NEW TECHNIQUES FOR CLUSTERING, SELECTION OF NUMBER OF HISTOGRAM BINS, AND ESTIMATION OF QUANTILES OF PAIRWISE DISTANCES

By

SAI VENU GOPAL LOLLA

Bachelor of Technology in Mechanical Engineering
Jawaharlal Nehru Technological University
Hyderabad, AP, INDIA
May, 2002

Master of Science in Mechanical Engineering
Oklahoma State University
Stillwater, OK, USA
May, 2005

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 2011

NEW TECHNIQUES FOR CLUSTERING, SELECTION OF NUMBER OF HISTOGRAM BINS, AND ESTIMATION OF QUANTILES OF PAIRWISE DISTANCES

Dissertation Approved:

Dr.Lawrence L. Hoberock

Dissertation Advisor

Dr.Prabhakar R. Pagilla

Dr.Jay C. Hanan

Dr.Guoliang Fan

Dr.Douglas R. Heisterkamp

Dr.Sheryl A. Tucker

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

CHAPTER 1

Introduction

In an earlier report [1], the development of a framework for a knowledge–based inference–driven vision system was proposed. The proposed framework was meant to: facilitate an approach towards integration of various vision techniques; serve as a means of communication between higher and lower–level algorithms to support feedback; and help towards the development of vision systems that can be ported to a variety of fields relatively easily.

The proposal for such a framework was motivated by: (1) The abundance and diversity of applications for vision systems; (2) The efficacy of the human visual system; (3) The fact that vision systems from almost all fields share several common needs [2]; and (4) The existence of generic systems/frameworks in areas such as microprocessors, high–level programming languages, and CAD Design Software where basic primitive operations are combined several times in several ways defining a hierarchical structure resulting in tools/devices whose efficacy increases rapidly.

The scope of vision systems developed from the framework was to be governed by: (1) Similarity in the type of images acquired - for low level image processing techniques; (2) Similarity or equivalence in the methods used to acquire the images - for any calibration procedures involved; and (3) Similarity in the content/structure expected in the image - for middle level and higher level processing. Four major functional requirements were identified: (1) A set of image analysis techniques that can identify graphic primitives such as points, lines, curves, ellipses, vertices etc.; (2) A notation/language that can describe the various properties (or metrics) of the graphic

primitives (or image segments) detected in the image. (3) A notation/language that permits representation of rules, which describe objects in terms of graphic primitives. (4) An Inference Engine that can produce logically correct conclusions to questions based on rules/knowledge provided to the system and the descriptions of the graphic primitives generated by the image analysis techniques.

It was noted that the proposed framework would operate by feeding the output of one layer of processing to another, and that any errors in processing would have a cascading effect. It was thus reasoned that the first layer of processing – that translates a given image into its analog equivalents of primitives found in the image – would be very crucial to the success of the framework. It was also noted that the first layer of processing is expected to be relatively domain–independent and context–free.

The process of converting a given image into its analog equivalents of primitives is referred to as "Segmentation", and marks the boundary at which pixel–level processing ends and object–level processing begins. Pixels are grouped into meaningful entities/segments. Segments might be considered as compact representations for certain data based on some metric of similarity. Meaningfulness of the segments is well–defined when the grouping is based on specific assumptions/rules, usually when segmentation is guided by domain-specific knowledge. In the absence of any explicit information relating to any particular domain (domain–independence), it was discovered that the human vision system exhibited certain preferences in grouping image points. These preferences are directed by what are known as Gestalt Laws or Gestalt Principles [3], [4]. Hence, it was also noted that the first layer of processing would be expected to display performance characteristics similar to "Gestalt Laws" [3] to mimic human visual processing.

Based on these observations, efforts were initiated toward emulating the "Proximity Law" of the "Gestalt Laws". The "Proximity Law" suggests grouping of all data points that are "nearby". This relates closely to the problem of "Clustering". Quickly

the efforts resulted in a search for a clustering technique that worked: (1) without the need for any input parameters that would govern the number of clusters; and (2) without any preference for a given segment/cluster shape. The time and resources required for this study indicated that the challenges involved in the development of the proposed framework in [1] were grossly underestimated.

A circuitous search coursing through various clustering techniques – most of which required either the number of clusters to be detected or some parameter that would govern the number of clusters detected – led to the finding of work done by Bicego et al. [5]. While their method did not need any pre–set parameters and had no specific preference for any cluster shape/structure, it was evident from the details that certain experimentally set values used in their method would result in degraded clustering performance. We developed a much improved approach, given in Chapter 2, over that given by Bicego et al. The improved clustering method employs the concept of scale to attain better clustering performance.

The development of the improved clustering method in Chapter 2 lead to the need for an appropriate metric to detect the right "scale" at which to detect clusters. In this process, two metrics we explored in great detail, Entropy [6] and Q_n [7] presented unexpected difficulties, computational and analytical. For the problem addressed in Chapter 2: (1) computation of entropy required computation of probability mass function values, which in turn required a sound technique to perform "data binning" (construct a histogram); and (2) computation of Q_n required an efficient technique for computation of quantiles of pairwise distances. While methods existed for both, opportunities for improvement were quite evident. Accordingly, following a review of other methods available for computing/estimating quantiles of pairwise distances, we developed in Chapter 3, a novel method to estimate quantiles of pairwise distances and present performance comparisons of this new method with existing methods. Following this, in Chapter 4 we present a brief review of existing methods available

for constructing histograms and development of a new and more accurate method for constructing histograms, together with performance comparisons with existing methods.

Computational difficulties encountered while attempting to apply the clustering method described in Chapter 2 to segment some images of fly-ash particles led to the development of an alternate implementation of the clustering method. The alternate implementation of the clustering method is described in Chapter 5. Performance comparisons between the two implementations of the clustering method are presented. A drawback with the clustering method that was discovered is also described in Chapter 5.

In Chapter 6, some conclusions are presented, and directions for future work are suggested.

CHAPTER 2

Improved Unsupervised Clustering Over Watershed–Based Clustering

2.1 Introduction

Clustering or Cluster Analysis refers to the process of classifying data into meaningful homogeneous groups, and is a type of unsupervised classification. Discriminant Analysis, which is a type of supervised classification, refers to a related problem where known groupings of observations govern the classification of other data, followed by evaluation of structure of the entire data. Clustering is a particularly difficult problem since the interpretation of the resulting clusters and their number depends upon domain–specific knowledge, practical experience, possible assumptions involved and human intuition [8]. It is known that no single currently existing clustering method is capable of handling all sorts of cluster structures due to variations of cluster shape, size, and density. An effective clustering method is often a well–balanced combination of data pre–processing methods, distance metrics, criterion functions, searching and sorting algorithms, and strategies to handle outliers and missing values [9]. Several articles in the literature provide an introduction to the vast amount of work done in this field [9–11].

Clustering algorithms are categorized into partitioning, hierarchical, density–based, grid–based, and model–based methods. Each method has its own set of advantages and disadvantages. Some work has also been devoted to combining several clustering methods into one algorithm [9]. There are two central issues that almost all clustering algorithms should address [12]:

- Into how many clusters should the data be classified?
- How should data be classified, once the number of clusters have been decided?

The former is considered to be a more difficult problem than the latter. Several clustering methods have been proposed that try to determine the "natural" / "optimum" number of clusters [8, 13–17]. To work properly, most algorithms require a parameter to be provided by the user – either the number of clusters present in the dataset, or a parameter that in turn governs the number of clusters that can be detected. Selecting the "right" value for such parameters might be trivial in some cases, but it can often become impractical and infeasible due to the size and dimensionality of the data or other constraints.

The clustering algorithm proposed in this chapter, and its predecessor [5], share some features with grid–based methods and density–based methods. A major advantage of both these methods is that they do not require the user to provide parameters.

The remaining sections of the chapter are organized as follows. In Section 2.2, the datasets used for evaluation of the proposed method are introduced. In Section 2.3, the currently existing Watershed–based method [5] is introduced and its major drawback is discussed. In Section 2.4, the proposed method is introduced. Section 2.5 presents the results of evaluation of the proposed method over the datasets. In Section 2.6, conclusions and scope for future work are presented. The work presented in this chapter was published in the Proceedings of the Ninth International Conference on Machine Learning and Applications (ICLMA'10) [18].

2.2 Datasets

A total of 12 synthetic datasets were used for the testing of the method proposed in this chapter. While 7 of the datasets (S1–S4, A1–A3) were imported [19], the other 5 datasets (V1–V3, Z1–Z2) were created by the authors to test particular aspects

of the proposed algorithm. Datasets S1–S4 have 15 clusters and 5000 points, each with various degrees of overlap. Datasets A1–A3 have varying number of data points and clusters [19]. Datasets V1–V3 have various degrees of overlap, and perhaps even multiple interpretations. Datasets Z1 and Z2 have interpretations that are scale–dependent. Visualizations of the datasets are provided in Figures 2.1a through 2.1l. Table 2.1 displays the "correct" number of clusters for each dataset.

While the aforementioned 12 datasets might not constitute a large enough test suite to measure the general performance of a clustering algorithm, it should be noted that the development of the clustering technique being presented here is motivated by an attempt to deliver a parameter-less mechanism to emulate the "Proximity" gestalt law. While there are several datasets available [20], many of those datasets have domain-dependent interpretations, and sometimes, the "correct" clustering configuration for these datasets does not agree with the configurations suggested by the human visual system. Thus for this work, it is preferable that the datasets used for testing purposes be relatively domain-independent and have "correct" clustering configurations that are compatible with those suggested by the human visual system.

Clusters in datasets S1–S4, A1–A3 could be termed as well–defined, since most human interpreters would draw the same conclusion about the number of clusters (and the cluster centers) without much doubt or difficulty. Since these datasets have "correct" clustering configurations that are compatible with those suggested by the human visual system, and were used by Zhao et.al. [17] to test a mechanism to detect number of clusters present in a dataset, they were included in the test suite. While these datasets have different number of datapoints, clusters, degrees of overlap etc., they do not present any ambiguity to the human visual system. Datasets V1–V3, Z1, and Z2 were created specifically to present some scale-related ambiguity to the human visual system while remaining domain-independent.

While Table–2.1 indicates 12 clusters for dataset V1, most human interpreters

Table 2.1: "Correct" number of clusters for each dataset

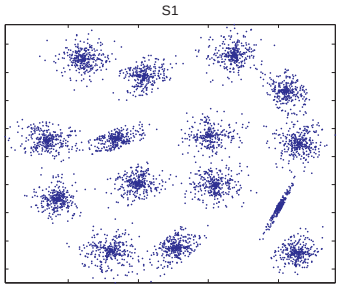
Dataset	Cluster Count	Dataset	Cluster Count
S1	15	A3	50
S2	15	V1	12
S3	15	V2	9
S4	15	V3	13
A1	20	Z1	1
A2	35	Z2	49

would perhaps perceive 11, 8, or 6 clusters, based upon apparently equally valid, but different interpretations. However, the dataset was created using 12 Gaussian distributions, hence the number 12 for dataset V1 in Table–2.1. Datasets V2 and V3 are also likely to draw differing human interpretations, but a majority would agree with the corresponding numbers in Table 2.1. Dataset Z1 has data points uniformly distributed over the feature space, such that most human interpreters would perceive 0 or 1 cluster. Dataset Z2 can be perceived as having 49, 7, or 1 clusters based upon the scale at which the interpreter chooses to group the data points.

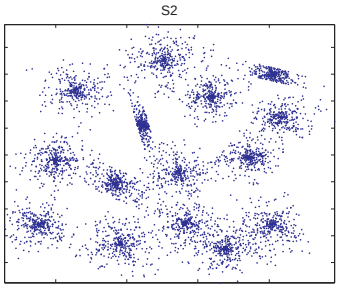
2.3 The Existing Method

The Watershed algorithm was developed from the fields of Image Processing and Mathematical Morphology, and is a region–based image segmentation method. The Watershed algorithm borrows its intuitive idea from geography – when a landscape or a topographic relief is flooded by water, water collects in *catchment regions*, and the catchment regions are divided by *watershed lines* [21].

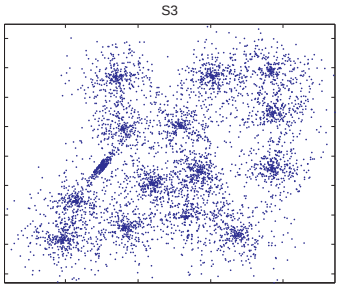
The existing method [5] proposes that a grid be constructed over the feature space and then a density function be defined over the grid. The density of each cell of the grid is treated as a height. Thus the density function takes on an interpretation of a landscape (3–D landscape for 2–D dataset). This landscape is then inverted and subjected to the Watershed algorithm. As a result of the Watershed algorithm,



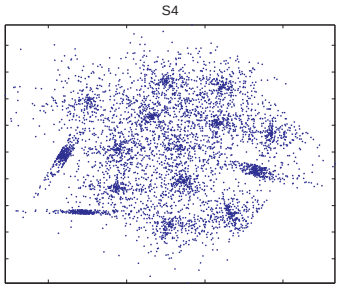
(a) Dataset - S1



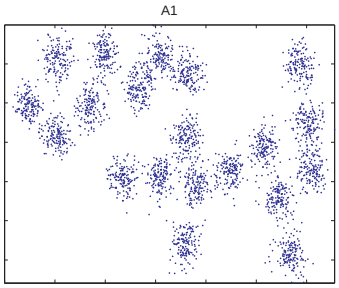
(b) Dataset - S2



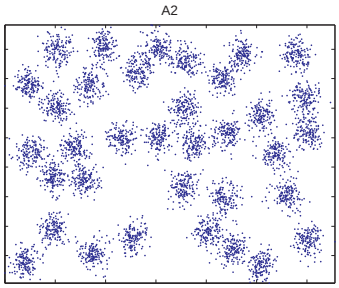
(c) Dataset - S3



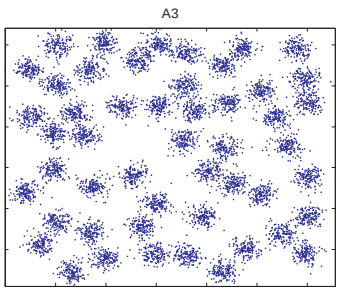
(d) Dataset - S4



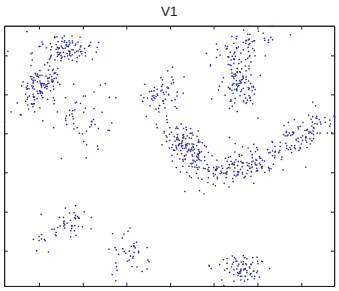
(e) Dataset - A1



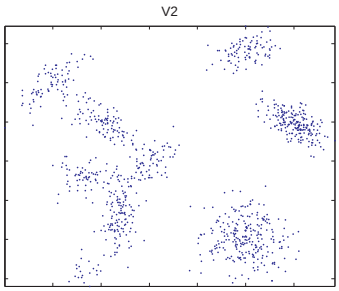
(f) Dataset - A2



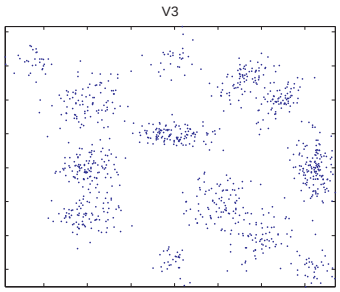
(g) Dataset - A3



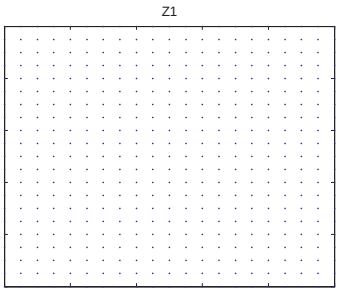
(h) Dataset - V1



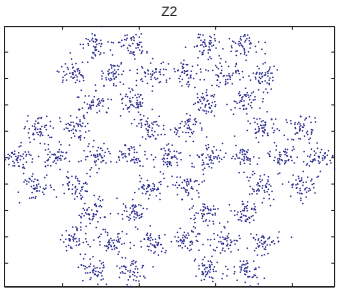
(i) Dataset - V2



(j) Dataset - V3



(k) Dataset - Z1



(l) Dataset - Z2

Figure 2.1: Datasets used for testing

the minima in the inverted landscape, corresponding to the high density regions in the grid, are detected. Thus clusters are implicitly defined as regions of high density in the feature space, and are marked by corresponding catchment regions in the inverted landscape. The number of catchment regions found is taken to be the number of clusters present, and the catchment region itself represents the region spanned by the corresponding cluster. The formal representation is reproduced as follows. Let $\mathbf{Y} = \mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}$ represent the dataset where each observation is $\mathbf{y_i} = y_{i,1}, y_{i,2}, \ldots, y_{i,D}$. A grid with cells as D–dimensioned hypercubes of fixed size l_R is defined over the feature space with an origin O:

$$O = [\min_n y_{n,1}, \min_n y_{n,2}, ..., \min_n y_{n,D}] \tag{2.1}$$

A cell in the position $\mathbf{i} = (i_1, i_2, ..., i_D)$ is denoted as $R(\mathbf{i}) = R(i_1, i_2, ..., i_D)$. Once l_R is chosen (a choice which is critical and discussed later), a grid with $\left(\frac{k}{l_R}\right)^D$ cells spans the feature space, where k represents the maximum dimension width of the feature space, and is defined as follows:

$$k = \max_d \left(\max_n y_{n,d} - \min_n y_{n,d} \right) \tag{2.2}$$

The height function for the grid for the Watershed algorithm to operate upon, is then defined: the value of the function $I(R(\mathbf{i}))$ in a cell is the number of points that belong to that cell. The function $I(R(\mathbf{i}))$ is defined as:

$$I(R(\mathbf{i})) = \sum_{\mathbf{y_n} \in \mathbf{Y}} \chi_{R(\mathbf{i})}(\mathbf{y_n}) \tag{2.3}$$

where χ is the characteristic function of the set $R(\mathbf{i})$, and is defined as:

$$\chi_{R(\mathbf{i})}(\mathbf{y_n}) = \begin{cases} 1 & \text{if } \mathbf{y_n} \in R(\mathbf{i}); \\ 0 & \text{otherwise;} \end{cases} \tag{2.4}$$

The function $I(R(\mathbf{i}))$ (landscape) is an approximation of the density properties of the feature space, with an underlying assumption that similar points (points that are to be grouped into the same cluster) are near in the feature space (Assumption–1). The function values are inverted so that the local maxima mark the minima and vice–versa. The Watershed algorithm then marks the catchment regions present in the inverted landscape and thus the number of clusters is obtained.

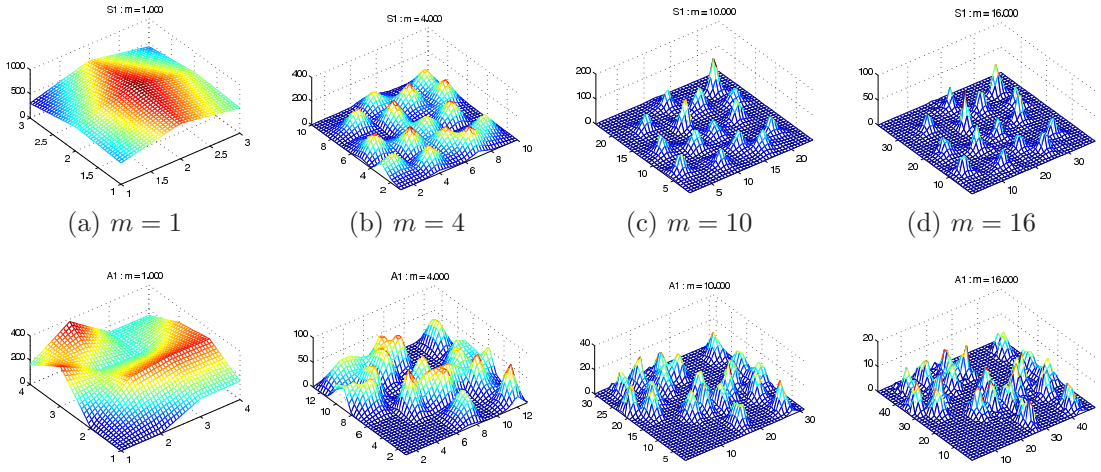
Choosing the right value for l_R is critical for the aforementioned method to work meaningfully. Choosing too large a value results in coarse–segmentation, and too small a value will result in over–segmentation. Either case will result in a clustering result that will not have much meaning. Consider the following: If $l_R = k$ then there will be only one cell, and only one cluster; If l_R is so small such that each cell contains only one point, then there could be as many clusters as the number of data points. Neither result is likely useful, but these extremes mark the limits within which values for l_R might lie that can result in meaningful clustering results.

Bicego et al. [5] suggest that the value of l_R should be estimated for the data that is to be clustered. It is stated that a "good" value for l_R can be obtained by using the median of pairwise distances between all points. All distances $d(\mathbf{y_i}, \mathbf{y_j})[\forall i, j \in 1...n]$ are computed and then the median of all those distances is computed. From the data, the value for l_R is calculated by:

$$l_R = \frac{median(d(\mathbf{y_i}, \mathbf{y_j}))}{m} \tag{2.5}$$

In Eq(2.5), m is a constant and is experimentally fixed at 4 for all the datasets

evaluated by Bicego et al. [5]. It is suspected that this experimentally fixed value of $m = 4$ may not work well for all datasets. This is demonstrated with some of the datasets introduced in Section 2.2. Figures 2.2a – 2.2d display the landscape for dataset S1 using $m = 1, m = 4, m = 10$ & $m = 16$ respectively. It can be seen from Figure 2.2a that $m = 1$ produces too coarse a grid for dataset S1, and hence information about the number of clusters is lost during the construction of the landscape. The value $m = 4$ produces a landscape, as seen from Figure 2.2b, which does not clearly show 15 peaks corresponding to the clusters. Figures 2.2c ($m = 10$) and 2.2d ($m = 16$) produce landscapes that clearly show peaks corresponding to the clusters. These landscapes are much more likely to report the correct cluster count when subject to the Watershed algorithm after inversion. Figures 2.2e – 2.2h display the landscape for dataset A1 using $m = 1, m = 4, m = 10$ & $m = 16$ respectively. Figure 2.2e reveals that $m = 1$ produces too coarse a grid for dataset A1, and hence information about the number of clusters is lost during the construction of the landscape. Figure 2.2f shows that $m = 4$ produces a landscape that does not show 20 peaks corresponding to the clusters. Figure 2.2g shows that $m = 10$ produces a landscape that displays 20 peaks corresponding to the clusters. Figure 2.2h shows that $m = 16$ produces a landscape displaying many more than 20 peaks. Landscapes were constructed for the remaining datasets and the results were similar, confirming the earlier suspicion that $m = 4$ does not work for all datasets. From Figures 2.2e – 2.2h it can be also seen that the landscape has a rather "abrupt" and "angular" nature as opposed to a "continuous" and "smooth" one, even for the cases where the number of peaks may be correctly perceived. The Watershed algorithm may be able to perform better, by detecting the peaks of the landscape more reliably, for smoother versions of the landscape. The following is surmised based on observations made from landscapes displayed in Figures 2.2e – 2.2h and landscapes generated for the other datasets:



(a) $m = 1$ (b) $m = 4$ (c) $m = 10$ (d) $m = 16$

(e) $m = 1$ (f) $m = 4$ (g) $m = 10$ (h) $m = 16$

Figure 2.2: Density Landscapes for S1 & A1 using the existing method

- The method should work better on a landscape in which the peaks are more clearly defined than on a landscape in which they are not clearly defined.

- If there are two landscapes for a given dataset, both displaying clearly perceivable peaks, the landscape that has a greater degree of smoothness and continuity should produce better results.

Since m influences the size of the cell on the grid and thus the construction of the landscape, fixing the value of m as a constant for all datasets is a major drawback of the existing method. While the aforementioned method presents a good attempt at rendering the whole process unsupervised, effort invested into finding methods that will choose cell sizes that are more appropriate might prove fruitful. More appropriate selection of cell sizes will result in construction of "better" grids, which in turn will result in construction of "better" landscapes, which will allow detection of clusters in a more accurate and reliable manner. Bicego et al. [5] duly acknowledge that future investigations should target construction of the grid to improve the existing method.

2.4 A New Proposed Method

In the method proposed here, the major emphasis is on selecting the most appropriate cell size, which will be shown to produce the most significant contribution. The method proposed here closely follows the central theme of the method described in Section 2.3, and can be viewed as an improved version.

It is known that data can display different structures at different scales [22]. The term "scale" as applied to a given dataset can be loosely interpreted as the size of the smallest spatial structure that can be perceived from the dataset. Any structure smaller than a given "scale" will have been suppressed in the rendering of the data at that scale. Since clustering can be interpreted as a method for detecting structure present in the data, different cluster configurations may be detected at different scales. For example: at very large scales all the data will be treated as one cluster, and at very small scales each data point can be treated as a cluster. Meaningful structures, and thus meaningful clusters will be perceived when operating at the "right" scale(s) for the data. Thus, clustering methods should consider scale to accurately detect the number of clusters while operating on a given dataset [15].

In the existing method described in Section 2.3 and the method proposed here, scale relates to the size of the cell, based upon which the grid is constructed. From here on, use of the term "scale" will loosely refer to the cell size used to construct the grid.

In order to handle scale, some sort of smoothing operations might need to be performed. It is also known that smoothing operations need to abide by certain scale–space axioms. The Gaussian kernel satisfies these axioms and hence is the kernel of choice for the work that follows. Use of other kernels for smoothing is possible [22].

2.4.1 Construction of Matrix Form for a dataset

To implement the smoothing operation using the Gaussian kernel, both the dataset as well as the Gaussian kernel should be transformed to matrix forms so that the convolution operation may be performed easily. To construct a matrix form for a given dataset, a grid is constructed over the feature space, in a fashion similar to that in Section 2.3. Let $\mathbf{Y} = \mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}$ represent the dataset where each observation is $\mathbf{y_i} = y_{i,1}, y_{i,2}, \ldots, y_{i,D}$. Then define d_{cw} as the cell size, given by:

$$d_{cw} = \frac{\min\limits_{\forall i,j \in 1 \ldots n} (d(\mathbf{y_i}, \mathbf{y_j}))}{(2 + \epsilon)} \tag{2.6}$$

where $d(\mathbf{y_i}, \mathbf{y_j})$ represents the distance between $\mathbf{y_i}$ & $\mathbf{y_j}$, and ϵ is any positive real number. This is inspired from the Nyquist–Shannon sampling theorem [23], and $\epsilon \to 0$ marks the limiting condition specified by the theorem for no loss of information. This should result in a grid $\mathcal{M}_{w_1, w_2, \ldots, w_D}$ with a size of w_i in the i^{th} dimension. $\mathcal{M}$ is a matrix representation of the dataset without any loss of information.

2.4.2 Gaussian Kernels for smoothing

Let K_t represent the sampled version of the Gaussian kernel G_t given by:

$$G_t(x) = \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} e^{(\frac{1}{2}(-x' \Sigma^{-1} x))} \tag{2.7}$$

where Σ is the covariance matrix for the Gaussian kernel given as:

$$\Sigma = \begin{pmatrix} t_1 & 0 & . & . & 0 \\ 0 & t_2 & 0 & . & . \\ . & 0 & . & 0 & . \\ . & . & 0 & t_{D-1} & 0 \\ 0 & . & . & 0 & t_D \end{pmatrix} \tag{2.8}$$

When employing K_t for smoothing $\mathcal{M}$, any spatial structural detail whose size is less than t_i will be suppressed in the i^{th} dimension. Also when operating upon $\mathcal{M}$, t_i (standard deviation of the i^{th} dimension) has two meaningful limits:

- The smallest meaningful value t_i can assume is 1, since the matrix would not contain any information about structures whose size is less than a single cell.
- The largest meaningful value t can assume is w_i, since the matrix would not contain complete information about structures whose size is larger than the matrix itself.

Samples from a span of $3t_i$ on either side of the mean of the Gaussian are used to generate the i^{th} dimension of K_t, which lie within 3 standard deviations, 3σ. Spans larger than $3t_i$ (on either side) may be used, but use of smaller spans is not recommended.

2.4.3 Generating Landscapes for a dataset

Define *scale index (SI)*, related to t_i, by:

$$\begin{gathered} t_i = \frac{w_i}{2^{(SI-1)}} \\ 1 \leq SI \leq [1 + \log_2(\min(w_1, w_2, ..., w_D))] \end{gathered} \tag{2.9}$$

where w_i is the width of $\mathcal{M}$ in the i^{th} dimension, and t_i is the scale for the i^{th} dimension. SI is used to ensure that all dimensions in the feature space (and hence all dimensions of $\mathcal{M}$) are given equal weight.

A landscape $\mathcal{L}_{SI}$ relating to scale index SI may be generated for a dataset by convolving the matrix representation of the dataset $\mathcal{M}$ with the sampled version of the D–dimensional Gaussian kernel. Since the Gaussian kernel is separable, implementing the convolution along each dimension with an appropriate 1–dimensional Gaussian kernel is usually much faster than convolution with the sampled version of a D–dimensional Gaussian kernel.

$$\mathcal{L}_i = \begin{cases} \mathcal{M} * K_{t_i} & \text{if i=1;} \\ \mathcal{L}_{i-1} * K_{t_i} & \text{otherwise;} \end{cases} \tag{2.10}$$

where the convolution operation "$*$" is performed along the i^{th} dimension with K_{t_i} (the sampled version of the 1–dimensional Gaussian kernel with standard deviation t_i). The final result ($\mathcal{L}_{SI}$) is the matrix resulting after the convolution is performed along the D^{th} dimension ($\mathcal{L}_D$).

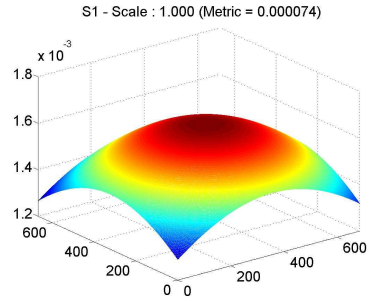
The landscape $\mathcal{L}_{SI}$ will have a structure that does not contain details smaller than t_i in the corresponding i^{th} dimension. The landscape also can be interpreted as a weighted density function. The weights are dictated by the values of the elements of the Gaussian kernel used in the process of convolution. So the landscapes will have flat regions where there are no data points and bumps/mounds where there are data points. The height of the mound at a particular point in the landscape is determined by the density of $\mathcal{M}$ at that point and the value of SI used to generate the Gaussian kernel, and in turn the landscape. Figures 2.3a – 2.3h display the landscapes for dataset S1 generated for $SI = 1, 2, \ldots, 10$ (Figure 2.3a displays the landscape for $SI = 1$, but the landscapes for $SI = 2, 3$ are very similar). Figures 2.3a – 2.3b show only one large mound, indicating that at those scales, the data points can all

be grouped into one cluster. Figures 2.3c ($SI = 5$) and 2.3d ($SI = 6$) clearly show 15 smooth mounds indicating that at these scales, the data can be grouped into 15 clusters. Figures 2.3e – 2.3h have 15 dominating peaks, but also contain several other "noisy" spikes. Construction of landscapes for other datasets using for a range of values of SI resulted in similar observations. So, based upon the two conjectures made in Section 2.3, the landscapes portrayed in Figures 2.3c & 2.3d should work best with the Watershed algorithm. Subjecting these landscapes to the Watershed algorithm loosely verifies the conjectures. Figures 2.4a – 2.4h display the results of subjecting the landscapes (Figures 2.3a – 2.3h) to the Watershed algorithm. It can be seen from Figures 2.4a – 2.4b that the corresponding landscapes resulted in too coarse a clustering (coarse–segmentation). Figures 2.4c ($SI = 5$) and 2.4d ($SI = 6$) show that the corresponding landscapes resulted in the "correct" number of clusters. Figures 2.4e – 2.4h show that the corresponding landscapes resulted in too fine a clustering (over–segmentation). The problem now lies in selecting the "optimal" value of SI to construct the "right" landscape on which to execute the Watershed algorithm.

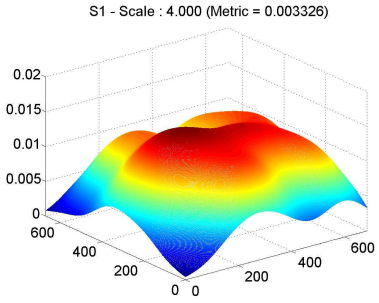
2.4.4 Selection of Optimal Scale Index

The selection of the optimal scale index SI_{opt} is critical as it governs the scale at which the landscape is generated for the given dataset. If the scale is selected appropriately, it will result in a landscape that reflects the underlying cluster structure "well". Observation of the landscapes for several datasets at several scale indices revealed the following qualitative aspects of landscapes that seemed to indicate which landscape would work most effectively with the Watershed algorithm to accurately reveal the number of clusters "present" in the dataset. In effect, these are heuristics:

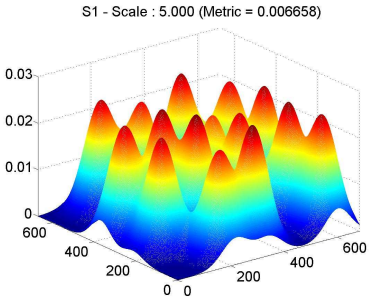
- Landscapes that resulted in the Watershed algorithm detecting too few clusters (coarse–segmentation) had mounds whose "bases" were very "wide" and the



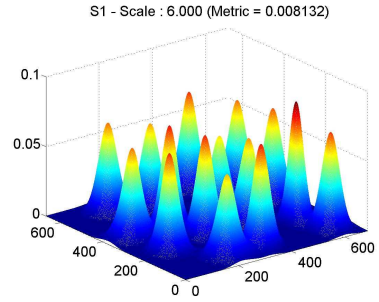
(a) $SI = 1, 2, 3$



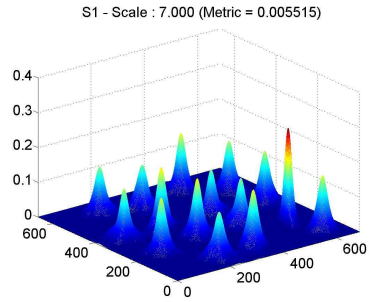
(b) $SI = 4$



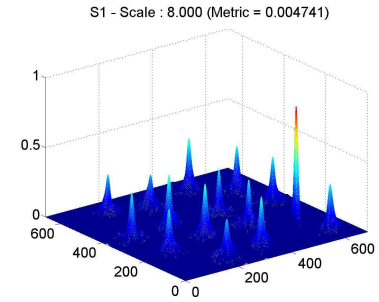
(c) $SI = 5$



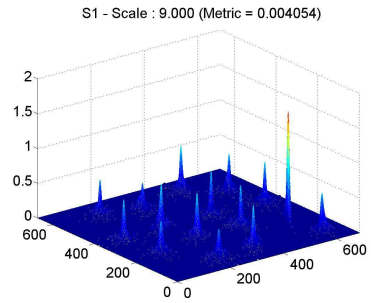
(d) $SI = 6$



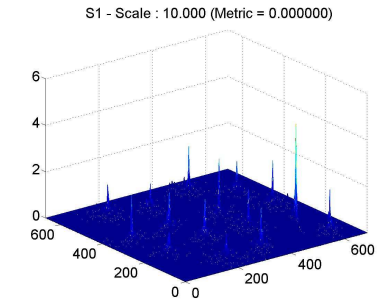
(e) $SI = 7$



(f) $SI = 8$

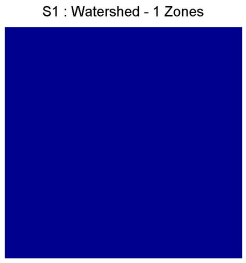


(g) $SI = 9$

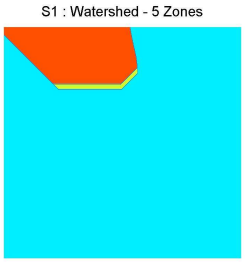


(h) $SI = 10$

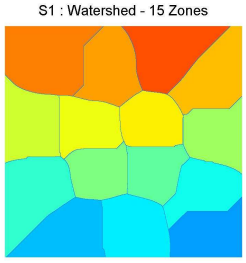
Figure 2.3: Density Landscapes for S1



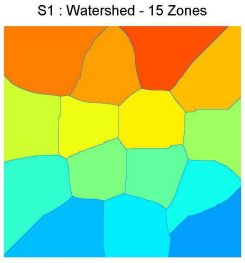
(a) $SI = 1, 2, 3$



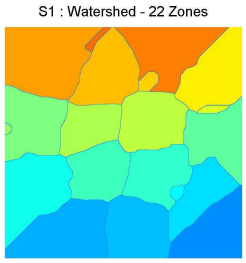
(b) $SI = 4$



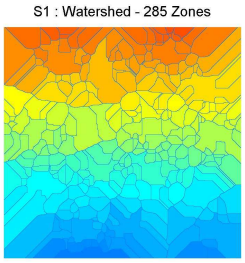
(c) $SI = 5$



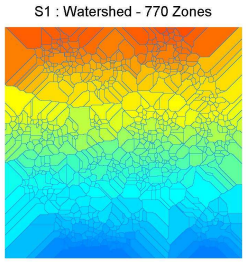
(d) $SI = 6$



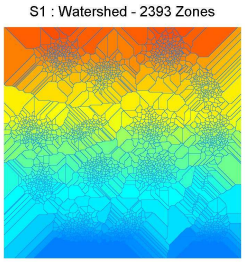
(e) $SI = 7$



(f) $SI = 8$



(g) $SI = 9$



(h) $SI = 10$

Figure 2.4: Watershed on Density Landscapes for S1

mounds did not look "tall".

- Landscapes that resulted in the Watershed algorithm detecting too many clusters (over–segmentation) had mounds whose "bases" were very "narrow" and the mounds were very "tall" resembling spikes.

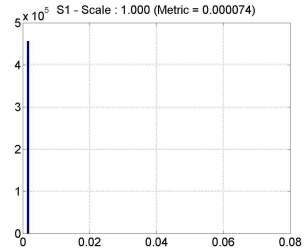
- Landscapes that resulted in the Watershed algorithm detecting the "correct" number of clusters had mounds whose "base size" was roughly equivalent to the "mound height". The mounds could be described as "balanced" or "well–rounded".

The rather excessive use of the quotations marks around several terms is to convey that these are qualitative perceptions that human interpreters would agree upon, but are difficult to quantify.

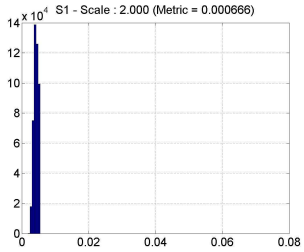
A heuristic from the observations made above is: *Select the scale index which generates the landscape in which the base–width appears to be roughly proportional to mound–height for the majority of the mounds perceived in the landscape.* A quantitative version (or an approximation) of this heuristic would assist in automating the selection of the optimal scale index (SI_{opt}), and this is given in what follows.

Observation of histograms of "height" data in the landscapes for several values of SI (Figures 2.5a – 2.5j) reveal that (1) histograms of the "height" data in the landscapes tend toward "well–spread" distributions for landscapes that result in the correct number of clusters being reported (2) histograms of the "height" data in the landscapes tend toward "spiked" distributions for landscapes that result in the incorrect number of clusters being reported. This observation hints that metrics of statistical dispersion should reach maxima for landscapes generated using the optimal scale index.

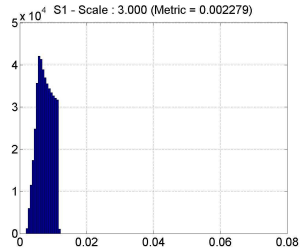
Due to its interpretation as being a "natural" measure of dispersion, Standard Deviation was the first variation index of choice. Standard Deviation and Range



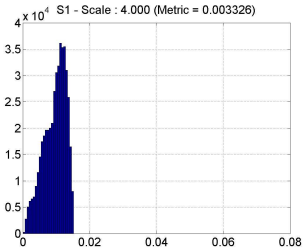
(a) $SI = 1$



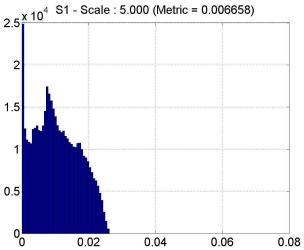
(b) $SI = 2$



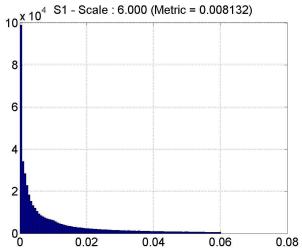
(c) $SI = 3$



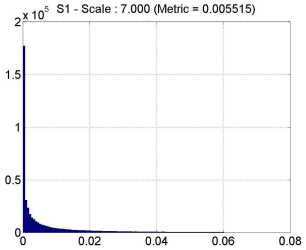
(d) $SI = 4$



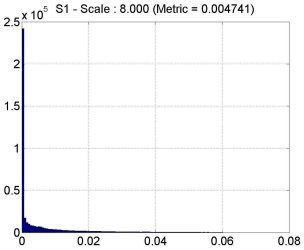
(e) $SI = 5$



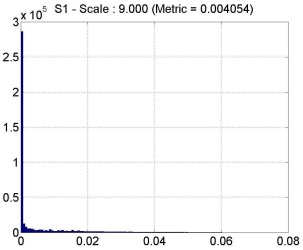
(f) $SI = 6$



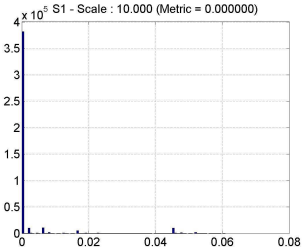
(g) $SI = 7$



(h) $SI = 8$



(i) $SI = 9$



(j) $SI = 10$

Figure 2.5: Histogram on Density Landscapes for S1

for the height data ($\mathcal{Z}$) increased monotonously with SI (Figure 2.6), and this did not produce the peak desired above. Several indices of qualitative variation due to Wilcox [24], MODVR, RANVR, AVDEV, MNDIF, VARNC, STDEV, and HREL were then computed for $\mathcal{Z}$ for various SI. These indices do not depend on the range of the data or the cardinal number of data, and they take on values in a standard range $[0, 1]$. However, the metrics did not result in any clearly defined peaks either (Figure 2.7). Inter–Quartile Range (IQR) [25] and Median Absolute Deviation (MAD) [25] computed for $\mathcal{Z}$ for various SI did reveal some clearly defined peaks (Figure 2.8). Since both IQR and MAD are robust measures of statistical dispersion, they are *outlier–resistant* [26]. The clearly defined peaks produced with IQR and MAD hinted that some sort of a filter should be applied to the height data $\mathcal{Z}$ before the variation indices are evaluated to make the variation indices less vulnerable to outliers in the height data. A technique developed by Tukey [25] is applied to the height data $\mathcal{Z}$. All values in $\mathcal{Z}$ in the range $[Z_{ll}, Z_{ul}]$ are accepted, and any values outside that range are filtered out during the evaluation of the variation index. Z_{ll} and Z_{ul} are defined as follows:

$$\begin{aligned} Z_{ll} &= Z_{25} - 1.5 \cdot IQR \\ Z_{ul} &= Z_{75} + 1.5 \cdot IQR \end{aligned} \tag{2.11}$$

where Z_{25} and Z_{75} are the 25 and 75 percentile values of $\mathcal{Z}$ respectively, and IQR is the Inter–Quartile Range of $\mathcal{Z}$.

When Standard Deviation was computed for filtered $\mathcal{Z}$ for several values of SI, well defined peaks were observed. This variation variation index VI_Z, is defined by:

$$VI_Z = SD(\,TF(\mathcal{Z})) \tag{2.12}$$

where SD represents the *Standard Deviation* function, and TF represents the *Tukey Outlier Filter* function. The scale index SI for which the variation index VI_Z is

maximized, is selected as the optimal scale index SI_{opt}.

An iterative search procedure is used to find the value of SI_{opt}. The procedure starts the search in range of values that SI can take on meaningfully and progressively narrows the search range. The procedure is terminated when further narrowing of the scale index search range does not cause a change in the size of the sampled version of the Gaussian kernels used for generating the landscapes. Figure 2.9 shows how the variation index changes with respect to the scale index, and this demonstrates the peaking we seek.

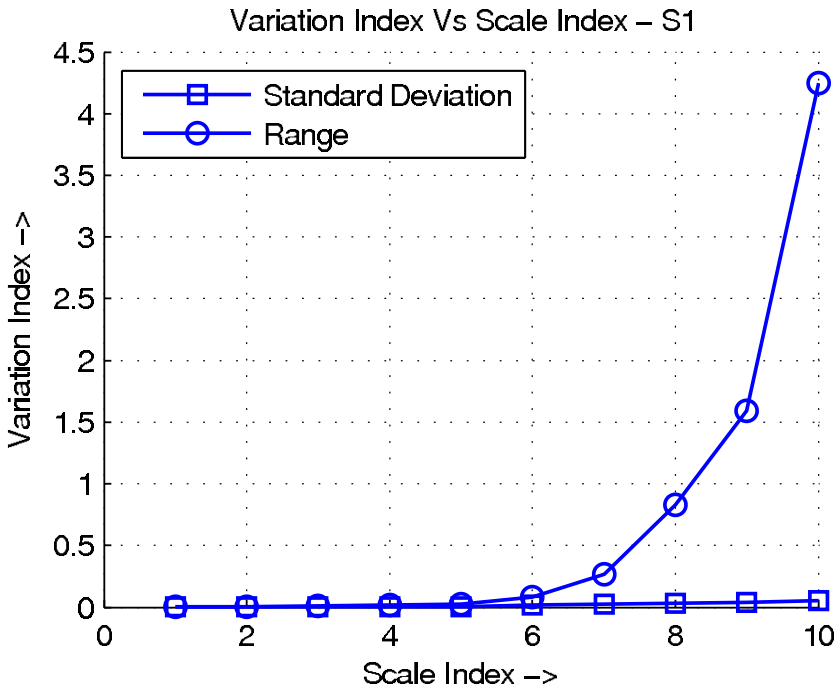


Figure 2.6: Standard Deviation & Range for various SI – Dataset S1.

2.4.5 Detection of Clusters – Count & Location

Once the optimal scale index SI_{opt} has been found, the related optimal landscape ($\mathcal{LS}_{opt}$) is generated. Figure 2.10 shows the optimal landscape for dataset S1. This landscape is then inverted as described by:

$$\mathcal{LS}_{inv} = max(\mathcal{LS}_{opt}) - \mathcal{LS}_{opt} \tag{2.13}$$

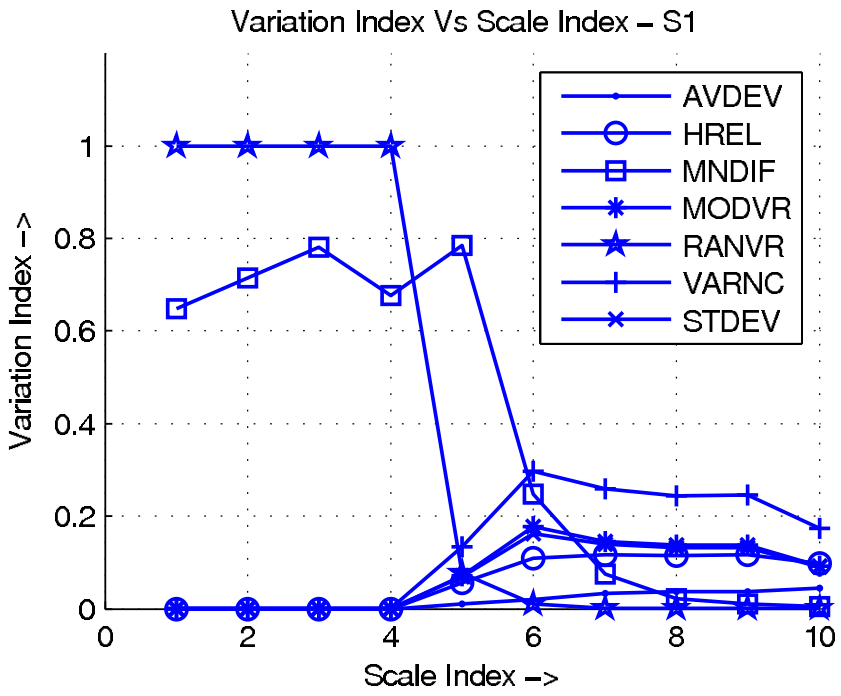


Figure 2.7: MODVR, RANVR, AVDEV, MNDIF, VARNC, STDEV, & HREL for various SI – Dataset S1.

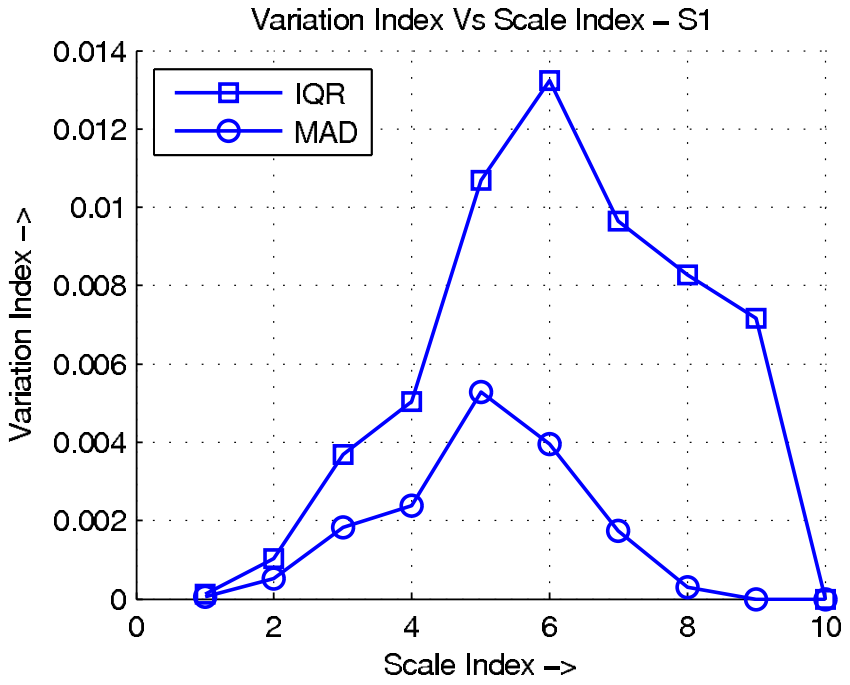


Figure 2.8: IQR & MAD for various SI – Dataset S1.

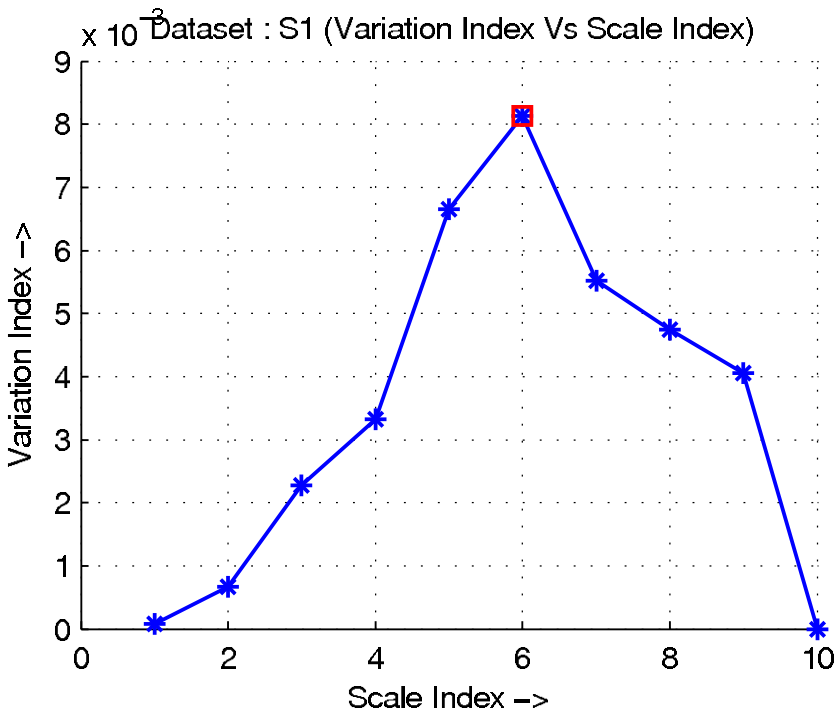


Figure 2.9: Standard Deviation after Filtering for various SI – Dataset S1.

The inverted landscape $\mathcal{LS}_{inv}$ can then be subject to the Watershed algorithm to detect the catchment regions. The number of catchment regions yields the number of clusters, and the cluster centers may be evaluated using the data points present in each catchment region. Based on Assumption–1 from Section 2.3, we assert that cluster centers must be regional maxima in $\mathcal{LS}_{opt}$. SI_{opt} indicates the minimum size of the structure that can be detected in the optimal landscape $\mathcal{LS}_{opt}$. Using this observation, if a peak on $\mathcal{LS}_{opt}$ has the maximum height in its neighborhood of size X, given by:

$$X = 2[t_1, t_2, t_3, ..., t_D] \tag{2.14}$$

centered at the peak, then that peak represents a cluster center. Eq(2.14) is constructed from a geometric interpretation for the condition to prevent overlapping clusters. The aforementioned might result in spurious clusters being identified due to isolated data points in the dataset, far removed from all the "real" clusters. The isolated point will cause a very small mound in the landscape, and if the point is

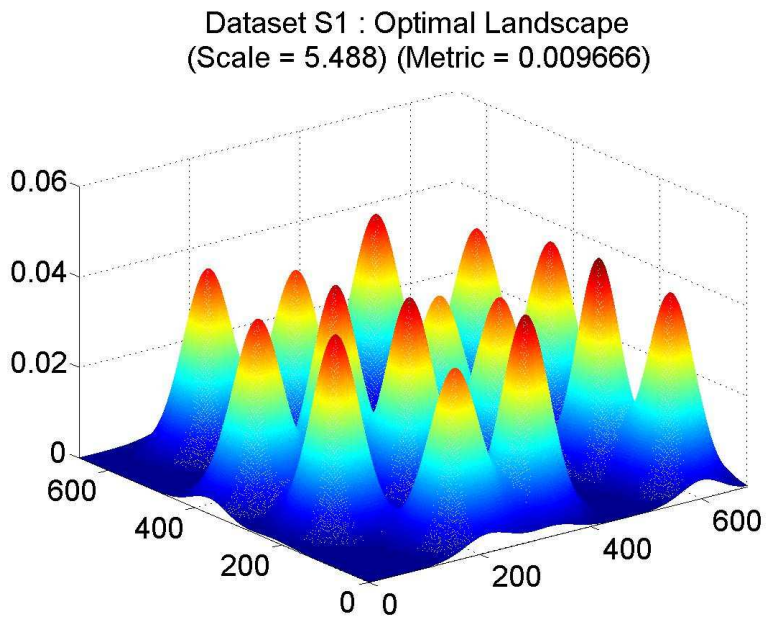


Figure 2.10: Optimal Landscape for Dataset S1.

sufficiently isolated, this mound can be a regional maximum. To avoid such spurious detections, only those maxima whose height is greater than the median of the height data $\mathcal{Z}$ are chosen for further processing. In other words, regional maxima whose height is less than the median of the height data $\mathcal{Z}$ will not be considered as clusters. We designate this as *median filtering*.

Accordingly, we can replace the Watershed algorithm with a computationally simpler regional maxima finding algorithm – we designate this as *Regional Maxima Finding* approach. For regional maxima that have a plateau structure, the centroid of the plateau is marked as the cluster center. It should be noted that the centroid for a plateau structure will make sense only for convex clusters. For non–convex or arbitrary shaped clusters it is recommended to revert to the Watershed algorithm to obtain a better description of the cluster.

Thus the number of regional maxima in $\mathcal{LS}_{opt}$ is the number of clusters "present" in the dataset, and the cluster centers are given by projecting the maxima locations from $\mathcal{LS}_{opt}$ to $\mathcal{M}$ and in turn to the feature space spanned by the dataset $\mathbf{Y}$. This

information (cluster count and cluster center locations) can then be used by any partitioning algorithm such as the *k–means* algorithm to determine cluster labels for all the data points.

2.5 Experiments & Results

The method developed in Section 2.4 was to be tested on the datasets introduced in Section 2.2. The algorithm was coded in MATLAB, and experiments quickly encountered computational difficulties.

The experiments were stalled by memory limitations while trying to compute $\mathcal{M}$ as described in Section 2.4.1. While the data could have been scaled down so as to overcome the memory limitations, what follows is an additional mechanism that may be used in conjunction with the proposed method to achieve a workable bypass, should similar limitations be encountered while applying the proposed method to other datasets.

A modified matrix representation $\mathcal{M}'$ is constructed instead of $\mathcal{M}$ and the rest of the algorithm proceeds as described earlier. $\mathcal{M}'$ is computed in a fashion similar to $\mathcal{M}$, but instead of using d_{cw} as described in Eq(2.6), d'_{cw} is used, given by:

$$d'_{cw} = SF\frac{P_{cp}(d(\mathbf{y_i}, \mathbf{y_j}))}{2} \tag{2.15}$$

where $P_{cp}(d(\mathbf{y_i}, \mathbf{y_j}))$ is the cp percentile value for the pairwise distances $d(\mathbf{y_i}, \mathbf{y_j})(\forall i, j \in 1...n)$, and SF is an arbitrary $s$hrinkage factor between $[0, 1]$. This definition will create $\mathcal{M}'$ with a much smaller memory requirement than $\mathcal{M}$. However, total preservation of structural information cannot be guaranteed. Structural details smaller than d'_{cw} will not be preserved. For all the tests conducted herein, cp was set at 1. It is contended that structural details with size less than 1 percentile of the pairwise distances in the dataset should not significantly affect the cluster structure. This contention

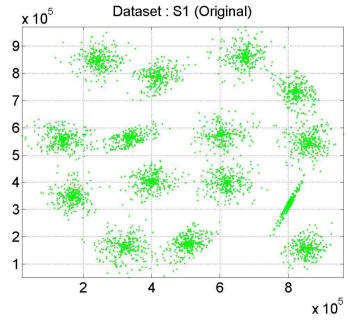
can be easily verified visually. The contention is further verified if the modified definitions produce the correct result with the chosen value of cp and several values of SF.

Figures 2.11a – 2.11f demonstrate cases where no perceivable differences are introduced between the original data and the modified matrix form of the data. However, there could be cases where some structural differences are perceived between the original data and the modified matrix form of the data – Figures 2.11g and 2.11h demonstrate one such case.

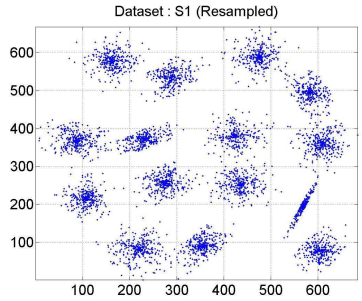
We note that this modification is proposed only for cases where a computational limitation restricts the original method altogether. Should a computational system be available such that a given dataset can be processed without running into storage limitations, this modification is not needed.

Table 2.2 displays the results obtained using the method proposed in Section 2.4 in conjunction with the *Regional Maxima Finding* approach for detecting cluster count and cluster centers. Table 2.3 displays the results obtained using the method proposed in Section 2.4 in conjunction with the *Watershed algorithm* approach for detecting cluster count and cluster centers. Entries displayed in bold red in Table 2.2 and Table 2.3 indicate cases where there is disagreement between the actual cluster count (Table 2.1) and the cluster count reported by the respective algorithms. Figures 2.12a through 2.12l display results of cluster detection (count & location) using the proposed method overlaid on the original datasets ($SF = 0.5$). (A standard MATLAB implementation of the *k–means* algorithm was used to determine the cluster labels for the data points.) Results for other values of SF are similar.

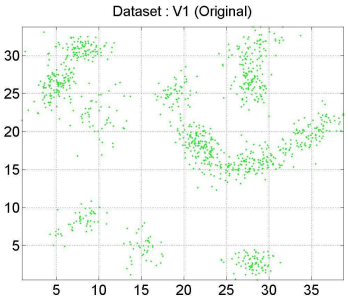
An overall comparison of Table 2.2 and Table 2.3 indicates that the proposed method variant using regional maxima finding approach with median filtering performs better than the proposed method variant using the Watershed algorithm. This is due to the intrinsic tendency of the Watershed algorithm to over–segment [5].



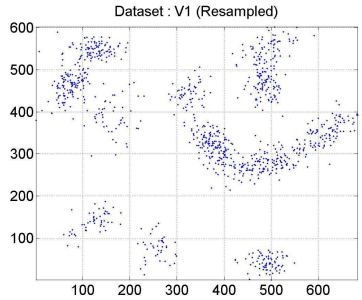
(a) S1 - Original



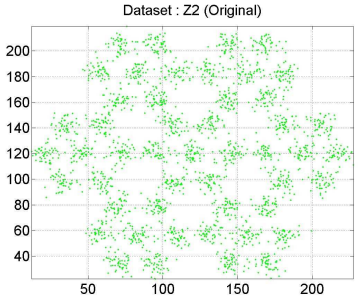
(b) S1 - $cp = 1$ & $SF = 0.5$



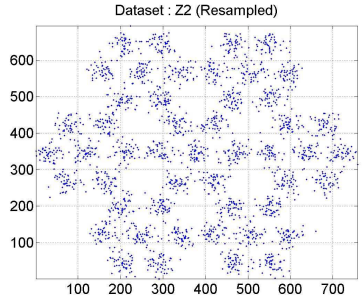
(c) V1 - Original



(d) V1 - $cp = 1$ & $SF = 0.5$

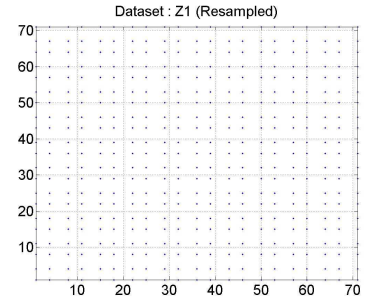


(e) Z2 - Original



(f) Z2 - $cp = 1$ & $SF = 0.5$

(g) Z1 - Original



(h) Z1 - $cp = 1$ & $SF = 0.8$

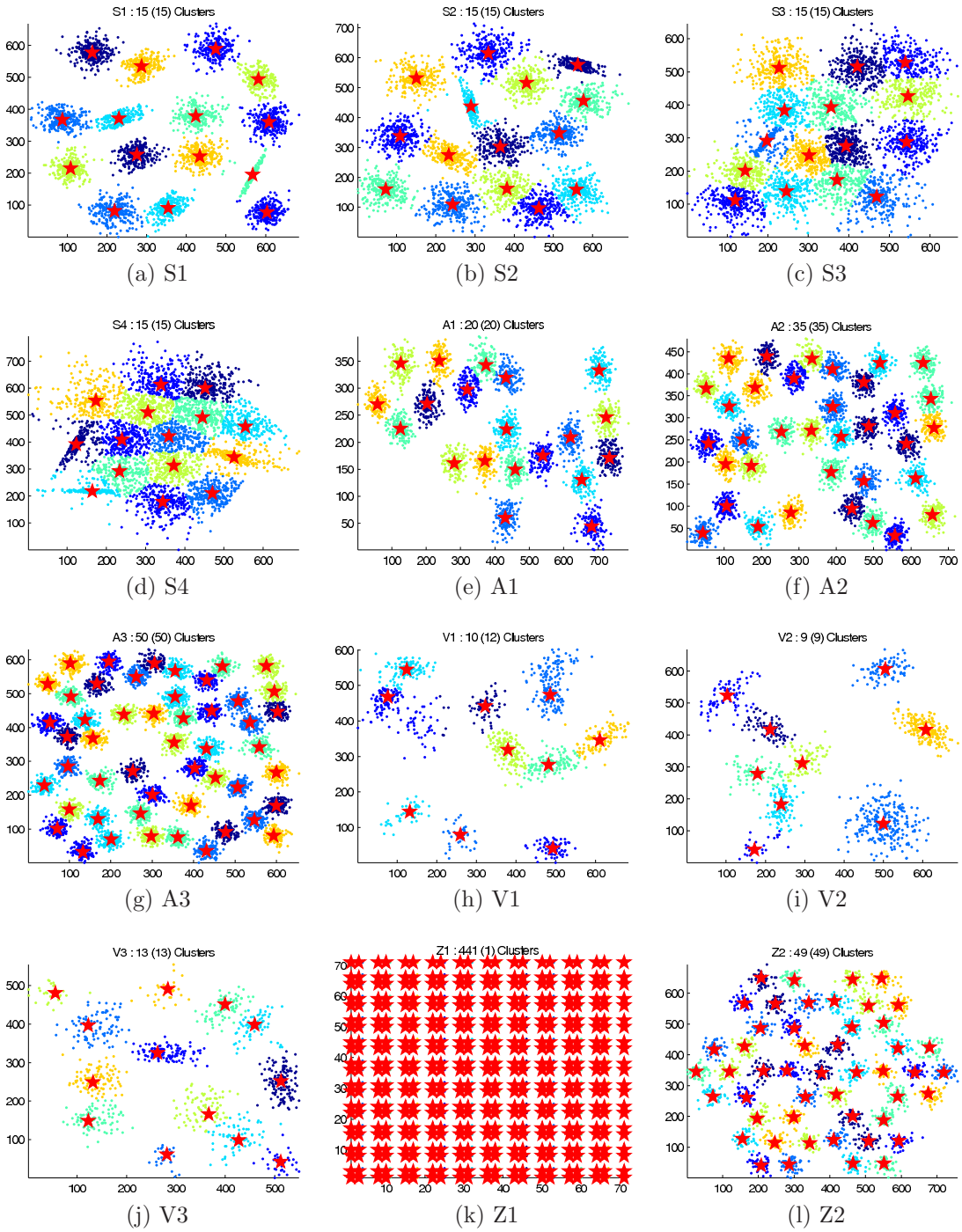
Figure 2.11: Original & Resampled: Datasets S1, V1, Z2 & Z1

Table 2.2 indicates that the method works well with S1–S4,A1–A3 & V2. The method consistently picks only 10 clusters for dataset V1 instead of the perceivable 11, which is likely due to the "low density" of the undetected cluster as compared to the other clusters in the dataset (See Figure 2.12h). Some experimental runs indicate an incorrect cluster count for dataset V3, and these are cases where median filtering fails to suppress the detection of spurious clusters. Figures 2.13a and 2.13b display two such cases. Dataset Z1 has a perfect square number of clusters in most cases. Figures 2.13c – 2.13f display such clustering results. It can be seen from these figures that these different perfect square counts are reported due to slight structural differences introduced during the construction of the modified matrix representation $\mathcal{M}'$ using different values for SF. The proposed method detects 49 clusters in dataset Z2 in most cases, but sometimes a cluster count of 50 is reported (See Figure 2.13g). It is suspected that this is also due to structural differences introduced during the construction of the modified matrix representation $\mathcal{M}'$ using different values for SF.

2.6 Conclusions

In this chapter, an improved approach toward Watershed–based clustering is presented. The improvements made are as follows:

- An automatic method is given for selecting cell size, based entirely on the data to be clustered itself, and eliminates the need for experimentally determined parameters.
- The computationally intensive Watershed algorithm is replaced with a much simpler regional maxima finding process.
- An approach toward incorporating the concept of scale while generating landscapes is introduced.



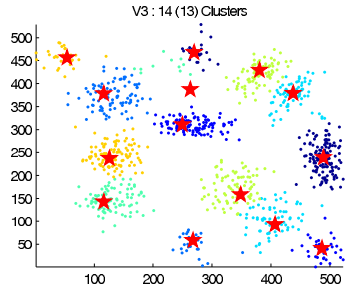
(a) S1 (b) S2 (c) S3

(d) S4 (e) A1 (f) A2

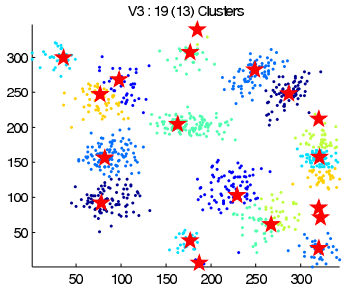
(g) A3 (h) V1 (i) V2

(j) V3 (k) Z1 (l) Z2

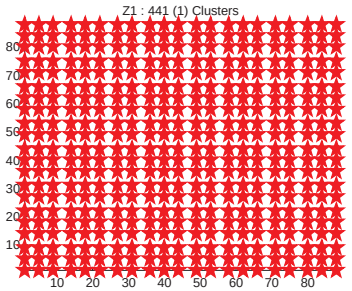
Figure 2.12: Clustering results using $SF = 0.5$ (Red points indicate cluster centers.)



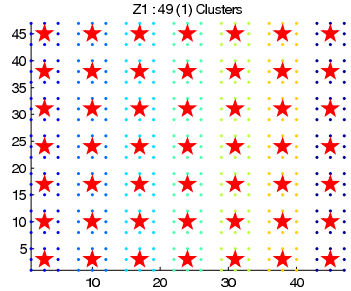
(a) V3 : $SF = 0.525$



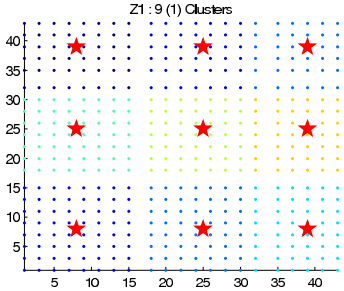
(b) V3 : $SF = 0.800$



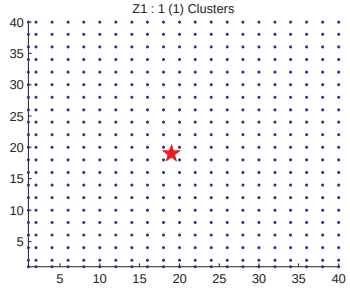
(c) Z1 : $SF = 0.400$



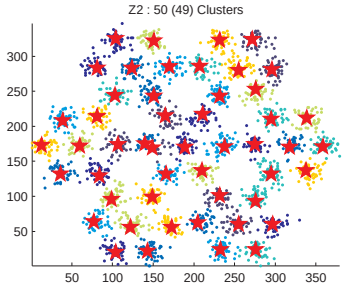
(d) Z1 : $SF = 0.750$



(e) Z1 : $SF = 0.825$



(f) Z1 : $SF = 0.900$



(g) Z2 : $SF = 1.000$

Figure 2.13: Clustering results – continued (Red points indicate cluster centers.)

Table 2.2: Results of Clustering – Regional Maxima with Median Filtering

SF \ Dataset	S1	S2	S3	S4	A1	A2	A3	V1	V2	V3	Z1	Z2
0.400	15	15	15	15	20	35	50	**10**	9	13	**441**	49
0.425	15	15	15	15	20	35	50	**10**	9	13	**441**	49
0.450	15	15	15	15	20	35	50	**10**	9	13	**441**	49
0.475	15	15	15	15	20	35	50	**10**	9	13	**441**	49
0.500	15	15	15	15	20	35	50	**10**	9	13	**441**	49
0.525	15	15	15	15	20	35	50	**10**	9	**14**	**441**	49
0.550	15	15	15	15	20	35	50	**10**	9	13	1	49
0.575	15	15	15	15	20	35	50	**10**	9	13	1	49
0.600	15	15	15	15	20	35	50	**10**	9	13	1	49
0.625	15	15	15	15	20	35	50	**10**	9	13	1	**50**
0.650	15	15	15	15	20	35	50	**10**	9	**14**	1	49
0.675	15	15	15	15	20	35	50	**10**	9	13	1	**50**
0.700	15	15	15	15	20	35	50	**10**	9	13	1	49
0.725	15	15	15	15	20	35	50	**10**	9	**15**	1	**50**
0.750	15	15	15	15	20	35	50	**10**	9	13	**49**	49
0.775	15	15	15	15	20	35	50	**10**	9	13	**36**	49
0.800	15	15	15	15	20	35	50	**10**	9	**19**	**16**	49
0.825	15	15	15	15	20	35	50	**10**	9	13	**9**	**50**
0.850	15	15	15	15	20	35	50	**10**	9	13	1	49
0.875	15	15	15	15	20	35	50	**10**	9	13	1	**50**
0.900	15	15	15	15	20	35	50	**10**	9	13	1	49
0.925	15	15	15	15	20	35	50	**10**	9	13	1	49
0.950	15	15	15	15	20	35	50	**10**	9	13	1	**50**
0.975	15	15	15	15	20	35	50	**10**	9	**14**	1	49
1.000	15	15	15	15	20	35	50	**10**	9	13	**30**	**50**
	← Number of Clusters →											

Bold red entries indicate disagreement between actual cluster count and cluster count reported by the algorithm.

Table 2.3: Results of Clustering – Traditional Watershed Approach

SF \ Dataset	S1	S2	S3	S4	A1	A2	A3	V1	V2	V3	Z1	Z2
0.400	15	15	**16**	**18**	20	35	50	10	**10**	**15**	**441**	**52**
0.425	15	15	15	**18**	20	35	50	**10**	**10**	**16**	**441**	**53**
0.450	15	15	15	**16**	20	35	50	**10**	**10**	**20**	**441**	**50**
0.475	15	15	15	**20**	20	35	50	**11**	9	**14**	**441**	**51**
0.500	15	15	15	**17**	20	35	50	**10**	**10**	**15**	**441**	**50**
0.525	15	15	**16**	**16**	20	35	50	**10**	**11**	**15**	**441**	49
0.550	15	15	15	**16**	20	35	50	**10**	**10**	**14**	1	**51**
0.575	15	15	15	15	20	35	50	**10**	9	**16**	1	**51**
0.600	15	15	15	15	20	35	50	**10**	**10**	**14**	1	**50**
0.625	15	15	15	**16**	20	35	50	**10**	9	**15**	1	**50**
0.650	15	15	15	15	20	35	50	**10**	9	**19**	1	**50**
0.675	15	15	15	15	20	35	50	**10**	9	13	1	**50**
0.700	15	15	15	15	20	35	50	**10**	**10**	**14**	1	**50**
0.725	15	15	15	15	20	35	50	**10**	9	**20**	1	**50**
0.750	15	15	15	15	20	35	50	**10**	9	**18**	**49**	**51**
0.775	15	15	15	15	20	35	50	**10**	9	**14**	**40**	**50**
0.800	15	15	15	15	20	35	50	**10**	9	**24**	**16**	**50**
0.825	15	15	15	15	20	35	50	**10**	9	13	**9**	**50**
0.850	15	15	15	15	20	35	50	**10**	9	**18**	1	**50**
0.875	15	15	15	15	20	35	50	**10**	9	**14**	1	**50**
0.900	15	15	15	15	20	35	50	**10**	9	**14**	1	**50**
0.925	15	15	15	15	20	35	50	**10**	9	**14**	1	**50**
0.950	15	15	15	15	20	35	50	**10**	9	13	**9**	**50**
0.975	15	15	15	15	20	35	50	**10**	9	**15**	1	**50**
1.000	15	15	15	15	20	35	50	**10**	9	**14**	**25**	**51**
	← Number of Clusters →											

Bold red entries indicate disagreement between actual cluster count and cluster count reported by the algorithm.

The main advantage of the proposed method is its unsupervised and automatic nature requiring no parameters to be tuned or to be determined experimentally.

Future investigations should explore issues such as: (1) optimal selection of SF and cp (where the modified matrix representation $\mathcal{M}'$ needs to be constructed) to minimize loss of structural information; (2) construction of statistical dispersion metrics that could be used to locate optimal scale indices with improved fidelity; (3) methodologies to handle spurious clusters in cases where median filtering fails; (4) use of non–Gaussian kernels; and (5) reduction of computational complexity.

CHAPTER 3

On Estimation Of Quantiles For Pairwise Distances

3.1 Introduction

Quantiles of pairwise distances for datasets are used to compute robust estimators of scale such as S_n and Q_n [7]. A naive implementation for computing quantiles for pairwise distances for a given dataset with n data points is given in Table 3.1. While this algorithm is relatively simple, the algorithm's time and memory requirements are order n^2 – $O(n^2)$. In fact Step–1 of the algorithm itself has time and memory requirements that are of order n^2. Thus the use of this algorithm to compute quantiles for pairwise distances becomes infeasible for large datasets. This chapter presents a novel approach to estimate quantiles for pairwise distances. The performance of the proposed method is compared against the performance of existing methods on four datafiles (described in Table 3.2, Table 3.3, and Figure 3.1). The work presented in this chapter was published in the Proceedings of the Ninth International Conference on Machine Learning and Applications (ICLMA'10) [27].

Table 3.1: A naive algorithm for computing quantiles of pairwise distances.

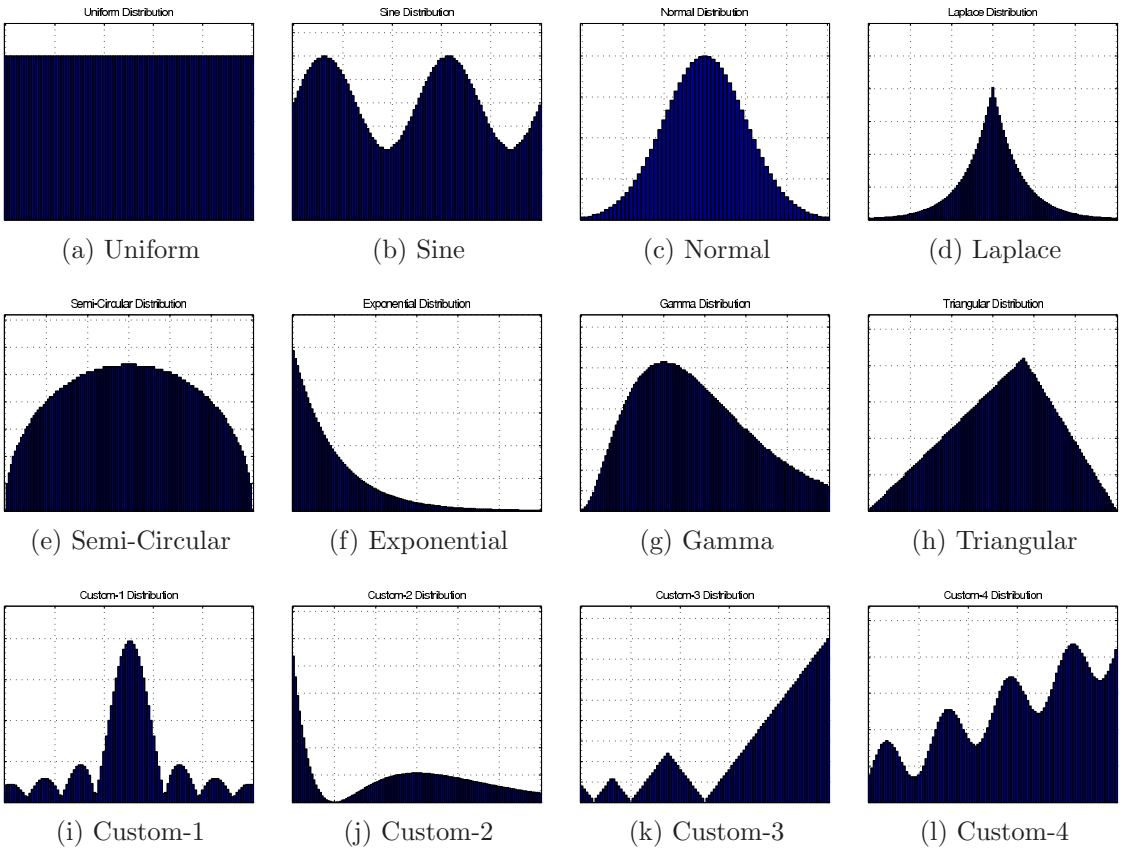
Input	Dataset X
Step 1	Compute pairwise distances for array X and store in array Y.
Step 2	Sort pairwise distances stored in array Y.
Step 3	Compute index for p^{th} quantile and select corresponding element.
Output	p^{th} quantile of pairwise distances of X

Table 3.2: Data files, Datasets & Data count

Datafile	# of Datasets	# of Data points
DF–1	12	≈100
DF–2	12	≈500
DF–3	12	≈1000
DF–4	12	≈5000

Table 3.3: Datasets & Data Distributions

Dataset	Distribution	Dataset	Distribution
DS–1	Uniform	DS–7	Gamma
DS–2	Sine	DS–8	Triangular
DS–3	Normal	DS–9	Custom–1
DS–4	Laplace	DS–10	Custom–2
DS–5	Semi–Circular	DS–11	Custom–3
DS–6	Exponential	DS–12	Custom–4



(a) Uniform (b) Sine (c) Normal (d) Laplace

(e) Semi-Circular (f) Exponential (g) Gamma (h) Triangular

(i) Custom-1 (j) Custom-2 (k) Custom-3 (l) Custom-4

Figure 3.1: Datasets used for testing

3.2 Currently Existing Methods

Several approaches have been outlined in the literature ([28], [29], [30], [31], [32], [33], [34], [35]) that present methods to estimate quantiles of datasets while requiring reduced memory. Most methods seem to be variants of the method proposed by Weide [28]. Most of the above referenced methods also place bounds on the error between the estimated and actual values of quantiles of pairwise distances. However, it should be noted that these methods target finding quantiles for a given dataset while using reduced memory and not *finding quantiles of pairwise distances* for a given dataset. Thus while most of the methods mentioned above do away with infeasible memory requirements, they still require that all the pairwise distances be computed, and hence are likely to have computational time requirements that are infeasible.

Johnson and Mizoguchi [36] proposed a method to select tuples based on their ranking, the k^{th} element in $X + Y$, defined as $\{x_i + y_i \mid x_i \in X, y_i \in Y\}$ where $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$. Here the rank of the tuple is defined as the value of the sum of all its elements. This method is later extended to $\sum_{i=2}^{m} X_i$ for $m > 2$. This method does not require that all the tuples be evaluated and sorted for selecting the k^{th} tuple. For the case of $m = 2$, the rank of the tuple is just the sum of two elements. The euclidean distance between two scalars is simply the difference in their magnitudes. Taking advantage of this, Croux and Rousseeuw [37] adapt the algorithm given by Johnson and Mizoguchi [36] such that it selects the k^{th} pairwise distance. This is done by redefining Y in the above algorithm as $Y = (-y_n, y_{n-1}, \ldots, y_1)$.

Since the modified algorithm by Croux and Rousseeuw [37] does not require that all the pairwise distances be calculated, its time requirements are less than requirements in other previously stated methods. It should also be noted that since the k^{th} pairwise distance is selected, rather than estimated, this method has practically no error in its output. Due to the nature of this algorithm, the memory requirements

for this method might exceed the requirements of other previously stated methods. Moreover, this method works only for scalar data and cannot be extended to data of higher dimensions, while the previously stated methods can, in theory, be extended to accommodate data of higher dimensions.

Most of the existing methods suffer from at least one of the following drawbacks: (1) infeasible computational time requirements, since all the pairwise distances must be computed; (2) infeasible memory requirements, since all or most of the data must be loaded into memory; and (3) in–extensibility of method to data of higher dimensions.

3.3 A New Proposed Method

The new method proposed here is an attempt to achieve a good trade–off between computation time and memory requirements in estimating quantiles of pairwise distances for a given dataset. The method proposed closely follows the theme of the method described by Schmeiser and Deutsch [30].

Let $X = (x_1, x_2, \ldots, x_n)$ be the given dataset (a collection of n scalars) for which the quantiles for pairwise distances are to be computed. Let the data be mapped into an "appropriate" histogram of m non–overlapping bins (not necessarily of uniform width) such that $B = (b_1, b_2, \ldots, b_m)$ represents the locations of the bin centers and $C = (c_1, c_2, \ldots, c_m)$ represents the bin counts, or the number of data points in each bin. The appropriateness of the histogram for the given dataset depends upon value of m. Choosing $m = 1$ results in all the data points being mapped into a single bin, resulting in a loss of distribution information. Choosing $m = n$ can result in each data point being mapped into a bin, yet again resulting in a loss of distribution information. Thus the value of m should lie in $[1, n]$ for the histogram to meaningfully capture the underlying distribution information present in the dataset. Literature that discusses selection of the number of bins can be found in [38], [39], [40], [41]. Our new proposed

method assumes that: *an "appropriate" histogram that reveals the salient features in the underlying data distribution can be (and will be) constructed for a given dataset* (Assumption–I). Assumption–I implies a Uniform Distribution of data within each bin.

The central idea of our proposed method is as follows: *Given B and C, (1) Compute number and average values of inter–bin and intra–bin pairwise distances for all bins; (2) Construct an approximate cumulative histogram based on those numbers; and (3) Interpolate the approximate cumulative histogram to estimate quantiles for pairwise distances.*

3.3.1 Average Values & Counts for Intra–Bin Pairwise Distances

Given a bin with bin count c_i, and bin width w_i, the intra–bin pairwise distances will follow a triangular distribution, as shown in Figure 3.2a. The number (Va_i) and the average value (Pa_i) for intra–bin pairwise distances for the i^{th} bin given as:

$$\begin{aligned} Va_i &= \tfrac{c_i(c_i-1)}{2} \\ Pa_i &= \tfrac{w_i}{3} \\ i &= [1, 2, \ldots, m] \end{aligned} \tag{3.1}$$

are computed for each of the bins. The value combinations (Va_i,Pa_i) are stored for later use in the construction of the approximate cumulative histogram for the pairwise distances of the dataset.

3.3.2 Average Values & Counts for Inter–Bin Pairwise Distances

Given two bins with bin counts c_i, c_j ($c_i \leq c_j$ without loss of generality), and bin width is w_i, w_j, the inter–bin pairwise distances will follow a trapezoidal distribution. Figure 3.2b shows the distribution for bins that are well separated, Figure 3.2c shows the distribution for bins that are adjacent, and Figure 3.2d shows the distribution for

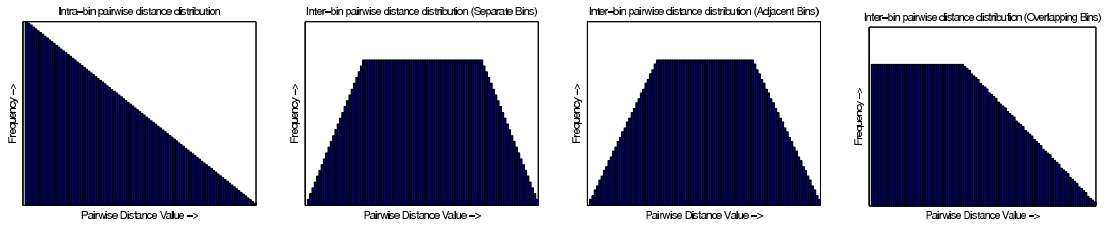
bins that are overlapping. Following Assumption–I the overlapping case is ruled out. The number (Vb_k) and the average value (Pb_k) for inter–bin pairwise distances for the pair of the i^{th} and j^{th} bins are computed by treating the trapezoidal distribution as a summation of a triangle on the left side, a rectangle in middle, and a triangle on the right side. Let $x1_i$,$x2_i$ mark the bin edges for the i^{th} bin. Similarly, let $x1_j$,$x2_j$ mark the bin edges for the j^{th} bin. We define D as the sorted array containing the pairwise distances between the bin edges:

$$D = sort([|x1_i - x1_j| \quad |x2_i - x1_j| \quad |x1_i - x2_j| \quad |x2_i - x2_j|]) \tag{3.2}$$

The first element of D marks the beginning point of the left triangle, the second element marks the beginning of the rectangle, the third element marks the beginning of the right triangle, and the fourth element marks the end of the right triangle. Vb_k and Pb_k are given as:

$$\begin{aligned}
Vl_k &= \tfrac{c_i(c_i-1)}{2} \\
Pl_k &= \tfrac{2}{3}D(2) + \tfrac{1}{3}D(1) \\
Vm_k &= c_i(c_j - c_i + 1) \\
Pm_k &= \tfrac{1}{2}D(2) + \tfrac{1}{2}D(3) \\
Vr_k &= \tfrac{c_i(c_i-1)}{2} \\
Pr_k &= \tfrac{2}{3}D(3) + \tfrac{1}{3}D(4) \\
Vb_k &= Vl_k + Vm_k + Vr_k \\
Pb_k &= \tfrac{Vl_kPl_k + Vm_kPm_k + Vr_kPr_k}{Vl_k + Vm_k + Vr_k} \\
k &= [1, 2, \ldots, \tfrac{m(m-1)}{2}]
\end{aligned} \tag{3.3}$$

are computed for each pair of bins. Using the combinations (Vl_k,Pl_k), (Vm_k,Pm_k), and (Vr_k,Pr_k) instead of the aggregate combination (Vb_k,Pb_k) for the construction



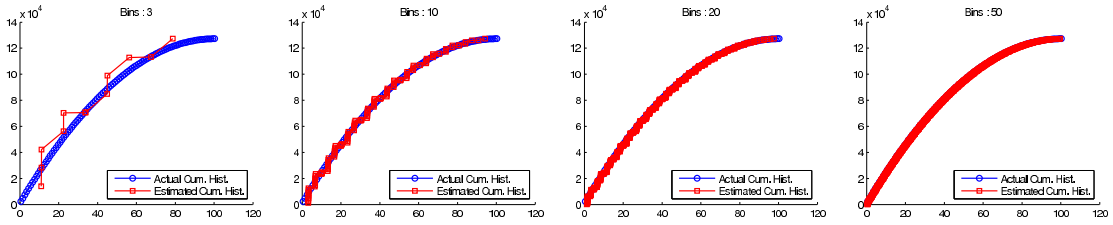
(a) Intra–Bin (b) Inter–Bin : Separated Bins (c) Inter–Bin : Adjacent Bins (d) Inter–Bin : Overlapping Bins

Figure 3.2: Intra–Bin (a) & Inter–Bin Pairwise Distance Distributions (b)–(d)

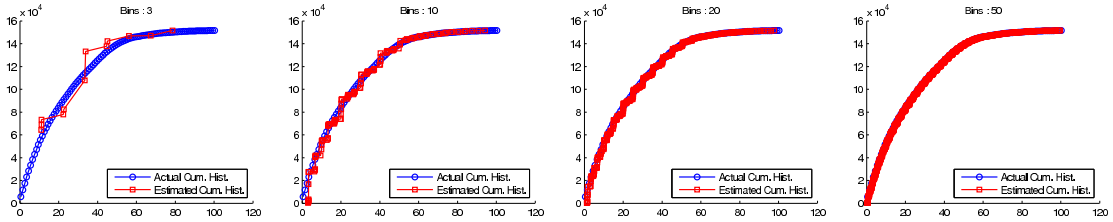
of the approximate cumulative histogram will result in slightly better approximation. The value combinations (Vl_k,Pl_k), (Vm_k,Pm_k), and (Vr_k,Pr_k) are stored for later use in the construction of the approximate cumulative histogram for the pairwise distances of the dataset.

3.3.3 Construction and Interpolation of an Approximate Cumulative Histogram

Once the number and average values for intra–bin and inter–bin pairwise distances are computed, the stored combinations are sorted on the basis of increasing average pairwise distance p (in Pa_i,Pl_k,Pm_k, and Pr_k). A table is constructed with the first column as the average pairwise distance p, the second column as the cumulative count of v (in Va_i,Vl_k,Vm_k, and Vr_k). Once this table is constructed, to estimate a particular quantile, a lookup or interpolation operation is performed over the table. Figure 3.3a – 3.3h show graphical representations of the approximate cumulative histograms constructed for datasets DS–1 and DS–9 from data file DF–2, with 500 data points in each dataset. Various numbers of histogram bins ($m = 3, 10, 20, 50$) are shown. The blue lines (with circular markers) in these figures are actual cumulative histograms for pairwise distances, and the red lines (with square markers) are approximate cumulative histograms constructed using our method. It can be seen that increasing the number of bins to construct the approximate histogram results in increasingly better approximations of the actual cumulative histogram for pairwise distances, and hence,



(a) DS–1 : $m = 3$ (b) DS–1 : $m = 10$ (c) DS–1 : $m = 20$ (d) DS–1 : $m = 50$



(e) DS–9 : $m = 3$ (f) DS–9 : $m = 10$ (g) DS–9 : $m = 20$ (h) DS–9 : $m = 50$

Figure 3.3: Approximate Cumulative Histograms constructed for DS–1 & DS–9 using $m = 3, 10, 20\&50$

will result in increasingly accurate estimates of the quantiles for pairwise distances. The least accurate and least time and memory consuming case occurs when all the data is grouped into one bin ($m = 1$). The most accurate and most time and memory consuming case occurs when each data point is mapped into a unique bin ($m = n$), and the algorithm is reduced to the naive implementation. Hence it can be seen that a good selection of the number of histogram bins (m) will result in a trade–off between the algorithm's accuracy and the algorithm's computation time and memory requirements.

3.4 Experiments & Results

The method developed in Section 3.3 was coded in MATLAB to be tested on the datasets introduced earlier. A naive implementation for computing the quantiles of pairwise distances, the method proposed by Weide [28], and the Croux and Rousseeuw's adaption [37] of Johnson and Mizoguchi's method [36] were also coded in MATLAB to compare against the performance of the proposed method. The number of histogram bins (m) was manually set at 50, since that was deemed to be an appropriate value

Table 3.4: Results for DF–1 (100 points)

	NM	WM			CRM			LoHoM		
Dataset	Time	Time	MSE	SMAPE	Time	MSE	SMAPE	Time	MSE	SMAPE
DS-1	0.003	0.037	129.6406	0.0892	0.145	0.0001	0.0001	0.092	0.0141	0.0023
DS-2	0.007	0.054	113.5292	0.0836	0.336	0.0000	0.0000	0.102	0.0161	0.0029
DS-3	0.005	0.047	14.9214	0.0869	0.111	0.0000	0.0001	0.072	0.0074	0.0036
DS-4	0.010	0.065	80.8357	0.1004	0.258	0.0000	0.0000	0.112	0.0393	0.0028
DS-5	0.007	0.057	101.0443	0.0881	0.263	0.0000	0.0001	0.114	0.0238	0.0037
DS-6	0.009	0.068	108.4192	0.0585	0.580	0.0000	0.0000	0.107	0.0496	0.0034
DS-7	0.007	0.061	79.2869	0.0795	0.286	0.0000	0.0000	0.109	0.0175	0.0035
DS-8	0.007	0.062	130.7961	0.0968	0.245	0.0000	0.0001	0.111	0.0239	0.0033
DS-9	0.006	0.054	57.1713	0.1122	0.171	0.0000	0.0001	0.089	0.0287	0.0038
DS-10	0.007	0.058	163.8101	0.0891	0.524	0.0000	0.0001	0.115	0.0589	0.0060
DS-11	0.006	0.055	322.3978	0.1799	0.189	0.0001	0.0001	0.088	0.0200	0.0022
DS-12	0.006	0.054	182.4313	0.1141	0.185	0.0000	0.0001	0.095	0.0159	0.0038
NM - Naive Method; WM - Weide Method; CRM - Croux Rosseeuw Method; LoHoM - Proposed Method Smaller MSE & SMAPE values indicate better performance. Smaller Time value indicates better performance.										

Table 3.5: Results for DF–2 (500 points)

	NM	WM			CRM			LoHoM		
Dataset	Time	Time	MSE	SMAPE	Time	MSE	SMAPE	Time	MSE	SMAPE
DS-1	0.068	0.220	133.6670	0.0925	1.536	0.0000	0.0000	0.130	0.0112	0.0019
DS-2	0.083	0.233	117.5170	0.0870	1.595	0.0000	0.0000	0.130	0.0119	0.0039
DS-3	0.078	0.219	9.4230	0.0859	0.432	0.0000	0.0000	0.100	0.0134	0.0094
DS-4	0.086	0.234	26.7054	0.1004	0.450	0.0000	0.0000	0.118	0.0492	0.0112
DS-5	0.087	0.233	118.3215	0.0904	1.385	0.0000	0.0000	0.123	0.0090	0.0035
DS-6	0.093	0.238	5.3208	0.0328	4.529	0.0000	0.0000	0.132	0.0788	0.0100
DS-7	0.085	0.228	51.4099	0.0793	1.490	0.0000	0.0000	0.123	0.0263	0.0033
DS-8	0.126	0.261	97.0951	0.0972	0.890	0.0000	0.0000	0.122	0.0196	0.0031
DS-9	0.083	0.229	33.0433	0.1273	0.753	0.0000	0.0000	0.132	0.0324	0.0048
DS-10	0.085	0.233	192.9634	0.1130	2.323	0.0000	0.0000	0.123	0.0358	0.0061
DS-11	0.079	0.225	301.1255	0.2247	0.437	0.0000	0.0000	0.116	0.0561	0.0079
DS-12	0.081	0.224	184.0531	0.1228	1.079	0.0000	0.0000	0.123	0.0191	0.0043
NM - Naive Method; WM - Weide Method; CRM - Croux Rosseeuw Method; LoHoM - Proposed Method Smaller MSE & SMAPE values indicate better performance. Smaller Time value indicates better performance.										

(satisfying Assumption–I). Computation times were measured using the computer's internal clock and are reported in seconds. The testing was done on a computer running Windows XP SP3, with an Intel Pentium 4 CPU (3.20 GHz) processor, and having 2.00 GB RAM. The accuracy of each method is measured by computing the metrics MSE (*Mean Squared Error*) and SMAPE (*Symmetric Mean Absolute*

Table 3.6: Results for DF–3 (1000 points)

	NM	WM			CRM			LoHoM		
Dataset	Time	Time	MSE	SMAPE	Time	MSE	SMAPE	Time	MSE	SMAPE
DS-1	0.284	0.753	132.8294	0.0928	5.023	0.0000	0.0000	0.137	0.0265	0.0040
DS-2	0.309	0.506	117.1777	0.0863	5.185	0.0000	0.0000	0.140	0.0184	0.0044
DS-3	0.300	0.499	9.0518	0.0856	0.993	0.0000	0.0000	0.126	0.0126	0.0096
DS-4	0.324	0.529	21.8617	0.0969	0.761	0.0000	0.0000	0.135	0.1005	0.0164
DS-5	0.317	0.511	119.9340	0.0910	3.633	0.0000	0.0000	0.193	0.0180	0.0045
DS-6	0.321	0.512	0.7568	0.0185	9.295	0.0000	0.0000	0.129	0.1474	0.0180
DS-7	0.324	0.513	48.2129	0.0797	3.638	0.0000	0.0000	0.131	0.0472	0.0055
DS-8	0.317	0.510	92.4086	0.0973	2.030	0.0000	0.0000	0.132	0.0113	0.0020
DS-9	0.315	0.519	31.1328	0.1293	1.907	0.0000	0.0000	0.135	0.0684	0.0095
DS-10	0.317	0.519	195.7374	0.1186	7.306	0.0000	0.0000	0.136	0.0497	0.0065
DS-11	0.310	0.519	295.8134	0.2289	1.498	0.0000	0.0000	0.124	0.0339	0.0080
DS-12	0.310	0.500	185.1643	0.1267	2.678	0.0000	0.0000	0.131	0.0261	0.0075

NM - Naive Method; WM - Weide Method; CRM - Croux Rosseeuw Method; LoHoM - Proposed Method
Smaller MSE & SMAPE values indicate better performance.
Smaller Time value indicates better performance.

Table 3.7: Results for DF–4 (5000 points)

	NM	WM			CRM			LoHoM		
Dataset	Time	Time	MSE	SMAPE	Time	MSE	SMAPE	Time	MSE	SMAPE
DS-1	8.839	8.299	133.5150	0.0931	104.834	0.0000	0.0000	0.468	0.0490	0.0061
DS-2	8.352	6.501	116.2811	0.0860	108.797	0.0000	0.0000	0.231	0.0201	0.0054
DS-3	8.576	6.426	8.8878	0.0862	25.639	0.0000	0.0000	0.223	0.0121	0.0102
DS-4	8.468	6.670	19.0011	0.0949	12.857	0.0000	0.0000	0.224	0.1326	0.0169
DS-5	8.321	6.505	118.6491	0.0913	76.683	0.0000	0.0000	0.225	0.0636	0.0073
DS-6	8.401	6.568	0.2518	0.0036	230.427	0.0000	0.0000	0.218	0.1954	0.0245
DS-7	8.550	6.372	45.4599	0.0798	71.489	0.0000	0.0000	0.220	0.0500	0.0036
DS-8	8.476	6.701	88.7781	0.0978	44.321	0.0000	0.0000	0.227	0.0857	0.0108
DS-9	8.397	6.657	29.3556	0.1304	40.085	0.0000	0.0000	0.224	0.0665	0.0149
DS-10	8.289	6.559	197.4254	0.1236	147.778	0.0000	0.0000	0.218	0.0335	0.0061
DS-11	8.573	6.467	288.9834	0.2319	20.876	0.0000	0.0000	0.223	0.1028	0.0091
DS-12	8.311	6.586	185.9443	0.1278	65.934	0.0000	0.0000	0.224	0.0375	0.0098

NM - Naive Method; WM - Weide Method; CRM - Croux Rosseeuw Method; LoHoM - Proposed Method
Smaller MSE & SMAPE values indicate better performance.
Smaller Time value indicates better performance.

Percentage Error) for each method. The metrics are defined as:

$$\begin{aligned} \hat{Q} &= [\hat{Q}_5, \hat{Q}_{10}, \hat{Q}_{15}, \ldots, \hat{Q}_{95}] \\ Q &= [Q_5, Q_{10}, Q_{15}, \ldots, Q_{95}] \\ MSE &= \tfrac{1}{N_q} \sum_{i=1}^{N_q} (\hat{Q}(i) - Q(i))^2 \\ SMAPE &= \tfrac{1}{N_q} \sum_{i=1}^{N_q} \frac{|\hat{Q}(i) - Q(i)|}{\hat{Q}(i) + Q(i)} \end{aligned} \tag{3.4}$$

where $\hat{Q}_k$ is the k^{th} percentile of the pairwise distances of the given dataset (actual value from the naive implementation), Q_k is the estimation of the k^{th} percentile (estimation from any of the estimation methods mentioned earlier), and N_q is the number of elements in the $\hat{Q}$ (and Q) vector. Increasing N_q increases the fidelity of the metrics. It can be seen from Tables 3.4 – 3.7 (through MSE & SMAPE columns) that the proposed method (LoHoM in the Tables) performs better than the Weide Method (WM in the Tables) in terms of accuracy. Since the Croux and Rousseeuw Method (CRM in the Tables) is a selection method, and not an estimation method, it is expected to perform better in terms of accuracy. While the Croux and Rousseeuw Method performs better in terms of accuracy, it can be seen that the proposed method performs much better in terms of time (see the Time column) than all the other three methods, including the Naive Method (NM in the Tables), especially as the number of data points in each dataset increases.

3.5 Conclusions

In this chapter a novel method to estimate quantiles for pairwise distances is introduced. The method is shown to perform better in terms of time and accuracy for large datasets as compared to the method proposed by Weide [28]. The method is also shown to have a significant advantage in computational time requirements as

compared to the method proposed by Croux and Rousseeuw [37]. Unlike the method proposed by Croux and Rousseeuw [37], the new method proposed can be generalized to accommodate data of higher dimensions.

Future investigations should explore issues such as: (1) Optimal selection of the number of histogram bins (m); (2) Generalization of the algorithm to accommodate data of higher dimensions; (3) Optimizing the proposed method to reduce time and memory requirements; (4) Performing a formal analysis of the proposed method to obtain bounds on time and memory requirements; (5) Performing a formal analysis to obtain theoretical bounds on the error in estimating quantiles for pairwise distances;

CHAPTER 4

On Selecting The Number Of Bins For A Histogram

4.1 Introduction

A histogram is a graphical representation of the frequency distribution of a dataset. Widely employed in exploratory data analysis, a histogram can be treated as a simple non–parametric density estimator. For a given dataset, a histogram can visually convey the information relating to shape, spread, location, modality and symmetry of the distribution of the underlying population, and are well suited for summarizing large datasets [42]. While more sophisticated kernel–based density estimators are available, histograms are widely employed due to the ease and simplicity of construction and interpretation [43], [44]. While histograms are used mainly for visualizing data and obtaining summary quantities such as entropy, the values of such quantities depend upon the number of bins used (or the bin width used) and the location of the bins [45].

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a univariate dataset with probability density function $f(x)$. We follow Martinez et al. [42]: To construct a histogram, an origin for the bins t_0 (also referred to as the anchor) and a bin width h are selected. Selection of these two parameters defines a mesh (position of all the bins) over which the histogram will be constructed. Each bin is represented by a pair of bin edges as $B_k = [t_k, t_{k+1})$, where $t_{k+1} - t_k = h$ for all k. Histograms using varying bin widths are not addressed in this chapter. Let c_k represent the number of observations in B_k (bin count for B_k)

given by:

$$c_k = \sum_{i=1}^{n} I_{B_k}(x_i) \tag{4.1}$$

where I_{B_k} is defined as:

$$I_{B_k}(x_i) = \begin{cases} 1 & x_i \text{ in } B_k \\ 0 & x_i \text{ not in } B_k \end{cases} \tag{4.2}$$

While the density estimate for the underlying population (c_k for all k) satisfies the non–negativity condition necessary for it to be a *bona fide* probability density function, the summation of all the probabilities do not necessarily add to unity. To satisfy that condition, the probability density function estimate, $\hat{f}(x)$, as obtained from a histogram, is defined as:

$$\hat{f}(x) = \frac{c_k}{nh} \qquad \text{for } x \text{ in } B_k \tag{4.3}$$

This assures that $\int \hat{f}(x)dx = 1$ is satisfied, and $\hat{f}(x)$ represents a valid estimate for the probability density function of the population underlying the dataset.

The information relating to shape, modality, symmetry and summary quantities estimated using a histogram will depend on the values that c_k (and $\hat{f}(x)$) assume, which in turn depend upon the parameters t_0 and h.

While histograms are commonly constructed using $t_0 = \min(X)$, it is known that modifying this parameter can sometimes cause a rather drastic change in the values assumed by c_k [46]. Simonoff et al. [44] provide a method to quantify the effects of changing the parameter t_0 during the construction of a histogram. However, in the work herein, we use $t_0 = \min(X)$.

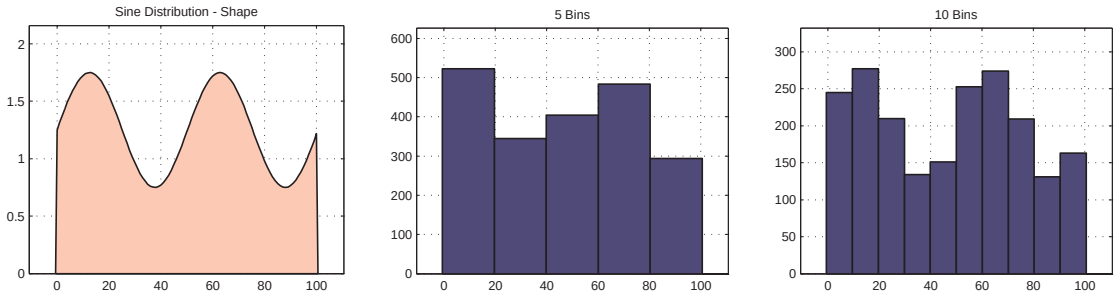
A common method to determine bin width h is:

$$h = \frac{\max(X) - \min(X)}{m} \tag{4.4}$$

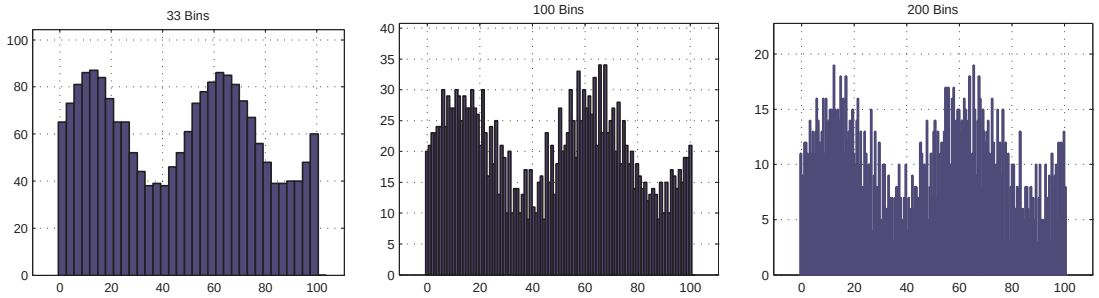
From (4.1), (4.2), (4.3) and (4.4) it can be seen that the number of bins used to construct a histogram will influence c_k (and $\hat{f}(x)$) and any further information derived from them. Consider the following two extreme cases: (1) Using only one bin ($m = 1$) will cause all the data points in X to map to that bin, and information relating to shape, modality, and symmetry will be lost (unless the underlying population distribution is Uniform); (2) Using n or more bins ($m \geq n$) will spread the data points over all the bins more or less uniformly, such that any information relating to shape, modality, and symmetry will again be lost. These two extreme cases suggest that an "optimal" number of bins should be used to construct a histogram that can effectively capture information relating to shape, modality, and symmetry and provide meaningful values for summary quantities. Using very few bins (small value for m) results in a large bin width, producing a histogram that captures the shape of the underlying distribution "coarsely". Using excessive bins (large value for m) results in a small bin width, producing a "noisy" histogram that captures the shape of the underlying distribution "finely" and typically "noisily". Figure 4.1 illustrates that arbitrarily increasing the number of bins to construct a histogram does not necessarily result in "better" histograms.

Thus, the problem of selecting an "optimal" number of bins refers to selecting an appropriate number of bins for constructing a histogram that achieves a "good" balance between "degree of detail" and "noisiness" for a given dataset. In other words, the number of bins should be large enough to capture all the major shape features present in the distribution, but small enough so as to suppress finer details produced due to random sampling noise [45].

Tables 4.1 and 4.2 and Figure 4.2 describe the datafiles and datasets used for testing our proposed method. The work presented in this chapter was published in the Proceedings of the Seventh International Conference on Data Mining (DMIN'11) [47].



(a) Original (b) 5 bins (c) 10 bins



(d) 33 bins (e) 100 bins (f) 200 bins

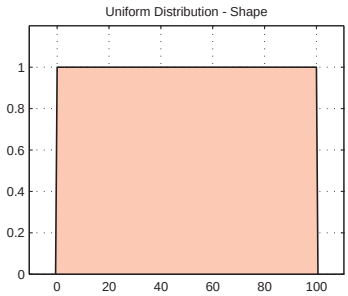
Figure 4.1: Original distribution and several histograms for a dataset ($\approx$ 2000 points)

Table 4.1: Datafiles used for testing

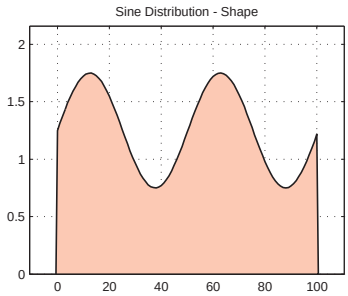
Datafile	# of Datasets	# of Data points
DF-1	12	$\approx$ 500
DF-2	12	$\approx$ 1000
DF-3	12	$\approx$ 2000
DF-4	12	$\approx$ 5000

Table 4.2: Datafiles used for testing

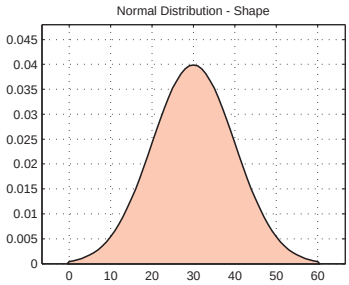
Dataset	Distribution	Dataset	Distribution
DS-1	Uniform	DS-7	Gamma
DS-2	Sine	DS-8	Triangular
DS-3	Normal	DS-9	Custom-1
DS-4	Laplace	DS-10	Custom-2
DS-5	Semi-Circular	DS-11	Custom-3
DS-6	Exponential	DS-12	Custom-4



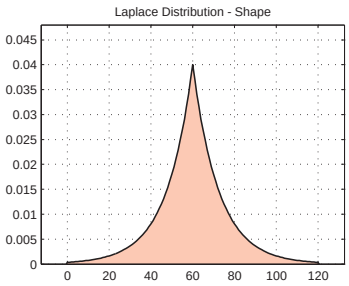
(a) Uniform



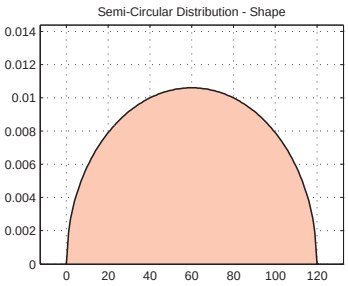
(b) Sine



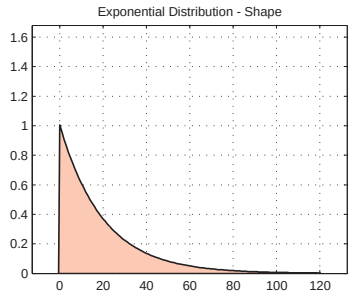
(c) Normal



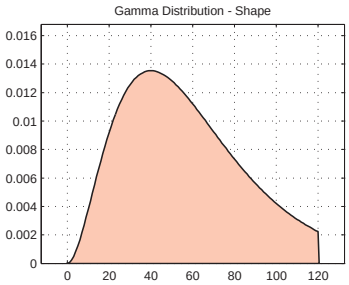
(d) Laplace



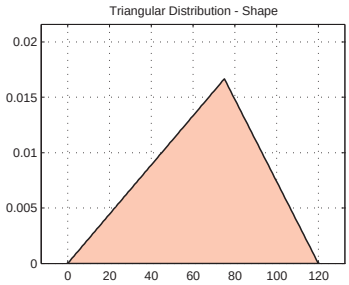
(e) Semi-Circular



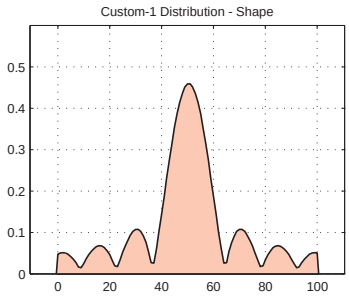
(f) Exponential



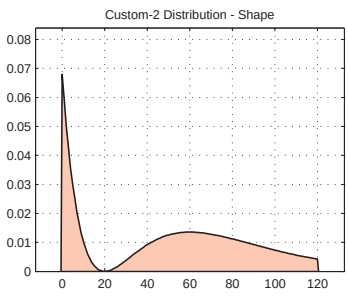
(g) Gamma



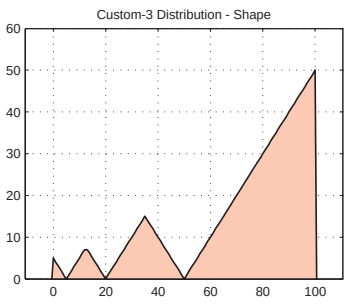
(h) Triangular



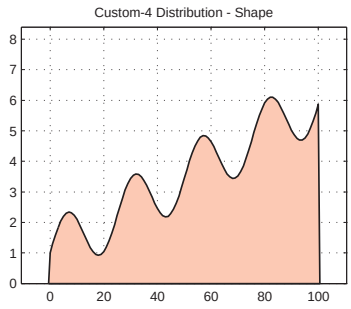
(i) Custom-1



(j) Custom-2



(k) Custom-3



(l) Custom-4

Figure 4.2: Datasets used for testing

4.2 Existing Methods

Perhaps the earliest reported method for constructing histograms is due to Sturges [48]. It is based on the assumption that a good distribution will have binomial coefficients $\binom{m-1}{i}, i = 0, 1, 2, \ldots, m-1$ as its bin counts. It suggests the number of bins to be used as:

$$m = 1 + \log_2 n \tag{4.5}$$

Hyndman [49] suggests that the argument used by Sturges [48] is incorrect and should not be used. Scott [50] uses IMSE (Integrated Mean Square Error – which is equal to Mean Integrated Square Error MISE [51]) as the measure of error between the estimated probability density ($\hat{f}(x)$) represented by the histogram, and the actual (and unknown) probability density ($f(x)$) of the underlying population. IMSE is defined as:

$$\begin{aligned} IMSE &= \int MSE(x)\, dx \\ &= \int E(\hat{f}(x) - f(x))^2\, dx \\ &= E \int (\hat{f}(x) - f(x))^2\, dx \\ &= MISE \end{aligned} \tag{4.6}$$

Using this error metric with Gaussian density as the reference for the actual probability density, Scott suggests the bin width to be used as:

$$h = \frac{3.49s}{n^{1/3}} \tag{4.7}$$

where s is the estimated standard deviation. Freedman et al. [52] suggests a similar formula with a slight modification:

$$h = \frac{2(IQR(X))}{n^{1/3}} \tag{4.8}$$

where $IQR(X)$ is the Inter–Quartile Range for the dataset X.

Methods proposed by Stone [53], Rudemo [54], and Wand [43] are also frequently encountered in the related literature. Stone [53] proposes a method based on minimization of a loss function defined on the basis of bin probabilities and number of bins. Rudemo [54] proposes a method based on Kullback–Leibler risk function and cross–validation techniques. Wand [43] extends Scott's method [50] to have good large sample consistency properties. Hall [55] investigates the use of Akaike's Information Criterion (AIC) and Kullback Liebler Cross Validation methods for constructing histograms.

More recently, Birge et al. [56] have proposed a method using a risk function based on penalized maximum likelihood. Knuth [45] has proposed a method based on maximizing the posterior probability for number of bins. Shimazaki et al. [38] have proposed a method based on minimizing an estimated cost function obtained by using a modified MISE. The method evaluates the estimated cost function using the implications of an assumption that the data are sampled independently of each other (assumption of a Poisson point process).

4.3 A New Proposed Method

Popular methods such as given by Scott [50], and Freedman et al. [52] try to asymptotically minimize MISE. These methods make certain assumptions to allow estimating the value of MISE, since the actual density function of the underlying population itself is unknown. Knuth [45] suggests that it is not reasonable to extend these assumptions for all datasets. It is also known that MISE does not necessarily conform with the human perception of closeness of a density function to its target [46]. Marron et al. [57] provide a good introduction to the disconnect between classical mathematical theory and the practice of non–parametric density estimation due to the non–conformance of human perception of closeness with metrics such as MISE and MIAE. Methods employing risk functions based on penalized likelihood functions need not make as-

sumptions about the underlying function, but their performance will depend upon the form of the risk function selected.

In the new method proposed here, error metrics are defined on quantities observable or computable from the dataset. An balance between the error and the cost of computing the histogram is used to select the number of bins.

Motivation: A histogram for a given dataset can be interpreted as a compact representation of the dataset itself, obtained by a lossy compression process. A good histogram will provide enough information to recreate data whose Cumulative Distribution Function (CDF) approximately matches the Cumulative Distribution Function of the actual dataset itself (Statement–I). Also, a good histogram will have no significant shape information inside any bin (Statement–II).

Statements I & II are axiomatic. They also indicate that data can be reconstructed from a given histogram. There are two simple ways to approximately reconstruct data from a histogram. For each bin B_k with bin count c_k: (1) recreate c_k data points equal to the bin center $((t_k + t_{k+1})/2)$ – equivalent to nearest neighbor interpolation; (2) recreate c_k data points spread uniformly over (t_k, t_{k+1}) – equivalent to linear interpolation.

Let $\hat{X}_{NN} = \{\hat{x}_{1_{NN}}, \hat{x}_{2_{NN}}, \ldots, \hat{x}_{n_{NN}}\}$ represent data reconstructed using the nearest neighbor equivalent described above, and let $\hat{X}_L = \{\hat{x}_{1_L}, \hat{x}_{2_L}, \ldots, \hat{x}_{n_L}\}$ represent data reconstructed using the linear interpolation equivalent. Figure 4.3 illustrates that for a histogram constructed using a given number of bins for a dataset, the CDF of the data recreated using linear interpolation matches the actual CDF more closely than the data recreated using the nearest neighbor approximation. Due to the Glivenko-Cantelli theorem [58], [59] both approximations will converge to the actual CDF itself as m increases.

Define the error metrics E_{NN} and E_L for the nearest neighbor and linear interpo-

lation reconstructions , respectively, by:

$$\begin{aligned} E_{NN} &= \sum_{i=1}^{n} |x_i - \hat{x}_{i_{NN}}| \\ E_L &= \sum_{i=1}^{n} |x_i - \hat{x}_{i_L}| \end{aligned} \tag{4.9}$$

Due to the aforementioned theorem, E_{NN} and E_L will converge to zero as the number of bins used to construct the histogram are increased ($m \to \infty$). In fact the convergence of the error metrics to zero is very likely once $m \geq n$. The CDF of data reconstructed using linear interpolation, which matches the actual data CDF more closely than the data reconstructed using the nearest neighbor approximation, indicates that E_L will converge faster than E_{NN}. Figure 4.4 shows plots of E_{NN} and E_L for various values of m. In Figure 4.4, the vertical axis represents the value of the error metrics and the horizontal axis represents the value of the computational cost. The computational cost involved in constructing a histogram using m bins for n points will at the most be of order $O(mn)$. Since we are trying to select m for the same n points, the computational costs will be proportional to m and hence m is used as the computational cost.

Figure 4.4 uses square markers to indicate "elbow points" for both error metric curves. An elbow point marks the region where incurrence of further "costs" does not result in any further significant "gains". Hence elbow points represent a trade–off between two conflicting quantities. The method is often traced to Thorndike [60] and has been used for similar purposes [17], [61]. The method described in [17] is used to compute the elbow points for the work done in this paper.

Let m_{NN} and m_L correspond, respectively, to the number of bins indicated by the elbow points on the E_{NN} and E_L metric curves. Using any m in [m_L,m_{NN}] will result in a histogram that offers a reasonably good trade–off between the error metrics and the cost involved. In all the histograms constructed using an m in [m_L,m_{NN}], the

histogram having the lowest roughness $\hat{R}$ is likely to be the most visually appealing. The roughness measure for a histogram is defined as [62]:

$$\hat{R} = \sum(\Delta^2 \hat{f}(x))h \tag{4.10}$$

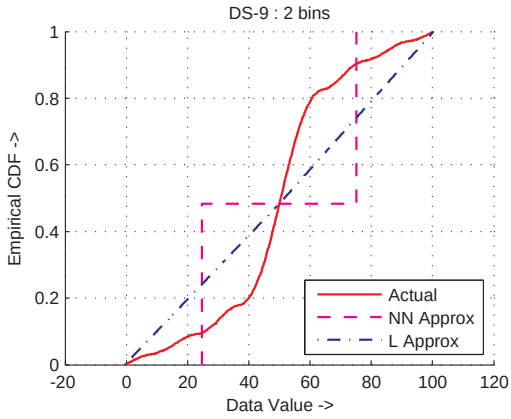
where Δ^2 represents the second order finite difference for $\hat{f}(x)$. Figure 4.5 shows Roughness measures for histograms constructed with m in the corresponding $[m_L,m_{NN}]$ for DS–7 & DS–8.

In summary, to construct a histogram using our new method: (1) Define $M_1 = \{1, 2, \ldots, \sqrt{n}, \frac{n}{\sqrt{n}}, \ldots, \frac{n}{2}, \frac{n}{1}\}$; (2) Construct a histogram for X with m bins for all m in M_1; (3) Construct E_{NN} and E_L for each histogram; (4) Compute m_{NN} and m_L for the E_{NN} and E_L metric curves; (5) Define $M_2 = \{m_L, m_L + 1, \ldots, m_{NN} - 1, m_{NN}\}$; (6) For each m in M_2 construct a histogram for X with m bins; (7) Compute roughness metric $\hat{R}$ for each histogram; (8) Select as the optimal number of bins m_{opt}, the value of m that has the lowest $\hat{R}$.

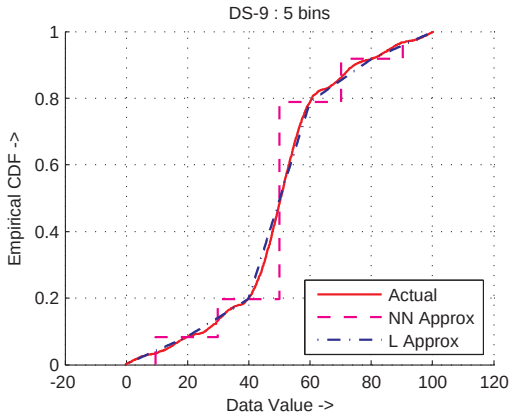
4.4 Experiments & Results

The method explained in Section 4.3 was coded in MATLAB for testing on the datafiles/datasets introduced in Section 4.1. Shimazaki et al. [38] and Knuth [45] provide MATLAB implementations of their methods. Methods due to Sturges [48], Scott [50], and Freedman et al. [52] were also coded in MATLAB. The testing was done on a computer running Windows XP SP3, with an Intel Pentium 4 CPU (3.20 GHz) processor, and having 2.00 GB RAM. All the methods were tested on datafiles DF–1, DF–2, DF–3, and DF–4.

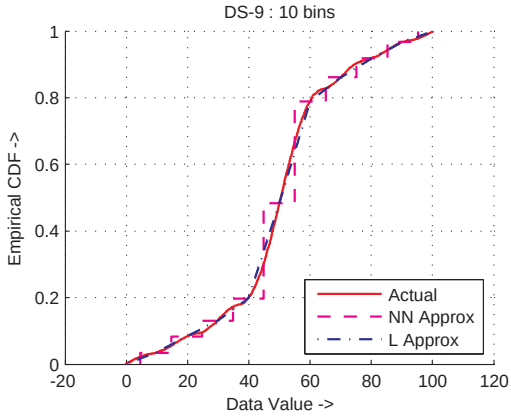
The following abbreviations are used in the tables and figures displaying results: StM – Sturges Method; ScM – Scott Method; FDM – Freedman Diaconis Method; SM – Shimazaki et al. Method; KM – Knuth Method; LHM – Lolla Hoberock Method



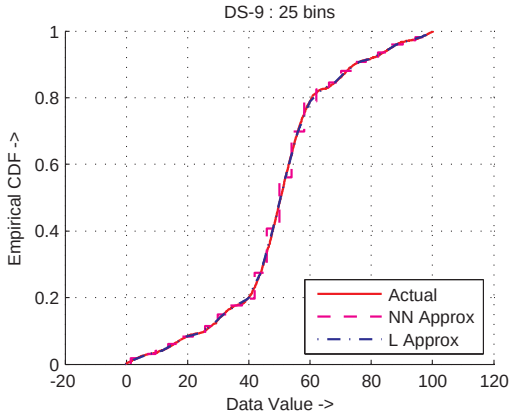
(a) $m = 2$



(b) $m = 5$

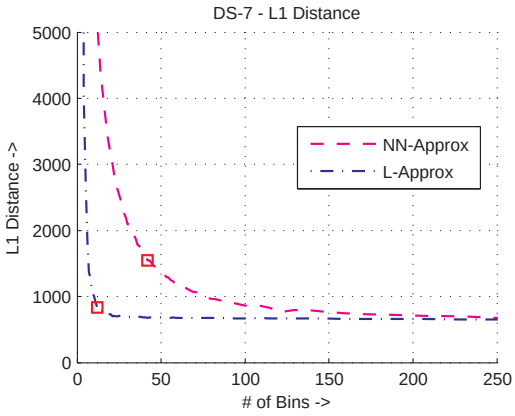


(c) $m = 10$

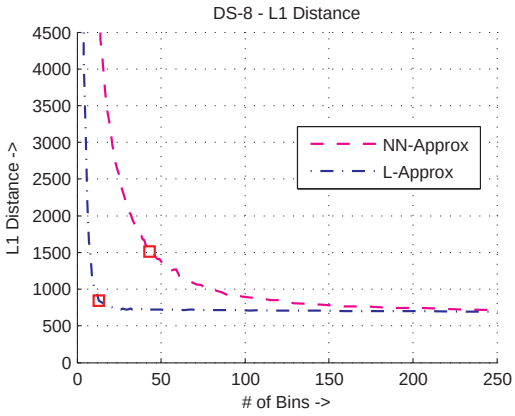


(d) $m = 25$

Figure 4.3: Empirical CDF: Data approximations using $m = 2, 5, 10, 25$ bins for DS–9

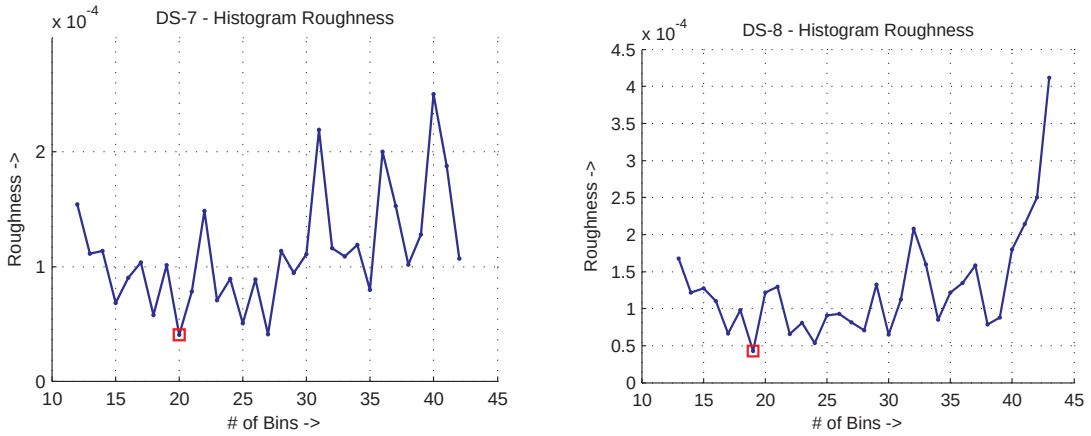


(a) DS-7



(b) DS-8

Figure 4.4: Error Metrics for DS–7 & DS–8



(a) DS-7 (b) DS-8

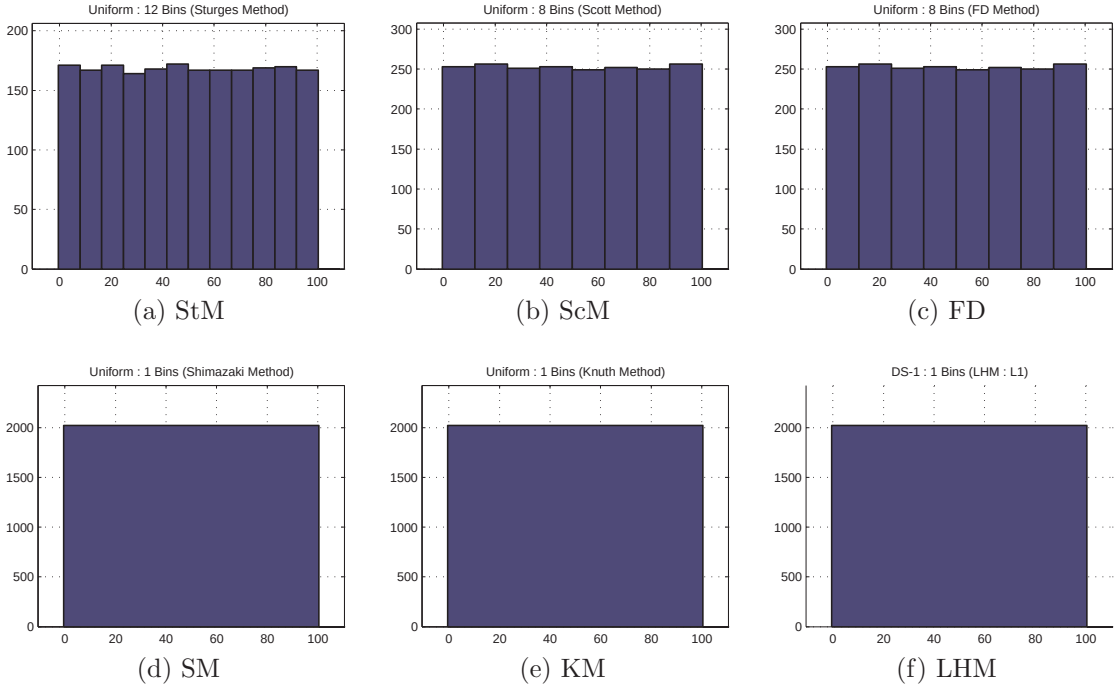
Figure 4.5: Roughness Measures for DS–7 & DS–8

(proposed in this chapter).

In order to measure the performance of the various methods mentioned above, the values of E_{NN}, E_L, and $\hat{R}$ are computed for the histograms generated by each method. It is desirable to have values as low as possible for all three metrics simultaneously. However, low values of $\hat{R}$ tend to result in relatively higher values of E_{NN} and E_L, and vice versa. E_{NN} and E_L indicate a given histogram's fidelity in representing the data, and $\hat{R}$ indicates the degree of over–fitting (or under–fitting) in the representation.

Tables 4.3 and 4.6 document values of m_{opt}, E_{NN}, E_L, and $\hat{R}$ for histograms generated by various methods for each dataset. The maximum values for m_{opt}, and the minimum values for E_{NN}, E_L, and $\hat{R}$ across all the methods are highlighted in blue boldface for easy reading. It can be seen from the tables that the method proposed herein (LHM) produces the lowest values of E_{NN}, E_L, and $\hat{R}$ simultaneously for a vast majority of the cases. This indicates that the proposed method does a better job of capturing shape-related information to a good degree of detail without admitting excessive noise as compared to the other methods.

Figure 4.6 to 4.17 display histograms constructed using various methods for the datasets in datafile DF–3. Visual examination of these plots and comparison to data distribution shapes in Figure 4.2 supports the aforementioned inference. Results for



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

Figure 4.6: Histograms generated for DS–1 (from DF–3) using various methods.

datasets in other datafiles were found to be similar.

The method proposed in this chapter also produces some results that the authors find less satisfying, in which case the shapes of the distributions underlying the population are not as well captured. However, as shown in Figure 4.18 to 4.22, results from the other methods are also less satisfying.

Table 4.3: Results for DF–1 using various methods

DS		StM	ScM	FDM	SM	KM	LHM
DS-1	m_{opt}	10	**13**	**13**	1	1	1
	E_{NN} (x 10^2)	12.75	**9.86**	**9.86**	127.46	127.46	127.46
	E_L (x 10^2)	1.84	**1.75**	**1.75**	1.90	1.90	1.90
	$\hat{R}$ (x 10^{-5})	2.19	16.95	16.95	**0.00**	**0.00**	**0.00**
DS-2	m_{opt}	11	12	12	4	4	**22**
	E_{NN} (x 10^2)	12.68	11.62	11.62	34.81	34.81	**6.43**
	E_L (x 10^2)	2.12	2.18	2.18	2.59	2.59	**1.87**
	$\hat{R}$ (x 10^{-4})	6.13	4.92	4.92	37.06	37.06	**2.20**
DS-3	m_{opt}	11	5	3	8	8	**20**
	E_{NN} (x 10^2)	7.39	15.91	26.28	10.04	10.04	**4.04**
	E_L (x 10^2)	1.97	4.81	9.00	2.30	2.30	**1.73**
	$\hat{R}$ (x 10^{-4})	18.00	142.05	499.85	43.61	43.61	**2.25**
DS-4	m_{opt}	11	8	5	15	11	**28**
	E_{NN} (x 10^2)	15.18	21.87	32.05	11.44	15.18	**6.17**
	E_L (x 10^2)	3.46	7.71	9.43	2.50	3.46	**2.03**
	$\hat{R}$ (x 10^{-3})	9.22	9.80	33.42	5.26	9.22	**1.05**
DS-5	m_{opt}	11	13	12	6	3	**15**
	E_{NN} (x 10^3)	1.53	1.27	1.40	2.77	5.43	**1.12**
	E_L (x 10^2)	2.36	2.07	**2.05**	3.14	9.08	2.10
	$\hat{R}$ (x 10^{-5})	14.59	12.91	19.88	19.04	166.64	**8.16**
DS-6	m_{opt}	11	11	7	14	7	**18**
	E_{NN} (x 10^2)	15.79	15.79	25.09	12.51	25.09	**9.87**
	E_L (x 10^2)	3.26	3.26	6.20	2.46	6.20	**2.33**
	$\hat{R}$ (x 10^{-4})	7.08	7.08	23.04	3.57	23.04	**2.03**
DS-7	m_{opt}	11	**12**	10	7	7	**12**
	E_{NN} (x 10^3)	1.56	**1.39**	1.67	2.41	2.41	**1.39**
	E_L (x 10^2)	2.77	**2.51**	2.62	3.65	3.65	**2.51**
	$\hat{R}$ (x 10^{-4})	1.67	**1.19**	2.69	8.47	8.47	**1.19**
DS-8	m_{opt}	11	11	9	9	6	**19**
	E_{NN} (x 10^2)	15.41	15.41	18.65	18.65	27.79	**8.92**
	E_L (x 10^2)	2.49	2.49	2.95	2.95	5.12	**2.13**
	$\hat{R}$ (x 10^{-4})	2.38	2.38	4.70	4.70	13.78	**1.17**
DS-9	m_{opt}	11	9	4	19	5	**27**
	E_{NN} (x 10^2)	12.61	15.43	40.00	7.30	25.34	**5.43**
	E_L (x 10^2)	4.47	5.05	20.38	2.35	5.54	**2.15**
	$\hat{R}$ (x 10^{-3})	12.56	27.60	6.77	5.83	51.06	**2.32**
DS-10	m_{opt}	11	15	17	**24**	16	23
	E_{NN} (x 10^2)	15.84	11.48	10.18	**7.29**	11.03	7.54
	E_L (x 10^2)	4.37	3.24	2.67	2.25	2.82	**2.17**
	$\hat{R}$ (x 10^{-4})	52.63	30.38	22.56	11.38	24.56	**8.65**
DS-11	m_{opt}	11	11	8	8	7	**20**
	E_{NN} (x 10^2)	12.57	12.57	17.50	17.50	20.02	**6.93**
	E_L (x 10^2)	2.75	2.75	3.76	3.76	4.20	**2.12**
	$\hat{R}$ (x 10^{-4})	15.65	15.65	19.44	19.44	36.89	**4.33**
DS-12	m_{opt}	11	**12**	11	4	4	4
	E_{NN} (x 10^3)	1.26	**1.16**	1.26	3.46	3.46	3.46
	E_L (x 10^2)	**2.25**	2.32	**2.25**	3.06	3.06	3.06
	$\hat{R}$ (x 10^{-6})	2036.87	1678.39	2036.87	**9.75**	**9.75**	**9.75**

Table 4.4: Results for DF–2 using various methods

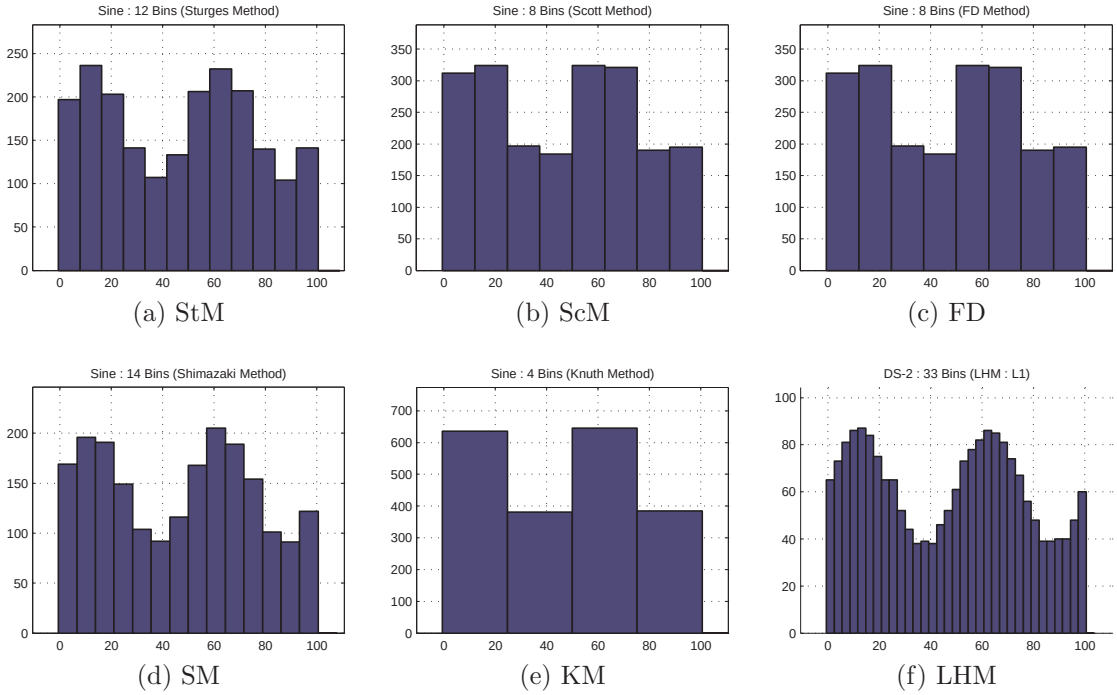
DS		StM	ScM	FDM	SM	KM	LHM
DS-1	m_{opt}	**11**	10	10	1	1	1
	E_{NN} (x 10^3)	**2.34**	2.57	2.57	25.50	25.50	25.50
	E_L (x 10^2)	3.41	3.48	3.48	**3.41**	**3.41**	**3.41**
	$\hat{R}$ (x 10^{-5})	2.58	5.25	5.25	**0.00**	**0.00**	**0.00**
DS-2	m_{opt}	12	10	10	4	4	**28**
	E_{NN} (x 10^2)	22.23	26.39	26.39	66.19	66.19	**9.76**
	E_L (x 10^2)	3.97	4.60	4.60	5.72	5.72	**3.54**
	$\hat{R}$ (x 10^{-4})	6.59	10.03	10.03	41.49	41.49	**1.13**
DS-3	m_{opt}	12	4	3	14	10	**18**
	E_{NN} (x 10^2)	13.41	38.86	51.60	11.44	15.80	**9.13**
	E_L (x 10^2)	4.06	16.06	19.13	3.70	4.27	**3.62**
	$\hat{R}$ (x 10^{-4})	14.10	169.16	514.27	6.76	19.19	**2.94**
DS-4	m_{opt}	12	6	4	21	11	**37**
	E_{NN} (x 10^2)	27.22	56.19	88.13	15.52	28.77	**9.06**
	E_L (x 10^2)	7.22	25.40	55.74	3.97	6.59	**3.75**
	$\hat{R}$ (x 10^{-4})	55.57	156.80	103.28	25.63	99.27	**5.61**
DS-5	m_{opt}	12	10	10	6	6	**18**
	E_{NN} (x 10^3)	2.65	3.11	3.11	5.16	5.16	**1.75**
	E_L (x 10^2)	4.03	4.65	4.65	6.33	6.33	**3.68**
	$\hat{R}$ (x 10^{-5})	5.02	5.56	5.56	22.77	22.77	**4.07**
DS-6	m_{opt}	12	8	5	18	9	**28**
	E_{NN} (x 10^3)	2.69	4.09	6.66	1.81	3.61	**1.17**
	E_L (x 10^2)	5.36	10.56	24.50	**3.75**	7.59	3.83
	$\hat{R}$ (x 10^{-4})	6.01	16.52	50.55	3.25	12.33	**1.00**
DS-7	m_{opt}	12	9	8	13	9	**19**
	E_{NN} (x 10^3)	2.66	3.54	3.96	2.47	3.54	**1.69**
	E_L (x 10^2)	4.78	5.60	6.53	4.31	5.60	**3.82**
	$\hat{R}$ (x 10^{-5})	12.98	36.91	59.32	12.62	36.91	**4.39**
DS-8	m_{opt}	12	9	7	10	10	**19**
	E_{NN} (x 10^3)	2.63	3.49	4.46	3.15	3.15	**1.67**
	E_L (x 10^2)	4.21	5.68	8.19	4.96	4.96	**3.62**
	$\hat{R}$ (x 10^{-5})	20.62	42.13	81.31	28.39	28.39	**4.80**
DS-9	m_{opt}	12	7	3	30	18	**36**
	E_{NN} (x 10^2)	22.30	41.00	69.76	9.32	14.99	**7.76**
	E_L (x 10^2)	6.92	8.41	29.66	3.84	4.67	**3.77**
	$\hat{R}$ (x 10^{-3})	17.98	42.55	30.02	2.04	7.68	**1.27**
DS-10	m_{opt}	12	12	14	**47**	17	30
	E_{NN} (x 10^2)	27.27	27.27	23.49	**7.26**	19.30	11.01
	E_L (x 10^2)	7.98	7.98	6.30	**3.59**	5.05	3.80
	$\hat{R}$ (x 10^{-4})	53.64	53.64	40.98	6.88	24.53	**5.71**
DS-11	m_{opt}	12	9	6	16	13	**25**
	E_{NN} (x 10^3)	2.23	3.00	4.46	1.66	2.08	**1.10**
	E_L (x 10^2)	4.85	6.83	13.73	4.19	4.43	**3.66**
	$\hat{R}$ (x 10^{-4})	14.30	14.98	16.36	7.15	12.02	**3.77**
DS-12	m_{opt}	12	9	9	8	4	**24**
	E_{NN} (x 10^3)	2.22	2.95	2.95	3.32	6.61	**1.15**
	E_L (x 10^2)	4.52	5.59	5.59	4.31	6.68	**3.77**
	$\hat{R}$ (x 10^{-6})	1949.62	1609.92	1609.92	4123.74	**4.91**	413.81

Table 4.5: Results for DF–3 using various methods

DS		StM	ScM	FDM	SM	KM	LHM
DS-1	m_{opt}	**12**	8	8	1	1	1
	E_{NN} (x 10^3)	**4.29**	6.40	6.40	51.00	51.00	51.00
	E_L (x 10^2)	6.65	6.74	6.74	**6.60**	**6.60**	**6.60**
	$\hat{R}$ (x 10^{-6})	12.62	5.58	5.58	**0.00**	**0.00**	**0.00**
DS-2	m_{opt}	12	8	8	14	4	**33**
	E_{NN} (x 10^3)	4.33	6.48	6.48	3.72	12.94	**1.68**
	E_L (x 10^2)	7.69	10.57	10.57	7.41	10.68	**6.95**
	$\hat{R}$ (x 10^{-5})	72.26	209.96	209.96	42.42	514.19	**6.35**
DS-3	m_{opt}	12	3	2	17	12	**31**
	E_{NN} (x 10^3)	2.65	10.19	16.79	1.91	2.65	**1.20**
	E_L (x 10^2)	7.64	37.73	144.58	7.14	7.64	**6.99**
	$\hat{R}$ (x 10^{-4})	12.04	552.52	**0.00**	4.65	12.04	1.77
DS-4	m_{opt}	13	5	3	25	17	**48**
	E_{NN} (x 10^3)	4.74	11.53	17.14	2.54	3.65	**1.40**
	E_L (x 10^2)	10.98	39.56	78.06	7.40	8.57	**7.07**
	$\hat{R}$ (x 10^{-4})	73.21	410.54	470.35	18.01	39.07	**2.82**
DS-5	m_{opt}	13	8	8	11	7	**17**
	E_{NN} (x 10^3)	4.75	7.64	7.64	5.60	8.66	**3.60**
	E_L (x 10^2)	8.02	10.86	10.86	8.31	11.48	**7.24**
	$\hat{R}$ (x 10^{-5})	6.38	12.81	12.81	7.21	16.79	**2.34**
DS-6	m_{opt}	13	6	4	23	17	**30**
	E_{NN} (x 10^3)	4.78	10.48	16.10	2.77	3.68	**2.05**
	E_L (x 10^2)	9.44	34.01	72.55	7.24	7.61	**7.01**
	$\hat{R}$ (x 10^{-5})	46.38	354.75	737.01	12.10	27.92	**3.48**
DS-7	m_{opt}	13	7	6	14	12	**20**
	E_{NN} (x 10^3)	4.78	8.80	10.22	4.41	5.12	**3.11**
	E_L (x 10^2)	8.16	13.66	18.73	7.56	8.34	**7.35**
	$\hat{R}$ (x 10^{-5})	11.14	99.50	155.56	11.34	15.44	**4.04**
DS-8	m_{opt}	13	7	6	13	13	**19**
	E_{NN} (x 10^3)	4.75	8.69	10.18	4.75	4.75	**3.28**
	E_L (x 10^2)	8.41	16.53	21.76	8.41	8.41	**7.55**
	$\hat{R}$ (x 10^{-5})	16.78	90.27	141.93	16.78	16.78	**4.30**
DS-9	m_{opt}	12	5	2	38	18	**47**
	E_{NN} (x 10^3)	4.37	9.32	33.70	1.50	2.93	**1.23**
	E_L (x 10^2)	13.95	20.20	236.71	7.05	8.96	**6.96**
	$\hat{R}$ (x 10^{-4})	190.43	626.38	**0.00**	13.42	82.32	6.93
DS-10	m_{opt}	13	10	11	**45**	23	37
	E_{NN} (x 10^3)	4.88	6.41	5.79	**1.52**	2.78	1.77
	E_L (x 10^2)	14.06	20.45	17.42	**7.00**	7.90	7.06
	$\hat{R}$ (x 10^{-4})	49.83	73.80	64.99	4.19	14.59	**3.25**
DS-11	m_{opt}	13	7	5	13	13	**34**
	E_{NN} (x 10^3)	4.04	7.55	10.44	4.04	4.04	**1.61**
	E_L (x 10^2)	8.51	14.55	27.88	8.51	8.51	**7.17**
	$\hat{R}$ (x 10^{-4})	12.27	37.72	35.01	12.27	12.27	**2.17**
DS-12	m_{opt}	13	7	7	8	4	**31**
	E_{NN} (x 10^3)	4.01	7.40	7.40	6.49	12.88	**1.76**
	E_L (x 10^2)	8.35	14.10	14.10	8.38	12.50	**6.93**
	$\hat{R}$ (x 10^{-6})	1710.36	775.42	775.42	3847.28	**2.82**	208.35

Table 4.6: Results for DF–4 using various methods

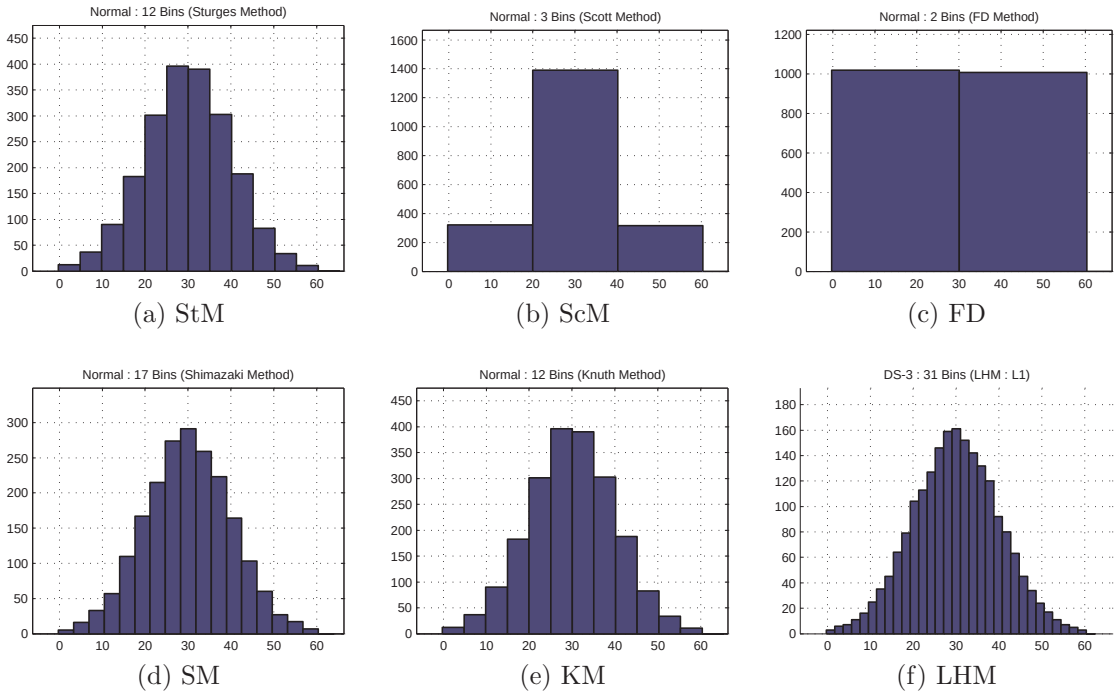
DS		StM	ScM	FDM	SM	KM	LHM
DS-1	m_{opt}	**14**	6	6	1	1	1
	E_{NN} (x 10^3)	**9.24**	21.29	21.29	127.53	127.53	127.53
	E_L (x 10^3)	1.72	1.72	1.72	**1.72**	**1.72**	**1.72**
	$\hat{R}$ (x 10^{-6})	9.75	5.31	5.31	**0.00**	**0.00**	**0.00**
DS-2	m_{opt}	14	6	6	18	4	**30**
	E_{NN} (x 10^3)	9.24	21.28	21.28	7.22	31.89	**4.50**
	E_L (x 10^3)	1.84	3.76	3.76	1.73	2.70	**1.69**
	$\hat{R}$ (x 10^{-5})	39.66	256.35	256.35	18.99	503.03	**5.20**
DS-3	m_{opt}	14	2	2	22	19	**61**
	E_{NN} (x 10^3)	5.68	41.89	41.89	3.74	4.29	**1.35**
	E_L (x 10^3)	1.82	36.47	36.47	1.69	1.70	**1.66**
	$\hat{R}$ (x 10^{-5})	67.85	**0.00**	**0.00**	15.97	22.60	5.60
DS-4	m_{opt}	14	3	2	54	21	**63**
	E_{NN} (x 10^3)	11.13	42.14	99.51	3.20	7.40	**2.88**
	E_L (x 10^3)	2.84	19.42	90.85	**1.70**	1.90	1.70
	$\hat{R}$ (x 10^{-4})	49.05	490.78	**0.00**	2.59	25.14	1.84
DS-5	m_{opt}	14	6	6	15	12	**25**
	E_{NN} (x 10^3)	10.82	24.93	24.93	10.12	12.59	**6.17**
	E_L (x 10^3)	1.86	3.31	3.31	1.80	1.91	**1.70**
	$\hat{R}$ (x 10^{-5})	4.52	23.42	23.42	3.94	4.85	**1.66**
DS-6	m_{opt}	14	4	3	28	17	**40**
	E_{NN} (x 10^3)	11.01	39.73	54.54	5.64	9.03	**3.88**
	E_L (x 10^3)	2.21	18.23	31.41	1.74	1.91	**1.72**
	$\hat{R}$ (x 10^{-5})	35.02	750.14	1023.42	9.27	20.94	**2.58**
DS-7	m_{opt}	14	5	5	21	16	**28**
	E_{NN} (x 10^3)	10.89	29.98	29.98	7.34	9.54	**5.57**
	E_L (x 10^3)	1.94	6.85	6.85	1.72	1.84	**1.72**
	$\hat{R}$ (x 10^{-5})	8.72	240.39	240.39	2.63	5.40	**2.43**
DS-8	m_{opt}	14	5	4	15	15	**33**
	E_{NN} (x 10^3)	10.79	29.97	36.79	10.12	10.12	**4.78**
	E_L (x 10^3)	1.95	7.89	10.33	1.90	1.90	**1.72**
	$\hat{R}$ (x 10^{-5})	11.66	235.58	509.14	10.42	10.42	**2.70**
DS-9	m_{opt}	14	4	2	51	30	**67**
	E_{NN} (x 10^3)	9.12	37.21	83.41	3.05	4.53	**2.48**
	E_L (x 10^3)	3.26	20.41	59.32	1.70	1.80	**1.69**
	$\hat{R}$ (x 10^{-4})	124.90	79.24	**0.00**	7.60	21.94	2.89
DS-10	m_{opt}	14	7	8	50	30	**59**
	E_{NN} (x 10^3)	11.14	22.96	19.91	3.42	5.32	**2.94**
	E_L (x 10^3)	3.13	9.23	7.15	1.73	1.83	**1.73**
	$\hat{R}$ (x 10^{-4})	44.75	90.91	92.32	2.16	6.00	**1.74**
DS-11	m_{opt}	14	5	3	26	17	**44**
	E_{NN} (x 10^3)	9.27	25.71	45.70	5.09	7.66	**3.27**
	E_L (x 10^3)	2.27	7.01	14.85	1.76	1.95	**1.71**
	$\hat{R}$ (x 10^{-5})	84.47	372.90	636.10	31.79	70.25	**8.79**
DS-12	m_{opt}	14	6	5	20	8	**43**
	E_{NN} (x 10^3)	9.20	21.31	25.56	6.54	15.97	**3.35**
	E_L (x 10^3)	2.01	3.99	4.98	1.75	2.08	**1.69**
	$\hat{R}$ (x 10^{-5})	145.12	30.04	**3.40**	69.38	396.55	8.77



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

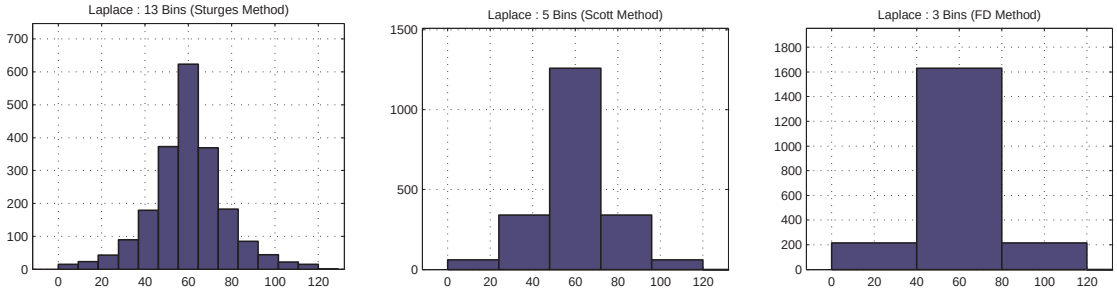
Figure 4.7: Histograms generated for DS–2 (from DF–3) using various methods.



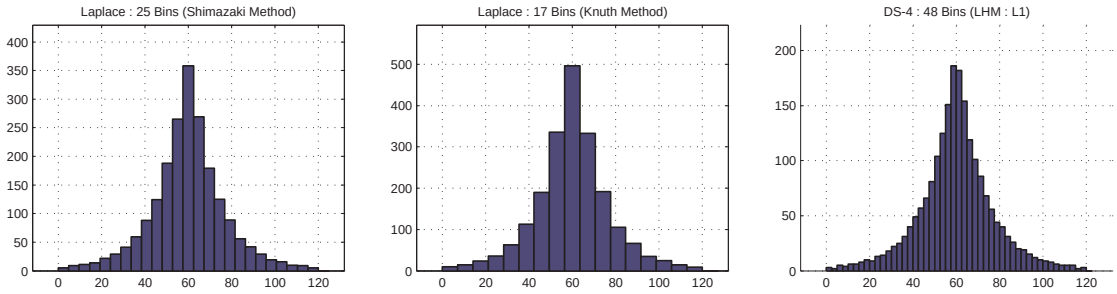
(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

Figure 4.8: Histograms generated for DS–3 (from DF–3) using various methods.

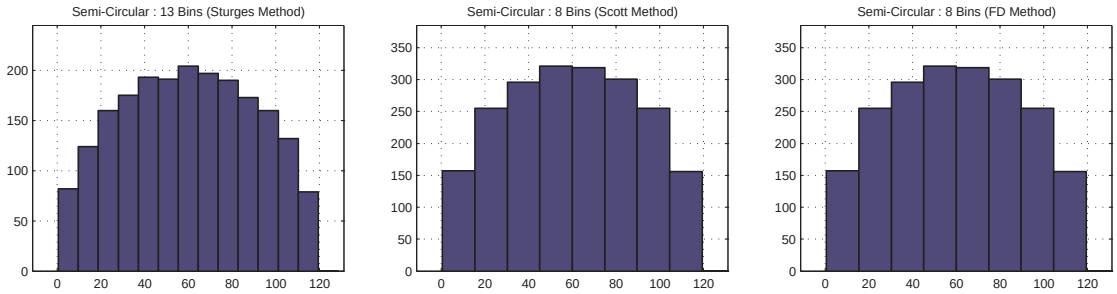


(a) StM (b) ScM (c) FD

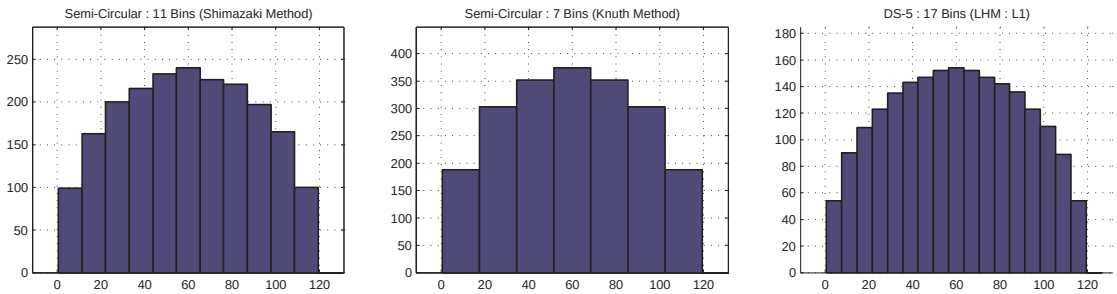


(d) SM (e) KM (f) LHM

Figure 4.9: Histograms generated for DS–4 (from DF–3) using various methods.

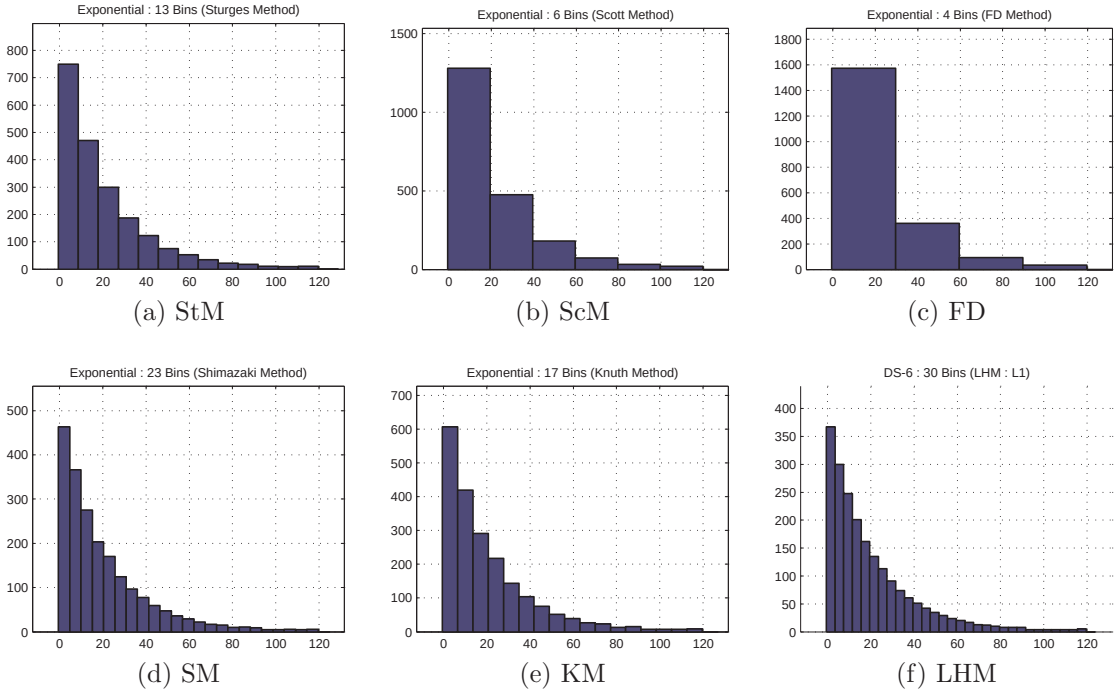


(a) StM (b) ScM (c) FD



(d) SM (e) KM (f) LHM

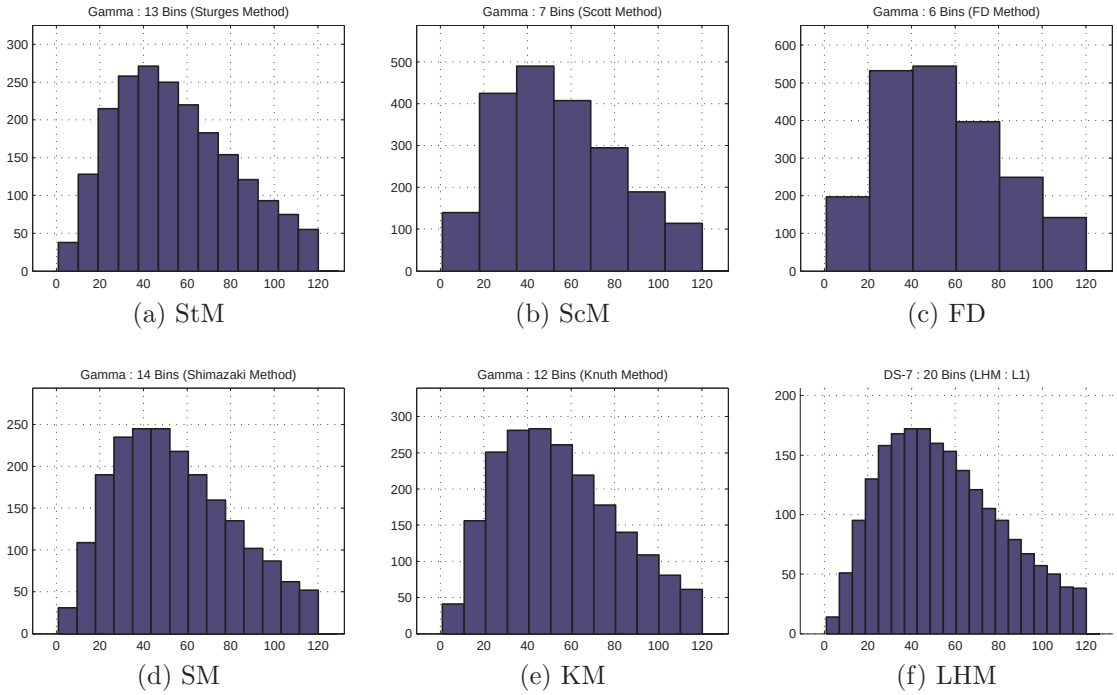
Figure 4.10: Histograms generated for DS–5 (from DF–3) using various methods.



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

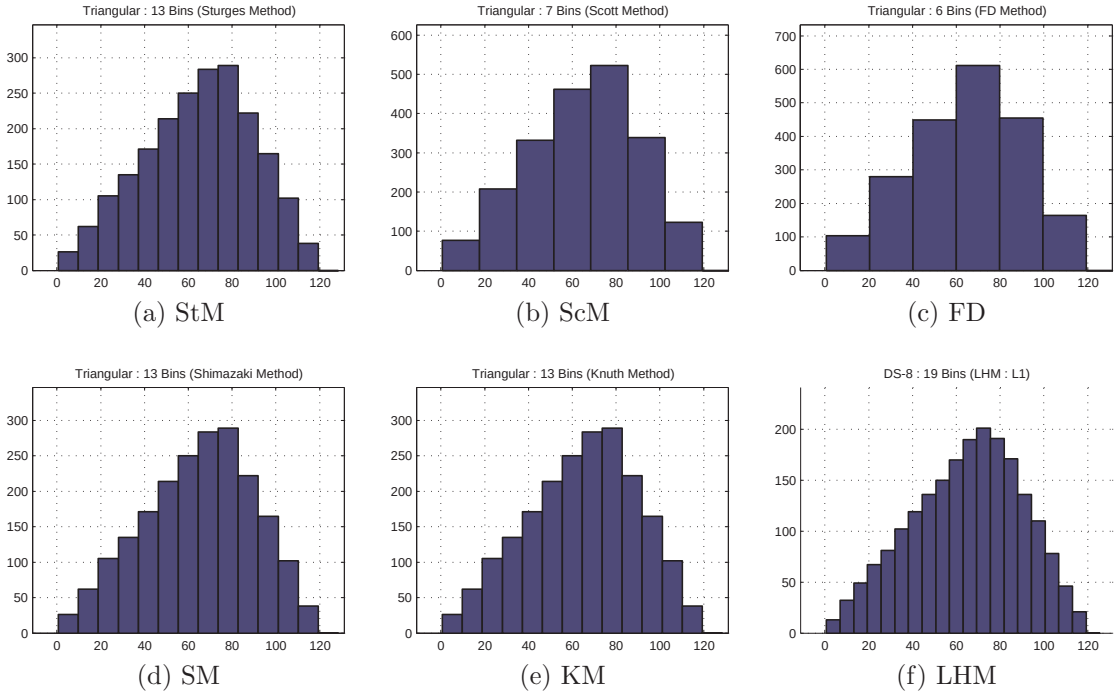
Figure 4.11: Histograms generated for DS–6 (from DF–3) using various methods.



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

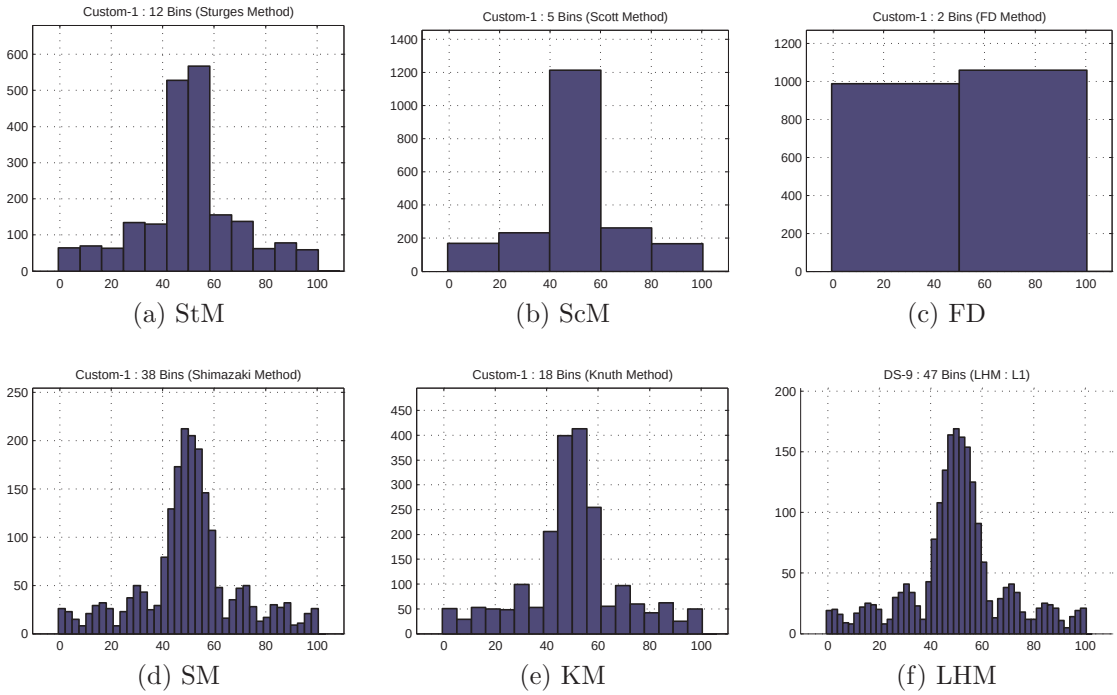
Figure 4.12: Histograms generated for DS–7 (from DF–3) using various methods.



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

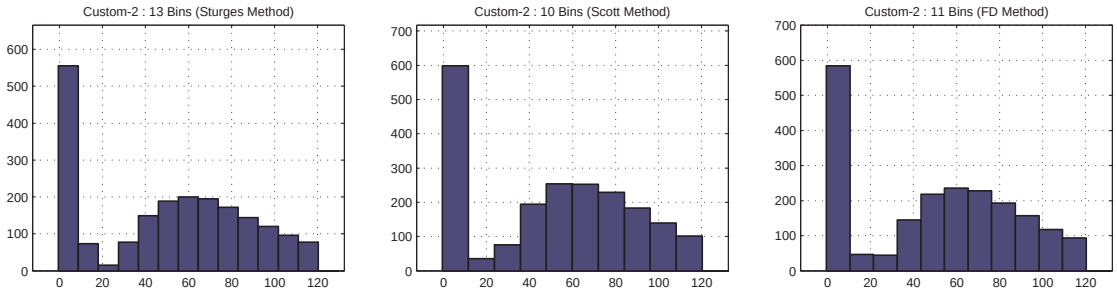
Figure 4.13: Histograms generated for DS–8 (from DF–3) using various methods.



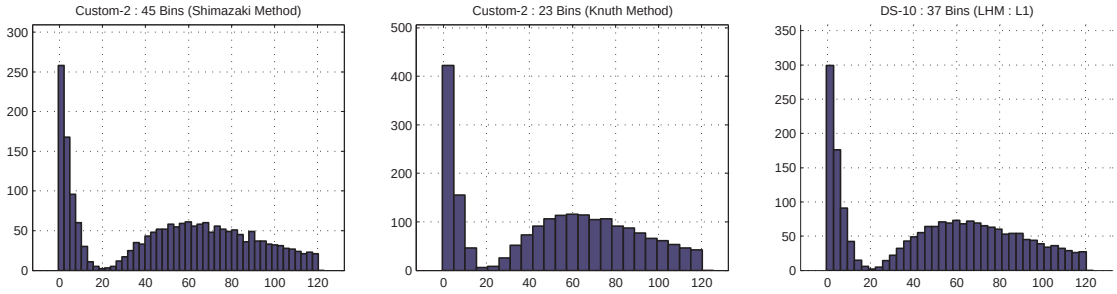
(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

Figure 4.14: Histograms generated for DS–9 (from DF–3) using various methods.

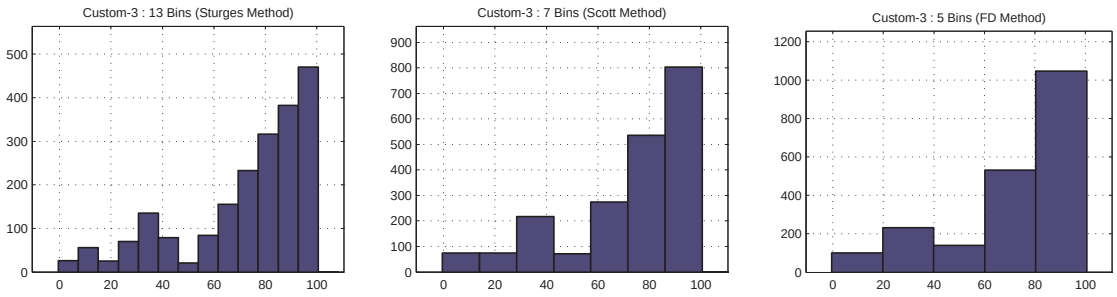


(a) StM (b) ScM (c) FD

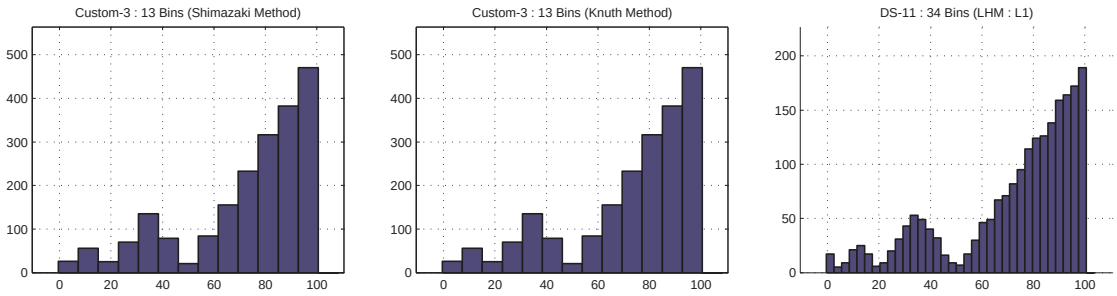


(d) SM (e) KM (f) LHM

Figure 4.15: Histograms generated for DS–10 (from DF–3) using various methods.



(a) StM (b) ScM (c) FD



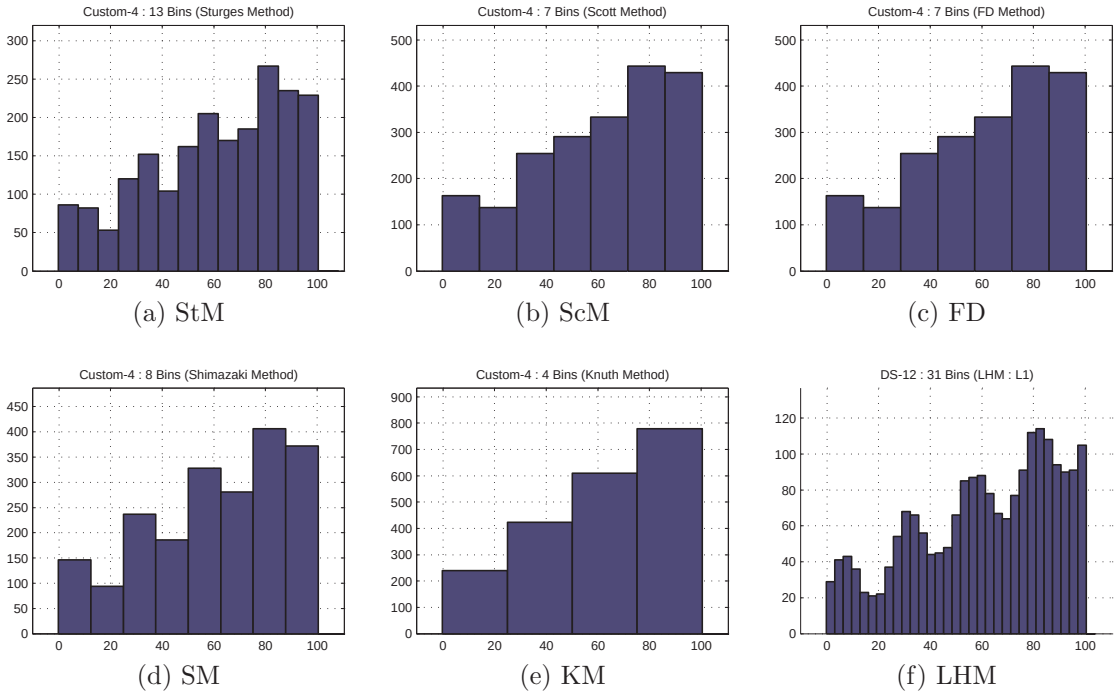
(d) SM (e) KM (f) LHM

Figure 4.16: Histograms generated for DS–11 (from DF–3) using various methods.

4.5 Conclusions

This chapter introduces a new method for selecting the number of bins for constructing a histogram for a given dataset. The performance of the proposed method is compared with the performance of five other methods in the literature. Comparison results show that the proposed method performs better than the other five methods, with the proposed method producing visually appealing histograms that reveal shape features of underlying distribution to a finer detail without admitting excessive noise.

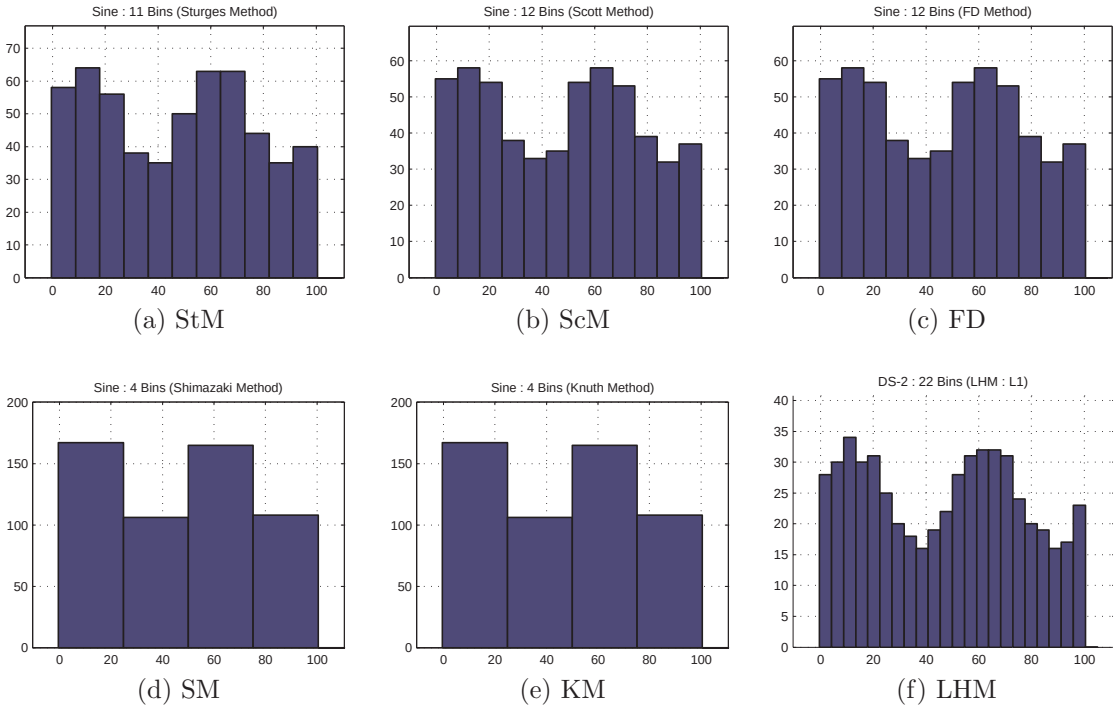
We suggest that future investigations should explore the following issues: (1) Designing a metric to measure the performance of a histogram as evaluated by human perception; (2) Extension of ideas proposed herein to higher dimensional data; and (3) Optimizing the proposed method to reduce time and memory requirements.



(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

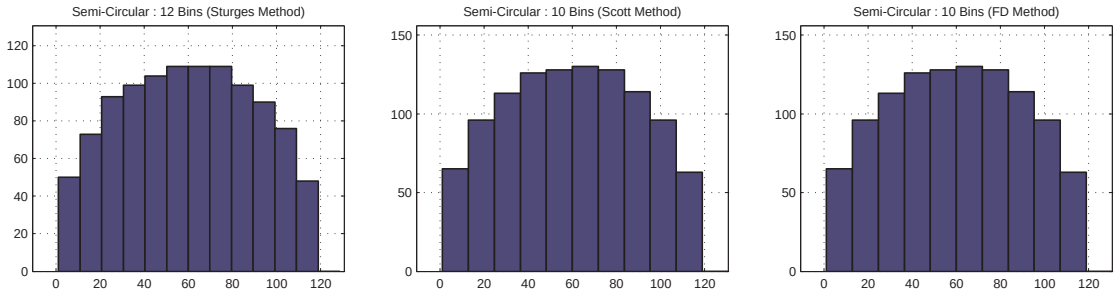
Figure 4.17: Histograms generated for DS–12 (from DF–3) using various methods.



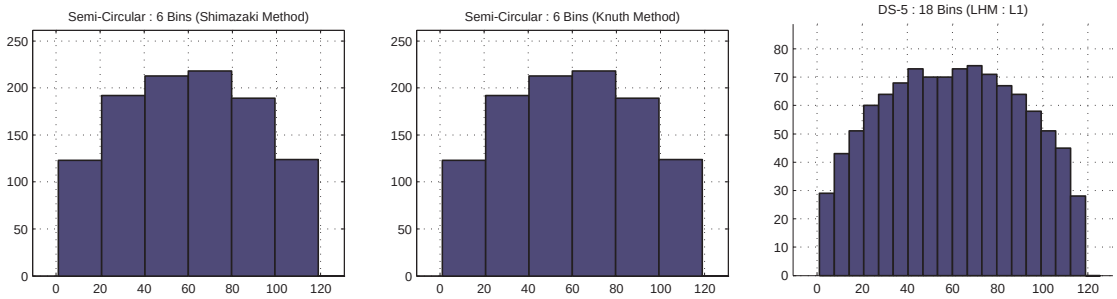
(a) StM (b) ScM (c) FD

(d) SM (e) KM (f) LHM

Figure 4.18: Less Satisfying Result (LHM): Undesirable spike on left mode (DS–2, DF–1).

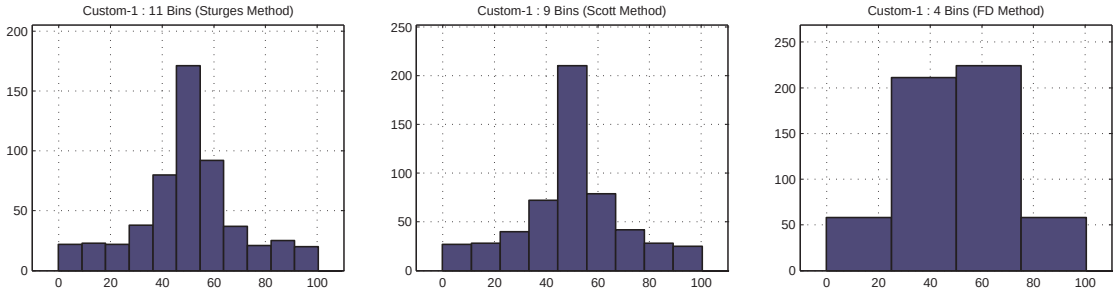


(a) StM (b) ScM (c) FD

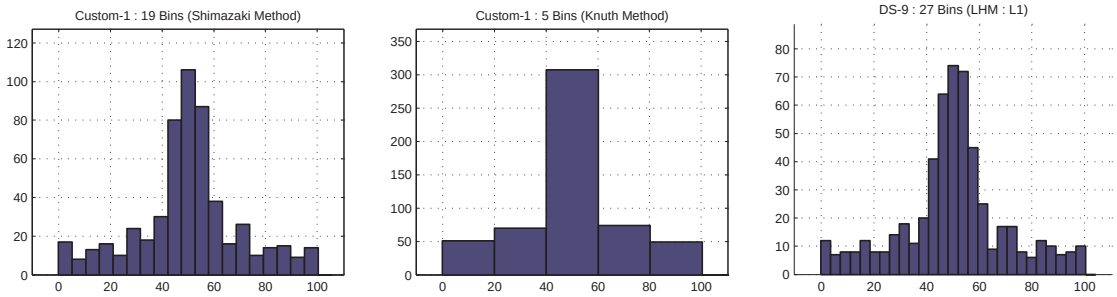


(d) SM (e) KM (f) LHM

Figure 4.19: Less Satisfying Result (LHM): Histogram could be "smoother" (DS–5, DF–2).

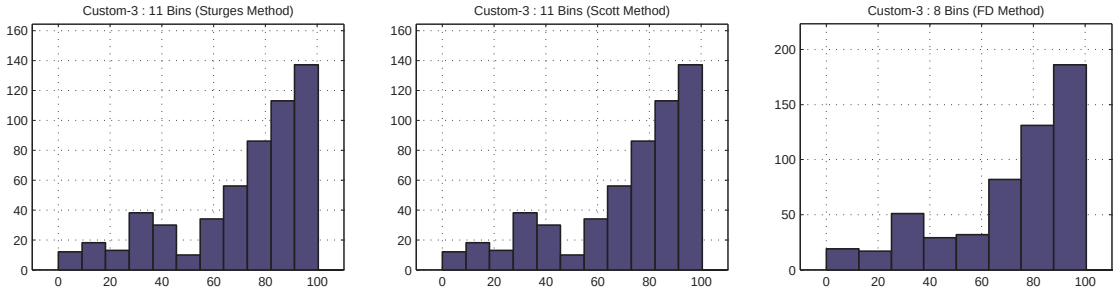


(a) StM (b) ScM (c) FD

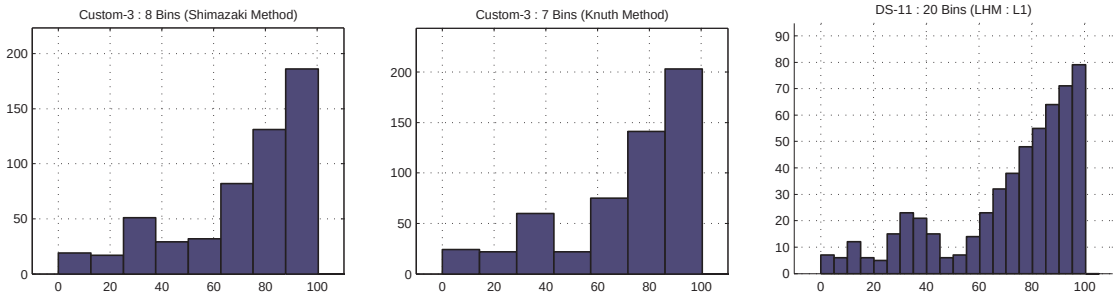


(d) SM (e) KM (f) LHM

Figure 4.20: Less Satisfying Result (LHM): Shape not captured "well" (DS–9, DF–1).

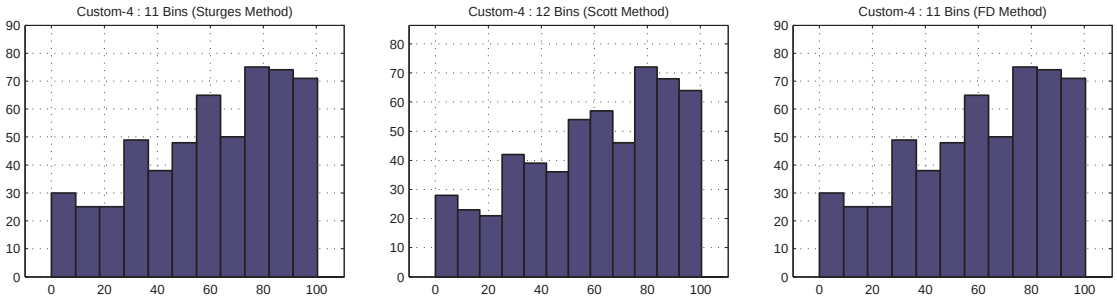


(a) StM (b) ScM (c) FD

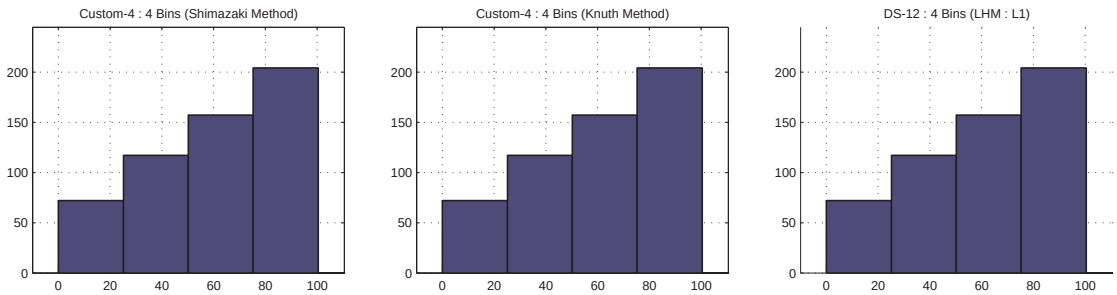


(d) SM (e) KM (f) LHM

Figure 4.21: Less Satisfying Result (LHM): Number of modes do not match original shape (DS–11, DF–1).



(a) StM (b) ScM (c) FD



(d) SM (e) KM (f) LHM

Figure 4.22: Less Satisfying Result (LHM): Shape not captured "well" (DS–12, DF–1).

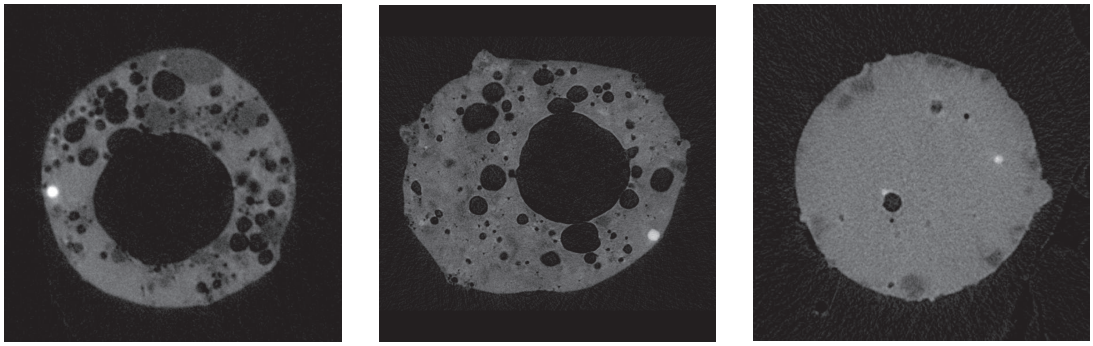
CHAPTER 5

Improved Unsupervised Clustering – An Alternate Implementation

5.1 Introduction

A new Clustering algorithm was described in Section 2.4. The performance of the Clustering algorithm was described in Section 2.5. Encouraged by some of the positive results obtained, the task of applying the Clustering algorithm to a particular application was taken up.

The selected application involved segmentation of images of fly-ash particles acquired using a micro Computed Tomography (μCT) imaging device (images supplied by Dr.Jay Hanan, Associate Professor, School of Mechanical & Aerospace Engineering, Oklahoma State University). Each particle produces a set of grayscale images, each image representing a particular slice of the particle. The desired segmentation operation should identify, group, and label regions of a given image based on "similarity" as perceived by human observers. While a feature set that will facilitate the measurement of such similarity is not readily available, it is most likely that a feature set based on pixel intensity, position, and texture values might suffice. The results of segmentation would then be used to estimate the chemical composition of the fly-ash particle based on comparison – after registration – with an image generated by a Scanning Electron Microscope (SEM) equipped with an Energy Dispersive Spectrometer (EDS). Figure 5.1 displays sample slice images of three fly-ash particles (Particle–1, Particle–2 & Particle–3). It can be seen that the μCT imaging process produces images with regions of different grayscale and texture. These different regions correspond to different chemical compositions (phases) present in the particle.

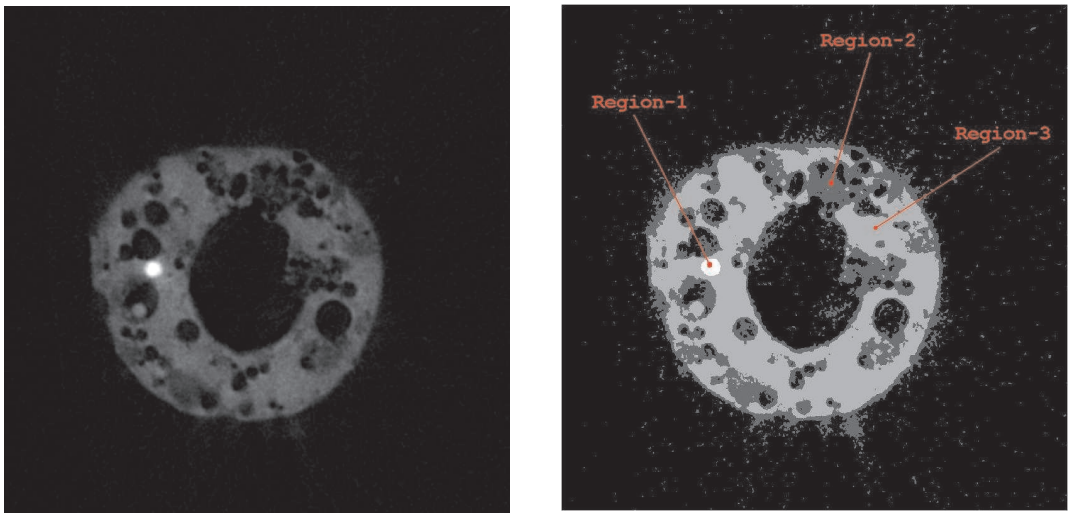


(a) Particle–1 (b) Particle–2 (c) Particle–3

Figure 5.1: Sample images of slices of three fly-ash particles

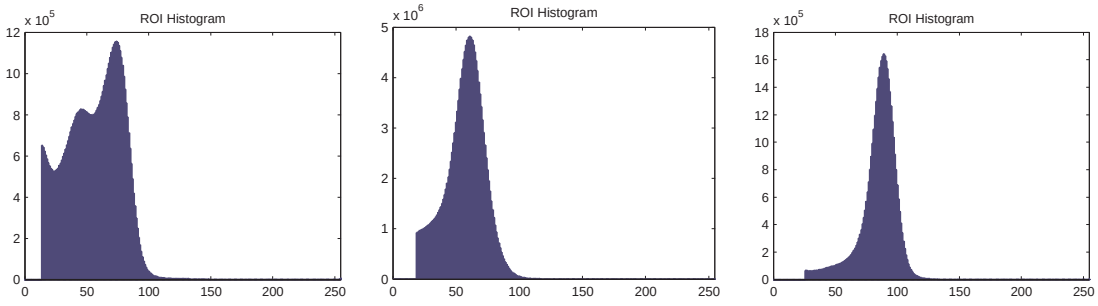
It was desired that: (1) the number of phases present in a given image be detected; and (2) each pixel in the image be associated with a particular phase. The Clustering algorithm described in Section 2.4 was employed on the images of Particle–1. The grayscale values of the pixels in the image were supplied as input to the Clustering algorithm. For Particle–1, the results were encouraging (Figure 5.2). However, for Particle–2 and Particle–3, the Clustering algorithm did not correctly detect the number of phases present in the particle. This could be explained, upon examining the histograms of the grayscale values of pixels in the Region-of-Interest (ROI) for both those particles. It can be seen from Figure 5.3 that while the ROI histogram for Particle–1 does clearly display several peaks corresponding to the various phases, such correspondence cannot be seen in the ROI histograms for Particle–2 and Particle–3. For these particles, the single–peak shape of the histogram is due to the particle having predominantly large proportions of one phase as compared to other phases. In other words, grayscale values of the pixels alone are not sufficient to determine the phase associated with a particular pixel in Particle–2 and Particle–3.

Visual examination of the images suggested that a texture-based feature set might probably separate the phases better. However, attempting to employ the Clustering algorithm on Gabor filter derived texture data (FS–1) ran into computational diffi-



(a) Original Image (b) Segmented Image (3 Regions)

Figure 5.2: Original & Segmented slice images – Particle–1



(a) Particle–1 (b) Particle–2 (c) Particle–3

Figure 5.3: ROI Histograms for "Particle–1", Particle–2" & "Particle–3"

culties. Construction of the modified matrix representation $\mathcal{M}'$ for the texture data required impractical amounts of memory. Thus data with another feature set (FS–2) consisting of intensity descriptors such as medial gray-level, minimum gray-level, maximum gray-level, and first 4 moments of intensity histogram for each (32 x 32) block in the images was chosen as input. However, this data also required very large amounts of memory to construct a matrix representation. Intending to overcome this problem, an alternate implementation of the Clustering algorithm was developed.

The alternate implementation is described in Section 5.2. A performance comparison of the two implementations for the datasets described in Section 2.2 is provided in Section 5.3. A significant drawback of the Clustering algorithm presented in Section 2.4 was discovered during experimentation, and is described in Section 5.4.

5.2 An Alternate Implementation

The Clustering algorithm implementation described in Section 2.4 and Section 2.5 operates by constructing several weighted density landscapes (at several scales) and then selecting an appropriate landscape (scale) based upon maximizing a particular metric (Section 2.4.4). Once this landscape is created, further processing is required to locate local maxima. This can be done using either a *Regional Maxima Finding* algorithm or a *Watershed* algorithm. Both these variants were used and the results were documented in Section 2.5.

This aforementioned implementation requires the construction of $\mathcal{M}'$ – a matrix representation for the dataset – based on two factors, cp and SF. In order to overcome the exorbitant memory requirement for the construction of $\mathcal{M}'$, an alternate implementation was developed. The alternate implementation addresses only the selection of an appropriate landscape. Once the alternate implementation selects an appropriate landscape, further processing required to locate local maxima – as in the case of the previous implementation – can be performed using either a *Regional*

Maxima Finding algorithm or a *Watershed* algorithm. The alternate implementation is described as follows.

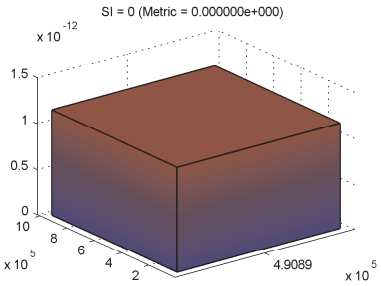
Let $\mathbf{Y} = \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_N}\}$ represent the dataset where each observation is $\mathbf{y_i} = \{y_{i,1}, y_{i,2}, \ldots, y_{i,D}\}$. The space spanned by the data can be represented by the hyper-cuboid with diagonal points $P1$ and $P2$, defined as follows:

$$\begin{aligned} P1 &= \{\min_n y_{n,1}, \min_n y_{n,2}, \ldots, \min_n y_{n,D}\} \\ P2 &= \{\max_n y_{n,1}, \max_n y_{n,2}, \ldots, \max_n y_{n,D}\} \end{aligned} \tag{5.1}$$

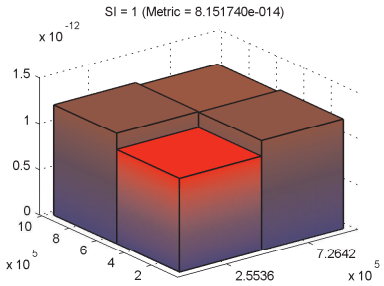
Since histograms can be treated as non–parametric density estimators [42], histograms can be constructed for the data using various bin–widths and then be interpreted as density landscapes. A histogram could be constructed using D–dimensional bins for the data using bin–widths $\mathbf{h} = \{h_1, h_2, \ldots, h_D\}$. Instead of using bin-widths $\mathbf{h}$ to describe the histogram construction, a vector $\mathbf{m} = \{m_1, m_2, \ldots, m_D\}$ could be used equivalently to describe the number of bins used in each dimension to construct the histogram. In order to treat all dimensions equally (because there is no *a priori* information to do otherwise), the dataset ranges can be normalized and then $m_i = m$ where $(i = 1, 2 \ldots D)$ could be used instead. Let $C_{\{i_1, i_2, \ldots, i_D\}}$ represent the bin count of the bin with index $\{i_1, i_2, \ldots, i_D\}$.

The choice of m governs the degree of details captured in the histogram as follows: (1) using a very small value for m will result in a coarse histogram and loss of structural detail; (2) using a very large value for m will result in a noisy histogram and capturing of excessive detail. Figure 5.4 displays several histograms created for dataset S1 (introduced in Section 2.2) using various values for m. For this implementation, the *scale index (SI)* is related to m by:

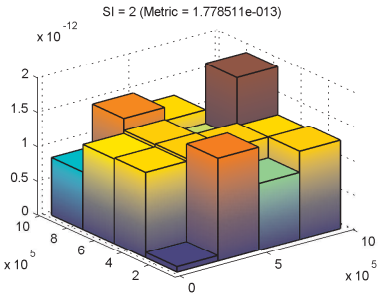
$$m = 2^{SI} \tag{5.2}$$



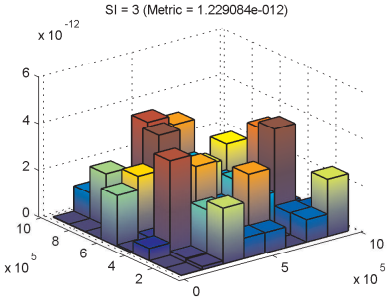
(a) SI = 0, m = 1



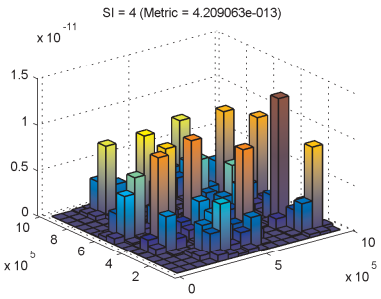
(b) SI = 1, m = 2



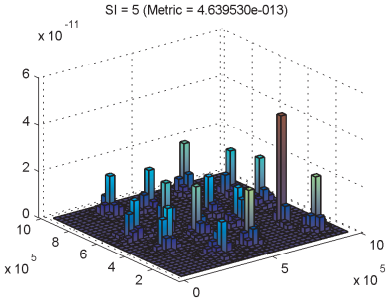
(c) SI = 2, m = 4



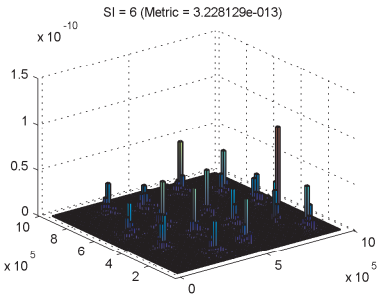
(d) SI = 3, m = 8



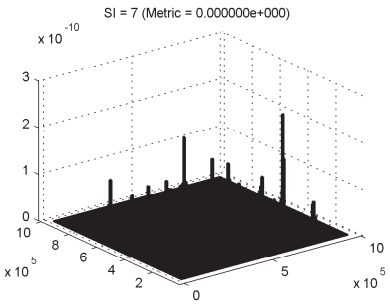
(e) SI = 4, m = 16



(f) SI = 5, m = 32



(g) SI = 6, m = 64



(h) SI = 7, m = 128

Figure 5.4: Histograms constructed using various values of SI

In order to select the landscape to be used for detecting clusters, the variation index VI_Z described in Section 2.4.4 can be used. The variation index VI_Z, is defined by:

$$VI_Z = SD(\,TF(\mathcal{Z})) \tag{5.3}$$

where SD represents the *Standard Deviation* function, and TF represents the *Tukey Outlier Filter* function. $\mathcal{Z}$ represents a set of probability density function values derived from the histogram bin–counts ($C_{\{i_1,i_2,\ldots,i_D\}}$) for each histogram as follows:

$$\begin{aligned} \mathcal{Z} &= \{\frac{C_{\{\ldots\}}}{N \cdot bv}\} \\ where \quad bv &= \frac{tv}{m^D} \\ and \quad tv &= \prod_{i=1}^{D} (\max_n y_{n,i} - \min_n y_{n,i}) \end{aligned} \tag{5.4}$$

This transformation ensures that the values of $\mathcal{Z}$ constitute a valid probability density function (PDF), and facilitates a meaningful comparison of the variation indices of various histograms.

The values of $\mathcal{Z}$ thus produced can be used to compute the variation index (VI_Z) in order to select the appropriate landscape. The variation index graph obtained (Figure 5.5) has a peak similar to the one obtained in the previous implementation (Figure 2.9). Let m_{opt} be the number of bins used to construct the histogram (H_{opt}) whose variation index (VI_Z) is maximized. This value is determined through an iterative search procedure.

The bin–width (h_{opt}) used to construct H_{opt} is representative of the scale determined to be most appropriate by the Clustering algorithm. At this scale, any structure smaller than h_{opt} will be suppressed. The next step towards detecting the underlying cluster structure is constructing an optimal landscape $\mathcal{LS}_{opt}$ and then subjecting it to further processing – *Median Filtering* followed by *Regional Maxima Finding* as described in Section 2.4.5.

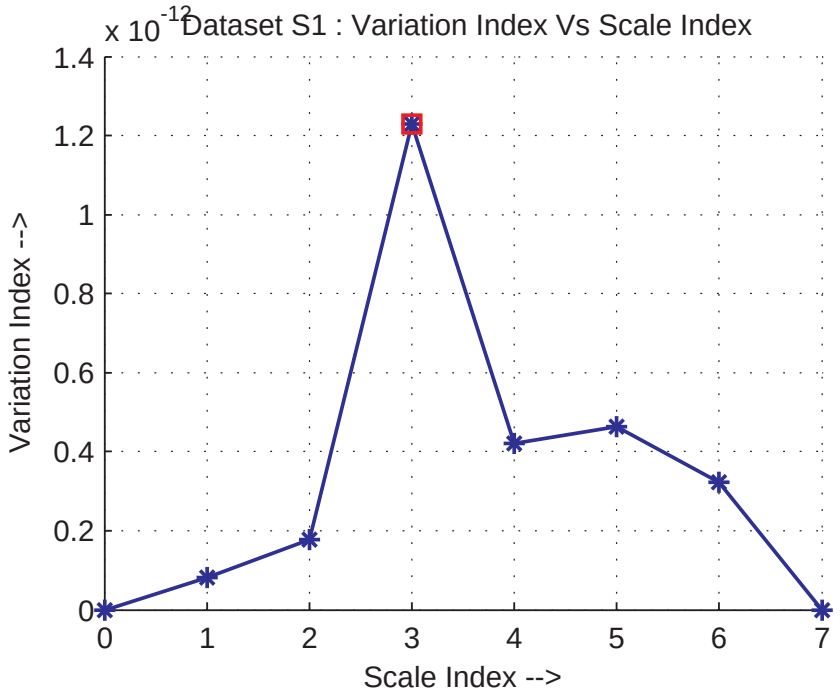


Figure 5.5: Variation Indices for Histograms using various SI - Dataset S1

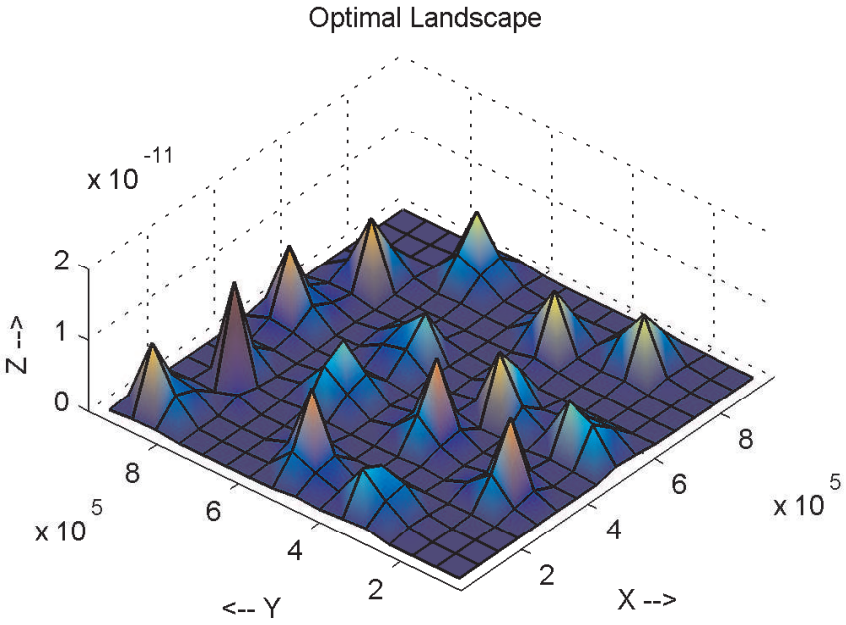


Figure 5.6: Optimal landscape – Dataset S1 (Histogram–based implementation)

In order to construct $\mathcal{LS}_{opt}$, the data needs to be smoothed with a Gaussian Kernel (D–dimensional) with a standard deviation equal to the h_{opt} (in each dimension) in order to suppress any smaller structural detail. To complete this operation, a matrix representation of the dataset needs to be constructed for convolution with the Gaussian kernel. Since the size of the smallest structure that will be retained in the optimal landscape is h_{opt}, sampling the feature space using cell sizes half that width ($0.5 \cdot h_{opt}$) should suffice to construct the matrix representation of the dataset ($\mathcal{M}$). This matrix representation $\mathcal{M}$ is effectively a histogram constructed using $2 \cdot m_{opt}$ bins ($0.5 \cdot h_{opt}$ bin–width). The matrix representation $\mathcal{M}$ thus constructed can be convolved with the aforementioned D–dimensional Gaussian Kernel to obtain $\mathcal{LS}_{opt}$. Figure 5.6 shows the optimal landscape generated for dataset S1. Further processing occurs as described in Section 2.4.5. This implementation eliminates the need for the parameters cp and SF (and associated assumptions) described in Section 2.5 in the construction of the matrix representation for a dataset.

5.3 Results & Comparison With Previous Implementation

After developing a new implementation for the Clustering algorithm, it was tested on the datasets described in Section 2.2 to study its performance. The alternate implementation described was coded in MATLAB; an N-Dimensional sparse matrix routines library [63] was used while coding the implementation.

Figures 5.7a through 5.7l display results of cluster detection (count & location) using the alternate implementation overlaid on the original datasets. (A standard MATLAB implementation of the *k–means* algorithm was used to determine the cluster labels for the data points.) Table 5.1 displays the results obtained using the alternate implementation (*Regional Maxima Finding* approach (A–RM) and *Watershed algorithm* approach (A–WS)) and compares them with some of the results obtained using the previous implementation as described in Section 2.5 ($cp = 1$ &

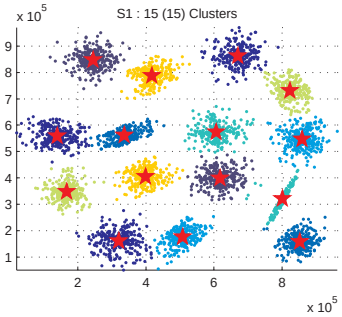
Table 5.1: Results of Clustering – Original & Alternate Implementations

DS	S1	S2	S3	S4	A1	A2	A3	V1	V2	V3	Z1	Z2
CNC	15	15	15	15	20	35	50	12	9	13	1	49
P–0.400	15	15	15	15	20	35	50	**10**	9	13	**441***	49
P–0.700	15	15	15	15	20	35	50	**10**	9	13	1	49
P–1.000	15	15	15	15	20	35	50	**10**	9	13	**30**	**50**
A–RM	15	15	15	15	20	35	50	**11**	9	**14**	**441***	49
A–WS	15	15	15	15	20	35	50	**11**	9	**14**	**441***	49
	← Number of Clusters →											

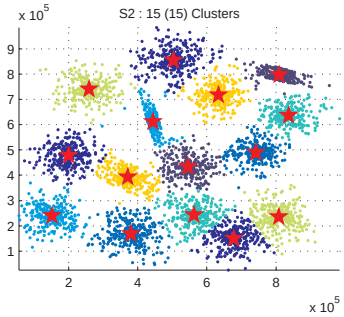
Bold red entries indicate disagreement between actual cluster count and cluster count reported by the algorithm.
CNC : Correct number of clusters for each dataset
P–0.400 : Previous implementation with $cp = 1$ & $SF = 0.400$
P–0.700 : Previous implementation with $cp = 1$ & $SF = 0.700$
P–1.000 : Previous implementation with $cp = 1$ & $SF = 1.000$
A–RM : Alternate implementation with *Regional Maxima Finding* approach
A–WS : Alternate implementation with *Watershed algorithm* approach
* : In Section 2.5, we explained that either 1, 441, or any squared number would be an acceptable interpretation.

$SF = 0.400, 0.700, 1.000$, *Regional Maxima Finding* approach).

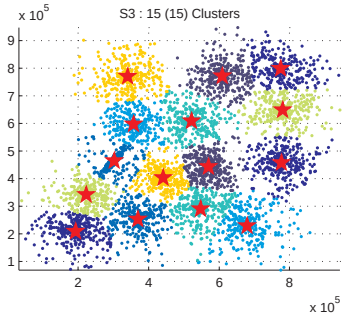
It can be seen from be seen from Table 5.1 that both implementations give identical results for cluster counts for datasets S1–S4, A1–A3, and V2. Both variants of the alternate implementation (A–RM & A–WS) detect 11 clusters in dataset V1 (Figure 5.7h). This is an improvement over the previous implementation; only 10 clusters are detected by the previous implementation (Figure 2.12h). The alternate implementation's performance deteriorates for dataset V3; 14 clusters are detected (Figure 5.7j) instead of the correct 13 detected by the previous implementation (Figure 2.12j). The alternate implementation reports 441 clusters for dataset Z1 – it marks each data point as a cluster. This is an acceptable interpretation since the dataset is a uniform distribution of data points. It can be seen that one variant of the previous implementation (P–0.400) also produces the same result. For dataset Z2, the alternate implementation reports 49 clusters. The previous implementation reports 49 or 50 clusters depending upon the SF value used.



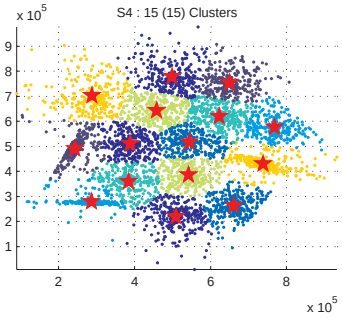
(a) S1



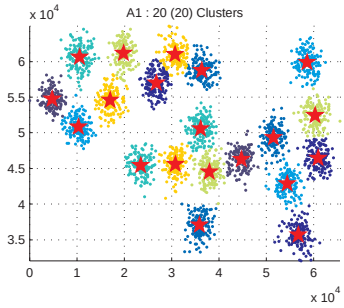
(b) S2



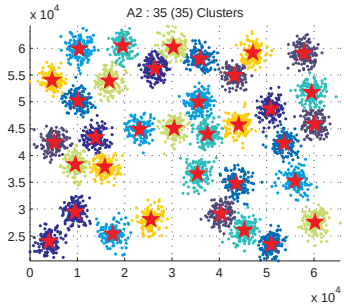
(c) S3



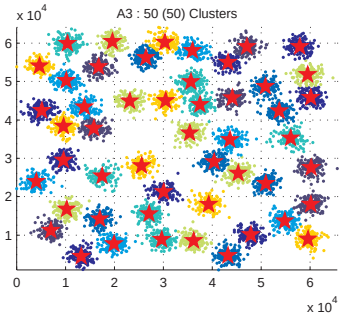
(d) S4



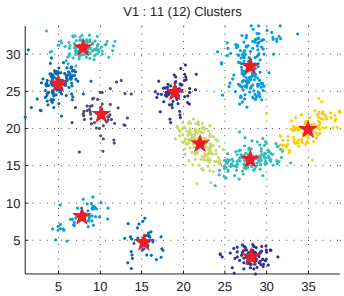
(e) A1



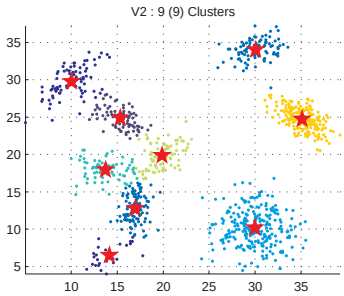
(f) A2



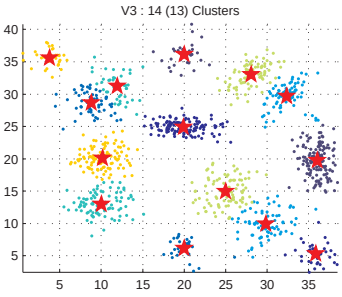
(g) A3



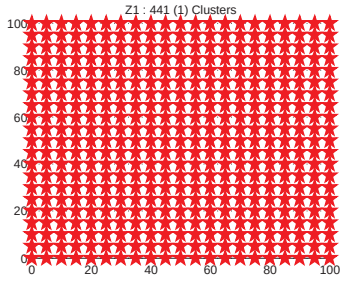
(h) V1



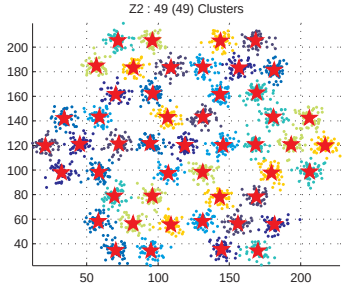
(i) V2



(j) V3



(k) Z1



(l) Z2

Figure 5.7: Clustering results using alternate implementation

Table 5.2: Computation Times for Original Implementation

	← RM Variant →			← WS Variant →		
SF / **Dataset**	**0.400**	**0.700**	**1.000**	**0.400**	**0.700**	**1.000**
S1	375.41	121.08	57.73	376.44	121.34	57.85
S2	388.31	132.61	67.19	389.33	132.87	67.31
S3	375.02	111.58	57.13	375.80	111.80	57.23
S4	446.95	146.57	69.83	447.96	146.87	69.96
A1	180.76	64.08	29.76	181.30	64.24	29.83
A2	327.48	117.82	50.85	328.19	118.01	50.93
A3	493.27	167.94	81.25	494.19	168.18	81.35
V1	170.93	61.31	26.87	171.79	61.52	26.97
V2	194.17	59.20	30.40	195.03	59.44	30.51
V3	114.89	41.41	19.31	115.56	41.63	19.38
Z1	1.71	0.52	0.29	1.34	0.52	0.28
Z2	297.79	96.46	47.40	298.96	96.78	47.52

$cp = 1.0$ for all cases
All times are measures in seconds
RM–Variant : *Regional Maxima Finding* approach
WS–Variant : *Watershed algorithm* approach

Table 5.3: Computation Times for Alternate Implementation

Variant / **Dataset**	← RM Variant →	← WS Variant →
S1	0.97	0.47
S2	0.85	0.45
S3	1.07	0.47
S4	1.51	0.48
A1	1.66	0.50
A2	2.71	0.60
A3	3.05	0.63
V1	1.30	0.46
V2	1.18	0.43
V3	1.32	0.45
Z1	9.76	0.64
Z2	2.57	0.57

All times are measures in seconds
RM–Variant : *Regional Maxima Finding* approach
WS–Variant : *Watershed algorithm* approach

It can also be seen from Table 5.1 that both the variants, A–RM and A–WS, produce identical results for all the datasets. This is not true in the case of the previous implementation. This could be due to the difference in the method used to construct the matrix representation for the datasets. In the alternate implementation, the matrix representation is only barely sufficient to capture the structural detail at the optimal scale. However, in the previous implementation, the matrix representation is sometimes constructed using a much more finely sampled grid than required to capture structural detail at the optimal scale. This could result in matrix representation capturing finer detail, which could in turn trigger the Watershed algorithm's intrinsic tendency to over–segment.

Tables 5.2 and 5.3 display the computation times for variants of both implementations. It can be seen from these tables that the alternate implementation is substantially faster than the previous implementation. The memory requirements of the alternate implementation are also usually much lower than that of the previous implementation.

It can be seen from Table 5.3 that the Watershed algorithm approach (A–WS) is faster than the Regional Maxima Finding approach (A–RM). This is most probably due to the MATLAB built–in Watershed algorithm implementation being more efficient than the custom–coded Regional Maxima Finding algorithm.

5.4 Drawback Discovered

The results described in Section 5.3 suggest the newly developed alternate implementation of the Clustering algorithm as being reasonably faithful. The dataset using feature set FS–2 was input to the newer implementation. The Clustering algorithm reported fewer clusters than present in the data.

In order to discover the reasons for the unsatisfactory behavior of the Clustering algorithm, it was necessary to track the algorithm's steps in a detailed fashion

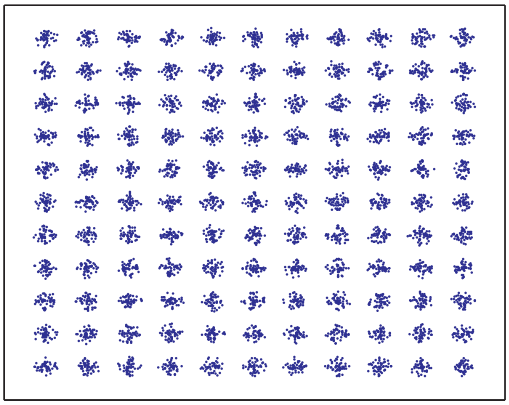
through its various stages. However, the texture data did not lend itself to an easy visualization, such that we were unable to examine the algorithm's steps using this data. Some synthetic datasets were constructed to aid in the examination (Figure 5.8). PGDS–1, PGDS–2, and PGDS–3 are two–dimensional (2–D) datasets, (x, y – position data) datasets with 6050, 6050, and 15400 data points respectively. PGDS–4, PGDS–5, and PGDS–6 are 5–D (x, y, r, g, b – position and color data) datasets with 7590, 1080, and 2220 data points. While comparing the algorithm's internal computations for two datasets PGDS–1 and PGDS–4, a problem was identified. While the datasets are of different dimensionality, it can be seen that both datasets have 121 non–overlapping clusters. The Clustering algorithm detects 121 clusters in PGDS–1; however, for PGDS–4 it reports only 4 clusters.

Investigation revealed that this behavior was due to the selection of an inappropriate scale for the detection of clusters in PGDS–4. The variation index (VI_Z) used for scale selection became ineffective with an increase in the dimensionality of the data from 2 to 5 dimensions.

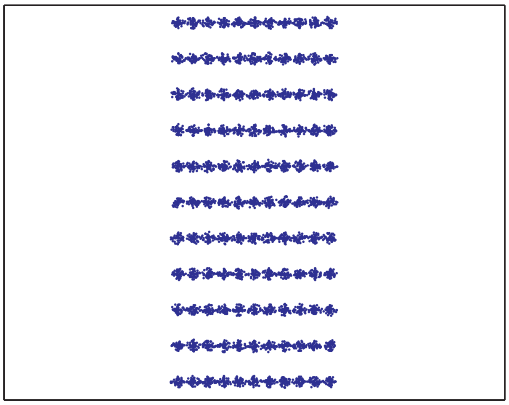
The variation index (VI_Z) used to select scale (described in Section 2.4.4) uses Tukey Outlier–Filtered height data ($\mathcal{Z}$). All values in $\mathcal{Z}$ in the range $[Z_{ll}, Z_{ul}]$ are used for computing (VI_Z), where Z_{ll} and Z_{ul} are defined in Eq(2.11) and repeated here as:

$$\begin{aligned} Z_{ll} &= Z_{25} - 1.5 \cdot IQR \\ Z_{ul} &= Z_{75} + 1.5 \cdot IQR \end{aligned} \tag{5.5}$$

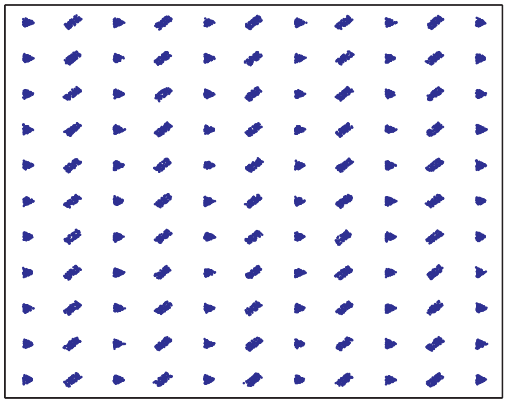
where as before, Z_{25} and Z_{75} are the 25 and 75 percentile values of $\mathcal{Z}$ respectively, and IQR is the Inter–Quartile Range of $\mathcal{Z}$. Values outside this range ($[Z_{ll}, Z_{ul}]$) are excluded from the calculation of VI_Z. The scale that maximizes the variation index (VI_Z) is iteratively searched for and selected. Figure 5.9a displays the VI_Z graph for the first iteration in the search for optimal scale for PGDS–1; Figure 5.9b shows the same for PGDS–4. It can be seen from Figure 5.9b that VI_Z takes on a zero value



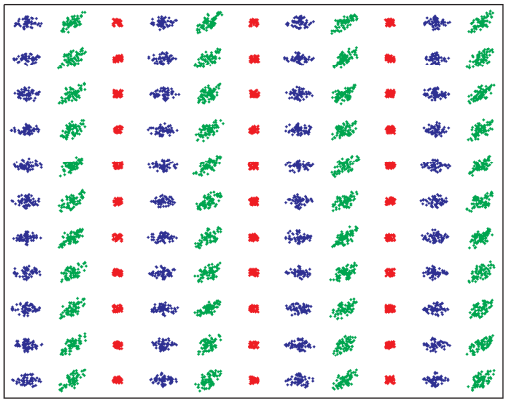
(a) PGDS–1



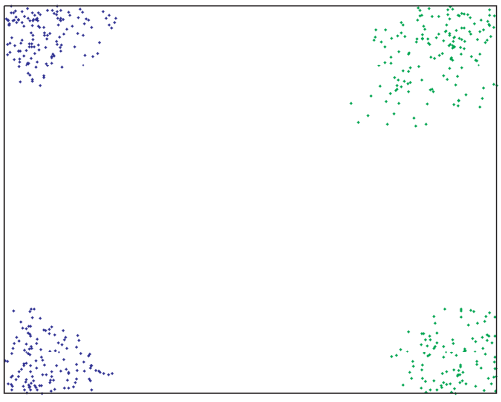
(b) PGDS–2



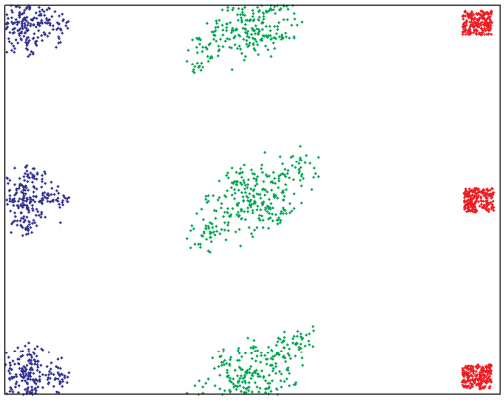
(c) PGDS–3



(d) PGDS–4

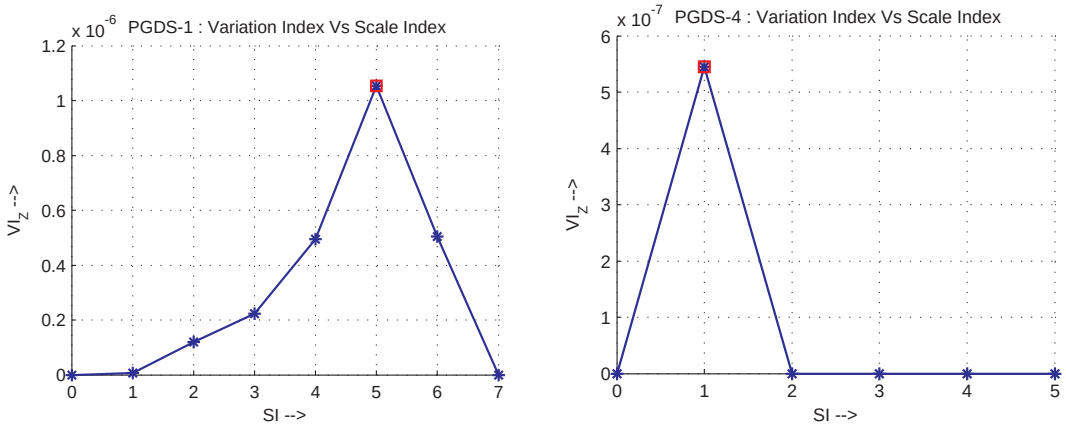


(e) PGDS–5



(f) PGDS–6

Figure 5.8: Datasets PGDS–1 to PGDS–6



(a) Variation Index Graph : PGDS–1 (b) Variation Index Graph : PGDS–4

Figure 5.9: Variation Index graphs for Datasets PGDS–1 & PGDS–4

Table 5.4: Internally computed values during Scale Selection – PGDS–1

SI	NB	TNB	ZBC	NZBC	PNZB	IQ·(10^{-7})	LL·(10^{-7})	UL·(10^{-6})	$VI_Z \cdot (10^{-7})$
0	1	1	0	1	100.00	0.00	8.94	0.89	0.00
1	2	4	0	4	100.00	0.11	8.68	0.91	.07
2	4	16	0	16	100.00	1.44	5.47	1.12	1.20
3	8	64	0	64	100.00	3.59	1.61	1.60	2.23
4	16	256	0	256	100.00	6.60	-5.11	2.13	4.94
5	32	1024	532	492	48.05	17.00	-25.50	4.25	10.5
6	64	4096	2698	1398	34.13	8.50	-12.70	2.12	5.03
7	128	16384	13237	3147	19.21	0.00	0.00	0.00	0.00

SI – Scale Index; NB – # bins per dimension; TNB – Total # bins
ZBC – # bins with zero data points in them; NZBC – # bins with some data points in them;
PNZB – % of bins with some datapoints in them; IQ – Inter–Quartile Range of Z values;
LL – Lower Cutoff according to Tukey Filter; UL – Upper Cutoff according to Tukey Filter;
VI_Z – Variation Index of Landscape at given Scale Index;

for scale indices 2 and higher.

Table 5.4 shows some values computed during the scale selection for PGDS–1. Table 5.5 shows the same values during the scale selection for PGDS–4. In Table 5.4, it can be seen from the last two rows (Scale Indices : 6, 7) that the Tukey Outlier Filter excludes some $\mathcal{Z}$ data causing VI_Z to be lower than the VI_Z associated for Scale Index 5. For Scale Index 7, the percentage of bins that contain at least one data point is 19.21%. Due to this, both Z_{25} and Z_{75} are zero; hence IQR is zero; because of that Z_{ll} and Z_{ul} also are zero; due to these limits, all non–zero values of $\mathcal{Z}$ are excluded and VI_Z goes to zero. This truncation is appropriate because the histogram

Table 5.5: Internally computed values during Scale Selection – PGDS–4

SI	NB	TNB	ZBC	NZBC	PNZB	IQ·(10^{-7})	LL·(10^{-7})	UL·(10^{-6})	$VI_Z \cdot (10^{-7})$
0	1	1	0	1	100.00	0.00	3.93	0.393	0.00
1	2	32	20	12	37.50	7.55	-11.3	1.89	5.44
2	4	1024	976	48	4.69	0.00	0.00	0.00	0.00
3	8	32768	32654	114	0.35	0.00	0.00	0.00	0.00
4	16	1048576	1048312	264	0.03	0.00	0.00	0.00	0.00
5	32	33554432	33553934	498	0.00	0.00	0.00	0.00	0.00

SI – Scale Index; NB – # bins per dimension; TNB – Total # bins
ZBC – # bins with zero data points in them; NZBC – # bins with some data points in them;
PNZB – % of bins with some datapoints in them; IQ – Inter–Quartile Range of Z values;
LL – Lower Cutoff according to Tukey Filter; UL – Upper Cutoff according to Tukey Filter;
VI_Z – Variation Index of Landscape at given Scale Index;

at Scale Index 7 would look very noisy – approximately 4/5 of the histogram bars have a height of zero, and the remaining are scattered over the data range. In the case of PGDS–1, VI_Z provides a mechanism to effectively identify the scale at which to detect clusters.

However, in Table 5.5, it can be seen from the last four rows ((Scale Indices : 2, 3, 4, 5)) that the Tukey Outlier Filter excludes all non–zero Z data causing VI_Z for all those Scale Indices to be zero. Due to this, $SI = 1$ is chosen for cluster detection, and that results in too few clusters being detected. It can be seen using Eq(5.2) in Figure 5.9b that using 2 bins per dimension will not provide enough resolution to detect all the clusters present in the dataset. The reason behind the Tukey Outlier Filter excluding all non–zero $\mathcal{Z}$ data for the mentioned Scale Indices is the rather large number of histogram bins (TNB) being created at those Scale Indices. Table 5.5 shows that TNB values grow at a power rate governed by the dimensionality of the input dataset. This causes the number of bins with no data points (ZBC) to increase much faster than the number of bins with at least one data point (NZBC). This results in Z_{25} and Z_{75} (and thus Z_{ll}, Z_{ul}) being zero and the Tukey Filter excluding all non–zero $\mathcal{Z}$ data.

It is clear that the Tukey Filter's upper and lower limits should be adjusted according to the dimensionality of the input data. Attempts at utilizing currently

existing modified Tukey Filter methods [64] based on *MedCouple* [65] did not yield any improvements – these methods also employed IQR, which is still affected by the aforementioned situation. Attempts to employ other outlier detection schemes [66] also did not meet with success.

Inter–Quartile Range (IQR) and Median Absolute Deviation (MAD), the two other metrics that displayed characteristics similar to the one employed – Variation Index (VI_Z) – also suffered from similar problems of ineffectiveness due to the rapidly increasing number of histogram bins with dimensionality of input data.

No clear model was discovered to govern any adjustments to the Tukey Filter's limits with a justifiable logic that would produce meaningful results. Since an appropriate improvement could not be made to the technique, a mechanism that would indicate whether or not the technique could be expected to work on a given dataset was deemed desirable. The following describes a mechanism that suggests whether the aforementioned clustering technique could be reliably applied to a given dataset.

As mentioned before, the performance of the variation index VI_Z is degraded in situations where the value of ZBC increases faster than the value of NZBC. This happens for datasets in which data points are sparsely distributed. The effect worsens with increasing dimensionality of the input dataset.

For a dataset with a finite number of data points, using an arbitrarily large number of bins to construct a histogram will drive the value of VI_Z to zero. Assuming that a given dataset has data points spread out more or less uniformly in all dimensions (a non-sparse dataset), consider the following. Eq(5.6) represents a condition for the minimum number of bins per dimension (m_{vi}) to ensure a histogram whose VI_Z *will* be zero.

$$\begin{aligned} 0.25(m_{vi})^D &= N \\ \Rightarrow \quad m_{vi} &= (4 \cdot N)^{\frac{1}{D}} \end{aligned} \tag{5.6}$$

where N is the number of data points in the dataset, and D is the dimensionality of

the dataset.

This implies that if the total number of bins in the histogram is four times the number of data points, then it is assured that at least 75 percent of histogram bin counts will be zero and this ensures that VI_Z will be zero. Thus, the first step in our diagnostic procedure is to compute m_{vi} from Eq(5.6) for the given dataset.

For the second step in our diagnostic procedure we compute a new quantity m_c for the given dataset (beginning with Eq(5.7)). Let a one dimensional histogram be constructed for each column of the dataset using m bins. For the i^{th} dimension, let nzc_i be the number of bins with non-zero bin counts, and let nzf_i represent the fraction of bins that have non-zero bin counts in the i^{th} dimension. The terms nzf_i and $tnzf$ are defined as:

$$\begin{aligned} nzf_i &= \tfrac{nzc_i}{m} \\ tnzf &= \prod_{i=1}^{D} nzf_i \end{aligned} \tag{5.7}$$

where $tnzf$ represents the maximum fraction of histogram bins that *can* take on a non-zero bin count in the D–dimensional histogram created using m bins per dimension. Let m_c be the value of m for which $tnzf$ is slightly less than 0.25; then the value of VI_Z goes to zero for a histogram created using m_c bins per dimension.

m_{vi} represents a threshold at which VI_Z will take on a value of zero for a dataset, irrespective of the distribution of its data. It is not affected by how sparsely or densely one dimension is populated compared with other dimensions. m_c represents another threshold at which VI_Z will take on a value of zero for a dataset; however, m_c is affected by the distribution of its data. For two datasets of equal cardinality and dimensionality, m_c will take on a larger value for the dataset in which data points are more or less uniformly spread out in all dimensions than for another dataset in which some dimensions are sparsely populated.

The third step in our diagnostic procedure is to compare m_c with m_{vi}. If m_c is less than m_{vi}, then it is likely that the clustering algorithm will not deliver satisfactory

results due to improper scale selection.

It should be noted that m_{vi} does not indicate any recommendations in the context of scale selection. It merely marks the threshold at which the scale selection metric (variation index VI_Z) loses its efficacy. In other words, m_{vi} marks a limit on the size of the smallest cluster structure that can be detected using the clustering algorithm. If a dataset has cluster structures whose sizes require a histogram that uses more than m_{vi} bins per dimension to be detected, then the clustering algorithm will not succeed in detecting the cluster structures; the scale selection metric VI_Z becomes ineffective for histograms constructed using more than m_{vi} bins per dimension.

Table 5.6 shows the results of running the diagnostic procedure on various datasets. The column titled "DPR" in Table 5.6 recommends whether or not the results of the clustering algorithm should be accepted as a valid clustering. A "reject" in the table column indicates that it is likely that the clustering algorithm will not process the dataset correctly due to relative sparsity of data in the dataset. An "accept" in the table column indicates that the clustering algorithm will not have any difficulty processing the dataset due to data sparsity issues.

Comparing the recommendations provided in Table 5.6 with the results of the clustering algorithm shown in Figure 5.7 and Figure 5.10 indicate that out of the 18 recommendations made by our diagnostic procedure, 17 are correct and 1 is incorrect.

For datasets PGDS–4, PGDS–5, and PGDS–6, a rejection of clustering algorithm output is recommended. The dimensions relating to the color data (r, g, b) for these datasets, are very sparsely populated; the data in these dimensions takes on values of either 0 or 1. This causes the variation index VI_Z to become ineffective in scale selection, leading to the clustering algorithm selecting inappropriate scales for cluster detection. These failures suggest that an implementation, in which, the D–dimensional histogram constructed with varying numbers of bins for each dimension might fare better than the proposed implementation. Using fewer number of bins for

Table 5.6: Results of running the diagnostic procedure on various datasets

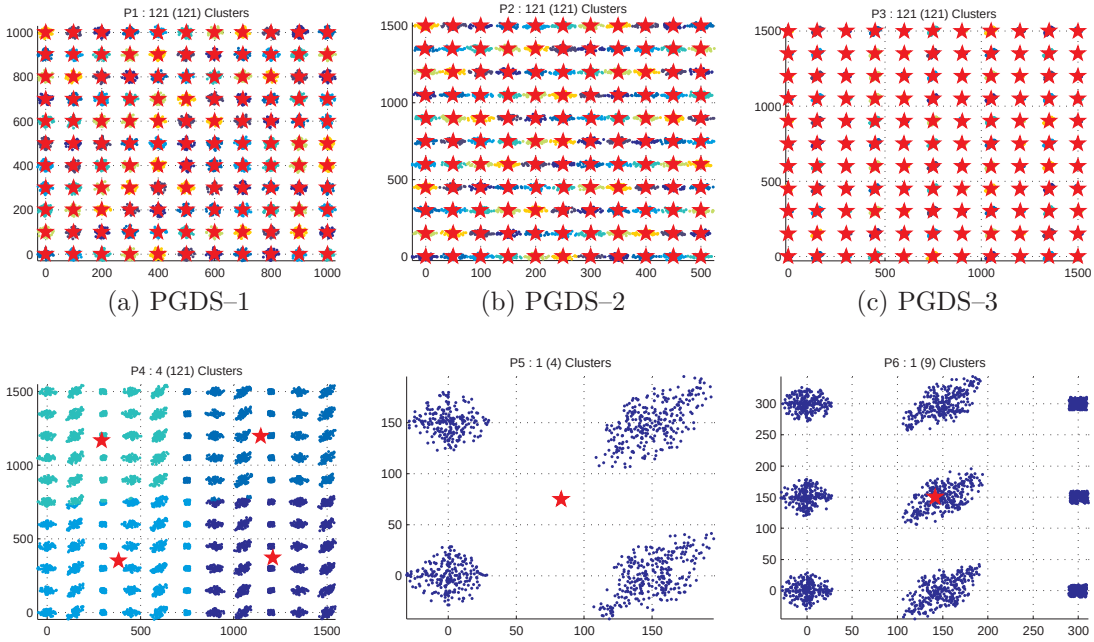
Dataset	**N**	**D**	m_{vi}	m_c	**DPR**
S1	5000	2	141	5367	Accept
S2	5000	2	141	5953	Accept
S3	5000	2	141	6001	Accept
S4	5000	2	141	5304	Accept
A1	3000	2	110	3685	Accept
A2	5250	2	145	6961	Accept
A3	7500	2	173	9954	Accept
V1	1065	2	65	1236	Accept
V2	995	2	63	1200	Accept
V3	1025	2	64	1285	Accept
Z1	441	2	42	42	Accept
Z2	2450	2	99	3071	Accept
PGDS–1	6050	2	156	3957	Accept
PGDS–2	6050	2	156	3189	Accept
PGDS–3	15400	2	248	67	Reject
PGDS–4	7590	5	8	4	Reject
PGDS–5	1080	5	5	3	Reject
PGDS–6	2220	5	6	4	Reject

N – Number of data points in the dataset
D – Dimensionality of the dataset
DPR – Recommendation provided by diagnostic procedure
m_{vi} – rounded to the closest integer

relatively sparse dimensions and a greater number of bins for relatively non–sparse dimensions might mitigate the problem caused by rapidly increasing ZBC values. For dataset PGDS–3, the diagnostic recommends rejecting the output of the clustering algorithm. However, it can be seen in Figure 5.10 that the clustering algorithm produces a valid result for PGDS–3. While this error is not as severe as accepting an incorrect clustering configuration, it indicates that the aforementioned diagnostic does not have 100% efficiency.

5.5 Conclusions

This chapter presents a newer (alternate) implementation for the new Clustering algorithm introduced in Section 2.4.



(a) PGDS–1 (b) PGDS–2 (c) PGDS–3

(d) PGDS–4 (e) PGDS–5 (f) PGDS–6

Figure 5.10: Clustering results for Datasets PGDS–1 to PGDS–6 (Red points indicate cluster centers.)

Performance comparisons between the previous implementation and the alternate implementation reveal that the alternate implementation is substantially faster than the previous implementation. The alternate implementation also demands less memory than the previous implementation. The *Watershed algorithm* variant of the alternate implementation is better equipped to overcome intrinsic over–segmentation tendencies. The alternate implementation also eliminates the need for parameters used for constructing the matrix representation for a dataset (SF and cp).

A major drawback was also discovered for the Clustering algorithm. It was discovered that the metric used for scale selection (VI_Z) loses its effectiveness with an increase in the dimensionality of the input data. No suitable adjustments or modifications could be developed or discovered to overcome this deficiency. A diagnostic mechanism is presented that recommends whether or not the output of the clustering algorithm should be accepted.

Future investigations should primarily explore: (1) modifications to the current

scale selection metric in order to overcome problems described in Section 5.4; (2) development of alternate dispersion metrics for use in scale selection; (3) development of alternate non–parametric outlier filters for use in metric computation; (4) development of another implementation which uses varying numbers of histogram bins for each dimension – where the number of bins used for a given dimension is governed by the sparsity of data in that dimension; and (5) development of a diagnostic procedure which recommends the correct course of action for any dataset.

CHAPTER 6

Conclusions and Ideas for Future Work

In this report, the following original contributions have been developed:

- An improved watershed–based clustering technique whose main advantage is its unsupervised and automatic nature, requiring no parameters to be tuned or to be determined experimentally.
- A new method to estimate quantiles for pairwise distances.
- A new method for selecting the number of bins for constructing a histogram for a given dataset.
- An alternate implementation for the aforementioned clustering technique that works faster and uses lesser memory.

Areas for future investigation have been outlined for each method in the individual chapters. Future effort should also be directed towards conducting formal analyses of the various techniques to furnish mathematical bounds and guarantees for the techniques' performance. Bounds on errors in results produced by the techniques, and bounds on time and memory requirements of the techniques will be significant additions to the work.

In an earlier report [67], it was proposed that the clustering algorithm should be adapted to tasks relating to "Proximity" and "Similarity" Gestalt Laws [3], [4], as a first step towards a framework proposed in [1]. It was suggested that a recursive application of the parameter–free clustering algorithm to datasets with appropriate feature sets could result in an emulation of the "Proximity" and "Similarity" laws.

To emulate the "Proximity" and "Similarity" laws for 2–D images, the feature set will need to include at least the following:

- 2 dimensions for position – (X, Y);
- 3 dimensions for color – (R, G, B) or an equivalent;
- at least 3 to 5 dimensions for shape – area, perimeter, area moments of inertia, and other such shape descriptors.

However, the drawback of the clustering algorithm described in Section 5.4 poses a significant obstacle to such an application. A dataset based on a feature set describing attributes such as position, color, and shape, will have a dimensionality at which the clustering algorithm is currently not effective at detecting the number of clusters present in the dataset.

It is imperative that future work should first be directed toward overcoming the clustering algorithm's drawback so as to make it effective with data of relatively high dimensionality. As mentioned in Section 5.5, there are several approaches that might improve the clustering algorithm's performance with higher dimensional datasets. While most of the suggestions require an element of discovery, the approach that suggests modifying the implementation to use varying number of bin numbers for each dimension for constructing a histogram is probably the easier one to implement. However, such an implementation will complicate the search procedure required to locate the configuration that maximizes the variation index (VI_Z).

Once the drawback is overcome, work can be directed back towards the emulation of the "Proximity" and "Similarity" laws.

BIBLIOGRAPHY

[1] S. V. G. Lolla, "A proposed knowledge–based, inference–driven vision system." Preliminary Exam Report, August 2008.

[2] S. Wrede, C. Bauckhage, G. Sagerer, W. Ponweiser, and M. Vincze, "Integration frameworks for large scale cognitive vision systems - an evaluative study," in *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1*, (Washington, DC, USA), pp. 761–764, IEEE Computer Society, 2004.

[3] Forsyth, David A. and Ponce, Jean, *Computer Vision: A Modern Approach*. Prentice Hall, August 2002.

[4] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*. Springer Publishing Company, Incorporated, 2007.

[5] M. Bicego, M. Cristani, A. Fusiello, and V. Murino, "Watershed-based unsupervised clustering," in Rangarajan *et al.* [68], pp. 83–94.

[6] Shannon, C. E., "A mathematical theory of communication," *Bell system technical journal*, vol. 27, 1948.

[7] C. Croux and P. J. Rousseeuw, "Alternatives to the median absolute deviation," *Journal of the American Statistical Association*, vol. 88, pp. 1273–1283, December 1992.

[8] G. Hamerly and C. Elkan, "Learning the k in k-means," in *In Neural Information Processing Systems*, p. 2003, MIT Press, 2003.

[9] S. B. Kotsiantis and P. E. Pintelas, "Recent advances in clustering: A brief survey," *WSEAS Transactions on Information Science and Applications*, vol. 1, pp. 73–81, 2004.

[10] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The Computer Journal*, vol. 41, pp. 578–588, 1998.

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

[12] R. L. Thorndike, "Who belongs in the family?," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[13] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD '99)*, (New York, NY, USA), pp. 49–60, ACM, 1999.

[14] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 226–231, 1996.

[15] R. Kothari and D. Pitts, "On finding the number of clusters," *Pattern Recognition Letters*, vol. 20, no. 4, pp. 405–416, 1999.

[16] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[17] Q. Zhao, M. Xu, and P. Fränti, "Knee point detection on bayesian information criterion," in *ICTAI '08: Proceedings of the 2008 20th IEEE International Conference on Tools with Artificial Intelligence*, (Washington, DC, USA), pp. 431–438, IEEE Computer Society, 2008.

[18] S. Lolla and L. Hoberock, "Improved unsupervised clustering over watershed-based clustering," in *In Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA'10)*, pp. 253 –259, Dec. 2010.

[19] P. Fränti, "Clustering datasets." `http://cs.joensuu.fi/sipu/datasets/`, 2006 (accessed February 19, 2010).

[20] "Uci machine learning repository." `http://archive.ics.uci.edu/ml/datasets.html`, 2007 (accessed October 19, 2010).

[21] J. B. T. M. Roerdink and A. Meijster, "The watershed transform: Definitions, algorithms and parallelization strategies," *Fundamenta Infomaticae*, vol. 41, no. 1-2, pp. 187–228, 2000.

[22] T. Lindeberg, *Scale-Space Theory in Computer Vision*. Norwell, MA, USA: Kluwer Academic Publishers, 1994.

[23] C. E. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[24] A. R. Wilcox, "Indices of qualitative variation," Tech. Rep. 3445605133753, Oak Ridge National Laboratory, 1967.

[25] D. C. Hoaglin, F. Mosteller, and J. W. Tukey, eds., *Understanding Robust and Exploratory Data Analysis*. Wiley Series in Probability and Mathematical Statistics, Wiley-Interscience, 1983.

[26] P. J. Huber, *Robust Statistics / Peter J. Huber*. Wiley, New York, 1981.

[27] S. Lolla and L. Hoberock, "On estimation of quantiles for pairwise distances," in *In Proceedings of the Ninth International Conference on Machine Learning and Applications (ICMLA'10)*, pp. 808 –812, Dec. 2010.

[28] B. W. Weide, "Space-efficient on-line selection algorithms," in *Proceedings of Computer Science and Statistics 11th Annual Symposium on the Interface*, pp. 308–311, March 1978.

[29] P. J. Rousseeuw and J. Gilbert W. Bassett, "The remedian: a robust averaging method for large data sets," *Journal of the American Statistical Association*, vol. 85, no. 409, pp. 97–104, 1990.

[30] B. W. Schmeiser and S. J. Deutsch, "Quantile estimation from grouped data: The cell midpoint," *Communications in Statistics: Simulation and Computation*, vol. B6(3), pp. 221–234, 1977.

[31] R. Jain and I. Chlamtac, "The p2 algorithm for dynamic calculation of quantiles and histograms without storing observations," *Commun. ACM*, vol. 28, no. 10, pp. 1076–1085, 1985.

[32] R. Agrawal and A. Swami, "A one-pass space-efficient algorithm for finding quantiles," in *In Proceedings of 7th International Conference on Management of Data (COMAD-95)*, 1995.

[33] C. Hurle and R. Modarres, "Low-storage quantile estimation," *Computational Statistics*, vol. 10, pp. 311–325, 1995.

[34] S. Battiato, S. Battiato, D. Catalano, D. Cantone, D. Catalano, G. Cincotti, and M. Hofri, "An efficient algorithm for the approximate median selection problem," in *In Proceedings of the Fourth Italian Conference, CIAC 2000*, pp. 226–238, Springer-Verlag, 1999.

[35] G. S. Manku, S. Rajagopalan, and B. G. Lindsay, "Approximate medians and other quantiles in one pass and with limited memory," *SIGMOD Rec.*, vol. 27, no. 2, pp. 426–435, 1998.

[36] D. B. Johnson and T. Mizoguchi, "Selecting the kth element in $x + y$ and $x_1 + x_2 + \cdots + x_m$," *SIAM Journal on Computing*, vol. 7, no. 2, pp. 147–153, 1978.

[37] C. Croux and P. J. Rousseeuw, "Time-efficient algorithms for two highly robust estimators of scale," *Computational Statistics*, vol. 1, pp. 411–428, 1992.

[38] H. Shimazaki and S. Shinomoto, "A method for selecting the bin size of a time histogram," *Neural Comput.*, vol. 19, no. 6, pp. 1503–1527, 2007.

[39] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[40] D. Freedman and P. Diaconis, "On the histogram as a density estimator:l2 theory," *Probability Theory and Related Fields*, vol. 57, pp. 453–476, December 1981.

[41] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[42] W. L. Martinez and A. R. Martinez, *Computational Statistics Handbook with MATLAB, Second Edition (Computer Science and Data Analysis)*. Chapman & Hall/CRC, 2 ed., December 2007.

[43] M. P. Wand, "Data-based choice of histogram bin width," *The American Statistician*, vol. 51, pp. 59–64, 1996.

[44] J. S. Simonoff and F. Udina, "Measuring the stability of histogram appearance when the anchor position is changed," *Comput. Stat. Data Anal.*, vol. 23, no. 3, pp. 335–353, 1997.

[45] K. H. Knuth, "Optimal Data-Based Binning for Histograms," *ArXiv Physics e-prints*, May 2006.

[46] M. P. Wand and M. C. Jones, *Kernel Smoothing (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1994.

[47] S. Lolla and L. Hoberock, "On selecting the number of bins for a histogram," in *In Proceedings of the Seventh International Conference on Data Mining (DMIN'11)*, pp. 344 –350, Jul. 2011.

[48] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[49] R. J. Hyndman, "The problem with sturges rule for constructing histograms," *Business*, pp. 1–2, July 1995.

[50] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[51] C. R. Rao, E. J. Wegman, and J. L. Solka, *Handbook of Statistics, Volume 24: Data Mining and Data Visualization (Handbook of Statistics)*. North-Holland Publishing Co., 2005.

[52] D. Freedman and P. Diaconis, "On the histogram as a density estimator:L2 theory," *Probability Theory and Related Fields*, vol. 57, pp. 453–476, December 1981.

[53] C. J. Stone, "An asymptotically optimal histogram selection rule," in *Proceedings of the Berkeley conference in honor of Jerzy Neyman and Jack Kiefer,*

Vol. II (Berkeley, Calif., 1983), Wadsworth Statist./Probab. Ser., pp. 513–520, Wadsworth, 1985.

[54] M. Rudemo, "Empirical choice of histograms and kernel density estimators," *Scandinavian Journal of Statistics*, vol. 9, no. 2, pp. 65–78, 1982.

[55] P. Hall, "Akaike's information criterion and kullback-leibler loss for histogram density estimation," *Probability Theory and Related Fields*, vol. 85, pp. 449–467, 1990.

[56] Lucien Birgé and Yves Rozenholc, "How many bins should be put in a regular histogram," *ESAIM: P&S*, vol. 10, pp. 24–45, 2006.

[57] J. S. Marron and A. B. Tsybakov, "Visual error criteria for qualitative smoothing," *Journal of the American Statistical Association*, vol. 90, no. 430, pp. 499–507, 1995.

[58] V. I. Glivenko, "Sulla determinazione empirica delle leggi di probabilita," *Giornale dell'Istituto Italiano degli Attuari*, no. 4, pp. 92–99, 1933.

[59] F. P. Cantelli, "Sulla determinazione empirica delle leggi di probabilita," *Giornale dell'Istituto Italiano degli Attuari*, no. 4, pp. 221–424, 1933.

[60] R. L. Thorndike, "Who belongs in the family?," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[61] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pp. 576 – 584, nov. 2004.

[62] P. J. Green and B.W.Silverman, *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*. Chapman and Hall/CRC, 1994.

[63] M. Jacobson and M. Vlker, "N–dimensional sparse arrays." `http://www.mathworks.com/matlabcentral/fileexchange/29832-n-dimensional-sparse-arrays`, Dec 2010 (accessed July 2011).

[64] M. Hubert and E. Vandervieren, "An adjusted boxplot for skewed distributions," *Comput. Stat. Data Anal.*, vol. 52, pp. 5186–5201, August 2008.

[65] G. Brys, M. Hubert, and A. Struyf, "A robust measure of skewness," *Journal of Computational and Graphical Statistics*, vol. 13, no. 4, pp. pp. 996–1017, 2004.

[66] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *SIGMOD Rec.*, vol. 30, pp. 37–46, May 2001.

[67] S. V. G. Lolla, "Some contributions to data analysis." Qualifying Exam Report, August 2010.

[68] A. Rangarajan, M. A. T. Figueiredo, and J. Zerubia, eds., *Energy Minimization Methods in Computer Vision and Pattern Recognition, 4th International Workshop, EMMCVPR 2003, Lisbon, Portugal, July 7-9, 2003, Proceedings*, vol. 2683 of *Lecture Notes in Computer Science*, Springer, 2003.

VITA

Sai Venu Gopal Lolla

Candidate for the Degree of

Doctor of Philosophy

Dissertation: NEW TECHNIQUES FOR CLUSTERING, SELECTION OF NUMBER OF HISTOGRAM BINS, AND ESTIMATION OF QUANTILES OF PAIRWISE DISTANCES

Major Field: Mechanical Engineering

Biographical:

Personal Data: Born in Rajahmundry, AP, India on April 29, 1980.

Education:

Completed the requirements for the Doctor of Philosophy degree in Mechanical Engineering at Oklahoma State University, Stillwater, OK, USA in December, 2011.

Completed the requirements for the Master of Science degree in Mechanical Engineering at Oklahoma State University, Stillwater, OK, USA in 2005.

Completed the requirements for the Bachelor of Technology degree in Mechanical Engineering at Jawaharlal Nehru Technological University, Hyderabad, AP, India in 2002.

Name: Sai Venu Gopal Lolla Date of Degree: December, 2011

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: NEW TECHNIQUES FOR CLUSTERING, SELECTION OF NUMBER OF HISTOGRAM BINS, AND ESTIMATION OF QUANTILES OF PAIRWISE DISTANCES

Pages in Study: 107 Candidate for the Degree of Doctor of Philosophy

Major Field: Mechanical Engineering

The emulation of Gestalt Laws was required for a proposed knowledge-based inference-driven vision system. Emulating the Proximity gestalt law required a clustering technique that worked without: (a) the need for any input parameters that would govern the number of clusters detected; and (b) without any preference for a given cluster shape. A watershed algorithm based clustering technique due to Bicego et. al. satisfies the aforementioned requirements. However, their algorithm's performance is degraded due to experimentally tuned internal parameters. An improved clustering technique was developed without the need for any such internal parameters by employing the concept of scale. Two implementations have been provided for the proposed clustering algorithm. A drawback affecting the proposed clustering algorithm's performance was also discovered. A diagnostic procedure that recommends whether or not the clustering algorithm's output should be accepted has been provided as an addendum.

While working on the problem of automatic scale detection for the clustering technique, certain analytical and computational difficulties were encountered while attempting to: (a) compute quantiles of pairwise distances for use in the calculation of Qn; and (b) construction of histograms for evaluating Entropy. A new method to estimate quantiles of pairwise distances was developed. Performance comparisons with other existing methods showed that the proposed method is: faster and more accurate than other estimation methods; and faster than other selection methods. A new method to select the number of bins for constructing a histogram was developed. Performance comparisons with existing methods showed that the proposed method produces visually appealing histograms that capture shape features of underlying distribution to a finer detail without admitting excessive noise.

ADVISOR'S APPROVAL: Dr.Lawrence L. Hoberock