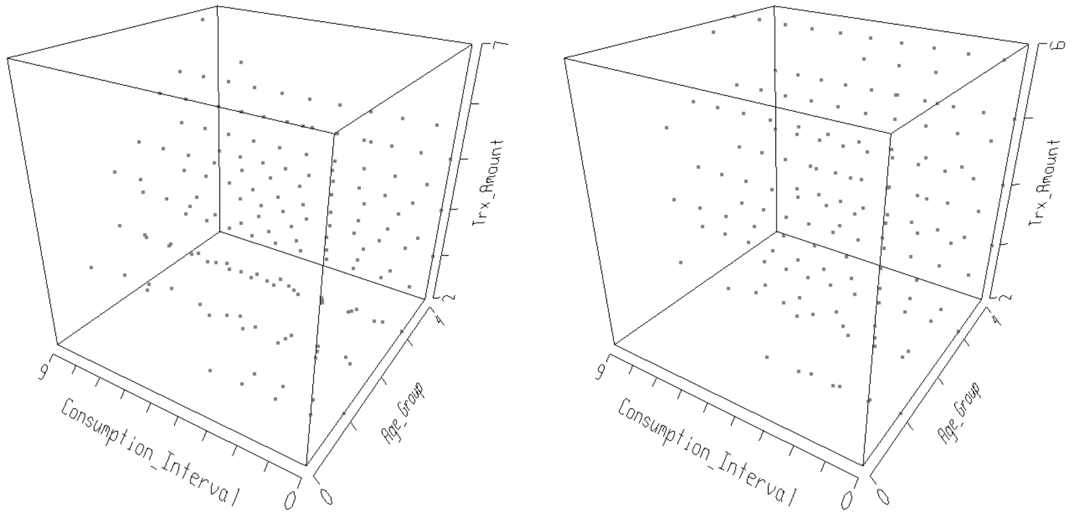
	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr	3320/98.8	1321649/92.2	**398.09**	7/100.0	15/100.0	32/100.0
Cl. 1	27/0.8	82075/5.7	**3039.81**	**3/42.9**	**3/20.0**	**7/21.9**
Cl. 2	13/0.4	29056/2.0	**2235.08**	**3/42.9**	**3/20.0**	**5/15.6**
Cl. All		**111131/7.8**	2778.28			

Table IV.28: Calculation Report of "Run N.VI" for "Month B Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr	3327/99.0	1233060/92.7	**370.62**	7/100.0	15/100.0	32/100.0
Cl. 1	26/0.8	80893/6.1	**3111.27**	**3/42.9**	**4/26.7**	**6/18.8**
Cl. 2	7/0.2	15528/1.2	**2218.29**	**3/42.9**	**3/20.0**	**3/9.4**
Cl. All		**96421/7.3**	2921.85			

➢ ***Large Bin Sizes Run (N.VII)***

In this run we used smaller number of bins for each dimension, namely 5, 10 and 10 for “Age Group”, “Consumption Interval” and “Transaction Amount” respectively. Besides we increased threshold values accordingly to 1500 and 3000 for preprocessing and main thresholds respectively. Visual outputs and calculation reports are shown in Figure IV.20 and Tables IV.29, IV.30 respectively.



(a) Month A Data (b) Month B Data

Figure IV.20: Segmentation Output of "Run N.VII"

Table IV.29: Calculation Report of "Run N.VII" for "Month A Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	500	1432780	2865.56			
Backgr.	339/67.8	101797/7.1	**300.29**	5/100.0	10/100.0	10/100.0
Cl. 1	161/32.2	1330983/92.9	**8266.98**	**5/100.0**	**10/100.0**	**6/60.0**
Cl. All		**1330983/92.9**	8266.98			

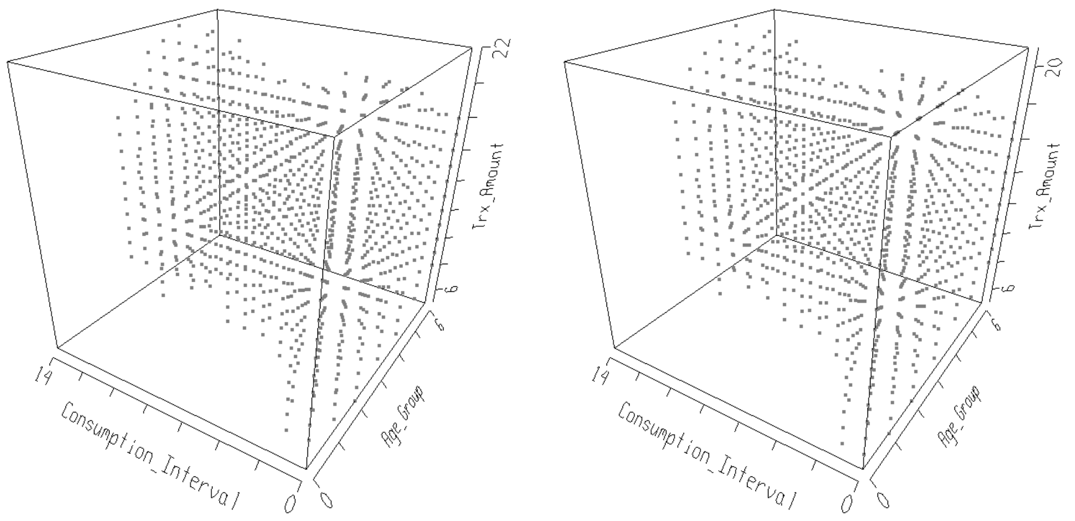
Table IV.30: Calculation Report of "Run N.CII" for "Month B Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	500	1329481	2658.96			
Backgr.	346/69.2	97379/7.3	**281.44**	5/100.0	10/100.0	10/100.0
Cl. 1	154/30.8	1232102/92.7	**8000.66**	**5/100.0**	**10/100.0**	**5/50.0**
Cl. All		**1232102/92.7**	8000.66			

Since we broke down pre-quantized data in an unnatural manner the clusters are unacceptable.

➢ ***Low Thresholds Run N.VIII (N.VIII)***

In this run we used smaller threshold values, 250 and 500 for preprocessing and main thresholds respectively. Rather larger and looser clusters are formed than to the reference run. Visual outputs and calculation reports are shown in Figure IV.21 and Tables IV.31, IV.32 respectively.



(a) Month A Data (b) Month B Data

Figure IV.21: Segmentation Output of "Run N.VIII"

Table IV.31: Calculation Report of "Run N.VIII" for "Month A Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr	2334/69.5	125604/8.8	**53.81**	7/100.0	15/100.0	32/100.0
Cl. 1	1026/30.5	1307176/91.2	**1274.05**	**7/100.0**	**15/100.0**	**18/56.2**
Cl. All		**1307176/91.2**	1274.05			

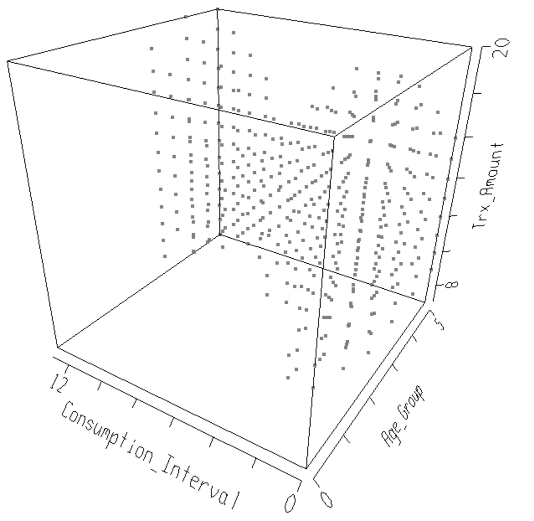
Selection of very low threshold values results in marking many data points as cluster members even though they are not.

Table IV.32: Calculation Report of "Run N.VIII" for "Month B Data"

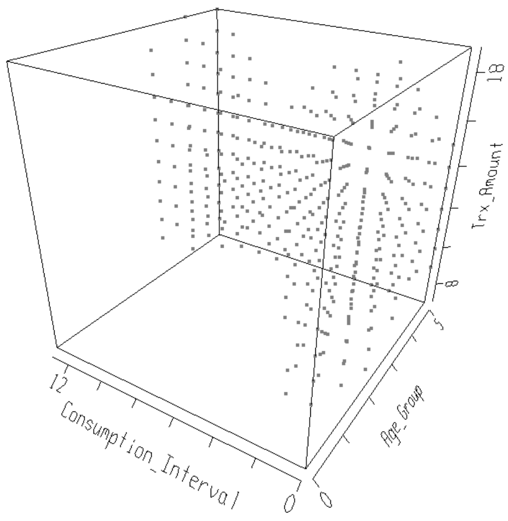
	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr	2390/71.1	120791/9.1	**50.54**	7/100.0	15/100.0	32/100.0
Cl. 1	970/28.9	1208690/90.9	**1246.07**	**7/100.0**	**15/100.0**	**17/53.1**
Cl. All		**1208690/90.9**	1246.07			

➢ ***No Bin Offsetting Run (N.IX)***

In this run we do not use bin offsetting for both dimensions “Age Group” and “Consumption Interval”. A large single cluster is formed. Visual outputs and calculation reports are shown in Figure IV.22 and Tables IV.33, IV.34 respectively.



(a) Month A Data



(b) Month B Data

Figure IV.22: Segmentation Output of "Run N.IX"

Table IV.33: Calculation Report of "Run N.IX" for "Month A Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	2688	1432780	533.03			
Backgr	2233/83.1	448469/31.3	**200.84**	6/100.0	14/100.0	32/100.0
Cl. 1	455/16.9	984311/68.7	**2163.32**	**6/100.0**	**14/100.0**	**14/43.8**
Cl. All		**984311/68.7**	2163.32			

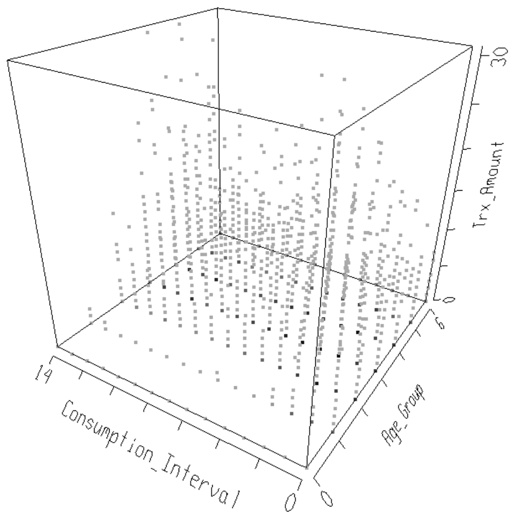
Table IV.34: Calculation Report of "Run N.IX" for "Month B Data"

	Member Bin Count/Per.	**Member Data Pt Count/Percent**	**Density**	**Age Group Range/Per.**	**Consumption Interval Range/Per.**	**Transaction Amount Range/Per.**
Overall	2688	1329481	494.60			
Backgr	2261/84.1	433164/32.6	**191.58**	6/100.0	14/100.0	32/100.0
Cl. 1	427/15.9	896317/67.4	**2099.10**	**6/100.0**	**14/100.0**	**13/40.6**
Cl. All		**896317/67.4**	2099.10			

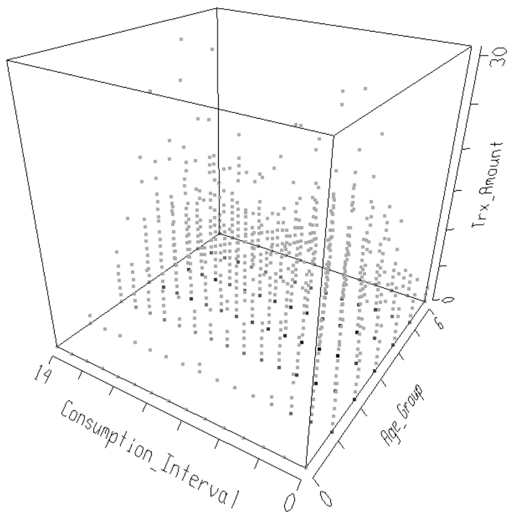
In our reference run, we used bin offsetting for pre-quantized dimensions as a requirement of our implementation in order to use same quantization. Disabling bin offsetting results in usage of an inappropriate quantization and production of an incorrect output.

➢ ***No Logarithmic Scale Run (N.X)***

In this run we do not use logarithmic scale on dimension "Transaction Amount". Since scaling selection is inappropriate no clusters are formed and no report is produced. In any case, we may examine the output of the binning stage, shown in Figure IV.23, in order to judge the scale selection.



(a) Month A Data



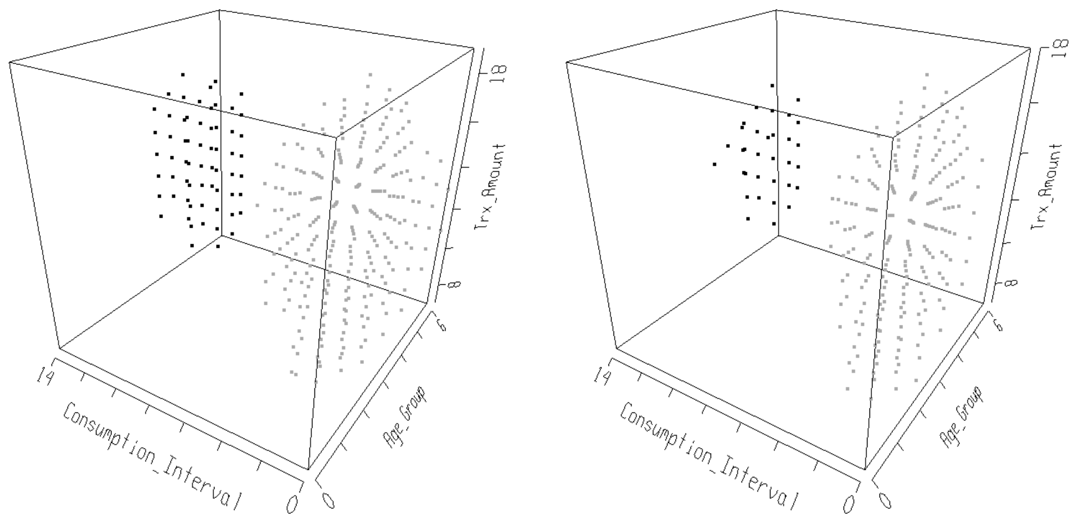
(b) Month B Data

Figure IV.23: Binning Output of "Run N.IX"

➢ ***No Threshold Distribution Run (N.XI)***

In this run we did not use variable threshold on dimension "Consumption Interval". This results in cluster extracted for the higher values of "Consumption

Interval" becoming smaller with respect to the reference run. Visual outputs and calculation reports are shown in Figure IV.24 and Tables IV.35, IV.36 respectively.



(a) Month A Data (b) Month B Data

Figure IV.24: Segmentation Output of "Run N.XI"

Table IV.35: Calculation Report of "Run N.XI" for "Month A Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1432780	426.42			
Backgr	3028/90.1	740631/51.7	**244.59**	7/100.0	15/100.0	32/100.0
Cl. 1	269/8.0	571297/39.9	**2123.78**	**7/100.0**	**9/60.0**	**13/40.6**
Cl. 2	63/1.9	120852/8.4	**1918.29**	**3/42.9**	**5/33.3**	**9/28.1**
Cl. All		**692149/48.3**	2084.79			

Table IV.36: Calculation Report of "Run N.XI" for "Month B Data"

	Member Bin Count/Per.	***Member Data Pt Count/Percent***	***Density***	***Age Group Range/Per.***	***Consumption Interval Range/Per.***	***Transaction Amount Range/Per.***
Overall	3360	1329481	395.68			
Backgr	3102/92.3	779795/58.7	**251.38**	7/100.0	15/100.0	32/100.0
Cl. 1	225/6.7	486395/36.6	**2161.76**	**7/100.0**	**8/53.3**	**12/37.5**
Cl. 2	33/1.0	63291/4.8	**1917.91**	**7/100.0**	**5/33.3**	**7/21.9**
Cl. All		**549686/41.3**	2130.57	3/42.9		

We used variable threshold in our reference run in order to highlight data points that belong to customers who have high consumption habits. After disabling variable threshold, this highlighting effect disappeared and one of the clusters, that stays on the higher end of the dimension “Consumption Interval”, became smaller. In any case, this run showed us that two valid clusters are present in the data and none of them are formed as a consequence of a parameter manipulation.

IV.4. Conclusion

In this study we developed an algorithm for clustering high dimensional data using grayscale mathematical morphology. We apply image processing operations on multidimensional signals derived from large data warehouses using the analogy between data warehouses and images. Morphology techniques have been useful for image processing applications to smoothen the image by filling out holes and cutting protrusions. A similar effect has been observed with multidimensional data. In particular, the clusters in data are more uniform and noise in the background regions is cleaned.

We first quantized the data to be processed. At this point we also handled two basic problems to be encountered during quantization process: dislocated bins and nonlinearly distributed data. We utilized two specific solutions to resolve these quantization issues respectively: bin offsetting and logarithmic scale binning.

Our purpose for applying grayscale mathematical morphology on a noisy multidimensional signal was to emphasize the signal and clean out the noise. We saw that we may achieve this goal with the help of selecting appropriate structuring element. Applying a threshold on this enhanced signal produced effective results by revealing dense regions in the data set. The quantization approach is the grid-based side of our algorithm and searching for dense zones in a data warehouse is the density-based side.

The proposed algorithm in this study approaches clustering problem somehow similar to the popular algorithms in the literature. For instance, WaveCluster also applies an image processing operation on quantized dataset and DENCLUE applies a threshold to the enhanced signal produced by the algorithm. The main contribution of our algorithm is the utilization of grayscale mathematical morphology on a multidimensional signal.

In the scope of the work we also implemented proposed algorithm using Java to observe the results of our approach. Strengths of our algorithm are, ability to extract arbitrary shaped non convex clusters, robustness against noise in data, independence from the order of data input, handling large datasets, support for high dimensional data and producing interpretable results.

Although the algorithm performs well on low dimensional data, its computational complexity increases exponentially with data dimensionality. The complexity problem is attributed to two phenomena: exponential increase of pixels to be processed in the multidimensional grid space and the increase of dimensions of the structuring element used in morphological operations. One drawback of our algorithm is the requirement of a number of parameters to be specified. These parameters cannot be set independently. Some guidelines are given in this work, but still they may need to be determined experimentally.

For future study, multiresolution (hierarchical) implementation and subspace processing of morphology could be used for computational efficiency and to cope with very high data dimensionality. Morphological operations are known to be suitable for hard-wired implementations. The prospects of this can be utilized for real-time clustering applications. In addition to the needed improvements to resolve time complexity issues, another future work area could be automatic selection of input parameters by analyzing the data.

As a conclusion, the proposed algorithm is observed to be comparable in many aspects to common clustering algorithms in the literature. Improvements on time complexity and parameter selection problems will enable the algorithm to be applied to a large number of application domains.

REFERENCES

[1]. Matheron, G., *Random Sets and Integral Geometry.* John Wiley and Sons, New York, 1975

[2]. Serra, J., *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982

[3]. Serra, J., *Image Analysis and Mathematical Morphology II.* Academic Press, London, 1988

[4]. Giardina, C. R., and Dougherty, E. R., *Morphological Methods in Image and Signal Processing.* Prentice-Hall, Englewood Cliffs, 1988.

[5]. Maragos, P., "Tutorial on advances in morphological image processing and analysis"*, Visual Communications and Image Processing*, SPIE Proceedings Vol. 707*,* 1986

[6]. Haralick, R. M., Sternberg, S. R. and Zhuang, X., "Image Analysis Using Mathematical Morphology", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4): 532-550, 1987.

[7]. Heijmans, H. J. A. M., *Morphological Image Operators,* Academic Press, Boston, 1994

[8]. Heijmans, H. J. A. M., "Mathematical morphology: basic principles." In *Proceedings of Summer School on "Morphological Image and Signal Processing"*. Zakopane, Poland, 1995 (http://citeseer.ist.psu.edu/heijmans95morphological.html)

[9]. Gonzales, C. G. and Woods, R. E., *Digital Image Processing*, Addisson-Wesley, 1993

[10]. Berkhin, P., "Survey of clustering data mining techniques", *Technical report*, Accrue Software, San Jose, California, 2002 (http://citeseer.ist.psu.edu/berkhin02survey.html)

[11]. Han, J. and Kamber, M., *Data Mining*. Morgan Kaufmann Publishers, 2001

[12]. Kolatch, E., "ClusteringAlgorithms for Spatial Databases: A Survey" , Dept. of ComputerScience, University of Maryland, College Park, 2001 (http://citeseer.ist.psu.edu/kolatch01clustering.html)

[13]. Hinneburg, A. and Keim, D. A., "An efficient approach to clustering in large

multimedia databases with noise", In *Knowledge Discovery and Data Mining*, pages 58-65, 1998 (http://citeseer.ist.psu.edu/hinneburg98efficient.html)

[14]. Sheikholeslami, C., Chatterjee, S., Zhang, A., "WaveCluster: A Multiresolution Clustering Approach for Very Large Spatial Database". *Proceedings of 24 th VLDB Conference*, New York, USA, 1998 (http://citeseer.ist.psu.edu/sheikholeslami98wavecluster.html)

[15]. Agrawal, R., Gehrke, J., Gunopolos, D. and Raghavan, P., "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Seattle, Washington, 1998 (http://citeseer.ist.psu.edu/agrawal98automatic.html)

[16]. Xu, Rui and Wunsch, Donald, "Survey of Clustering Algorithms", In *IEEE Transactions On Neural Networks*, Vol 16, No 3, May 2005

[17]. Kotsiantis, S.B and Pintelas P.E., "Recent Advances in Clustering: A Brief Survey", In *WSEAS Transactions on Information Science and Applications*, Vol 1, No 1, 2004

[18]. Postaire, J.G. and Zhang, R.D. and Lecocq-Botte, C., "Cluster Analysis by Binary Morphology", In *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol 15, No 2, February 1993

[19]. Luo, H. and Kong, F. and Zhang, K. and He, L., "A Clustering Algorithm Based on Mathematical Morphology". *Proceedings of the 6th World Congress on Intelligent Control and Automation*, Dalian, China, June 2006

[20]. MATLAB 6.5: http://www.mathworks.com/, December 2006

[21]. VisAD 2.0: http://www.ssec.wisc.edu/~billh/visad.html, December 2006

[22]. Eclipse 3.2: http://www.eclipse.org/, December 2006

[23]. Java3D API: https://java3d.dev.java.net/, December 2006

[24]. MIPAV 3.0.1: http://mipav.cit.nih.gov/, December 2006