Keywords: Clustering, Path Length, Consensus, N-Dimensional, Line of Sight

High Dimensional Cluster Analysis Using Path Lengths

Kevin L. McIlhany MCILHANY@USNA.EDU
Physics Department
U. S. Naval Academy, Stop 9c
572c Holloway RD
Annapolis, MD 21402-5002, USA

Stephen Wiggins S.WIGGINS@BRISTOL.AC.UK
School of Mathematics
University of Bristol
University Walk
Bristol, BS8 1TW,
United Kingdom

October 16, 2017

Editor:

Abstract

A hierarchical scheme for clustering data is presented which applies to spaces with a high number of dimension ($N_D > 3$). The data set is first reduced to a smaller set of partitions (multi-dimensional bins). Multiple clustering techniques are used, including spectral clustering, however, new techniques are also introduced based on the path length between partitions that are connected to one another. A Line-Of-Sight algorithm is also developed for clustering. A test bank of 12 data sets with varying properties is used to expose the strengths and weaknesses of each technique. Finally, a robust clustering technique is discussed based on reaching a consensus among the multiple approaches, overcoming the weaknesses found individually.

1. Introduction

Clustering is a fundamental technique and methodology in data analysis and machine learning. The explosion of the field of data science has, consequently, led to an expansion in how this notion is applied. In this respect, it would be more appropriate to refer to clustering as data organization, which would encompass the ideas of 1) data reduction, 2) data identification, 3) data clustering, and 4) data grouping.

Data reduction is the process of converting raw data into a form that is more amenable for the application of a specific analytical and/or computational methodology. Data identification is the process of analysing trends or distributions within the data. Data clustering is the process of associating data through proximity, similarity, or dissimilarity. Data grouping refers to breaking down data into groups according to a criterion that is appropriate for the specific application under consideration.

The literature on clustering is extensive and it is beyond the scope of this paper to provide an adequate review of this topic. However, the following papers Jain et al. (1999); Ng et al. (2002); Barbakh et al. (2009); Jain (2010), for background on the clustering methods in this paper and the

book Kaufman and Rousseeuw (2009) provides a broad overview of clustering methodologies, as well as their numerical implementation.

There is no single algorithm that realizes all four of these aspects of data organization. The approach to this problem pursued in this paper is to develop a hierarchical scheme leading to a cluster analysis that encompasses the issues raised above and is adapted to high dimensional spaces.

The data analysis scheme presented in this paper uses a blend of traditional data analysis via a multi-variate histogram along with standard clustering techniques, such as k-means, k-medoids and spectral clustering. By binning the data onto a multi-dimensional grid, data is partitioned into regions on the grid which may be connected or separated depending on the character of the data set. Data reduction is realized by only retaining bins that have a population above a user selected threshold. The resulting multidimensional bins are referred to as partitions. The passage to partitions is the data reduction step.

Data indentification is the process of assigning known data distributions (parent) to an entangled set of data. Typical examples are found in the literature of Bayesian analysis Binder (1978); Kass and Raftery (1995), however, this pursuit dates farther back to earlier attempts to understand how to distinguish data from two or more distributions with overlapping tails. In more difficult scenarios, several distributions might overlap within the peak regions, changing the problem to the identification of subdomains of the mixed versus non-mixed distributions.

Data clustering traditionally refers to assigning data to subsets based on the proximity of data to one another. The goals of the field of data clustering have expanded from this definition, taking on some of the other roles identified here. For the purposes of this study, the term clustering will refer to both the overall techniques applied as well as the specific property a set has when its members are close to on another when appropriate. In the broadest sense, a cluster is simply a label given to data to identify common features.

Data grouping is the process of assigning labels to data, without regard for proximity or parent distributions. An example might be to segregate a class of thirty 2^{nd} grade children into five sub-groups before entering a museum for a tour. How the larger group is broken apart is unimportant, merely that the larger group is distributed into smaller groups.

In this study, standard clustering techniques are applied such as kmeans, kmedoids and spectral clustering, along with new path-based approaches. After data reduction, data within partitions may be connected in regions where a path length can be calculated along the grid of partitions between any two data. Several new clustering algorithms have been developed using the path length. Further, if two partitions are visible to each other by a Line-Of-Sight criteria, the relationship between them is given additional significance. These ideas are used, in conjunction with standard clustering techniques, to construct 26 different clustering algorithms.

An analysis configuration is the set of all 26 clustering techniques along with the choices made for which variables represent the data manifold as well as how the data space is partitioned. The choice of variables used to represent the data defines the number of dimensions as well as the character of the data manifold formed. In some cases, two or more variables may provide redundant information, while other choices may add noise. Changes to the resolution of how the data space is partitioned may lead to changes in a datum's cluster assignment. For each choice of clustering technique, variables used (dimensions) and resolution (partitioning), each datum is assigned to a cluster. When data consistently cluster in one arrangement across multiple analysis configurations, the data is assigned robustly to its cluster. To determine a *robust* clustering assignment, a polling technique is used to arrive at a consensus amongst the clustering algorithms. While any one

technique has faults, the consensus of techniques overcomes any one failure mode, giving the best all-round identification Strehl and Ghosh (2003).

This paper is organized as follows, section 2 defines the basic components used in this study. Section 3 shows the calculations of several values used throughout the analysis. Section 4 outlines the strategy taken for this study and it lists the comprehensive set of arrays calculated that are needed for the suite of algorithms. This section also introduces a test-bank of data sets used for clustering. Section 5 presents each algorithm, with details left for the appendix. Section 6 shows the results for each clustering algorithm, discussing the strengths and weaknesses of each approach. Section 7 introduces the approach to robust clustering, employing multiple techniques and how a consensus is reached. Section 8 concludes with suggestions for extending this suite of clustering techniques. In the appendix, Sec. A discusses algorithms used by more than one clustering technique. The next appendices show the details of the clustering algorithms as applied in this study for: clustering to maxima globally and using path length, Sec. A.1, as well as clustering based on Line-Of-Sight, Sec. B.

2. Terminology, Definitions, Notation and Data Reduction

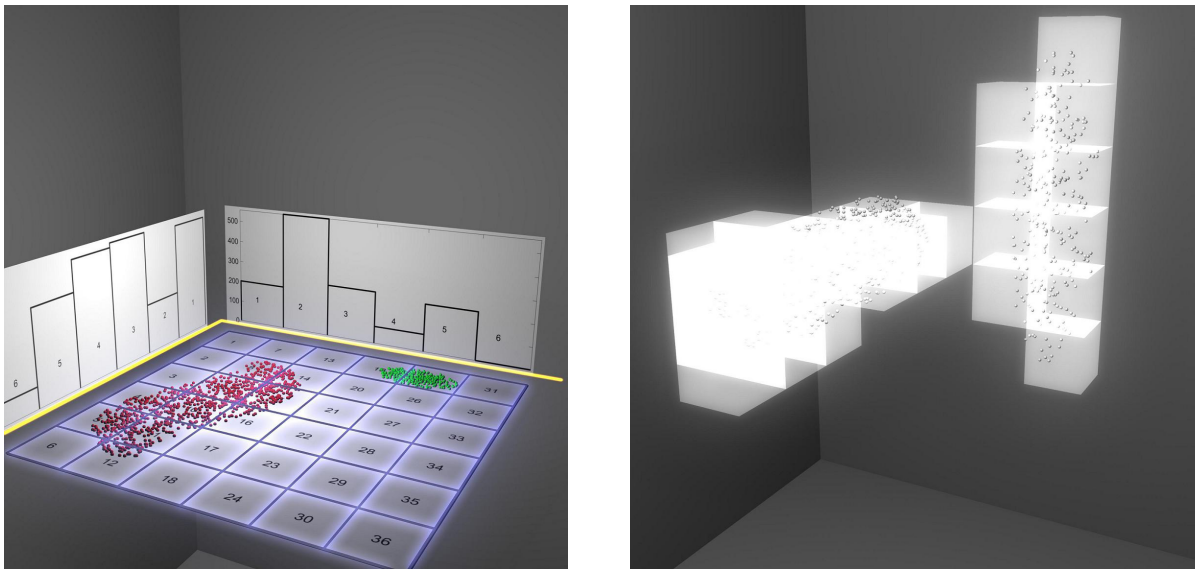
This study has five basic components that discussed in detail; data, partitions, clusters, clustering algorithms and configurations of analysis. The definitions for data, partitions and clc usters are given in this section, while the clustering algorithms and configurations of analysis given in Sec. 5 and Sec. 7 respectively. Quantities denoted by a tilde refer to the original data set while quantities without a tilde symbol refer to partitions (a reduced data set defined in Sec.2.3).

2.1 Data

The term data used in this study will refer to a set of measurements taken of a system. The values of the measurements may vary in nature, from logical values, lists of characters, images, vector or scalar values, either real or imaginary. In the case of complex values or vectors, each component will be taken separately as a real valued scalar. For parts of the data which are non-numerical, the values of the measurements will be mapped to a numerical value. A *data vector* is formed whose components are the set of numerical measurements. The collection of these data vectors form the data set. For each component, the range is determined either by the extrema taken from the data set or by the limits based on the functional form of the measurement. The *data space* is the space spanned by the set of data vectors.

The total number of data vectors in the set is denoted by "N". Each component of the data vector represents a dimension of the data space, with the total number of components, N_D. The data vector formed will be denoted by a capital "$\mathbf{X}$" and the components of the vector will be denoted in lower case, "x", together given as $\tilde{\mathbf{X}}(\tilde{x})$.

Definition 1 (Data, Data Vector, Data Space)
Data: *Collection of events, $\mathcal{D}$, forming a set, described by a set of variables. The variables are often mapped to numerical values.*
Data Vector: *Vector of data values, $\tilde{\mathbf{X}} = [\tilde{x}_1, \tilde{x}_2, ...\tilde{x}_{N_D}]$.*
Data Space: *Space which contains the set of data vectors; however, the data space may extend further to the natural limits for each variable.*



(a) (b)

Figure 1: Panel 1(a) illustrates the idea of partitioning the data into bins. Each axis is binned with an index from 1 to 6, leading to the combined binning from 1 to 36, with data residing in bins: 4,5,8,9,10,11,12,14,15,16,19 and 25. Panel 1(b) illustrates the same data set, where a third dimension has been added, leading to 216 possible bins, filling 16 of them. The data represents two groups of data, shown in red and green, yet the task of separating the data is the goal of the clustering algorithm. As such, data initially is presented without knowledge of what distributions it breaks into.

Definition 2 (Numbering)
N $\leftarrow$ *the number of data within* $\mathcal{D}$.
N_D $\leftarrow$ *the number of dimensions (data vector components).*
$N_{B,i}$ $\leftarrow$ *the number of bins per component, indexed by* i*, defined in Sec. 2.2*
$N_{B,tot}$ $\leftarrow$ *the highest possible bin index value,* $N_{B,tot} = \Pi_{i=1}^{N_D} N_{B,i}$
N_P $\leftarrow$ *the number of partitions, defined in Sec. 2.3.*
N_{CL} $\leftarrow$ *the number of clustering techniques, defined in Sec. 2.4*
$N_{C,m}$ $\leftarrow$ *the number of clusters for a technique, with techniques indexed by* m.
N_{sought} $\leftarrow$ *the number of clusters sought using k-means and k-medoids algorithms.*
N_{spec} $\leftarrow$ *the number of spectral clusters sought after.*

2.2 Multi-variate Histogram, Bin Addresses and Data Reduction

The primary mechanism to reduce the total data set to a smaller size is to employ a single multi-dimensional histogram. The details of forming the bins of the multi-variate histogram are given below as they relate to how neighboring bins are determined. A histogram is a frequency distribution along one or more dimensions. For a histogram along one component, each datum is assigned a bin

number based on the minimum and maximum value, $[x_{min}, x_{max}]$, and number of bins given along each component, (N_B). The bin value for a datum along one dimension is then given by:

Definition 3 (Bin Index, Bin Address, $\tilde{b}, \tilde{\mathbf{B}}$) *Vector of bin indices,* $\tilde{\mathbf{B}} = [\tilde{b}_1, \tilde{b}_2 ... \tilde{b}_i, ... \tilde{b}_{N_D}]$.

$$\tilde{b} = \text{ceil}\left[\frac{\tilde{x} - x_{min}}{x_{max} - x_{min}} N_B\right] \tag{1}$$

The bin indices $(\tilde{b})$ form a vector, the "bin address", $\tilde{\mathbf{B}}(\tilde{b})$, assigning each datum to a multi-dimensional bin. A serial bin address index, $\tilde{k}$, is then formed from the bin address vector.

Definition 4 (Serial Bin Address Index - Datum, $\tilde{k}$) *Serialized bin index mapping the bin address vector to a single value,* $\tilde{k}$.

$$\tilde{k} = \sum_{i=1}^{N_D} \left[\left(\tilde{\mathbf{B}}(\tilde{b}_i) - 1 \right) * \prod_{q=0}^{i-1} N_{B,q} \right] \; + \; 1 \tag{2}$$

with each component indexed by i or q and $N_{B,0} = 1$.

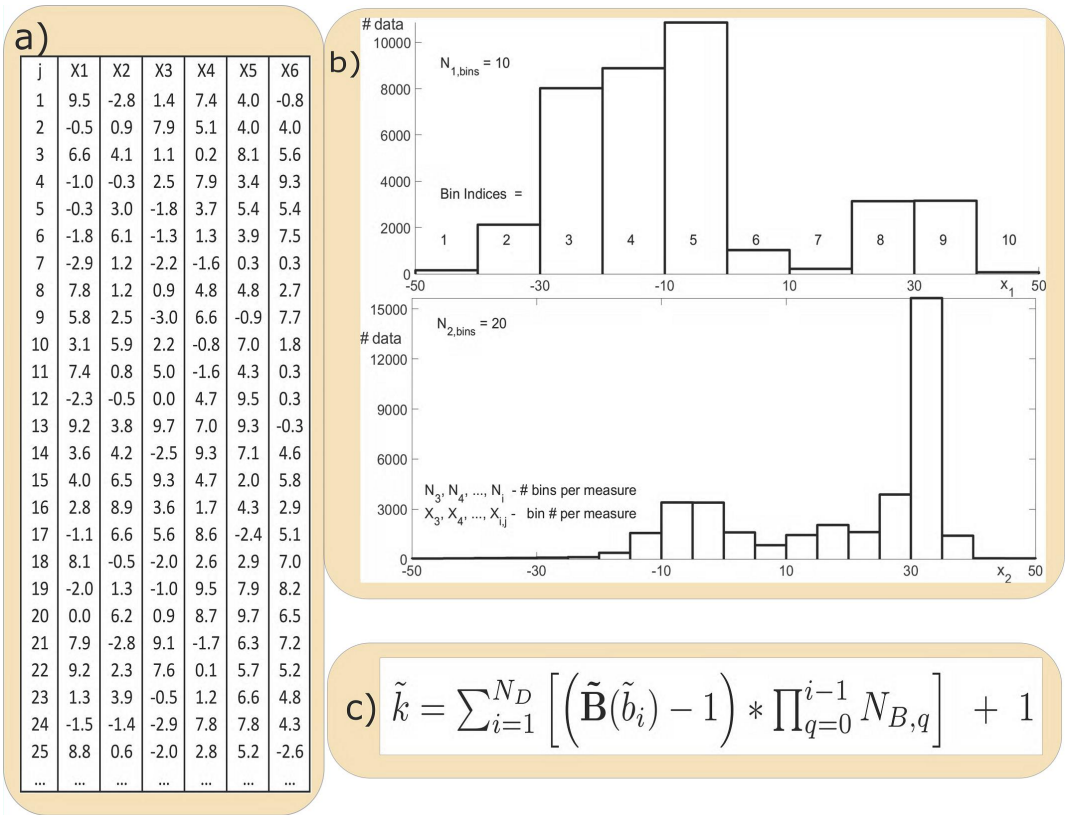
Data are grouped into multi-dimensional bins in this manner. Figure 1 illustrates the labelling of bins in 2D and 3D for a sample data set. Each multi-dimensional bin will contain a subset of the data, whose population will be used as a weighting factor in later analyses. The range of values for the bin address index can be quite large, given a high number of dimensions. The possible range of values for this index is: $1...N_{B,tot}$, where the maximal value is simply the product of the number of bins used along each dimension. Further, this index will have large gaps in regions representing empty space.

Once a serial bin address index is assigned to each datum, the population of each bin is found by summing over the data with the same bin address index, $\tilde{k}$. This can be shown as a 1D histogram using the serial bin address index as the bins and setting the number of bins equal to the largest value of $\tilde{k}$. An example of this histogram is shown in Fig. 2(d), where the vertical axis of this histogram is the population of each bin.

This process reduces the data set from N data down to a smaller set of bins containing the data, where each bin is identified by two numbers, the serial bin address index and the weight. Because each datum within a bin has the same address index, *all* data within the bin is simply defined by two values $[\tilde{k}, \tilde{w_k}]$. The number of multi-dimensional bins should be significantly less than the number of data, provided the data are not homogeneously distributed across the space.

As a possible means to reduce the data set further, only bins with a higher population will be considered for further analysis. Bins not considered are found either by selecting bins with specific low populations (bins with one datum, two data, etc...), or by finding the cumulative set of all bins whose summed population is some small percentage of the total data size. By defining the parameter, Θ_{low}, as a percentage of the total data set size, only consider bins above this threshold to define a new data set, $\mathcal{D}'$. The complementary dataset, $\mathcal{D}'' = \mathcal{D} - \mathcal{D}'$, either represents noise in the data set or low population data bins compared to the larger data set. Figure 2(e) shows an example histogram and thresholds, where the sum of all bins below threshold is less than $\Theta_{low}\ N$. The data set formed above this lower threshold, $\mathcal{D}'$, is defined as:

$$\mathcal{D}' = \left\{ \tilde{x}, \tilde{w} \in \mathcal{D} \;\middle|\; \mathcal{F}(\tilde{k}) > \Theta_{low} \mathcal{F}(N_{B,tot}) \right\}, \tag{3}$$



j	X1	X2	X3	X4	X5	X6
1	9.5	-2.8	1.4	7.4	4.0	-0.8
2	-0.5	0.9	7.9	5.1	4.0	4.0
3	6.6	4.1	1.1	0.2	8.1	5.6
4	-1.0	-0.3	2.5	7.9	3.4	9.3
5	-0.3	3.0	-1.8	3.7	5.4	5.4
6	-1.8	6.1	-1.3	1.3	3.9	7.5
7	-2.9	1.2	-2.2	-1.6	0.3	0.3
8	7.8	1.2	0.9	4.8	4.8	2.7
9	5.8	2.5	-3.0	6.6	-0.9	7.7
10	3.1	5.9	2.2	-0.8	7.0	1.8
11	7.4	0.8	5.0	-1.6	4.3	0.3
12	-2.3	-0.5	0.0	4.7	9.5	0.3
13	9.2	3.8	9.7	7.0	9.3	-0.3
14	3.6	4.2	-2.5	9.3	7.1	4.6
15	4.0	6.5	9.3	4.7	2.0	5.8
16	2.8	8.9	3.6	1.7	4.3	2.9
17	-1.1	6.6	5.6	8.6	-2.4	5.1
18	8.1	-0.5	-2.0	2.6	2.9	7.0
19	-2.0	1.3	-1.0	9.5	7.9	8.2
20	0.0	6.2	0.9	8.7	9.7	6.5
21	7.9	-2.8	9.1	-1.7	6.3	7.2
22	9.2	2.3	7.6	0.1	5.7	5.2
23	1.3	3.9	-0.5	1.2	6.6	4.8
24	-1.5	-1.4	-2.9	7.8	7.8	4.3
25	8.8	0.6	-2.0	2.8	5.2	-2.6
...	...	...	...	...	...	...

Figure 2: Data reduction process shown starting from a data set (a) to the individual histograms (b), where the bin indices are collected, leading to the calculation (c) of the unique bin address, k_j, assigned to each datum.

$$\text{where} \quad \mathcal{F}(\tilde{k}) \;=\; \sum_{\tilde{k}'=1}^{\tilde{k}} \tilde{w}(\tilde{k}') \qquad \text{for} \quad \tilde{k} = \{1...N_{B,tot}\}. \tag{4}$$

2.3 Partitions

The set of multi-dimensional bins above threshold, $\mathcal{D}'$, contains a majority of the data, defined by their bin index and population, $[\tilde{k}, \tilde{w_k}]$. Within each bin, the data all share a common bin index, $\tilde{k}$. As such, the bins of $\mathcal{D}'$ will be referred to as "partitions" for the remainder of the analysis. The tilde is dropped to indicate the new data set formed from the bins themselves, $\mathcal{P}(\mathbf{k}, \mathbf{w})$. The index, k, is the mapping from the serial bin index per datum, $\tilde{k}$ to a sequential bin index, where any gaps in the values of $\tilde{k}$ are removed, giving: $k \in \{1, 2, ..., N_P\}$, with N_P the number of bins above threshold (equal to the number of partitions). Each partition is defined by two values, the partition index, k, and the population of that partition, w_k. A partition address vector is obtained by mapping the partition index back to the original multi-dimensional bin in the data space, giving

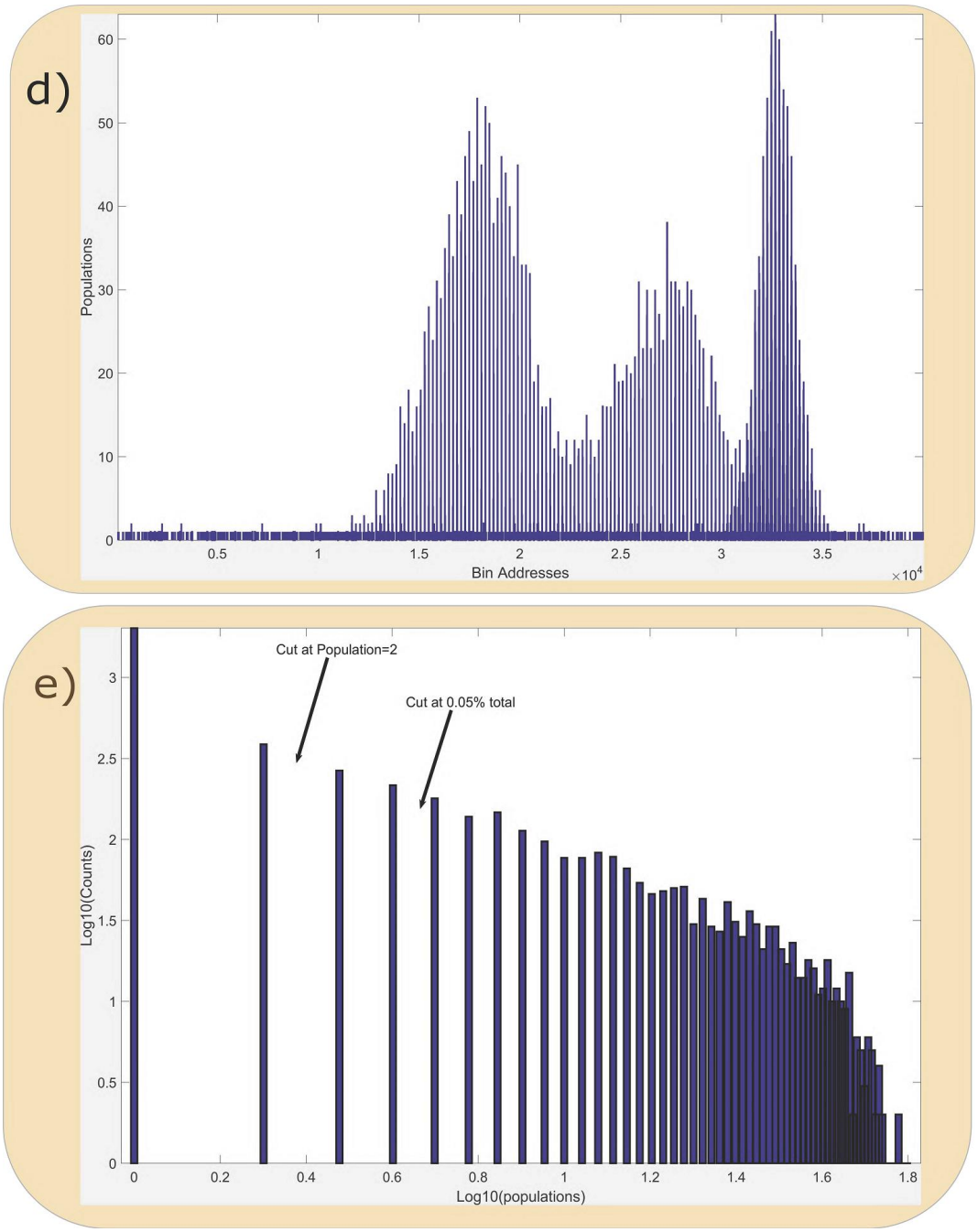


Figure 2: Continuation of the data reduction process shown. The last two panels show the histogram of the bin addresses (d) which leads to the last histogram of the bin address frequencies, where a cut is placed to eliminate the cumulative group of partitions below the lower distribution threshold, $\Theta_{low} = 0.05$ *or* a direct cut on the population of the bins at two or less.

the binning of each partition, $\mathbf{B}(b)$ and its location within the space, $\mathbf{X}(x)$. The set of partitions, $\mathcal{P}$, will be defined on the bin index space, where each partitions' width has unit length.

Definition 5 (Partition) *Set of multi-dimensional bins, $\mathcal{P}$, populated with high density data, whose population is the number of data located within a bin, effectively weighting the bin.*

Definition 6 (Sequential Bin Index - Partition, k) *Sequential bin index, $k \in \{1...N_P\}$, obtained from the serial bin index, $\tilde{k}$ removing any gaps and renumbering sequentially.*

Definition 7 (Partition Bin Address, B) *Vector of bin indices for a partition, mapping from $k \to \tilde{k} \to \mathrm{b_i}$, the individual bin indices per component, forming $\mathbf{B} = [\mathrm{b}_1, \mathrm{b}_2 ... \mathrm{b}_i, ... \mathrm{b}_{N_D}]$.*

Definition 8 (Partition Population, w) *Number of data located within a partition.*

Definition 9 (Partition space) *Space containing the set of partitions with axes defined by bin indices, $\mathbf{B}$, where each partition is defined as a unit hypercube. The partitions are arranged to form a multidimensional grid, with most of the grid empty (no partitions).*

During this analysis, values computed between any two partitions form a matrix with indices given by the current partition being investigated, k, and another partitions, labeled, ℓ. Each calculation is then represented by either a vector of values with k as the index or as a matrix of values with $[k, \ell]$ as the indices. Unless required for clarification, these array values will be shown without indices.

2.4 Clusters

The idea of a cluster is that data can be grouped based on common features found within subsets. When data is described by several variables, one possible definition is that data near one another within the data space belong to a *cluster*. Clustering can also be defined as a simple grouping of the data, which could be based alphabetically, by income, or some property that is difficult to map numerically such as an objects shape. Proximity of data to one another is a common feature of data clustering. Proximity alone can fail to cluster data that has a direction with respect to its relation to other data, such as a directed graph. By altering the definition of "proximity" to include distance measures such as path length, clustering can still be viewed as a local grouping. Broadly, clustering refers to any choice of grouping of data into subsets. This paper explores multiple clustering algorithms to later sort the clustering assignments into groupings reached by consensus.

Definition 10 (Cluster) *A subset of data defined by a common feature.*

3. Intermediary Calculations - Euclidean Distance, Nearest Neighbors, Path Length

Several calculations are common to multiple techniques which require only the partition bin address vector. These low level calculations define geometrical features of how the partitions are related to one another. These calculations are represented by a matrix, where each row represents a partition and the columns represent all other partitions. Specific algorithms for each calculation can be found in the appendix.

The Euclidean distance is calculated between all partitions within the partition space. First, the difference between two partitions' bin address are calculated for one component, $\mathbf{\Delta b_i}$. The Euclidean distance is then calculated from the sum over $\mathbf{\Delta b}_i$:

$$\mathbf{\Delta b}_i = b_{i,k} - b_{i,\ell} \tag{5}$$

$$\mathbf{\Delta R} = \sqrt{\Sigma_{i=1}^{N_D} \mathbf{\Delta b}_i^2} \tag{6}$$

with indices (k, ℓ) representing the two partitions. As each partition is a unit hypercube, the distances range from $\{1...\sqrt{N_D}\}$.

The First Nearest Neighbor matrix, **NN1**, defines the Euclidean distance between any two bins that are in contact with one another. Two partitions are in contact with one another if there exists no bin address component difference greater than one in magnitude, leading to the interpretation that they share a common geometric feature; a point, line, area, etc... The elements of the matrix given by:

$$\mathcal{I} \equiv \begin{cases} 0 & \mid \mathbf{\Delta b}_i \mid > 1, \quad \text{for any } i \\ 1 & \mid \mathbf{\Delta b}_i \mid \leq 1, \quad \text{for all } i \end{cases} \tag{7}$$

$$\mathbf{NN1} = \mathcal{I} \circ \mathbf{\Delta R}. \tag{8}$$

The **NN1** matrix is the adjacency matrix weighted by Euclidean distance, $\mathbf{\Delta R}$.

In this analysis, the usage of the Path Length is an integral part of many of the clustering algorithms. Because the data has been reduced to a set of partitions defined by a grid of integer bin address, the Path Length is the distance between any two partitions taken by stepping from one partition to another through $NN1$ steps, summing the Euclidean distance (L2-norm) on this grid *between* each step along the path:

$$\mathbf{\Delta L} \;=\; \sum_{j=k}^{\ell} \mathbf{NN1}_{j,j+1} \tag{9}$$

where the initial partition is k, interim partitions, j, up to the final partition, ℓ. Partitions are *connected* when a path is found, and for partitions that have no connecting path, the path length is set to ∞. The number of steps taken between any two partitions is the Path Count, $\mathbf{\Delta L}_C \equiv P_C$, where P_C is used when a fixed pair of partitions has been established for a calculation.

Definition 11 (Path) *A set of partitions where each member has a minimum of one shared $NN1$ partition in common with another partition in the set.*

Definition 12 (Path Length) *The sum of $NN1$ steps from one partition to another, with ∞ assigned when two partitions have no path between them.*

Definition 13 (Connected) *Set of all partitions having a non-infinite path length between them.*

Definition 14 (Line-Of-Sight: LOS) *State of a path between two partitions having the minimal L2 path length, being closest to the straight line path, which does not intersect any empty multi-dimensional bins.*

Definition 15 (Visibility) *The number of partitions that are Line-Of-Sight to a partition.*

Between any two partitions, many different paths may be taken to connect them. In order to find if two partitions have a Line-Of-Sight (LOS) to one another, only paths that fall within a rectangular convex hull are considered. Placing one partition as the origin with the second partition as the distant corner, the convex hull is the set of all partitions whose bin address indices are equal to or fall between the two partitions along each component.

Six values are calculated for specific paths unique to the LOS criteria. The true path length is the distance between any two partitions taken by stepping from one partition to another through the *least* number of $NN1$ steps and the minimal sum of Euclidean distances from step to step along the path:

$$\mathbf{\Delta L2}_T \;=\; \min\left[\sum_{j=k}^{\ell} \mathbf{NN1}_{j,j+1}\right] \tag{10}$$

with indices for the initial partition, k, final partition, ℓ, and the interim partitions up to the final partition, j. For partitions that have no connecting path, the true path length is set to ∞.

The Summed L1-norm, $\mathbf{\Delta SL1}$, is calculated between any two partitions, using the initial partition as the origin to the convex hull. Starting from the origin, the summed L1 distance from origin to each path step is calculated:

$$\mathbf{\Delta L1} = \sum_{i=1}^{N_D} \mid \mathbf{\Delta b}_i \mid \tag{11}$$

$$\mathbf{\Delta SL1} = \sum_{j=k}^{\ell} \mathbf{\Delta L1}_{kj} \tag{12}$$

Similar to the case with the Path Length, a Minimal L1-norm Path Length as well as the True L1-norm Path Length are needed to establish the LOS criteria. The Minimal Summed L1-norm is the length of the path whose summed L1-norms add together giving the least value between the two partitions, while the True Summed L1-norm is the sum of L1-norm values taken along the True Path established in Eqn.10. Finally, the squared difference along a path of the summed L1-norm to the True Summed L1-norm is used to find the true path. Details of these calculations will be given in Appendix A.2.

$$\mathbf{\Delta SL1}_{min} = \min\left[\sum_{j=k}^{\ell} \mathbf{\Delta L1}_{kj}\right] \tag{13}$$

$$\mathbf{\Delta SL1}_T = \sum_{j=k}^{\ell} \mathbf{\Delta L1}_{kj,true} \tag{14}$$

$$\mathbf{SL1VAR} = \sum_{j=k}^{\ell} \mid \mathbf{\Delta SL1}_{kj} - \mathbf{\Delta SL1}_{T,kj} \mid^2 \tag{15}$$

By finding these six values from all paths connecting any pair of partitions, the Line-Of-Sight criteria is formed using the True Path Length, Path Count, Summed L1-norm, True Summed L1-norm, Minimum Summed L1-norm and the Summed L1-norm Variance, where the Line-Of-Sight condition is discussed in section 5.4 and appendix A.2.

4. Strategy

The clustering strategy presented in this paper uses a blend of traditional data analysis via a multi-variate histogram along with standard clustering techniques such as k-means, k-medoids and spectral clustering. By binning the data onto a multi-dimensional grid, data is partitioned into regions on the grid which may be connected or separated depending on the character of the data set. A novel approach is taken by calculating the path length between any two populated partitions provided they are connected. Further, if two partitions are visible to each other by a Line-Of-Sight criteria, LOS, the relationship between them is given additional significance.

The initial data set, $\mathcal{D}$, is reduced to a smaller set of partitions based on the number of variables chosen to represent the data, the number of bins for each variable and a threshold placed on each bin to ensure the population of the bins is above a minimal value. Taking only the set of data with higher population bins to analyze further, $\mathcal{D}'$, the data has effectively been reduced to a set of *partitions*, $\mathcal{P}$, whose weight represents the population of data in the partition. Each partition is given a unique address identifying its location within the data space.

Matrices used in this analysis each have rows and columns representing the partition set, with off-diagonal entries representing values which depend on two partitions. Based on any given partitions proximity to another partition, a first nearest-neighbor, **NN1**, matrix is created where the neighborhood is defined by fixing one partition (a row) as the center of the neighborhood and defining partitions that share a common geometrical border with the center partition as non-zero, with the entries along the columns representing the Euclidean distance of the $NN1$ to the center partition.

From the set of variables defined in Sec. 4.1,3, twenty-six differing clustering techniques are employed to determine any given partitions' overall cluster identity. To *robustly* determine a final clustering assignment, a polling technique is used to arrive at a consensus amongst the clustering algorithms. While any one technique has faults, the consensus of techniques overcomes any one failure mode, giving the best all-round identification.

The group of clustering techniques applied to the analysis is called an *analysis configuration*. Analysis configurations include the 26 clustering techniques along with variations in binning choice and variable choice. As the binning is changed, resolution of the partitions changes, leading to possible changes in cluster assignment. Also, the choice of variables used to represent the data defines the number of dimensions as well as character of the data manifold. When agreement is reached across multiple techniques, differing bin resolutions as well as data manifolds, if data consistently cluster in one manner, then the data is assigned *robustly* to its cluster. This study presents 26 different clustering algorithms, multiplied by each choice in binning and variables, leading to new variations which will likely lead to new cluster assignments to individual data. Each choice of (variables, binning, clustering algorithms) creates a configuration for an analysis. For each configuration, the consensus approach then determines clusters based on the best agreement between the assignments. The best choice for clustering can then be determined by the analyst of the data as their needs might favor one configuration over another.

4.1 Arrays: Indices, Scalars, Vectors and Matrices

Adding to the definitions already given, a comprehensive list of several arrays integral to the clustering algorithms are given here:

Definition 16 (Indices)

i $\leftarrow$ *component index (vectors), dimension index (space)*
k, ℓ, j $\leftarrow$ *partition indices*
m $\leftarrow$ *clustering algorithm index*

Definition 17 (Scalars)

$\tilde{x}_i$ $\leftarrow$ *values of the data for each component*
$\tilde{b}_i$ $\leftarrow$ *bin index value (data) for component i*
$\tilde{k}$ $\leftarrow$ *serialized bin index value for each datum*
b_i $\leftarrow$ *bin index value (partition) along component i*
k $\leftarrow$ *sequential bin index value for a partition*

Definition 18 (Vectors)

$\mathbf{k}$ $\leftarrow$ *array of indices for partitions*
$\mathbf{w}$ $\leftarrow$ *array of weights (populations) for partitions*
$\mathbf{k}'$ $\leftarrow$ *array of partition indices for maximal weights*

Definition 19 (Matrices)

$\tilde{\mathbf{X}}$ $\leftarrow$ *Data values, size:* (N, N_D)*, requires:* $\tilde{x}_i$
$\tilde{\mathbf{B}}$ $\leftarrow$ *Bin indices, size:* (N, N_D)*, requires:* $\tilde{b}_i$
$\mathbf{X}$ $\leftarrow$ *Partition values, size:* (N_P, N_D)*, requires:* x_i
$\mathbf{B}$ $\leftarrow$ *Partition bin indices, size:* (N_P, N_D)*, requires:* b_i
$\mathbf{\Delta b}_i$ $\leftarrow$ *Partition bin index differences, size:* (N_P, N_P)*, requires:* b_i
$\mathbf{\Delta R}$ $\leftarrow$ *Bin based Euclidean distance between two partitions, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta b_i}$
$\mathbf{NN1}$ $\leftarrow$ *First Nearest Neighbor, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta b_i}$
$\mathbf{\Delta L}$ $\leftarrow$ *Path Length - distance between two partitions, size:* (N_P, N_P)*, requires:* **NN1**
$\mathbf{\Delta L2}_T$ $\leftarrow$ *True Path Length - minimal distance two partitions, size:* (N_P, N_P)*, req:* **NN1**
$\mathbf{\Delta L}_C \equiv \mathbf{P_C}$ $\leftarrow$ *Path Length count - # of steps between two partitions, size:* (N_P, N_P)*, req:* **NN1**
$\mathbf{\Delta L1}$ $\leftarrow$ *L1-norm between two partitions, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta b_i}$
$\mathbf{\Delta SL1}$ $\leftarrow$ *Summed L1-norms for all steps along the path, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta L1}$
$\mathbf{\Delta SL1_{min}}$ $\leftarrow$ *Minimal Summed L1-norms steps along the path, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta L1}$
$\mathbf{\Delta SL1}_T$ $\leftarrow$ *True Summed L1 - distance along L2 true path , size:* (N_P, N_P)*, req:* $\mathbf{\Delta SL1, SL1_{min}}$
$\mathbf{SL1VAR}$ $\leftarrow$ *Summed L1-norm variance path to true path, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta SL1, SL1_T}$
$\mathbf{LOS}$ $\leftarrow$ *Line-Of-Sight condition of two partitions, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta L, SL1}$
$\mathbf{LAPNN1}$ $\leftarrow$ *Numerical Laplacian using* $NN1$*, size:* (N_P, N_P)*, requires:* **NN1**
$\mathbf{LAPLOS}$ $\leftarrow$ *Numerical Laplacian using LOS, size:* (N_P, N_P)*, requires:* **LOS**
$\mathbf{LAPGAU}$ $\leftarrow$ *Numerical Laplacian of Gaussian, size:* (N_P, N_P)*, requires:* $\mathbf{\Delta R}, \sigma$
$\mathbf{CONN}$ $\leftarrow$ *Connection between two partitions, size:* (N_P, N_P)*, requires:* **NN1**
$\mathbf{CLUS_m}$ $\leftarrow$ *Cluster for technique m with members for each row per cluster, size:* $(N_{CL}, N_{C,m})$

For matrices used to represent partition calculations (N_P, N_P), each row represents one partition and each column represents all other partitions.

4.2 Data Test Cases - 12 Shapes

A test bank of data sets was used to develop the algorithms for this study comprised of various shapes, both connected and disconnected as well as a point cloud in both 2D and 3D. In each

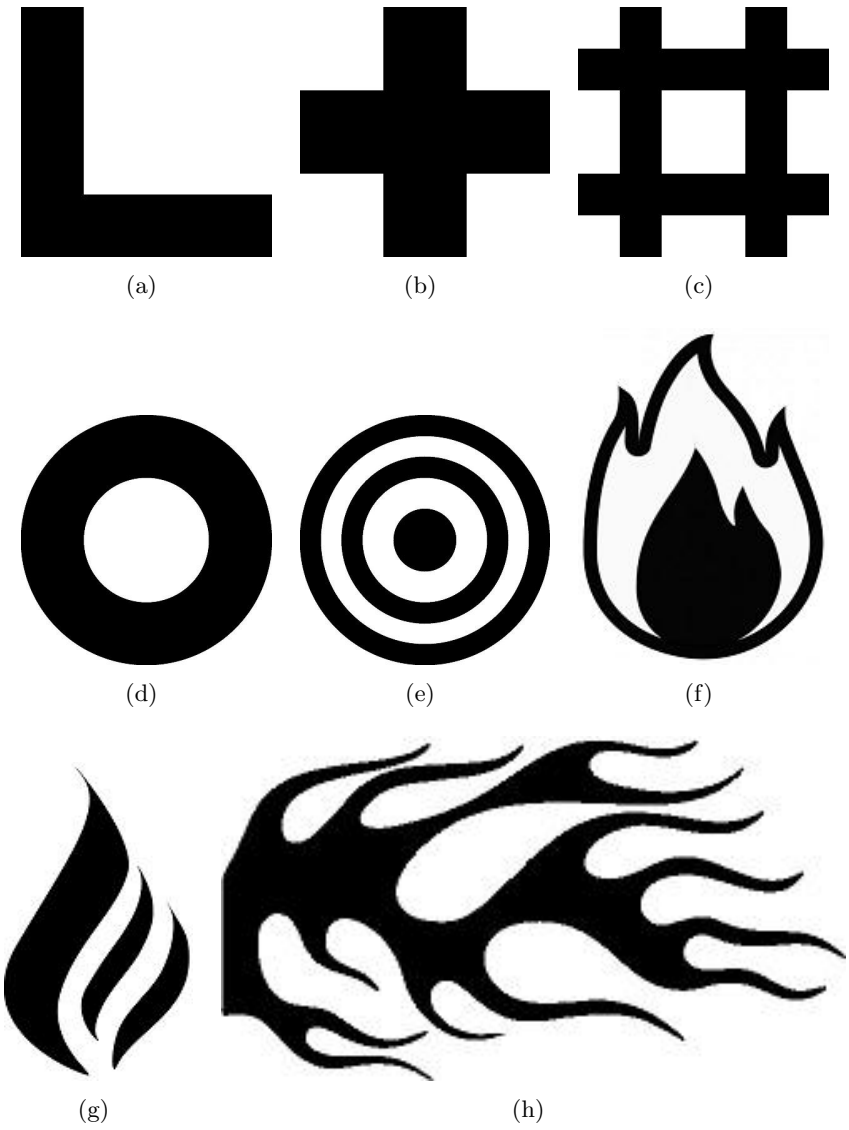


Figure 3: Test bank of 8 shapes: L, Plus-1, Plus-2, Concentric-1, Concentric-2, Flame-1, Flame-2, Flame-3.

of the point clouds, four gaussian distributions were placed near one another, with three densely populated regions and a fourth low density gaussian which spans the domain. The point clouds were further varied in 2D by creating two differing sets, one where the three dense set of points are clearly separated from each other and another set where the three dense populations have overlapping regions. Similarly in 3D, two point clouds were made where the first has the three high density regions fully separated and the second has two of the three overlapping with a lone third set. Figures 3,4 illustrate the test bank used. For each test, a differing feature was sought to examine. Table 1 lists the testbank set as well as the features sought to examine in each case. The first test is the simple L as discussed in section 5.5. The *Plus1* and *Plus2* cases are extensions to

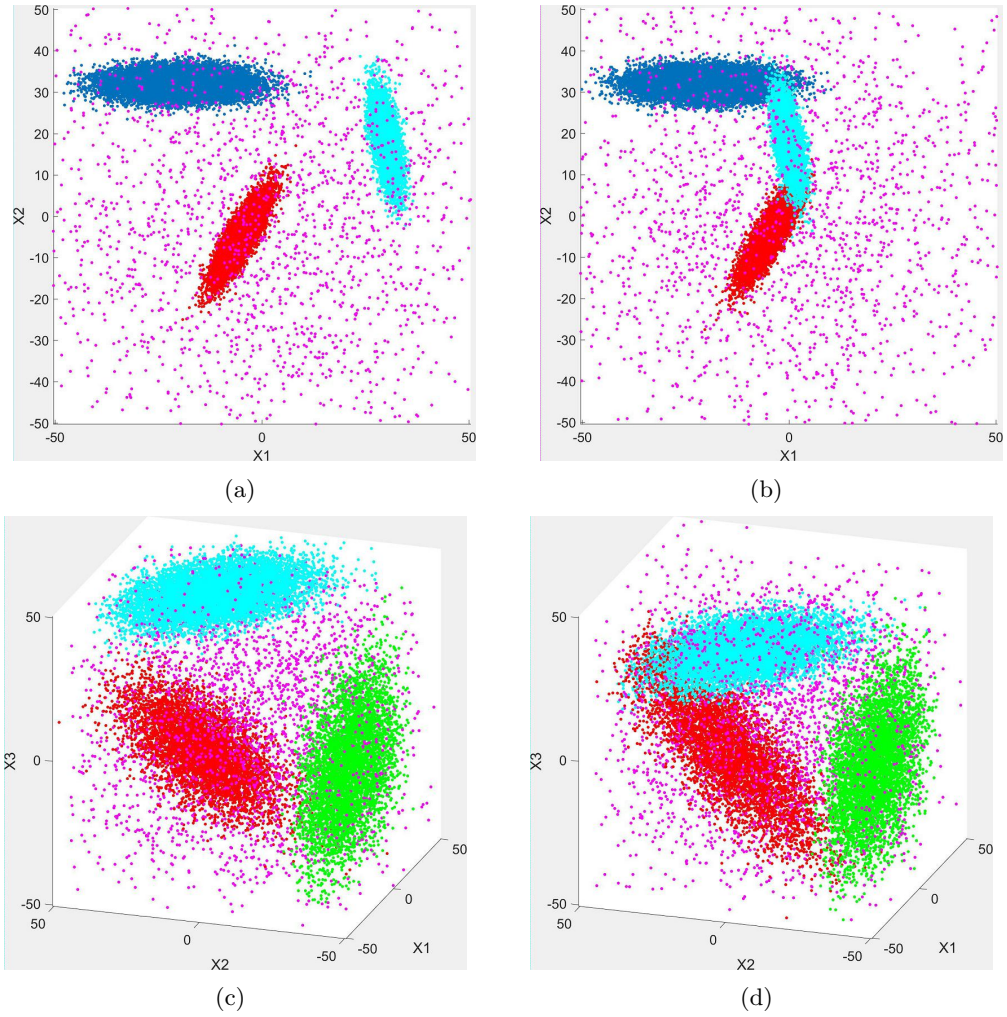


Figure 4: Test bank of 4 distributions: Data2D-1 (a), Data2D-2 (b), Data3D-1 (c) and Data3D-2 (d).

the L case where the symmetry of the algorithms can be understood as well as how the routines respond to void regions in the data, *Plus2* case. *Concentric1* and *Concentric2* test how the routines respond to curved domains with symmetry and whether the domain is connected or not. *Flame1, Flame2* and *Flame3* test how asymmetry is dealt with as well as connected versus disconnected regions. *Flame3* also tests how well "tendrils" or filamentary data is handled. As a test of a 2D point cloud, *Data2D-1* and *Data2D-2* test how well four gaussian point clouds can be clustered for the case of three separated clusters,*Data2D-1*, and three close-by clusters, *Data2D-2*, where the fourth gaussian is evenly distributed across the domain as noise. *Data3D-1* and *Data3D-2* show the point cloud in 3D of four gaussian distributions similar in definition to the 2D cases, where the first are three disconnected elliptical distributions with a fourth acting as a background of noise, while the second shows the same three elliptical distributions moved closer to one another such that two of the tails overlap. For all cases other than the point clouds, the data is derived from an image,

		Test Bank Data Sets					
Labels	#	dim	size (pixels/pts)	connected	symmetry	plateau	filamentary
L	1	2D	1200x1200	√	X	√	X
Plus1	2	2D	1200x1200	√	√	√	X
Plus2	3	2D	1200x1200	√	√	√	X
Concentric1	4	2D	1200x1200	√	√	√	X
Concentric2	5	2D	1200x1200	X	√	√	X
Flame1	6	2D	1200x1200	√	X	√	X
Flame2	7	2D	1200x1200	X	X	√	X
Flame3	8	2D	1200x1200	√	X	√	√
Data2D-1	9	2D	200,000	X	X	X	√
Data2D-2	10	2D	200,000	√/X	X	X	√
Data3D-1	11	3D	200,000	X	X	X	√
Data3D-2	12	3D	200,000	√/X	X	X	√

Table 1: Test bank data sets.

where a binary set of points is established for all 8-bit grey-scale values above 100 (1) or below (0). The image sizes when possible are 1200x1200, unless the aspect ratio prevented that exact size. The point clouds are based on four distributions with a summed value of 200,000 points. Figures 3(a)-4(d) show the test bank in this order: *L, Plus1, Plus2, Concentric1, Concentric2, Flame1, Flame2, Flame3, Data2D-1, Data2D-2, Data3D-1, Data3D-2.*

5. Clustering Algorithms

This section discusses the clustering algorithms used in this approach. Some techniques are standard approaches, but several are variations on existing techniques with new approaches. The new approaches involve changing the distance metric used from a traditional L2-norm to a path length along a grid of partitions. Along with investigating path length based clustering, a Line-Of-Sight, LOS, criteria is also developed. An alternative approach to using spectral clustering is also used, utilizing a different set of eigenvectors to establish clusters, as well as an alternative to the traditional Laplacian operator. Once all twenty-six clustering techniques are used to assign a cluster identity, an overall cluster identity is given to each data based on the consensus of the set of techniques, similar to how ensemble modeling reduces systemic errors for simulation. Table 2 lists the twenty-six techniques used in this study.

5.1 K-Means and K-Medoids Clustering - KMEANS, KMEDOIDS

K-means is a well established clustering technique, seeking from a data set, the lowest possible distance to a set of mean positions based on the overall positions of the data. As an input, the user is required to provide a number of means to seek ($k \equiv N_{sought}$), at which point, the algorithm proceeds to find exactly that number of mean positions, regardless of whether the data actually cluster into as many groups. As a result, k-means suffers from an inability to "stop early" in its search for clusters. Silhouette plots help determine the appropriate number of means to seek, making this process computationally expensive as it requires several passes to find reasonable clustering. Finally, data is not always meaningful as a continuous valued set, making the concept of a "mean" of the data invalid. Examples include data sets based on finite categorizations or possible

mixed data sets of discreet and continuous data. Finally, the mean location may be located outside of the data set, making the position of the mean difficult to interpret in terms of the data axes, examples include a concentric distribution, where two differing data sets are interweaved, yet share similar mean locations in the space. Several variants on k-means exist which address many of its problems, however, its interpretation remains problematic.

K-medoids is similar in its approach to clustering as k-means, yet assigns positions based on locations of data points *within the set* and not mean locations (medoids). Further, k-medoids minimizes the dissimilarity for each medoid compared to other data points within a proposed cluster compared to other clusters. As a result, k-medoids tend to provide more meaningful cluster definitions which are less sensitive to noise Kaufman and Rousseeuw (2009). Initially, a number of medoids is sought after ($k \equiv N_{sought}$), however, once k-medoids has found the cluster definitions that have the least dissimilarity, the search ends, allowing the algorithm to stop early.

By shifting the analysis from individual datum to partitions with weights, the K-means and K-medoids algorithms are adjusted to accommodate the weighted bins. All calculations for distance between two partitions are multiplied by the weight of each partition and any centroid calculation must be treated as a weighted value.

5.2 Maxima Clustering - Global And Path Length

Section 2.3 discussed how the initial data set is reduced to a smaller set of multi-dimensional bins referred to as "partitions". Each partition has a unique location in the bin address space formed from the bin index given along each component. Further, each partition has a "weight" which is equivalent to the population of data located in that particular bin.

Clustering by proximity is a standard approach to finding which data are similar to one another based on how close the data are within the space. This approach can be problematic in that data from one distribution can be near data from a separate distribution, yet might be clustered together due to their mutual closeness. Two algorithms are applied that cluster partitions based on the weights (population) of the partitions and their proximity to a nearby local maxima of weights. A cluster which is *global* in scope has a clustering relationship which does not require the partitions be in direct contact to one another, in contrast to the *path length* analysis which requires a cluster to be contiguous. The global scheme can associate partitions together even if they are not connected to one another by using the Euclidean distance between partitions as the distance metric. In contrast, the path length algorithm only associates partitions that are connected to one another via a path and uses the path length as the distance metric. Partitions in both schemes are assigned to peaks in the weight distribution based on the distance from any given partition to a local maxima as well as the slope to the nearest peak. Local maxima, peaks and slopes are defined in the appendix A.1. A peak and all of the partitions associated with it are then assigned a cluster identification number.

5.2.1 Global Maxima Clustering - MAXGLOB

The MAXGLOB scheme seeks to form clusters based on the proximity of a partition to local maxima among the population of the partitions. Appendix A.1 details how peaks are determined relative to nearby partitions using two values, the Euclidean distance to a peak, $\mathbf{\Delta R}$, as well as the slope between the value of the current partitions weight and the weight of a nearby local maxima. When several peaks are considered to be associated with a partition, a hierarchy exists among the peaks

		Clustering Algorithms							
Labels	#	connected	balanced	LOS	sens. noise	fixed N	ID	Cluster	Group
KMEANS	1	X	X	X	X	√	X	√	X
KMEDOIDS	2	X	X	X	X	√	X	√	X
MAXGLOB	3	X	X	X	X	X	X	√	X
MAXPATHL	4	√	X	X	X	X	X	√	X
CONN	5	√	X	X	X	X	X	X	X
LOS-MAXVIS	6	√	X	√	X	X	√	X	X
LOS-MUTUAL	7	√	√	√	X	X	√	X	X
SPEC-NN1-12-2DHIST	8	√	X	X	√	X	X	X	X
SPEC-NN1-12-KMEANS	9	√	X	X	√	√	X	X	X
SPEC-NN1-12-KMEDS	10	√	X	X	√	√	X	X	X
SPEC-NN1-23-2DHIST	11	√	√	X	√	X	X	X	√
SPEC-NN1-23-KMEANS	12	√	√	X	√	√	X	X	√
SPEC-NN1-23-KMEDS	13	√	√	X	√	√	X	X	√
SPEC-LOS-12-2DHIST	14	√	X	√	√	X	√	X	X
SPEC-LOS-12-KMEANS	15	√	X	√	√	√	√	X	X
SPEC-LOS-12-KMEDS	16	√	X	√	√	√	√	X	X
SPEC-LOS-23-2DHIST	17	√	√	√	√	X	√	X	√
SPEC-LOS-23-KMEANS	18	√	√	√	√	√	√	X	√
SPEC-LOS-23-KMEDS	19	√	√	√	√	√	√	X	√
SPEC-GAU-12-2DHIST	20	X	X	X	X	X	X	√	X
SPEC-GAU-12-KMEANS	21	X	X	X	X	√	X	√	X
SPEC-GAU-12-KMEDS	22	X	X	X	X	√	X	√	X
SPEC-GAU-23-2DHIST	23	X	√	X	X	X	X	X	√
SPEC-GAU-23-KMEANS	24	X	√	X	X	√	X	X	√
SPEC-GAU-23-KMEDS	25	X	√	X	X	√	X	X	√
LMH-POS	26	X	X	X	X	√	X	√	X

Table 2: Clustering techniques for 26 algorithms highlighting requirements, pros and cons in each case. Some algorithms required partitions to be connected in order to search for clustering within the connected region. Clusters can be feature driven or can even the distribution of cluster assignments (balanced, group). LOS is a criteria for some clustering, which in turn can help identify data distributions. Finally, some algorithms treat isolated partitions on equal footing with larger connected regions, making the clustering sensitive to noise.

to properly associate any given partition to the correct peak. The results of the MAXGLOB scheme is shown in Sec. 6.1 and most closely resemble clustering done via k-means as well as k-medoids. The advantage of using the MAXGLOB approach is that clustering is performed on partitions ($\mathcal{P}$) and not directly on the original dataset ($\mathcal{D}'$), giving it a significant computational advantage.

5.2.2 Path Length Maxima Clustering - MAXPATHL

The MAXPATHL scheme seeks to form clusters based on the closest distance from partitions to local maxima in the populations of partitions using the path length, $\mathbf{\Delta L}$, as the distance metric. The clustering relationship is based on proximity *within a connected* group of partitions, making the algorithm sensitive to the shape of the distribution by using the path length. Similar to the MAXGLOB algorithm, the path length and slope are used in conjunction to determine which peak

is best associated with a partition where $\mathbf{\Delta R}$ is simply replaced by $\mathbf{\Delta L}$. Figure 7 in Sec. 6.1 compare the results of MAXGLOB and MAXPATHL to KMEANS, KMEDOIDS.

5.3 Clustering via Connection - CONN

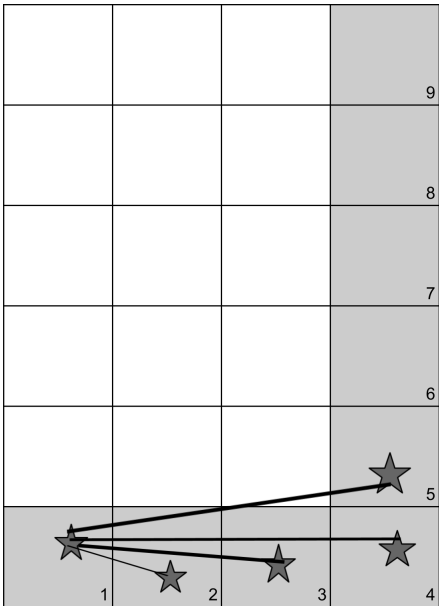
In cases where local clusters of partitions are sparsely found within the data space, a simple clustering algorithm is to determine which partitions are connected to one another using first nearest neighbor steps, **NN1**. All partitions connected to one another, **CONN**, are given a single cluster ID. Any partitions that are alone, disconnected from all other partitions, are given their own cluster ID of zero. By giving all isolated partitions a cluster ID of zero, allows for a simple cluster designation based on a logical value ($\mathbf{CONN} > 0$) which then distinguishes between connected regions in the partition data space and isolated regions, which may be viewed as noise. An isolated partition may contain a large amount of data, in which case, care should be taken to consider the weight of each partition such that a better method of looking for noise would be the condition $((\mathbf{CONN} = 0) AND (\mathbf{w} < \Theta_{noise}))$, where Θ_{noise} is a user defined threshold.

The simplest technique for determining whether a partition is connected to another within a set of partitions is to begin with the first nearest neighbor matrix, **NN1**. If this matrix is block diagonal, then all partitions within a block on the diagonal are connected to each other and are disconnected from any partitions in a separate block. A **NN1** matrix is most likely not block diagonal initially, however, it can be readily made block diagonal using a Dulmage-Mendelsohn decomposition (1958). A **NN1** matrix is most likely not block diagonal initially, however, can be readily made block diagonal using the decomposition of Dulmage and Mendelsohn (1958). In this approach, for the symmetric **NN1** matrix, a series of row and column interchanges occurs until the matrix has become block diagonal, at which point, the blocks represent subsets of connected partitions.

Many problems in clustering are made difficult by having multiple clusters in close proximity to one another, yet not being contiguous. By using the connection criteria, as long as the data space has been resolved well enough by choosing appropriate bin sizes along each axis, clusters should be resolvable to within the resolution established across the space.

5.4 Clustering by Line-Of-Sight - LOS

Clustering by Line-Of-Sight is motivated by the idea that data within a convex hull has a higher chance of being correlated than data separated by distance and visibility. Although distributions of data may follow traditional gaussian shapes, it is also possible for a distribution to be bent within the data space. Distributions of data can appear to follow a curve within the space which may simply reflect a functional dependence between one or more variables, yet it may also form when two or more distributions have means near one another and tails that overlap, leading to the appearance of a single bent distribution. Clustering via **CONN** will associate all data in the bent distribution, however, checking whether two data lie within a convex hull more closely associates those data with one another. A Line-Of-Sight, LOS, criteria determines which partitions are convex to one another. As examples, figures 4(a)-4(d) illustrate several distributions which have both convex regions as well as overlapping tails of distributions. In this discussion, the term *visibility* refers to the number of partitions that are LOS to a specific partition. A detailed discussion is given in appendix B of the algorithms used.



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

$NN1$ matrix

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

"L" LOS matrix

$$\begin{pmatrix} 5 & 5 & 5 & 5 & 5 & 3 & 3 & 3 & 3 \\ 5 & 5 & 5 & 5 & 5 & 3 & 3 & 3 & 3 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 3 & 3 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 3 & 3 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 3 & 3 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 3 & 3 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \end{pmatrix}$$

LOS^2 matrix

$$\begin{pmatrix} 5 & 5 & 5 & 5 & 5 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 5 & 5 & 0 & 0 & 0 & 0 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 5 & 5 & 9 & 9 & 9 & 7 & 7 & 7 & 7 \\ 0 & 0 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 0 & 0 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 0 & 0 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \\ 0 & 0 & 7 & 7 & 7 & 7 & 7 & 7 & 7 \end{pmatrix}$$

$LOS^2 \cdot LOS \equiv LLL$ matrix

Figure 5: Simple example to illustrate the ideas behind LOS clustering. This "L" shaped domain has nine populated bins. Starting from the bottom left to right then and moving upwards,the bins are numbered initially by rasterizing the domain, then contracting the bin indices to simply number from #1-9. Beginning with bin #1, the line-of-sight bins are indicated by the starred (★) bins. Matrices calculated for the simple example, the **NN1** (upper left) **LOS** (upper right), $\mathbf{LOS^2}$(lower left), $\mathbf{LOS^2} \cdot \mathbf{LOS} \equiv \mathbf{LLL}$(lower right).

The **LOS** matrix is formed where each row represents a partition and each column represents all other partitions where a logical value indicates whether the two are LOS. The **LOS** matrix is symmetric, further, squaring the **LOS** matrix, $\mathbf{LOS^2}$, gives a matrix whose values along each row tally the number of partitions which are mutually LOS to one another. The $\mathbf{LOS^2}$ matrix may contain non-zero values for partitions that share a mutual visibility with any given row, but are not LOS to the current partition. An example will given next to expand on this idea. In order to handle these entries in $\mathbf{LOS^2}$ that are not present in the **LOS** matrix, a Hadamard product is taken between **LOS** and $\mathbf{LOS^2}$ yielding a third matrix, **LLL**. The LOS criteria is detailed in the appendix B.

5.5 Simple Example: L

A simple example can serve to realize the idea of these matrices and how they interact with one another. Consider a simple distribution of partitions forming a 6x4 grid connected to each other in an "L" configuration as shown in Fig. 6. In order to follow the serialization of partitions given in Sec. 2.2, it helps to invert the L from right to left ($\leftrightarrow$). Numbering the partitions along the bottom row first (1-4), then along the vertical right side (5-9). In this case, there are only nine partitions connected to each other, requiring a 9x9 matrix to represent the information. As each partition is connected to all of the other partitions, the **CONN** matrix is full, with values of one. The **NN1**

matrix reflects which partitions share a common geometrical feature. The **LOS**, **LOS^2** and **LLL** matrices show which partitions can "see" another partition. Note that partition five is visible to partition one, meaning that partitions can see the edges of one another. From the matrices shown, partitions (3,4,5) form a cluster with the maximal visibility, followed by partitions (6,7,8,9) then (1,2) (LOS-MAXVIS). Partitions (3,4,5,6,7,8,9) form a cluster with the highest mutual visibility followed by (1,2) with the lowest (LOS-MUTUAL).

In the language established in previous sections, the data points are pixels gathered from an image where two axes are used to describe the image, making the data space 2D. The data are integers for the (x_1, x_2) positions ranging from $(0...4], (0...6]$ respectively. Each axis is binned horizontally (4 bins) and vertically (6 bins). The data bin addresses are:

$$[(1,1),(1,2),(1,3),(1,4),(2,4),(3,4),(4,4),(5,4),(6,4)]$$

and the serial bin address ranges from $[1...24]$, of which only: $[1, 2, 3, 4, 8, 12, 16, 20, 24]$ are filled. These filled bins become the partitions, labeled by the partition sequential bin addresses: $[1...9]$. The partition space is also 2D with partition bin addresses having the same values as the data bin addresses.

5.5.1 LOS Clustering With Maximal Visibility - LOS-MAXVIS

The **LOS** matrix contains for each row the logical status of which partitions are LOS to the current partition. Further, the **LLL** matrix shows the number of mutually visible partitions within LOS of the current. From the **LLL** matrix, two values can be used to determine clustering using LOS. The highest value in the **LLL** matrix indicates which partitions are within LOS of the *most* other partitions. These highest valued **LLL** partitions have the *maximal visibility*, LOS-MAXVIS, of the set of partitions that are LOS. An example would be any partition that is located at an intersection of several distributions of partitions. Consider the test cases: *L* and *Plus1*, where the corner of the *L* and the center of the *Plus1* will have maximal visibility.

Clustering by LOS-MAXVIS is achieved by taking the diagonal from **LLL**, which gives the total visibility of each partition. A histogram of the visibility is shown in Fig. 7. The horizonal axis indicates the number of partitions LOS to others. Being a histogram, the vertical axis is the number of partitions sharing a common visibility value. Starting from the maximal value of the visibility, a cluster is formed by taking all partitions sharing the maximum or nearby, defined by including all bins in the histogram starting from the topmost until a minimum in the bins is reached. In the case of the simple L, the most visible partitions are the corner partitions with a **LLL** value of nine. Further clusters are then identified by taking partitions associated with the next highest visibility bin in the histogram, beginning where the last set left off, and including all partitions with successively lower visibilities until the next minimum in the bins is reached. This process continues until the set of partitions is fully associated with clusters.

5.5.2 LOS Clustering With Mutual Visibility - LOS-MUTUAL

The **LLL** matrix can alternatively be used to cluster partitions with the *highest mutual visibility* (LOS-MUTUAL) by selecting clusters with the most common shared **LLL** value instead of the maximal value. In this manner, clusters are formed around partitions that can mutually see each other the most. Clustering by LOS-MUTUAL is achieved by first creating a histogram of **LLL** values over all partitions. Figure 6 shows the histogram of visibility values for the *Data3D2* test case. The horizonal axis indicates the visibility of partitions. The vertical axis is number of partitions

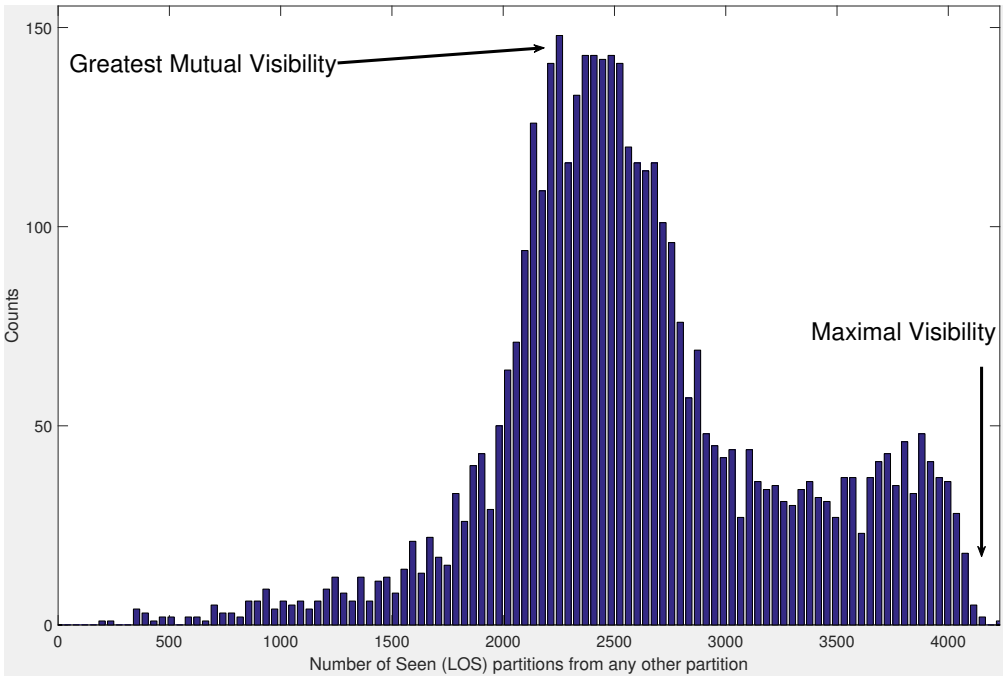


Figure 6: Histogram of **LLL** values, the mutual visibility from one partition to all others for the *Data 3D-2* case. The horizonal axis indicates the number of mutually seen LOS to this partition. The vertical axis is number of other partitions LOS to the current. From the horizontal axis, this partition is visible to ≈ 4100 other partitions, giving it a MAXVIS of 4100. Also, this partition is part of a group of partitions that maximally and mutually see each other with a visibility of ≈ 2200 whose set size is ≈ 150 partitions in membership, starting off the search for MAXMUT clusters. In both cases, the starting point for the cluster search defines which other partitions are near to the goal, either maximal visibility or highest mutual visibility. The clusters are formed by grouping LOS partitions to the current around the closest peak in the histogram, where peaks are separated by the basins.

sharing a common visibility value. Starting from the bin with the most frequent visibility, a cluster is formed by seeking the minima on both sides of the peak in the histogram nearest the most populated bin. Once the lower and upper bins are found, all partitions which have any visibility values in **LLL** within this range are clustered together. After identification, these clusters are removed from future searches by setting the rows and columns of the **LLL** matrix to zero for these partitions. The process is repeated until all partitions are identified. LOS-MUTUAL clustering finds the largest sets of partitions that are LOS to each other first, then searches for the next largest set of partitions that do not include the first set and so on.

In the case of the simple L, the highest mutually visible partitions are the partitions forming the long arm of the L. For the Chesapeake Bay, all partitions with an visibility between 1700 up to 3000 are included in the first cluster found.

5.6 Spectral Clustering

Spectral clustering [Chung (1997); von Luxburg (2007)] represents data as a graph, where data becomes vertices and relationships between data points are represented by edges and weights in the graph. This analysis uses the **NN1** matrix, the **LOS** matrix as well as a Gaussian radial basis function to form a graph. A Laplacian operator is formed by setting the degree of vertices along the diagonal of the Laplacian matrix with off diagonal elements set to a negative weight factor. When using the **NN1** matrix, the Euclidean distance is given a binary value whereas the **LOS** matrix uses the binary connections between partitions that are LOS to one another. For the Gaussian radial function, the Euclidean distance between any two partitions is used as the basis to an exponential term with a constant allowing control over the degree of locality. Radial spectral clustering differs from the other two methods in that **NN1** and **LOS** clustering require a connection to exist between two partitions, limiting the scope of how the clusters form. Radial clustering is performed over all pairs of partitions regardless of connection. Given two parameters, $(\sigma_{rad}, \Theta_{rad})$, the Gaussian radial approach can be limited by setting the falloff (σ_{rad}) as well as a distance cutoff (Θ_{rad} - measured in units of σ_{rad}), where all distances greater than the cutoff set the Laplacian of Gaussian term to zero. In all cases, the analysis that follows is similar. The eigenvectors are calculated for the Laplacian, where the lowest two eigenvectors are typically used to define a *new data space* using each eigenvector as a basis. The partitions are then mapped to the eigenspace and clusters within the space are sought using novel 2D clustering techniques, either KMEANS, KMEDOIDS or a simple 2D histogram over the domain.

This analysis employs all three clustering techniques in the eigenspace as well as explores using two differing sets of eigenvectors, the lowest pair (1,2) as well as the next two lowest pair (2,3). In the first case using eigenvectors (1,2), the first eigenmode accentuates a single large feature within the eigenspace, where the second eigenvector segments the space into two symmetric regions. By using the next lowest pair of eigenvectors, surpassing the lowest eigenmode, the partitions are segregated differently, more evenly distributed, leading to a different interpretation of clustering.

5.6.1 Spectral Cluster Gathering By K-Means, K-Medoids or 2D Histograms

Once the eigenspace has been populated with the partitions, k-means, k-medoids as well as traditional 2D histograms can be used to collect the partitions and assign them to cluster IDs. K-means and k-medoids have been discussed earlier in Sec. 5.1 as to their strengths and weaknesses. As an alternative approach to finding the clusters within the eigenspace, simply histogram the 2D eigenspace and assign to each non-zero bin a different cluster ID (2DHIST). This approach has the advantage of simplicity and finds exactly the number of clusters that fill bins within the eigenspace, not requiring an initial guess as the number of possible clusters, as in the case of k-means or k-medoids, however a maximum possible count of clusters is set by the bin size of the eigenspace (N_{spec}, N_{spec}).

5.7 Clustering By Coarse Position (LMH-POS)

The most obvious form of clustering is to associate a partition solely by its *position* (LMH-POS) using a coarse binning within the partition space. By setting the number of bins along each dimension to three, the bins are interpreted as being *low, medium* or *high* for the values represented along each axis. In this case, the *sequential partition bin index*, **k**, becomes the cluster ID, with the maximum number of possible clusters at 3^{N_D}, for the three bins along each axis. This approach is

a coarse designation for clustering as it employs no complicated algorithms, and data with similar values are associated irrespective of all other factors. When handling large data sets, this approach allows for a quick look at where the data reside within the larger space.

This approach suffers from many problems in that data in one bin will not be clustered with data from a neighboring bin no matter how close in proximity the two are to one another. In the extreme case of using only three bins per axis, the data are characterized in the crudest sense with no refinement for the shape of a distribution or even the relative sizes of the distribution.

One advantage to this approach is that it is easy to understand, even while spanning multiple dimensions. As an entry point for a discussion about the data, the LMH-POS approach eschews complication for simplicity and frames the discussion to evolve towards the nuances within the data set, its shape, dimension, span, relative size, etc...

5.8 User Choice in Clustering - Variables, Thresholds, Binning

Throughout this study, several parameters have been defined which affect the outcome of the clustering algorithms. Each of these user defined values reflects how knowledge of the data set can lead to appropriate choices for clustering. A list of the user defined variables, thresholds and binning choices is given here:

Definition 20 (Variables)

x_i $\leftarrow$ *variables chosen forming the data space.*

N_D $\leftarrow$ *the number of dimensions.*

Definition 21 (Binning)

$N_{B,i}$ $\leftarrow$ *the number of bins for component i.*

N_{spec} $\leftarrow$ *the number of spectral clusters sought after.*

Definition 22 (Thresholds)

Θ_{low} $\leftarrow$ *threshold for bins with low population, set either as an integer cutoff for the per bin population, or as a percentage cutoff for the sum of all low population bins compared to the total population.*

$\Theta_{pathmin}$ $\leftarrow$ *threshold similar to* Θ_{low} *to ignore partitions whose maximum pathcount is low, effectively removing isolated partitions.*

$\Theta_{overlap}$ $\leftarrow$ *threshold set for LOS maximum visibility gathering process requiring an percentage of overlap between smaller clusters and total visible set of partition from larger clusters, with typical values set at 90% or above.*

σ_{rad} $\leftarrow$ *sets distance scale used in spectral clustering with a radial basis (Gaussian), typically set at the longest distance scale in the domain,* $\mathbf{\Delta R_{max}}$.

Θ_{rad} $\leftarrow$ *threshold for the value for the Laplacian of the Gaussian used in spectral clustering with a radial basis.*

Θ_{cons} $\leftarrow$ *threshold for a consensus to be reached, where a simple majority is 50%.*

6. Results

6.1 Global and Path Length Maxima Clustering

Results from MAXGLOB clustering are shown in Figs:7(a)-7(d), for KMEANS, KMEDOIDS, MAXGLOB and MAXPATHL maxima clustering for the *Data2D-1* data set. Global clustering

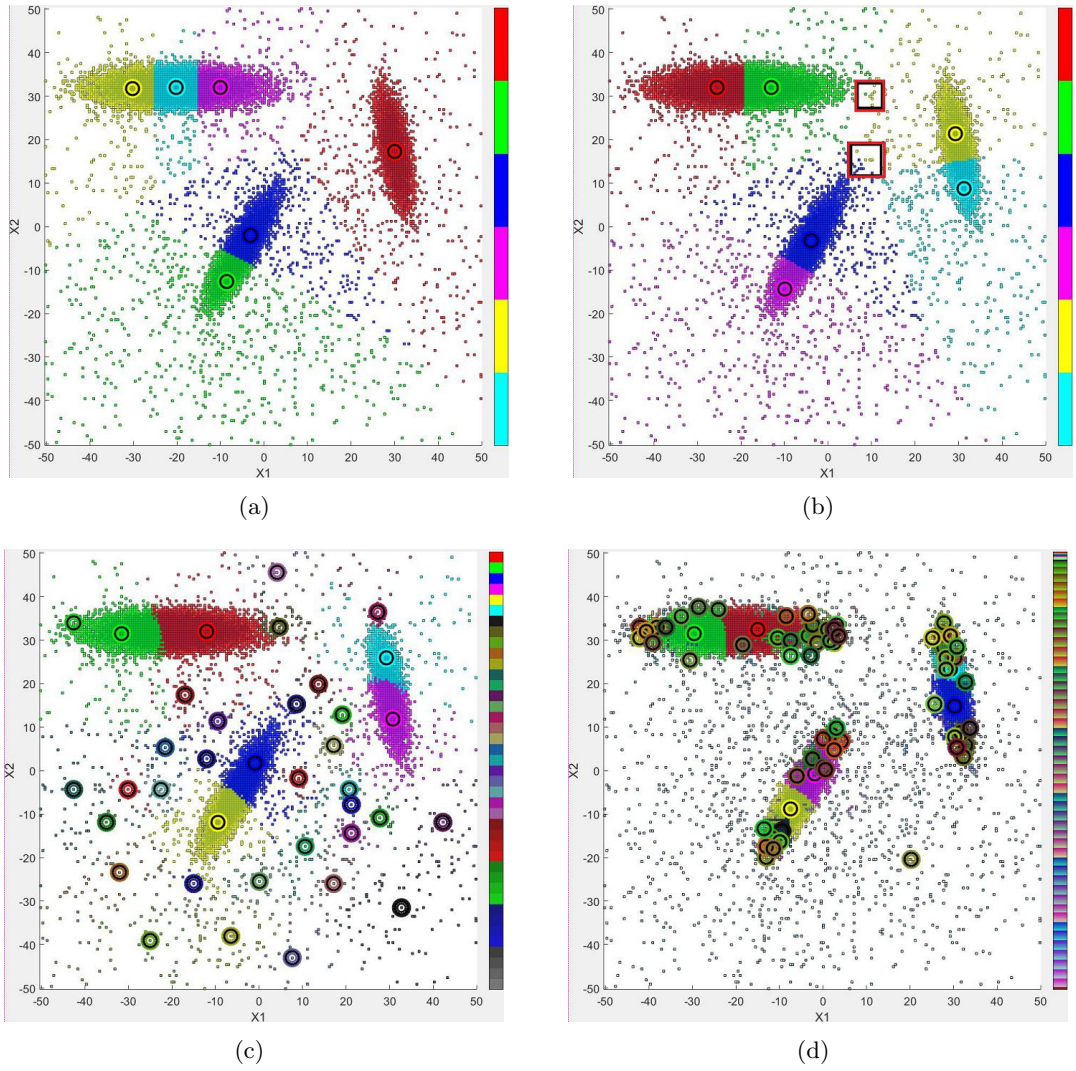


Figure 7: Comparison of k-means 7(a), k-medoids 7(b), MAXGLOB 7(c), and MAXPATHL 7(d). In the figure for k-medoids, two red/black squares have been added to illustrate a failure of the method, namely, elongated distributions can falsely associate with nearby data, when the data is transverse and far from the short axis of an ellipsoidal distribution while being close to the longitudinal axis of another.

performs similarly to KMEANS and KMEDOIDS, however, this approach does not require an initial guess at how many clusters may be present, which is a problem at times for k-means. The MAXPATHL approach uses the same algorithm as MAXGLOB, however, uses the path length as the distance metric, allowing it to associate clusters following the envelope of the distribution, avoiding confusion between nearby, yet disconnected dense regions.

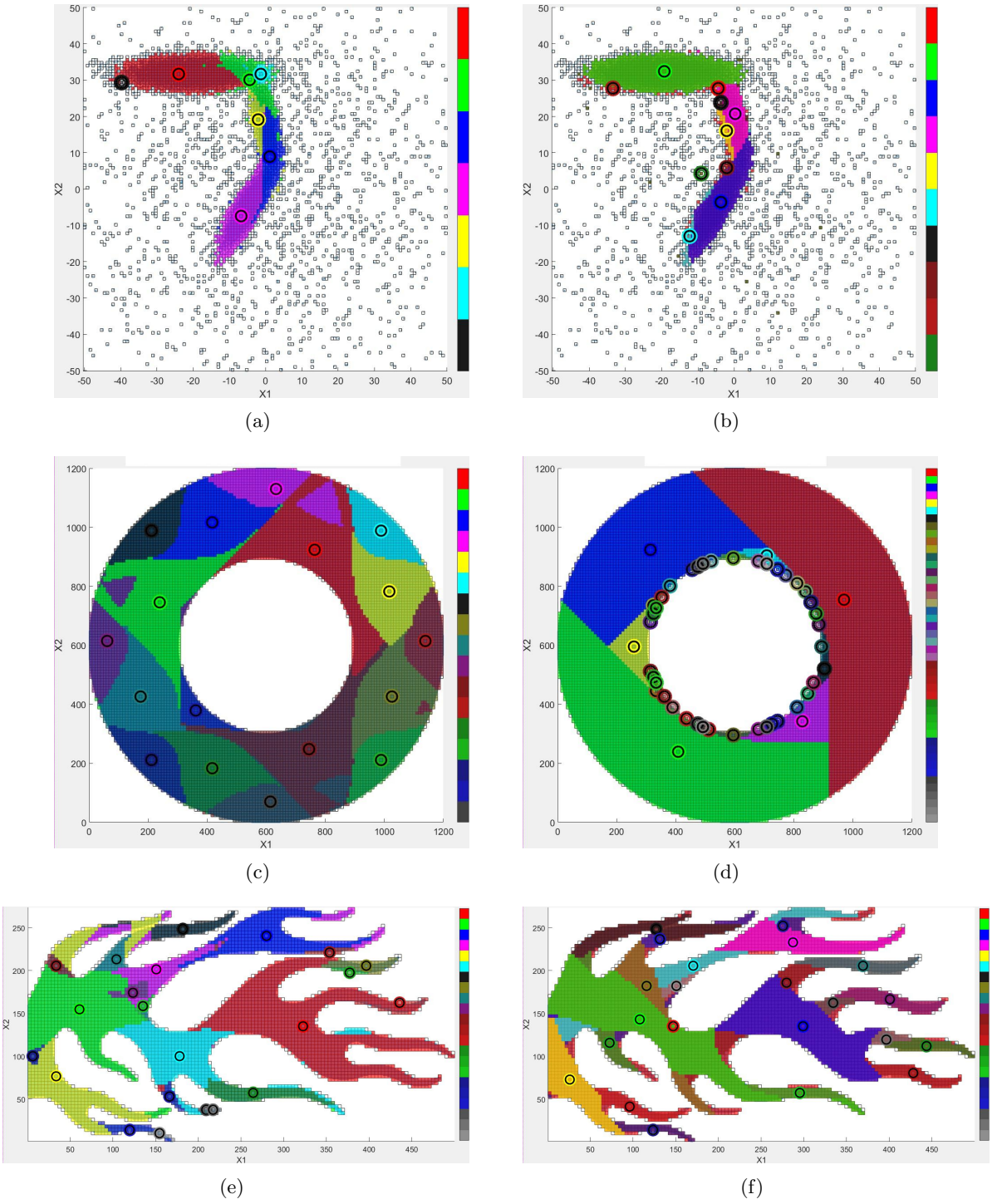


Figure 8: Panels 8(a),8(c),8(e) illustrate LOS-MAXVIS clustering. Panels 8(b),8(d),8(f) show the LOS-MUTUAL clustering applied to the same test cases. The test cases chosen illustrate how the LOS criteria affects ellipsoidal distributions, cases with a high degree of symmetry and filamentary cases.

6.2 LOS Clustering - Maximal Visibility and Highest Mutual Visibility

Results from LOS-MAXVIS and LOS-MUTUAL clustering are shown in Figs:8(a)-8(f), for clustering for the *Data2D-1* data set. Maximum visibility finds clusters initially in regions with the highest coverage, establishes a cluster, then successively removes those partitions from further searches. The process is repeated until all partitions are assigned to clusters. This process can find a high number of clusters depending on how the partitions are arranged. It is useful to then gather the large number of initial clusters found and require that the smaller clusters regroup with a larger cluster. The condition for regrouping is that the smaller cluster share a large percentage of membership from its partitions to the larger clusters *visible* partition set, typically requiring more than 90% overlap from the smaller cluster to the larger. Figure 8(a) illustrates the property of LOS-MAXVIS to find the regions with the largest coverage, which tend to be the corners or elbows of a set of partitions. The maximum visibility algorithm also finds symmetry in groups based on visibility as is shown in Fig. 8(c). LOS-MAXVIS is also useful in finding clusters along filamentary distributions as is shown in Fig. 8(e).

LOS-MUTUAL shares many properties with LOS-MAXVIS, however, the order of grouping is performed differently. The largest mutual visible group is clustered first, regardless of the degree of coverage. When a partition has a visibility that is shared in common with the most amount of other partitions, a cluster is formed. The effect is shown in Figs. 8(b)-8(f). In this approach, the largest common group of visible partitions is assigned to a cluster first, then the next largest group and so on. This has the effect of grabbing a single large group of partitions near one another, such as taking the long arm of the simple "L" example, then leaving the short arm to be taken as the second cluster, ending the search for clusters. By comparison, the LOS-MAXVIS algorithm finds the corner region as a cluster first, then searches along the arms. LOS-MUTUAL finds clusters along filamentary distributions, yet tends to break symmetry by taking the largest pieces first.

Figures 8(a),8(c),8(e) illustrate LOS-MAXVIS clustering. In the first case (*Data2D2*), the maximum visibility identifies the corner regions separately from the centers of the ellipsoids. For the *Concentric1* case, maximal visibility first finds eight symmetric smaller regions hugging the outer radius, then finds interior regions extending to the inner radius, each with lesser and lesser visibility. The last case *Flame3* shows how maximal visibility finds the central regions first, then finds successive filamentary regions.

Figures 8(b),8(d),8(f) show the LOS-MUTUAL clustering applied to the same test cases. In each case, the largest common visibility region is found first, then assigned a cluster ID. Each subsequent search grabs the next largest regions to cluster until no further regions exist. This approach identifies larger distributions then proceeds to track down the remaining ones, also finding clusters within filaments.

6.3 Spectral Clustering

Spectral clustering is a commonly used approach, generally using the first two eigenvectors formed from a numerical Laplacian based on a graphs first nearest neighbors. In this study, the graph would be based on the lattice of partitions. The definition of a neighbor can be simple, such as a geometric neighbor on a grid (**NN1**) or it may be more abstract, such as all partitions visible to one another **LOS**. Further, the neighborhood may be defined globally by taking the Laplacian of a Gaussian for the distance between two neighbors, using the L2-norm $\mathbf{\Delta R}$.

The choice of eigenvectors applied changes the interpretation of the results of clustering within the eigenspace formed from the two eigenvectors. When using the lowest two eigenvectors to form the eigenspace, the lowest eigenvector has the longest feature scale in the eigenspace, where its values cluster near the largest feature seen in the domain of the partitions. The result is that a large cluster is formed near the largest subregion in the partition space. Further clusters are found at increasingly smaller feature sizes, as is shown in Figs. 9(a)-9(f). Clustering values in the eigenspace correlates to identifying differing length scales in the position space, with oscillatory behavior along the length of any subdomains in the position space.

A problem can occur when the partition domain is segregated by disconnected regions. When solving for eigenvectors, through a similarity transformation the Laplacian matrix is effectively diagonalized so that the eigenvectors formed associate with the subblocks along the diagonal. When the partition domain is fully connected, the subblocks become clusters *within* the connected region. When the partitions are disconnected however, this process associates each subblock with a cluster, leaving the interior structure of the subblocks unassigned to clusters. The effect is seen in Figs. 9(f), 10(f), 11(f), where the larger structure is assigned a single cluster due to the small disconnected clusters near its boundary. This sensitivity to noise is the partition set is failing of spectral clustering, in that it requires a fully connected domain in order to resolve interior structure in the presence of ancillary distributions nearby.

Figures 9(a)-9(f) show the results on select test cases for spectral clustering where the Laplacian is based on the **NN1** matrix and the first two eigenvectors were used for gathering within the eigenspace using K-means assuming 16 possible clusters (*SPEC-NN1-12*).

Figures 10(a)-10(f) show the results using a Laplacian based on the **NN1** matrix with the second and third eigenvectors used for defining the eigenspace using K-means for gathering assuming 16 possible clusters (*SPEC-NN1-23*).

Figures 11(a)-11(f) show the results using the LOS criteria to define neighbors on a graph that the Laplacian is created from. The first two eigenvectors define the eigenspace used for gathering with a 2D histogram with 4x4 bins (*SPEC-LOS-12*).

Figures 12(a)-12(f) show the results from using the Laplacian based on a Guassian, making it global in scope. The first two eigenvectors were used for gathering and the eigenspace formed was gathered using K-medoids assuming 16 possible clusters (*SPEC-GAU-12*).

6.4 Positional Clustering (LMH-POS)

Figure 13 shows positional clustering applied to two of the data test cases. A low density threshold was applied so that only higher density partitions are considered for clustering. In the two dimensional case, Fig. 14(a), only four of the possible nine positional clusters are found, while in the three dimensional case, Fig. 14(b), only 21 out of the possible 27 clusters are found. As the number of dimensions grows, the number of possible clusters to be found increases as 3^{N_D}, yet in most cases, the data will likely fill only a fraction of the total possible set of clusters, making the LMHpos clustering a quick view of where data can be found within the data space.

7. Robust Clustering over Multiple Algorithms

In this paper, multiple clustering algorithms have been presented and applied to several testcases. Each technique has strengths as well as weaknesses which have been exposed through the cases

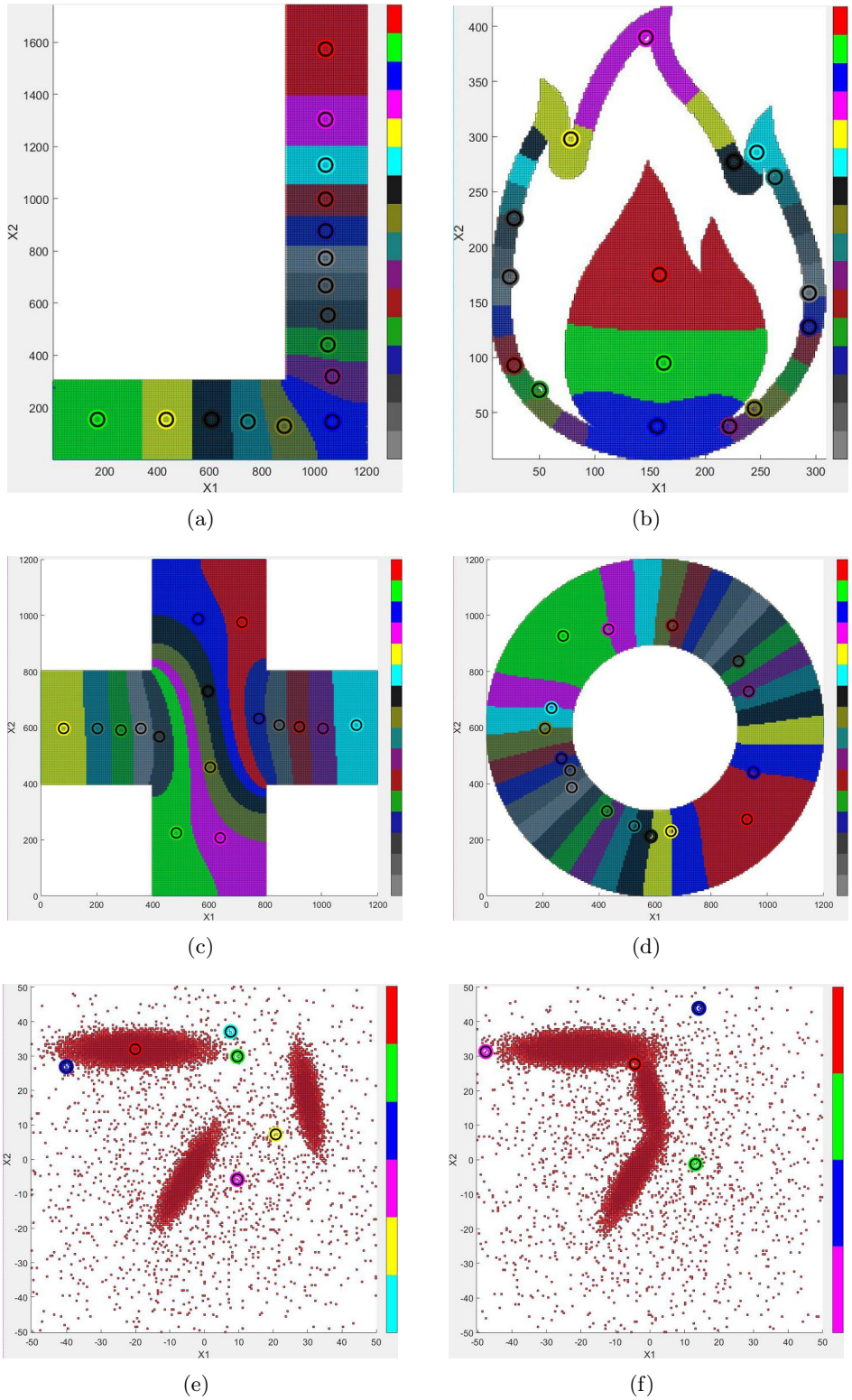


Figure 9: Test bank data sets for *L, Plus1, Concentric1, Data2D1, Data2D2* and *Flame1* are shown using spectral clustering with a Laplacian based on $NN1$ using eigenmodes one and two collected in the eigenspace using K-means seeking 16 possible clusters.

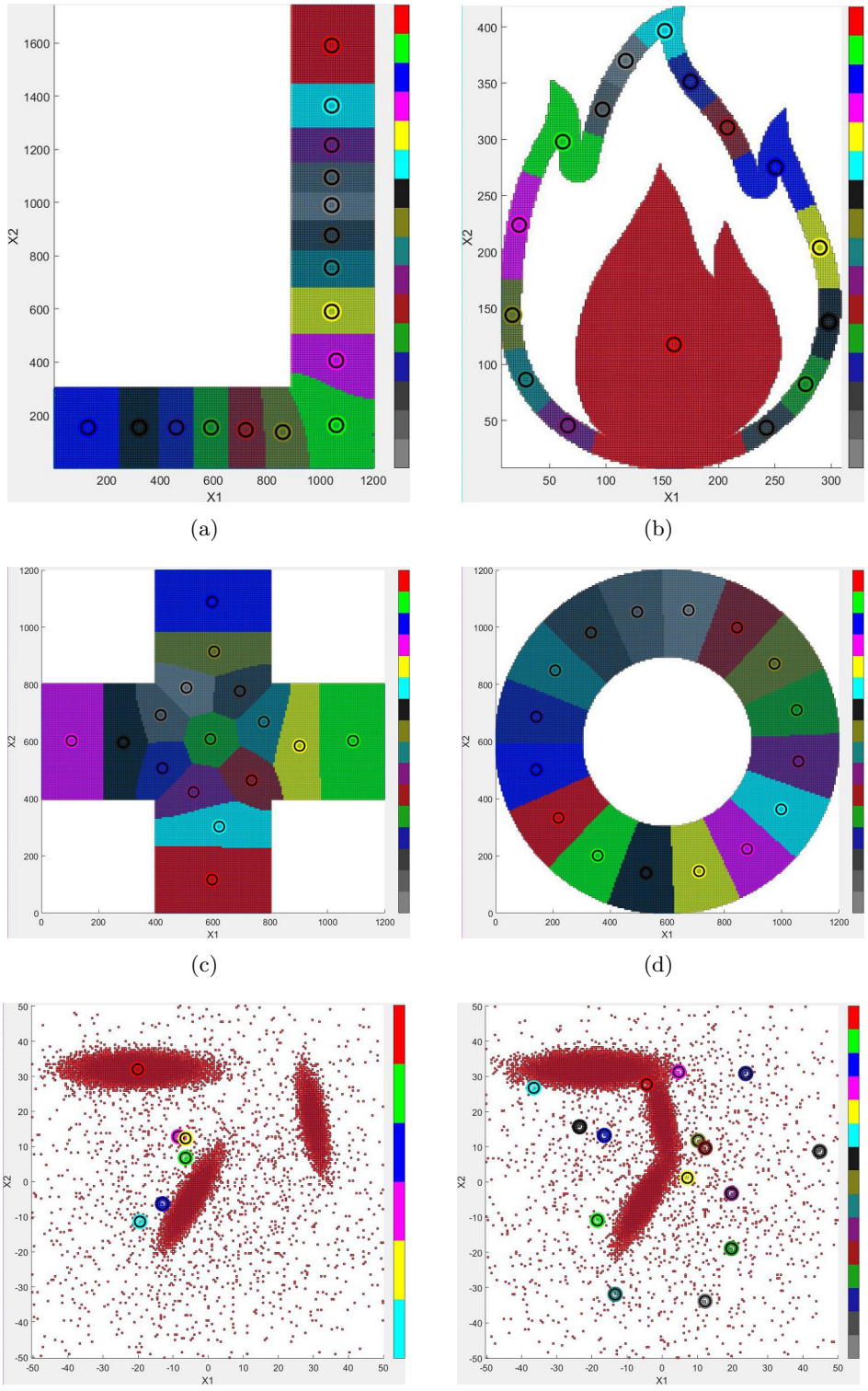


Figure 10: Test bank data sets for *L, Plus1, Concentric1, Data2D1, Data2D2* and *Flame1* are shown using spectral clustering with a Laplacian based on $NN1$ using eigenmodes two and three collected in the eigenspace using K-means seeking 16 possible clusters.

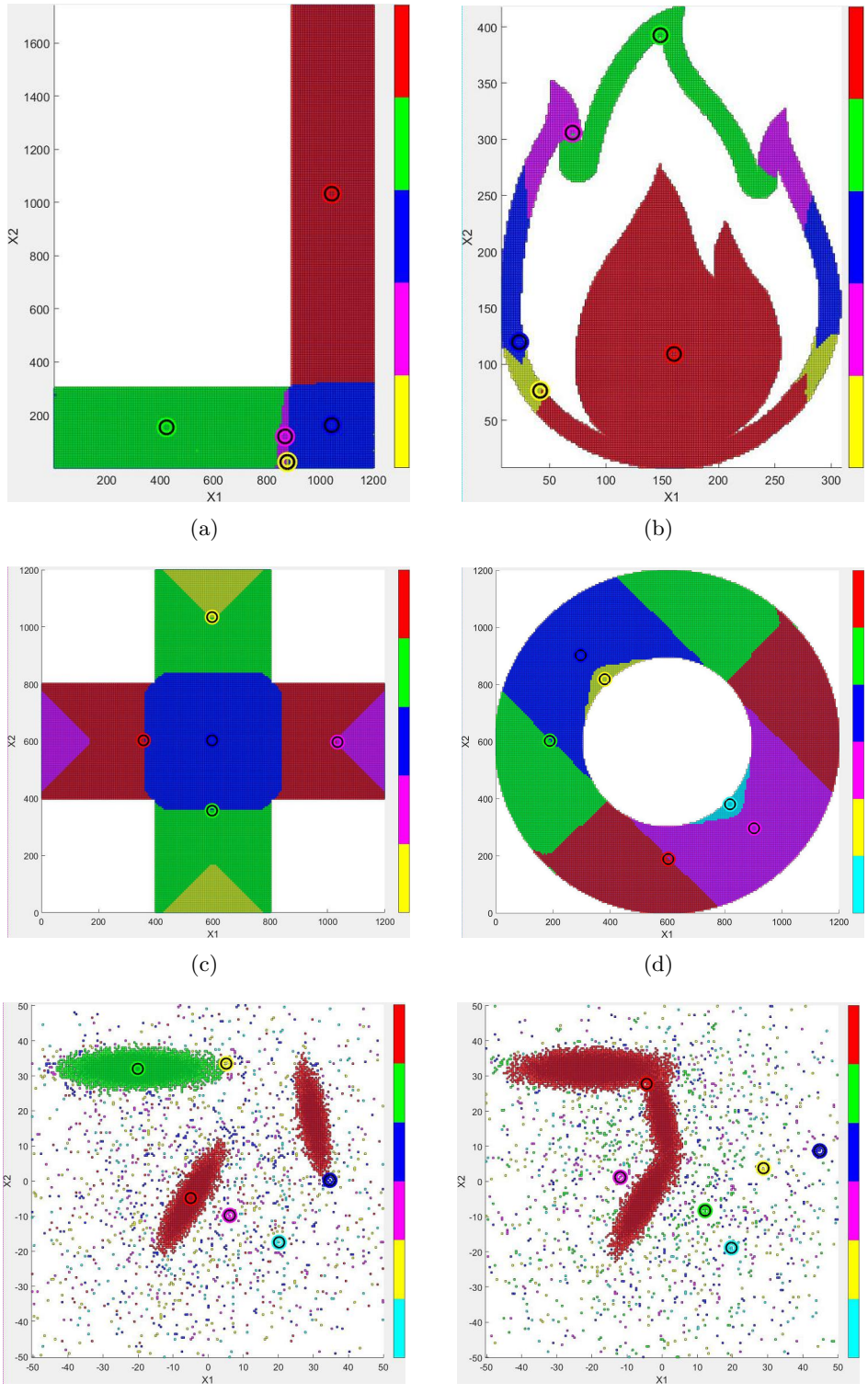


Figure 11: Test bank data sets for *L, Plus1, Concentric1, Data2D1, Data2D2* and *Flame1* are shown using spectral clustering with a Laplacian based on LOS using eigenmodes one and two collected in the eigenspace using a 2D histogram seeking 16 possible clusters.

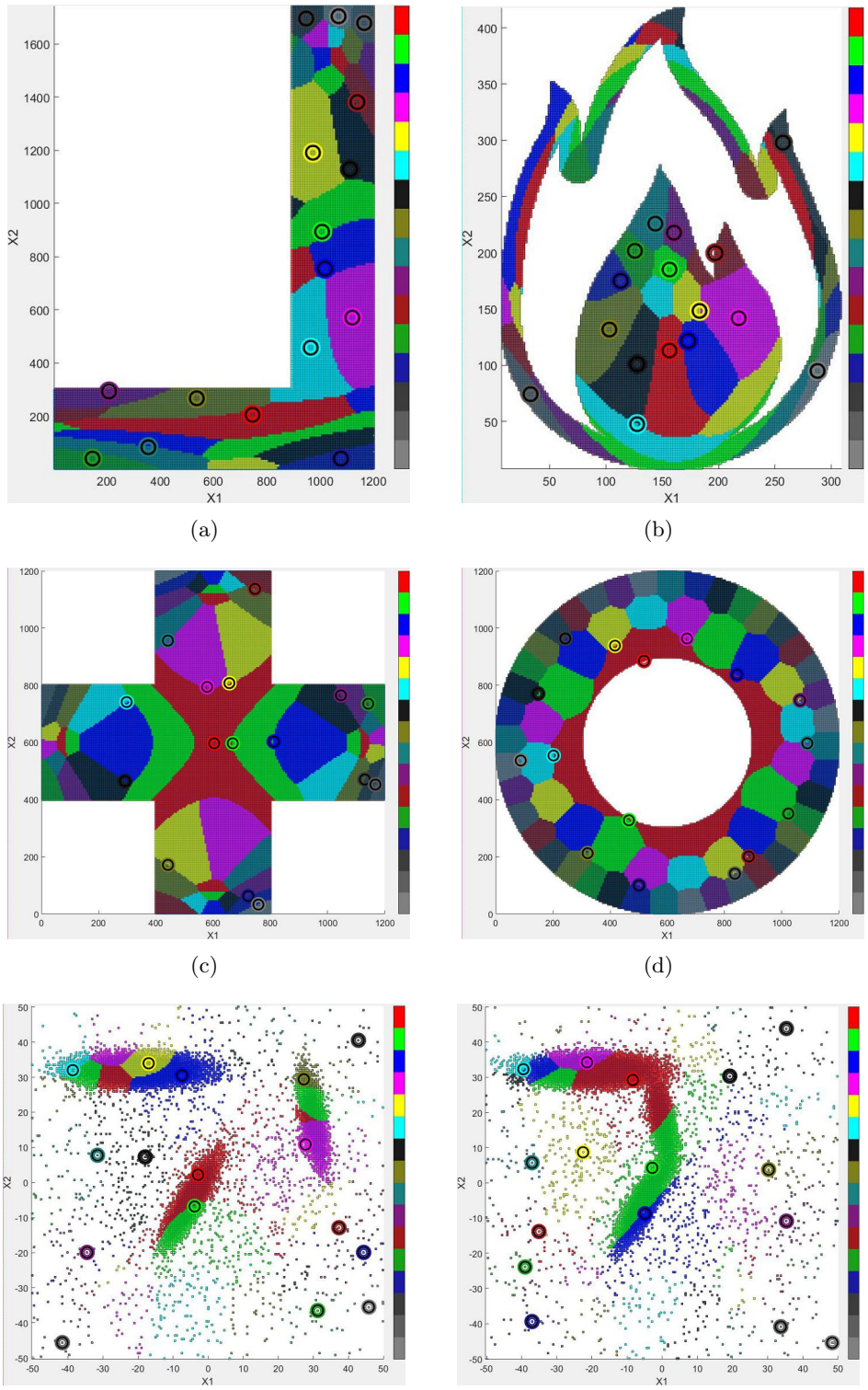
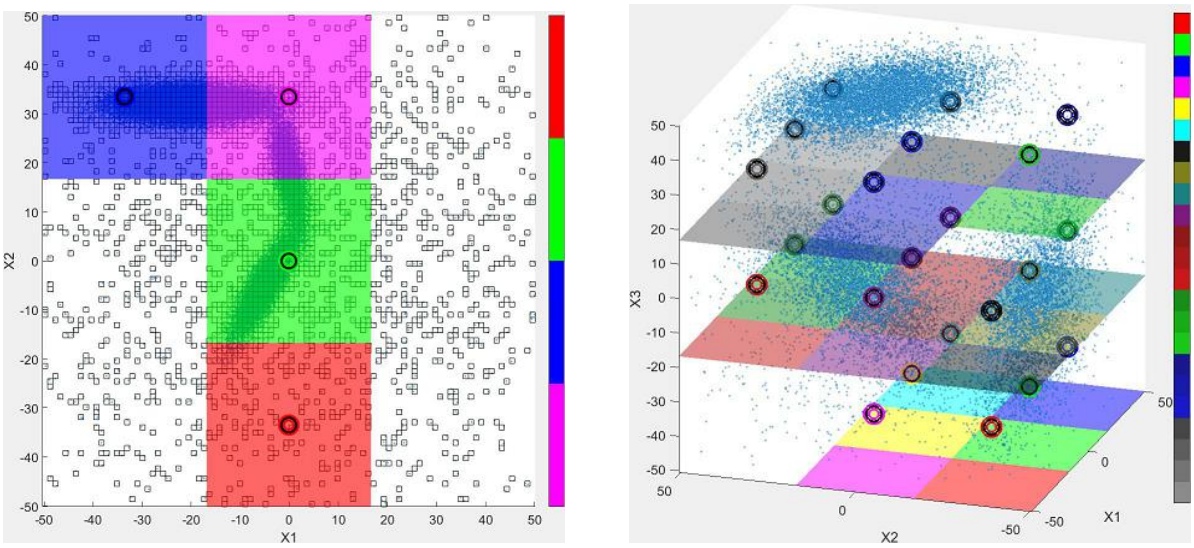


Figure 12: Test bank data sets for *L, Plus1, Concentric1, Data2D1, Data2D2* and *Flame1* are shown using spectral clustering with a Laplacian of a Guassian using eigenmodes two and three collected in the eigenspace using K-means seeking 16 possible clusters.

Figure 13: Positional clustering applied to the 2D data set (left) and the 3D data set (right).



(a) (b)

presented. When using multiple techniques, the possibility exists to leverage the information gathered from all techniques to arrive at a final cluster designation, based on the level of agreement or disagreement found between the algorithms Strehl and Ghosh (2003). This approach is comparable to ensemble modeling used in various fields Hansen (2002); Tebaldi and Knutti (2007). This section proposes four possible robust ways to gather the cluster information.

In each approach taken, the cluster information for the partitions is represented by a matrix of cluster IDs, where each row represents a single cluster algorithm and each column is a partition. The values along each row is then the cluster ID given to each partition. The matrix formed is called the Cluster ID matrix and is (N_C, N_P) in size. In order find the agreement or disagreement between cluster IDs across many techniques, the columns are rearranged so that the cluster IDs are sequential starting from the first row and maintaining the ordering as each row is subsequently reordered until all rows have been processed. Table 3 illustrates this process for a sample of 40 partitions using six cluster algorithms. The top matrix is the initial partition cluster ID matrix unsorted. The second matrix is the sorted cluster ID matrix described above. Finally, the third matrix from the top shows the differences in cluster IDs *along each row*, where a one represents a change in cluster designation for that rows technique. The process of assigning cluster IDs to partitions begins with the lowest numbered cluster IDs over all algorithms, and proceeds in increasing cluster ID order. In the table shown, this is equivalent to following the partitions from left to right across the page.

As examples of robust clustering, Fig. 14 as well as Tab. 3 are provided to illustrate the process. The figures shown in Fig. 14 are all clustering techniques excluding the LMHpos algorithm for the *Data2D2* test case with no minimal population set for the partitions. The LMHpos technique was excluded as its partition definitions do not align with the remaining 25 algorithms. In cases where multiple techniques are compared using differing partition sizes, the robust technique is then

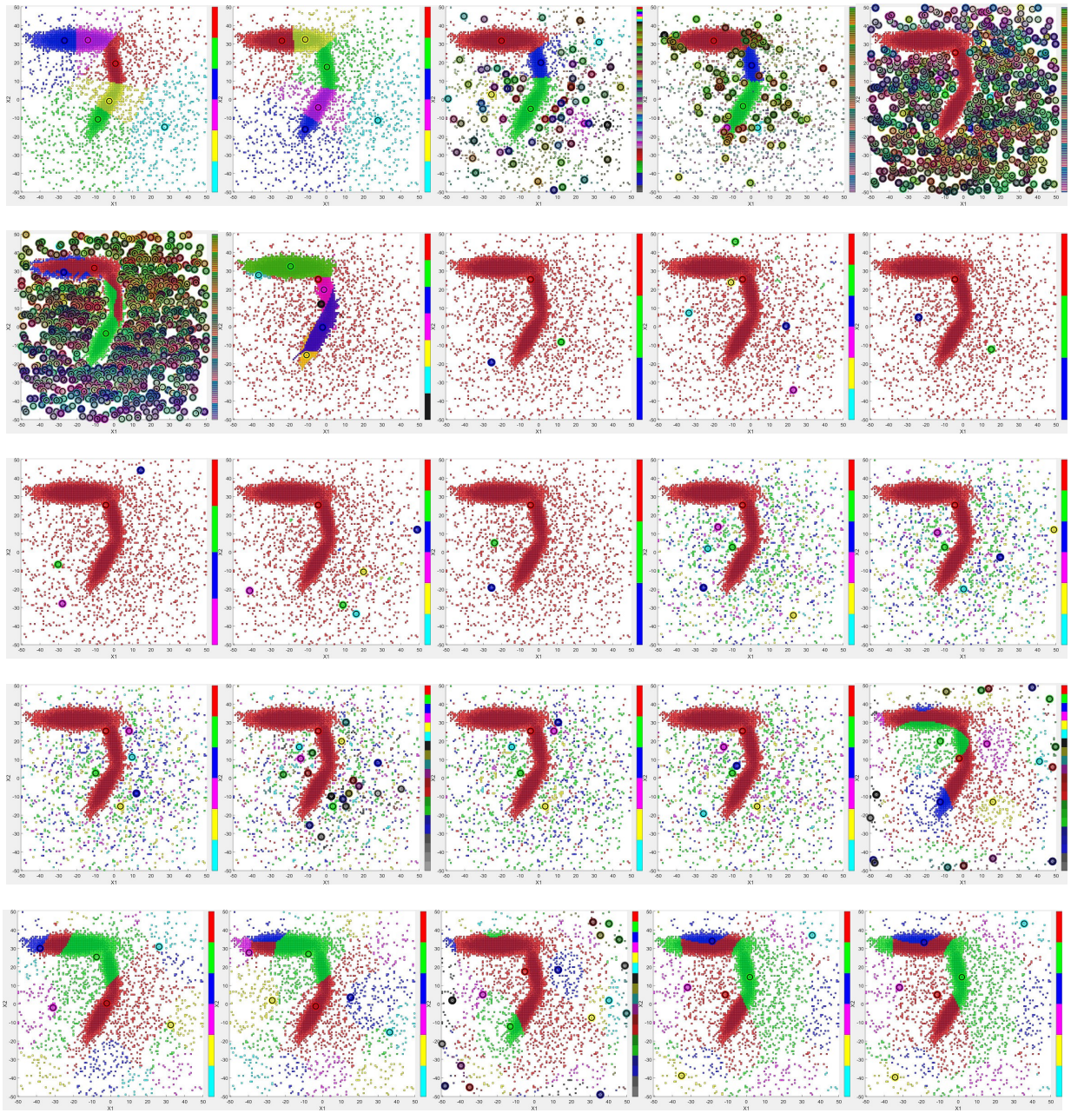


Figure 14: Clustering techniques applied to the *Data2D2* set with no lower bound for partition population. All techniques are shown excluding LMHpos clustering as it has differing partitions. Clustering techniques are shown in the following order: KMEANS, KMEDOIDS, MAXGLOBAL, CONN, MAXPATHL, LOS-MAXVIS, LOS-MUTUAL, SPECTRAL01, SPECTRAL01-18.

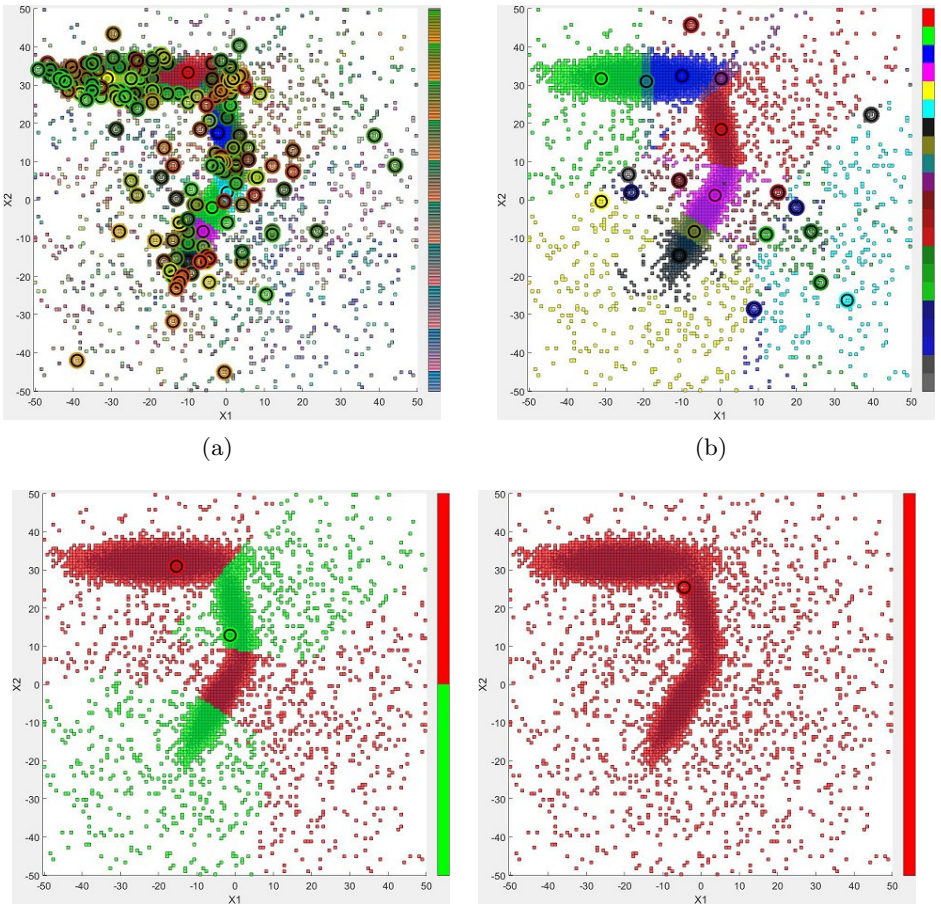


Figure 15: Robust clustering results based on clustering techniques shown in Fig. 6.4. Panel 15(a) shows the clusters formed when any change across all clustering techniques occurs. Panel 15(b) shows clusters formed using a consensus with 75% taken as the majority threshold. Panel 15(c) shows clusters formed once all techniques (100%) have encountered a change. Panel 15(d) shows the cluster formed when requiring that all techniques register a change simultaneously. For K-means and K-medoids, a k-value of three was used.

applied *per datum*, using the same procedures, however, the sorting is performed over all data instead of partitions.

7.1 Fractured Cluster ID

The **Fractured** robust designation results by assigning each partition a new cluster ID starting from one and increasing the cluster ID each time *any* technique changes its ID. This results in the largest set of clusters found. This approach is the most sensitive to changes in the cluster designations. Figure 15(a) shows the clusters formed, leading to the largest set of the robust techniques, also the most sensitive to changes in the partitions grouping.

7.2 Majority Changed Cluster ID

The **Majority Changed** robust designation results by assigning each partition a new cluster ID starting from one and increasing the cluster ID each time the accumulated number of algorithms changing reaches a majority of the total number of algorithms. Once the cluster ID has been changed, the accumulated sum of changes is reset to zero. This results in a medium sized set of clusters found, where a significant number of algorithms found a change, however, not all algorithms are required to note the change in ID. Figure 15(b) shows the clusters formed using a consensus among the techniques with a 75% threshold applied. A simple majority places the threshold for consensus at 50%, however, other values can be used to attain consensus. Ideally, the best value would create the largest number of clusters with the highest average membership.

7.3 All Changed Cluster ID

The **All Changed** robust designation results by assigning each partition a new cluster ID starting from one and increasing the cluster ID each time the accumulated number of algorithms changing reaches the total number of algorithms. Once the cluster ID has been changed, the accumulated sum of changes is reset to zero. This results in a small-medium sized set of clusters found, where every algorithm found a change, however, the changes may not have been at the same partition number, merely, that the total set of changes across all algorithms eventually required a change of ID. Figure 15(c) shows the clusters formed by requiring that all techniques register a change in cluster definition before a partition is given a new identity. This is equivalent to setting the consensus threshold to 100%. Once a clustering technique has changed a partitions cluster ID, any further changes from that technique are not registered until *all* techniques have shown a change as well. This approach limits the number of final clusters formed.

7.4 No Overlap Cluster ID

The **No Overlap** robust designation results by assigning each partition a new cluster ID starting from one and increasing the cluster ID each time the total number of algorithms changes designation simultaneously. This results in a smallest sized set of clusters found, where every algorithm found a change. Figure 15(d) shows the cluster formed by requiring all techniques to simultaneously register a change in cluster identity. Ideally, this would happen for each disconnected group of partitions, however, several techniques are "global" in scope and do not require a connection to exist to form clusters, leading - in this case - to a single large cluster.

	40 Partition Cluster IDs																																							
Alg_1	5	7	2	4	7	1	2	4	1	1	2	3	8	6	4	1	5	5	4	1	1	4	8	8	9	6	9	4	1	2	4	4	4	1	7	4	3	3	4
Alg_2	7	7	3	7	7	1	3	7	3	1	3	4	7	7	6	1	7	7	5	1	2	5	7	7	7	7	7	5	1	3	6	7	6	1	7	7	5	3	5
Alg_3	6	6	2	6	7	1	3	6	2	1	3	3	8	6	5	1	6	6	5	1	1	5	8	8	8	8	6	8	5	1	2	5	6	6	1	6	6	3	3	5
Alg_4	4	5	2	4	6	1	2	4	2	1	2	2	6	4	4	1	4	4	2	1	2	3	6	6	6	5	6	2	1	2	3	4	4	1	5	4	2	2	2
Alg_5	5	6	1	4	6	1	1	5	1	1	1	2	6	6	4	1	6	5	3	1	1	4	6	6	6	6	6	4	1	1	4	4	4	1	6	4	2	1	4
Alg_6	6	8	1	5	8	1	2	5	1	1	3	4	8	7	4	1	7	5	4	1	1	4	9	9	9	7	9	4	1	1	4	5	5	1	7	5	4	3	4
	40 Partition Cluster IDs - Resorted by Partitions in Ascending ID Order																																								
Alg_1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	6	6	7	7	7	8	8	8	9
Alg_2	1	1	1	1	1	1	2	3	3	3	3	3	3	4	5	5	5	5	5	5	6	6	6	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
Alg_3	1	1	1	1	1	1	1	2	2	2	3	3	3	3	3	4	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	6	6	7	8	8	8	8
Alg_4	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	3	3	4	4	4	4	4	4	4	4	4	4	5	5	5	6	6	6	6	6	6
Alg_5	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	3	3	4	4	4	4	4	4	4	4	4	5	5	5	6	6	6	6	6	6	6	6	6	6
Alg_6	1	1	1	1	1	1	1	1	1	2	3	3	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	6	7	7	7	7	8	8	8	9	9	9	
	40 Partition Cluster Difference Flags (Logical) for Sorted IDs																																								
Alg_1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	1	0	0	1
Alg_2	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alg_3	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
Alg_4	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0
Alg_5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
Alg_6	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	0
	40 Partition Cluster Fractured IDs																																								
Rob_1	1	1	1	1	1	1	2	3	4	4	5	6	7	8	9	10	11	12	12	13	14	15	16	17	17	17	18	19	20	21	22	23	24	25	26	27	28	28	29
	40 Partition Cluster Majority Changed IDs																																								
Rob_2	1	1	1	1	1	1	1	2	2	2	3	3	3	4	4	5	5	6	6	6	6	6	7	7	7	7	7	8	8	8	9	9	9	10	10	11	11	11	11
	40 Partition Cluster All Changed IDs																																								
Rob_3	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4
	40 Partition Cluster No Overlap IDs																																								
Rob_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3: A sample set of partitions that have had six differing cluster algorithms applied. In each case, the cluster algorithm identified up to nine different clusters. The set contains 40 partitions. The top table represents the data initially unsorted. Each row is a different cluster algorithm and each column is a partition where a cluster ID has been assigned. The second table has sorted the each row while maintaining the assignments to each partition. The third table from the top are the differences in cluster ID assignments from one column to the next. The fourth table is the final cluster assignment given to the partitions when any one change occurs (a disagreement) between the cluster algorithms. The fifth table requires a majority of the cluster algorithms to change (cumulatively) before a new cluster assignment is designated. The sixth table only changes the cluster assignment once all cluster algorithms cumulatively have changed. Finally, the last table requires that all algorithms change assignments simultaneously before a new cluster ID is designated (the clusters are disjoint - with no overlap).

7.5 Strategy with Robust Clustering

Several of the clustering techniques used in this study require either a guess or fore-knowledge of the number of clusters sought, such as KMEANS and KMEDOIDS. Further, the spectral methods applied k-means and k-medoids in order to identify clusters within the eigenspace formed from the eigenvectors chosen. In these cases, an initial guess at the number of clusters sought is required, the *k-value*. Robust clustering can provide a reasonable guess for the k-value, by first attaining consensus over all techniques that do *not* use a k-value, which are: MAXGLOBAL, MAXPATHL, CONN, LOS-MAXVIS, LOS-MUTUAL and all spectral methods which use 2D histogram binning, although the 2D histogram binning itself requires a guess as to the number of bins to use; however, once the bins have been established, the 2D histograms simply find clusters which populate those bins, often finding fewer clusters than bins. Using the ROBUST2 technique with a suitable choice

in consensus threshold, Θ_{cons}, a number of clusters will be found. Taking the number of clusters found and setting k-value to this number then allows a reasonable guess to re-run the analysis utilizing the full complement of techniques.

This approach also accommodates using differing sets of variable choices as well as different binning across each dimension. In this case, the partition definitions will not be the same from one set of analysis choices (variables, binning, techniques); however, each datum will still be assigned a cluster ID. Robust clustering is then applied for each datum, leading to considerably longer sort times, however, the robust approach is still applicable, allowing various consensus clustering to performed.

8. Conclusions

A study using 26 clustering techniques has been performed over 12 test cases to illustrate both the strengths and weaknesses of clustering algorithms. A robust form of clustering is achieved through consensus over all techniques, helping reduce clustering problems by finding consistent clustering definitions across many approaches.

8.1 Overview - Multiple Clustering Techniques Employing Path Lengths

The approach taken by this study utilizes four main ideas to produce a robust clustering analysis:

- Reduce a large data set by binning the space into multi-dimensional partitions and create a serial index for each filled partition.
- Algorithms use the path length between any connected partitions as well as traditional distance metrics (L1, L2, etc...).
- Employ multiple clustering techniques to the set of partitions based on first nearest neighbors, distance weighted factors and geometrical properties of the set.
- Establish a final cluster ID based on all the consensus of techniques employed as well as multiple resolutions and differing variable sets.

The combination of multiple clustering techniques, various distance metrics and traditional data reduction lead to a robust set of clusters defined.

From the arrays defined in Sec. 4.1, the following clustering techniques are employed:

- Clusters are sought by proximity using K-means and K-medoids.
- Clusters are sought globally, finding local peaks based on the maxima of the weights.
- Clusters are sought by path length, finding local peaks based on the maxima of the weights.
- Clustering is calculated based on whether partitions are connected or not.
- Clustering is calculated based on whether partitions are within LOS of each other using two gathering methods based on maximal visibility and highest mutual visibility.
- Spectral clustering established from a $NN1$ Laplacian based on eigenmodes 1 and 2 or on eigenmodes 2 and 3 using either k-means, k-medoids or 2D histograms for gathering.

- Spectral clustering established from a LOS Laplacian based on eigenmodes 1 and 2 or on eigenmodes 2 and 3 using either k-means, k-medoids or 2D histograms for gathering.
- Spectral clustering established from a Laplacian of a Guassian based on eigenmodes 1 and 2 or on eigenmodes 2 and 3 using either k-means, k-medoids or 2D histograms for gathering.
- Clusters are calculated based on a course binning (Low-Medium-High) of the absolute position of data within the data space.
- A final cluster ID is assigned from one of four different robust techniques based on reaching consensus amongst the clustering algorithms.

8.2 Analysis Choices

By choosing to use the bin address space of filled bins only (partitions), the multidimensional nature of the study is reduced to working on a integer based rectilinear grid of bins, facilitating the choices in distance metrics used as well as their interpretation. Keeping the distance metrics to either L1 or L2 norms and working within a global domain, using $\mathbf{\Delta R}$, or a local region, using $\mathbf{\Delta L}$, further simplifies the interpretation of the clusters identified.

The novel approach to data reduction used assists in reducing the computational cost associated with Big Data analysis. The introduction of weighted partitions requires alterations to standard techniques as the proximity metrics used must be re-evaluated. The computational advantage generally is orders of magnitude greater as most clustering problems are of the order N^2, where N is reduced from the data set size to the partition set size.

Beyond standard clustering algorithms currently employed in the field, a path length approach as well as Line-Of-Sight conditions were added to this study. Using path lengths provides greater sensitivity to the shape of a distribution, while using LOS criteria identifies convex hull regions within distributions. Both additions were then further employed within standard techniques, enhancing their capability to find clusters.

Overall, the robust approach accomplishes its goals, where the second approach is the most appropriate in general. The other three robust techniques help bound the result found in Robust-2. Iterative applications of the robust approach further help to establish a reasonable k-value for those algorithms requiring an initial guess.

The inclusion of both path length based and LOS criteria enhance the ability to cluster data, accomplishing both the goals of data clustering by proximity as well as data identification by finding subdomains within a distribution that are either in the bulk of a group of data, or are in an overlapping region of two or more distributions.

Acknowledgments. SW acknowledges the support of ONR Grant No. N00014-01-1-0769. KM acknowledges the support by ONR grants: N00014-15-WX-01814, N00014-16-WX-01705 and N00014-17-WX-01705 as well as the Kinnear Fellowship from the USNA Foundation.

Appendix A. General Algorithms

A.1 Finding Local Maxima - FINDMAX

Consider a contour map based on the weights of the partitions, similar to a multi-dimensional topographic map. The peaks of the set of weighted partitions fall into one of three categories: distinct peaks, ridges and plateaus Given that the bin resolution may be course over any one dimension, it is difficult to apply a critical value criteria to the weights of the partitions. A simpler approach to finding the peaks is to first find all local maxima of the partitions by considering the first nearest neighbors ($NN1$) for a given partition. If all of its neighbors have weights less than or equal to the center, then the center partition is a local maxima. Among the local maxima, those that have partitions with neighbors of equal weight are set apart as plateau maxima, while the others are peak maxima.

When attempting to classify the various local maxima into categories, it is helpful to first calculate the standard deviation of the weights across the domain, $\sigma(wgt_k)$, where the index k represents the partitions. Once the standard deviation is calculated, a threshold parameter, P_{peak} is set which together form a threshold, $\Theta_{peak} = P_{peak}\ \sigma(wgt_k)$. Similarly, a plateau parameter is also set, leading to the definition of plateaus, with the difference being that the band of weights is much smaller and in both directions, such that a "plateau" is defined as all of the connected partitions to a given local maxima within the threshold:

$$wgt_q - \Theta_{plat} < wgt_k \leq wgt_q + \Theta_{plat} \tag{16}$$

where $\Theta_{plat} = P_{plat}\ \sigma(wgt_k)$. Finally, a check to ensure that all found maxima are connected to one another is performed. All maxima found by this criteria are considered part of one plateau and given a single cluster index for the set. A special plateau is formed from the set of all local maxima whose weight is close to zero (low population). The same criteria given above is used to define this plateau, however, partitions found within this plateau may be released from this designation to allow these minimally populated partitions the chance to be associated with other higher maxima. Once a partition has been classified as part of a plateau, it is no longer used in searches for further peaks, leaving the remaining peaks to be either distinct or part of a ridge.

For each peak, a slope, s_q, is found which will be used to determine when another partition is close by or not. The slope is used both to find ridges as well as determine clusters for partitions to peaks. Figure 16(a) illustrates the means of determining the slope for a given peak, labeled "q". Taking the set of all local maxima found, for each maxima (q), a set of slopes can be found to all other local maxima, q'(including maxima found in plateaus). The greatest slope in magnitude is taken as the slope assigned to each peak:

$$s_{qq'} = \left| \frac{\Delta wgt_{qq'}}{\Delta R_{qq'}} \right| \tag{17}$$

$$s_q = Max[\ s_{qq'}\ ] \tag{18}$$

Although clustering can be restricted to just the definitions of peaks and plateaus, when a group of peaks are close together, they can act as a single entity called a *ridge*. Clustering to a ridge is possible provided a distance can be found which determines when one peak is close to another. For the purpose of determining whether a peak is distinct or part of a ridge, a radius is found for each peak by taking a weight difference and dividing by the slope. Starting from the highest weighted

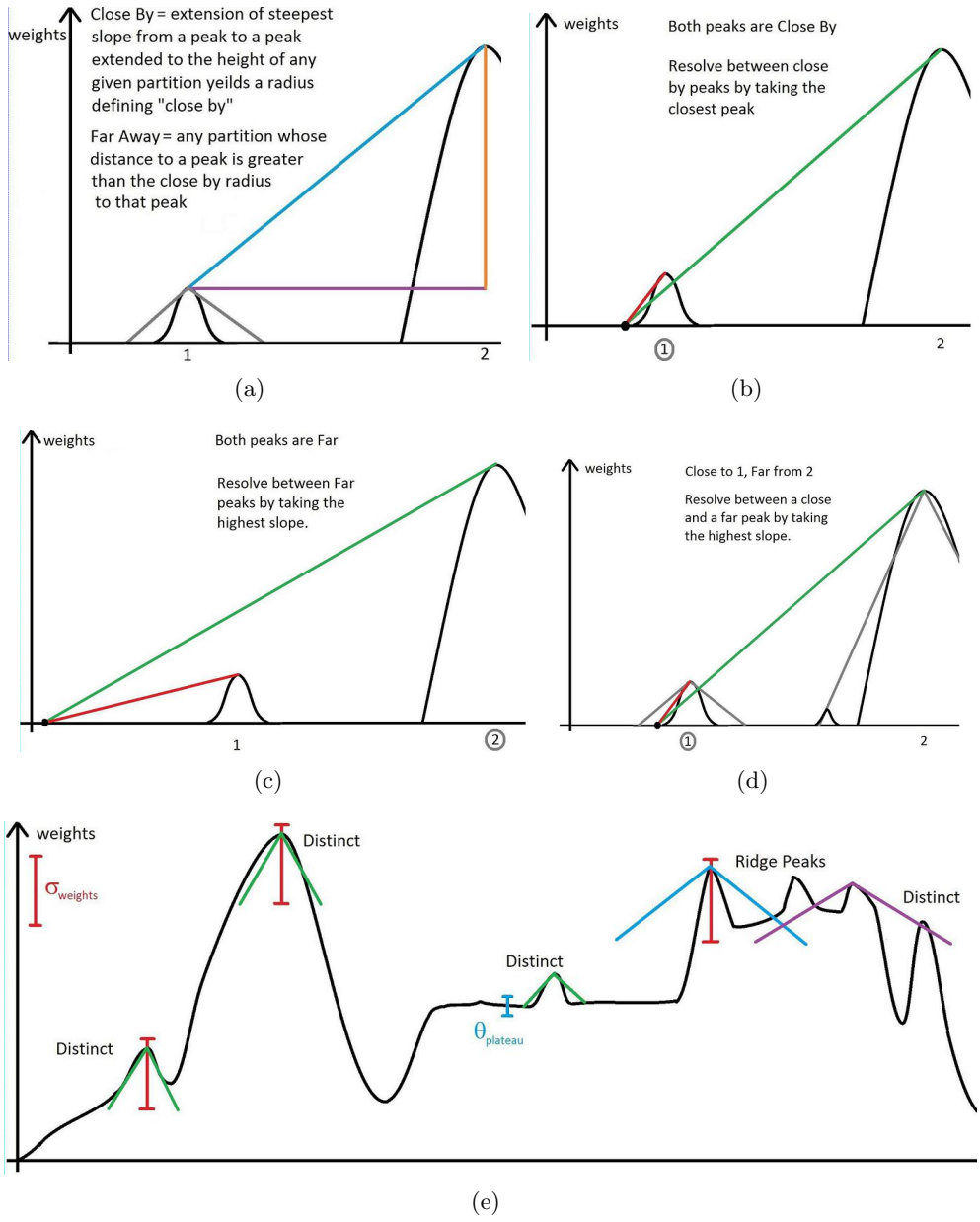
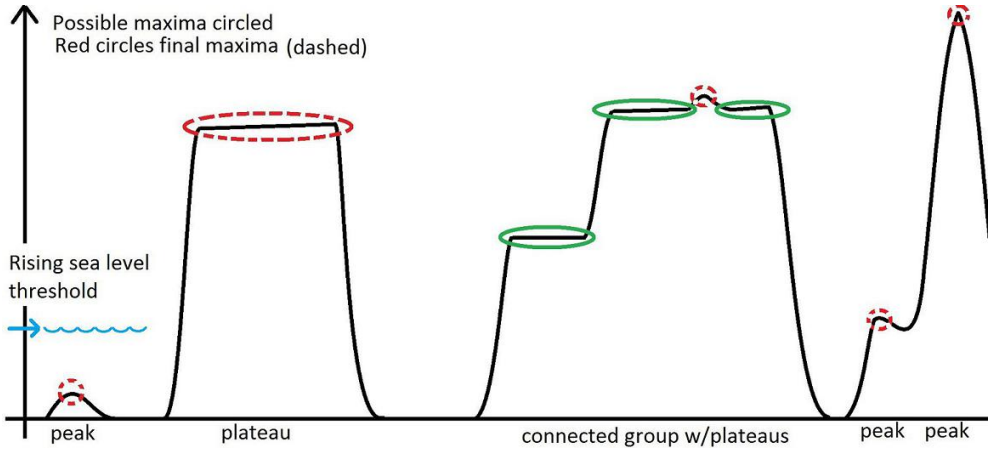


Figure 16: Definition of the slope between peaks (shown in cyan). Once a slope has been assigned to a peak, s_q, the closeness of a partition can be determined by calculating a radius for each peak, R_{kq} compared to the distance to the peak, ΔR_{kq}. The slopes in the later panels are indicated by the grey lines. Determination of a peak, ridge or a plateau is based on the standard deviation of the weights across the domain, $\sigma(wgt_k)$. The slopes are indicated by the lines stemming from each peak. The radii are found by dividing the weight threshold by the slope, $R_q = \Theta_{peak}/s_q$. Peaks are *distinct* if no other maxima are found within a radius of the peak, R_q. A *ridge* is found when multiple peaks are connected within a range of weights and within the radius of each peak. Finally, *plateaus* are the set of local maxima found within a narrow band of weights, Θ_{plat}.

Algorithm 1 Find Maxima algorithm - Global and Path Length (uses FastConn)

procedure FINDMAX($k, w_k, \Delta w_{k\ell}, \Delta r_{k\ell}, \Delta \ell_{k\ell}$) ▷ Requires a list of partitions, weights,
▷ weight differences, distance differences
  $\sigma_P \leftarrow stddev(w_k)$ ▷ Calculate standard deviation of weights
  $\Theta_{plateau} \leftarrow 0.02\sigma_P$ ▷ Threshold for weights to be within a *plateau*
  $ik' \leftarrow 1$ ▷ index for possible maxima
  for all $partitions, k$ **do** ▷ Compare center to first nearest neighbors, $NN1$
    if $w_{NN1,\ell} <= w_k$ **then** ▷ If center has the greatest weight (or tied)
      $possiblemax[ik', 1] \leftarrow k$ ▷ Add center to list of possible maxima
      $possiblemax[ik', 2] \leftarrow w_k$ ▷ Add center weight to list of possible maxima
      $ik' \leftarrow ik' + 1$ ▷ k' Index for *possiblemax*
    end if
  end for ▷ See Fig.17 where *possiblemax* are circled.
  for all *possiblemax* **do** ▷ Adds *plateaus* to *possiblemax*
    if $\text{abs}(\Delta w_{k'\ell}) < \Theta_{plateau}$ **then** ▷ Find other partitions weights within a
      $possiblemax[ik', 1] \leftarrow \ell$ ▷ narrow range of the current *possiblemax*
      $possiblemax[ik', 2] \leftarrow w_\ell$
      $ik' \leftarrow ik' + 1$ ▷ Add these partitions, ℓ, to set of *possiblemax*
    end if
  end for
  ▷ "Rising sea" criteria to find one maxima per group of *possiblemax*
  $possiblemax \leftarrow sort(possiblemax, 2, ascend)$ ▷ Reorder by ascending weights
  $removemax \leftarrow zeros$ ▷ Initialize removal array to zero
  $ik' \leftarrow 1$
  for all $possiblemax(ik')$ **do** ▷ Find set of all partitions higher than
    $partcheck \leftarrow k(\ w_\ell > possiblemax(ik', 2)\ )$ ▷ current possiblemax, $w_\ell > w_{k'}$
    $connpartcheck \leftarrow FastConn(partcheck)$ ▷ Find all connected partitions to k'
    for all $connpartcheck(\ell \neq k')$ **do**
      if $connpartcheck \subset possiblemax(\ell)$ **then** ▷ Set current *possiblemax*, k', for removal
        $removemax(ik') \leftarrow 1$ ▷ if there is a higher $possiblemax_\ell$ in the
      end if ▷ connected group of partitions
    end for
    $ik' \leftarrow ik' + 1$
  end for
  $possiblemax(removemax) \leftarrow [empty]$ ▷ Remove all maxima from a group except highest
  $maximafinal \leftarrow possiblemax$
  return $maximafinal$ ▷ Returns list of partition maxima.
end procedure



(a)

Figure 17: Types of possible maxima, plateaus and peaks. While searching for the maxima within each group of partitions, first potential maxima are identified (shown circled). By finding all connected partitions higher than a possible maxima, if any other maxima are connected to the lower maxima, the lowest is removed from consideration, leaving only successively higher and higher maxima until only one maxima is left per connected group of partitions.

partition, a band of partitions is found for all whose weights are between the given maxima weight down to the threshold:

$$(wgt_{min} \equiv wgt_q - \Theta_{peak}) < wgt_k \leq wgt_q \tag{19}$$

with q being the index of the current maxima and k being the index of all connected partitions to q. The set of partitions is further restricted by requiring that the distance between the maxima and the surrounding partitions be within the radius set by:

$$\text{Radius peak} \quad \equiv \quad R_q = \frac{wgt_q - wgt_{min}}{s_q} \tag{20}$$

Finally, a check to determine if the set of partitions is connected to the maxima is done. A *distinct* peak is then any maxima which is a single maxima found within a connected group of partitions within a band of weight values which are close by to the peak, like the Matterhorn, Switzerland. If other maxima are found within the connected set of partitions and within the radius, then the peak and others found form a ridge. Subsequent checks on the other peaks are made to determine if the ridge continues to find further peaks. When determining the radius for subsequent peaks in the ridge, the same lower value is used in the weight band, wgt_{min}. In this manner, the full ridge is found by cascading the peak search to all other peaks found until the search within the set of ridge peaks is exhausted, at which point, a single cluster identification is given to the entire ridge, examples would include the Grand Tetons, USA or Valais, Switzerland. Figure 17 illustrates how local maxima are classified into plateaus, distinct peaks or ridges.

Following the analogy to a height field, a criteria is established to determine which peaks are the most significant to their surrounding partitions. By determining the set of partitions the most

closely associated with a peak, a cluster is formed around that peak. The criteria for determining which peak is the most appropriate is based on the maximal slope of any given peak to another peak - including the plateau local maxima. When considering which peak a partition in the domain is associated with, the slope and distance to peak are the two parameters used. Figures 16(b)-16(e) show the scenarios that a partition may encounter. In each case, the distance between a partition and a peak, Δr_{kq}, is found to be either "close by" or "far away" based on a radius determined by the slope for each peak. The radius is found by taking the partitions weight difference and dividing by the slope for the peak:

$$\text{Radius} \equiv R_{kq} = \frac{wgt_q - wgt_k}{s_q} \tag{21}$$

$$\text{Close By} \equiv \Delta R_{kq} \leq R_{kq} \tag{22}$$

$$\text{Far Away} \equiv \Delta R_{kq} > R_{kq} \tag{23}$$

$$\tag{24}$$

Algorithm 2 Connection Cluster algorithm (Matlab notation)

```
1: procedure CONNCLUSTER(NN1)
2:     NN1tmp ← NN1 + eye(N_P)                          ▷ Add the Identity matrix to NN1
3:     [p, q, r, s, cc, rr] = dmperm(NN1tmp)    ▷ Routine dbperm permutates the NN1 matrix
4:                                          ▷ p, array of all partitions ordered by connected clusters.
5:                                          ▷ r, array of starting indices for each connected cluster.
6:     clustercnt ← 0
7:     for all i ∈ r (except last entry) do
8:         clustercnt ← clustercnt + 1
9:         CONN(clustercnt, :) ← p(r(i) : r(i + 1) − 1)
10:    end for
11:    return CONN                    ▷ Returns matrix of clusters of connected partitions.
12: end procedure
```

A.2 Path Length Based Calculations

This study employs clustering techniques utilizing the path length between two partitions. When two partitions cannot form a path between them, the path length matrix entry is set to infinity. Dijkstra's algorithm (1959) and variations of it are used to calculate the shortest path length. The first nearest neighbor matrix, **NN1**, is a weighted adjacency matrix used to search for the path starting from one partition (k) continuing making NN1 steps until reaching a final partition (ℓ).

When partitions are connected it is useful to reorder the partitions so that their values in matrix form appear block-diagonal. The algorithm of Dulmage and Mendelsohn permutes rows and columns until a block form is reached (1958). Algorithm 2 shows how partitions are reordered as well as assigned to clusters via connection.

Definition 23 (Path Type Values)
Stepwise $\leftarrow$ *A value defined between nearest neighbors summed as a path is traversed.*
Pathwise $\leftarrow$ *A value defined between the originating partition and each subsequent partition along a path which is summed as the path is traversed.*

The path length calculation uses the **NN1** matrix which gives the L2 distance from one partition to a neighboring partition. When using the **NN1** matrix with Dijkstra's algorithm, the path length that is calculated sums the step-by-step distance when going from $[k, \ell]$, leading these calculations to be called *stepwise*. Other path-based calculations needed in this study require distances to be summed from the k^{th} partition to each subsequent partition along the path, dubbing these calculations *pathwise*. The criteria used to establish a Line-Of-Sight (LOS) between two partitions relies on the summation of L1 distances to points along the path taken from $[k, \ell]$. This distance has the property that when traversing a grid from $[k, \ell]$, the distance calculated is different than when returning from $[\ell, k]$. The asymmetry of this measure proves useful in determining the LOS condition. Figure 18 illustrates how the distance is asymmetric with regard to the path taken.

Algorithm 3 Path Length Algorithm (Matlab notation)

procedure PATHLENGTH(**NN1**) ▷ Compute a stepwise L2 path length.
  $GraphPath \leftarrow graph(\mathbf{NN1})$ ▷ Form graph from symmetric weighted adjacency matrix.
  PathL $\leftarrow distances(GraphPath)$ ▷ Use Dijkstra's Algorithm for distance.
  return PathL $\equiv \mathbf{\Delta L}$ ▷ Matrix of distances between partitions (stepwise).
end procedure

Algorithm 4 Path Length Algorithm (Asymmetric) (Matlab notation)

procedure PATHLENGTHASYM(**NN1**, **AsymMatrix**)
  ▷ Compute a pathwise, asymmetric path length.
  for all $partitions, k$ **do** ▷ Loop through each partition, row by row.
    $AsymRow \leftarrow$ **AsymMatrix**$(k, :)$ ▷ Copy the k^{th} row to fill a square matrix
    AdjacencyAsym $\leftarrow repmat(AsymRow, N_P, 1) \circ logical(\mathbf{NN1})$
    ▷ Create an asymmetric Adjacency matrix.
    $GraphPath \leftarrow graph(\mathbf{AdjacencyAsym})$
    ▷ Forms a directed graph from adjacency matrix.
    PathLAsym$(k, :) \leftarrow shortestpath(GraphPath, k, :)$ ▷ Shortest distance from k to rest.
  end for
  return PathLAsym ▷ Matrix of pathwise distances between partitions.
end procedure

In path-based calculations, a graph is formed based on an adjacency matrix which will either be the **NN1**, **SL1** or **SL1VAR** matrices. The **NN1** matrix is symmetric and stepwise, which is used as a weighted adjacency matrix to form an undirected graph which is then used to calculate the path length based on L2 steps taken from $[k, \ell]$. This application of the adjacency matrix to form a graph is a standard approach, outlined in algorithm 3.

The **SL1** distances form an asymmetric and pathwise matrix, which requires an alteration to how Dijkstra's algorithm is applied to calculating the summed L1 path length. Starting from the first partition, the starting partition is fixed (k) and the top row of the **L1** matrix is copied to fill a square matrix. This matrix represents the L1 distances from k to all other partitions. To insure steps are taken by nearest neighbors only, the **NN1** matrix is first changed to logical values then multiplied element by element with the $\mathbf{L1_k}$ matrix just created for partition k only. By applying

this matrix as the weighted adjacency matrix, a directed graph is formed, where each step along the path from $[k, \ell]$ is then summed using the $\mathbf{L1_k}$ value, but in each case, when taking a new step utilizing Dijkstra, the value used is always fixed to the distance from the initial partition, making this calculation pathwise rather than stepwise. After all path distances have been calculated from the current partition to all others, the process is repeated for all other rows by fixing the next partition (row) and copying that row to fill a square matrix, at which point Dijkstra's algorithm is applied giving the pathwise summed L1 distances from the new partition to all others. The final collection of summed L1 path distances form the matrix **SL1**.

The **SL1VAR** matrix is asymmetric, pathwise and treated in a similar fashion as the **SL1** matrix in order to calculate the minimal variance of the summed L1 distance from the true summed L1 distance for a linear path. Algorithm 4 outlines the procedure for treating asymmetric path calculations using Dijkstra. The usage of **SL1** and **SL1VAR** in calculating the LOS criteria are discussed in App. B.

Appendix B. Line-of-Sight (LOS) Criteria

A Line-Of-Sight algorithm determines whether two partitions can be connected by an unbroken line. The **LOS** matrix produced is a logical matrix indicating no direct path (false) or direct path (true) exists for the $LOS_{k\ell}$ element. The three LOS criteria will be discussed in this section, leading to the formation of the LOS matrix. Each criteria places tighter restrictions on partitions to be LOS:

1. A path or set of paths must exist between $[k, \ell]$ equal to the true path length, ΔL_T. The set of all paths sharing this length form a parallelotope with partitions k and ℓ placed at the far corners of the region, called the $L2_T$ *convex hull*. This criteria is the least restrictive and requires the least computational effort.

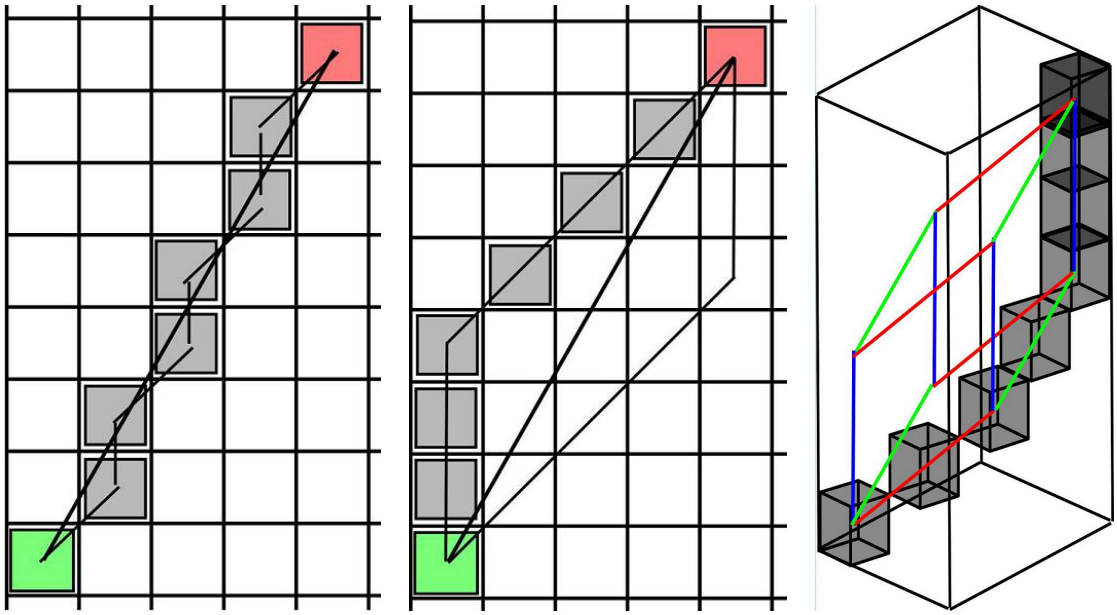
2. The summed L1 distance between $[k, \ell]$ along all paths within the $L2_T$ convex hull must be less than or equal to the true summed L1 distance, $\Delta SL1_T$. This is a strong restriction which removes all paths taken around a corner between $[k, \ell]$. The computational time is greatest for this criteria.

3. The sum of the squared differences of the summed L1 distance and true summed L1 distance, $SL1VAR(k, \ell)$ must be less than the pathcount. This is the most restrictive criteria demanding that a path form a line between $[k, \ell]$. The computational time is generally less than the second criteria but can be as much.

The first criteria determines whether two partitions are connected within a trapezoidal region, defined by the vector of differences of their individual bin addresses, $\mathbf{\Delta b}$. In 2D, this forms a parallelogram while in N_D this is a parallelotope. If two partitions are connected within the parallelotope by some path, see Figs. 18(a)-18(c), they may be LOS, however, when a path is taken outside of this region, the two partitions are not LOS to each other.

B.0.1 Path Length Calculations

All LOS criteria listed above require several calculations to be done based on two types of path lengths, L1 and L2 norms. There are five path based calculations needed (in all cases between $[k, \ell]$): the true L2 path length, $\mathbf{\Delta L_T}$, the true summed L1 path length, $\mathbf{\Delta SL1_T}$, the minimal



(a) (b) (c)

Figure 18: Illustrating the True Path Length from the k^{th} partition (green) to the ℓ^{th} (red). The left panel shows the direct path and the nearest neighbor steps required to follow that path. The middle panel shows that the same path length can be taken along the edge of a parallelogram, bounded by the rectangular region between $[k, \ell]$. The right panel shows how to extend the idea into higher dimensions.

summed L1 path length, $\mathbf{\Delta SL1_{min}}$, the maximal summed L1 path length, $\mathbf{\Delta SL1_{max}}$, and the summed L1 path difference, $\mathbf{\Delta SL1_{\Delta}}$. Also essential to these calculations is the orthogonal convex hull formed between $[k, \ell]$, which defines the set of partitions that can be considered for a path that is LOS. The true L2 path length is the shortest path length possible taking NN1 L2 steps, previously defined in Sec.3. The true summed L1 path length has a similar definition, being the summed L1 distances taken along the straight line path. The minimal summed L1 path is the path taken which minimizes the summed L1 distance and the maximal summed L1 path maximizes this length. The L1 path difference is the difference between the maximal summed L1 and minimal summed L1 path distances. In addition to the path based calculations, four vectors are required based on the orthogonal convex hull; the *corner rank vector*, **CR**, which is simply $[1...N_D]$, which delineates the types of corner steps taken along a path, the *convex hull length* vector, **CVL**, which is the difference of the bin address vectors of $[k, \ell]$, the reverse sorted convex hull length, **RSCVL**, and the forward difference reverse sorted vector of the orthogonal convex hull length, **ΔRSCVL**.

$$\mathbf{CVL} = \mathbf{b}_k - \mathbf{b}_\ell \tag{25}$$

$$\mathbf{RSCVL} = \text{sort}\left(\mathbf{CVL},\ 'descend'\right) \quad \text{(Matlab notation)} \tag{26}$$

$$\mathbf{CR} = [1...N_D] \tag{27}$$

$$\mathbf{\Delta RSCVL} = \text{diff}\left([\ \mathbf{RSCVL}\ 0]\right) \quad \text{(Matlab notation)} \tag{28}$$

Table 4: Table of steps, coordinates and the summed L1 path distance for the two examples given by Figs. 18(b) and 18(c). The top three rows represent the number of steps taken along each dimension *from one partition to the next*. The next three rows are the coordinates of the partitions along the paths taken, remembering that the path starts with the k^{th} partition at the origin. The final row is the summed L1 path distance.

dims	Steps Taken Along Each Dimension - k^{th} partition is the origin																													
	2D Case (min path)							2D Case (max path)							3D Case (min path)							3D Case (max path)							
x_1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
x_2	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	
x_3															0	0	0	0	0	1	1	1	1	0	0	0	0			
	Coordinates Of Path Along Each Dimension - k^{th} partition is the origin																													
	2D Case (min path)							2D Case (max path)							3D Case (min path)							3D Case (max path)							
x_1	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7	1	2	3	4	5	6	7		
x_2	0	0	0	1	2	3	4	1	2	3	4	4	4	4	0	0	0	1	2	3	4	1	2	3	4	4	4	4		
x_3											$\Uparrow x_2$ holds at 4				0	0	0	0	0	1	2	1	2	2	2	2	2	2		
																							$\Uparrow x_3$ holds at 2							
																												$\Uparrow x_2$ holds at 4		
SL1	1	3	6	11	18	27	38	2	6	12	20	29	39	50	1	3	6	11	18	28	41	3	9	17	27	38	50	63		

Figures 18(a)-18(c) illustrate how the path based calculations are made. The two partitions in this calculation form an orthogonal convex hull, with the k^{th} partition in one corner and the ℓ^{th} partition in the farthest other corner. The centers of each square (partition) are the bin indices, beginning with the lower left as (1,1) to the upper right at (5,8) for the 2D case; for the 3D case, the lower left bin is (1,1,1) and the upper right is (3,5,8). The convex hull lengths, **CVL** are [4,7] and [2,4,7] respectively and the sorted and descending convex length differences, **ΔRSCVL**, are [3,4] and [3,2,2] respectively. The reverse sorted convex hull length difference vector is the number of steps required to take at each successive corner rank, with a corner rank of one being a straight step, rank two a step in two dimensions, etc...

As figures 18(a),18(b) show in 2D, the path length along the direct path between $[k, \ell]$ is the exact same length as if taken along the edge of a parallelogram bounded within the convex hull. In 3D, Fig.18(c), the same is true, however, two differing kinds of corner steps can be taken. In higher dimensions, the set of all paths taken from $[k, \ell]$ that have the same L2 path length form a parallelotope. The straight line path between $[k, \ell]$ shares the same path length as a path moving along the edge of the parallelotope, making the calculation for the true L2 path length simple by taking the dot product of the reverse sorted difference convex hull length vector with the square root of the corner rank vector:

$$\mathbf{\Delta L2}_T \;=\; \mathbf{\Delta RSCVL} \cdot \sqrt{\mathbf{CR}} \tag{29}$$

The first LOS criteria demands that partitions considered in further LOS tests must have the path length equal to the true path length, $\mathbf{\Delta L} \;=\; \mathbf{\Delta L2_T}$, such that these partitions are located within the parallelotope, forming the more constrained $L2_T$ *convex hull*. The path length algorithm has been discussed in Sec.A.2.

The summed L1 path length is a pathwise length described in Sec. A.2, where the L1 norm is calculated for a set of partitions. Using a modified Dijkstra's algorithm, the minimal path is found of the summed L1 norms taken along the path from $[k, \ell]$. The minimal summed L1 path, $\mathbf{\Delta SL1}_{min}$, does not take steps directly from k to ℓ, rather, it takes steps which minimize the L1 norm by moving in a line taking no corners for as long as possible. After exhausting all steps with

no corners taken, the algorithm then finds the path with the most steps taken with the nest lowest rank corner, a 2D corner. After exhausting this option, further corner steps are taken until the final partition, ℓ, is reached. Table B.0.1 outlines this process. The columns in the table represent steps taken. The rows represent the number of corners taken per step. Initially, all steps are 1D (a line). The number of possible steps that can be taken at each corner rank is given by the forward difference reverse sorted convex hull length vector, $\mathbf{\Delta RSCVL}$. For the examples given, in the 2D case, the minimal summed L1 path will take three 1D steps followed by four steps in 2D. For the 3D case, three steps will be 1D followed by two 2D steps then two 3D steps. The table represents these steps by assigning a one or a zero for the number of dimensions used for the corner of that particular step. The minimal summed L1 path takes the larger corners last in the sequence, which prevents the summed L1 value from becoming large. In Fig. 18(b) the minimal path corresponds to the upper edge of the parallelogram.

By contrast, the maximal summed L1 path, $\mathbf{\Delta SL1}_{max}$, takes the highest rank corners first and ends with the lowest rank steps. This path corresponds to the lower edge of the parallelogram in Fig. 18(b). Table 4 shows the effect of taking the higher ranked corners first. The number of steps at each rank is the reverse order of $\mathbf{\Delta RSCVL}$. By taking the largest ranked corners first, once the steps are exhausted for these ranked corners, the L1 value for that dimension is held constant for the remainder of the summed L1 path length. In the 2D case, four steps are taken at rank two, then all further steps have the value four as part of the L1 norm. Finishing the steps in the example, the last three steps each have four added. In the 3D case, two steps at rank three are taken, then two steps at rank two then three steps at rank one. The middle three rows are the coordinates at each step. The last row shows the running sum along each dimension, which is added together to form the summed L1 path distance.

The straight line path between $[k, \ell]$ will have the *true* summed L1 path distance, $\mathbf{\Delta SL1_T}$, which is the average of the two extrema, $\mathbf{\Delta SL1}_{min}, \mathbf{\Delta SL1}_{max}$. The difference between the two extrema is also necessary to calculate, $\mathbf{\Delta SL1}_{\Delta}$. From table 4, the calculations for the extrema are easiest to see if calculated *along each dimension*. In the minimal summed L1 case, this is simply the cumulative summation of ones for the convex hull length along each dimension. In the maximal summed L1 case, the calculation is similar, yet the highest coordinate reached along each dimension is now held for the remainder of the sum, indicated in the table. In each calculation, the total number of steps is simply the path count, P_C, defined in Sec. 3.

$$\begin{aligned}
\mathbf{\Delta SL1}_{min} &= \sum_{i=1}^{N_D} \frac{1}{2} RSCVL_i \left(RSCVL_i + 1\right) \\
&= \tfrac{1}{2} \left(\mathbf{\Delta SL1} + \mathbf{\Delta L2}^2\right) && (30) \\
\mathbf{\Delta SL1}_{max} &= \sum_{i=1}^{N_D} \frac{1}{2} RSCVL_i \left(RSCVL_i + 1\right) + \sum_{i=1}^{N_D} (P_C - RSCVL_i) RSCVL_i \\
&= \left(P_C + \tfrac{1}{2}\right) \mathbf{\Delta SL1} - \tfrac{1}{2} \mathbf{\Delta L2}^2 && (31) \\
\mathbf{\Delta SL1}_T &= \tfrac{1}{2} \left(P_C + 1\right) \mathbf{SL1} && (32) \\
\mathbf{\Delta SL1}_{\Delta} &= \mathbf{\Delta SL1}_{max} - \mathbf{\Delta SL1}_{max} \quad = \quad P_C \mathbf{\Delta SL1} - \mathbf{\Delta L2}^2 && (33)
\end{aligned}$$

When using Dijkstra's algorithm to find a path length, the minimal value at any given step along the path is always found, meaning that in the case of the summed L1 path distance, the algorithm will find the lowest value it can going from $[k, \ell]$. If all paths within the $L2_T$ convex hull are available

to traverse, then Dijkstra will find the path with $\mathbf{\Delta SL1}_{min}$. When returning from $[\ell, k]$ Dijkstra's algorithm will find a new minimal pat, which is equivalent to the **maximal** path when going from $[k, \ell]$. The asymmetry Dijkstra's algorithm creates is useful in finding the correct straight line path, illustrated in Fig. 17. Due to this asymmetry, two values are considered when determining LOS, $\mathbf{\Delta SL1}$ for $[k, \ell]$ as well as the transpose, $\mathbf{\Delta SL1}^{\dagger}$ for $[\ell, k]$. The minimal and maximal paths in 2D form a plane which passes through the true summed L1 path. In higher dimensions, the minimal and maximal paths are not constrained to a plane, however, they do follow two extreme edges of the $L2_T$ convex hull. Fitting a line through each extreme path again gives a plane whose center line is the true summed L1 path. The values of the all paths within the $L2_T$ convex hull are either below the true value (closer to the minimal path) or above the true value (closer to the maximal path) as shown in Fig. 17.

The second criteria to determine LOS compares the two values $\left(\mathbf{\Delta Sl1}, \mathbf{\Delta SL1}^{\dagger}\right)$ previously mentioned to determine if a path turns a corner or not. When a limited number of paths are available between $[k, \ell]$, three possibilities can exist. The first case is if the two partitions are around a corner from each other, where one of the two summed L1 distances will be forced to be greater than $\mathbf{\Delta SL1}_T$, which means that there is no LOS path from $[k, \ell]$. The second case could be that two paths exist which give both summed L1 values below $\mathbf{\Delta SL1}_T$, however, a straight line path is not possible due to an empty bin along the straight path. In this case, both paths encompass the empty bin, such that the criteria $\left(\mathbf{\Delta SL1}, \mathbf{\Delta SL1}^{\dagger}\right) < \mathbf{\Delta SL1}_T$ is a false positive, it fails LOS. The third case is that one of the two summed L1 values is equal to $\mathbf{\Delta SL1}_T$, in which case, the two partitions are LOS to each other and no further criteria is required.

In order to resolve the second case, Dijkstra's algorithm is employed a third time, using the asymmetry from the two values given from $[k, \ell]$ and $[\ell, k]$, a new pathwise value is applied to Dijkstra's algorithm, the squared difference of the summed L1 path distance from the true summed L1 distance, called the summed L1 variance:

$$\mathbf{SL1VAR} \;=\; \sum_{j=k}^{\ell} \mid \mathbf{\Delta SL1}_{kj} - \mathbf{\Delta SL1}_{T,kj} \mid^2 \tag{34}$$

The value of **SL1VAR** is similar to a variance in that it is minimized around the true value. For this reason, using **SL1VAR** as the pathwise value applied to Dijkstra's algorithm will seek a minimal path from $[k, \ell]$ along the path closest to the straight path. For each step along the *straight line path*, the difference between the summed L1 distance and the true distance should never be greater than one or one-half, such that the summation of steps along a straight line path should not be greater than the path count, P_C. The third LOS criteria is that the summed L1 variance should be: $\mathbf{SL1VAR} \leq P_C^2$. This criteria is too conservative as the $L2_T$ convex hull may be constrained such that of the paths available, all are closer to the straight path than this limit, in which case, an adaptive criteria for LOS is that $\mathbf{SL1VAR} \leq \frac{1}{4}\mathbf{\Delta SL1}_{\Delta}$ which is halfway between the minimal summed L1 distance and the true summed L1 distance.

With respect to Dijkstra's treatment of **SL1VAR**, the value is both asymmetric and pathwise and requires caution as some of its elements will be zero. In this case, the value being summed may be zero, which confuses its interpretation as an adjacency matrix, which typically has zero values for non-connected entries. In order to avoid this confusion, a small non-zero value is applied to each entry in **SL1VAR** that is intended to be used as part of the path sought, keeping zero values for non-connected partitions. Further, Dijkstra's algorithm treats multiple paths with exact same values by selecting one of the paths at random. The small non-zero value applied can be used to

force Dijkstra's algorithm to choose one path which might be slightly lower due to the artificial bias introduced by the small value. To apply the small value, ϵ, effectively, each entry in **SL1VAR** is summed with $(-1)^{L1}\epsilon$, which slightly increases and decreases values along the grid of partitions, which adjusts to the convex hull.

The three LOS criteria can be written as follows:

1) $\mathbf{\Delta L} = \mathbf{\Delta L2}_T$	fastest,	forms the $L2_T$ convex hull
2) AND $\left(\mathbf{\Delta SL1} \leq \mathbf{\Delta L1}_T \ , \ \mathbf{\Delta SL1}^{\dagger} \leq \mathbf{\Delta L1}_T\right)$	slowest,	eliminates corner paths
3) $\mathbf{SL1VAR} \leq P_C^2$ OR $\mathbf{SL1VAR} \leq \frac{1}{4}\mathbf{\Delta SL1}_{\Delta}$	moderate,	seeks linear path

Dijkstra's algorithm is used in each of the three LOS criteria as it is well-known and many optimized routines have been written for high computational efficiency. The first criteria uses Dijkstra in the traditional symmetric, step-wise fashion, using a single adjacency matrix, running very quickly, typically 15-20 times faster than the pathwise application. The second and third application of Dijkstra runs slower as each computation is pathwise requiring a new adjacency matrix to be generated as an input to the algorithm. Each LOS criteria further restricts the number of partitions that are LOS to each other, suggesting that as time the user may choose the level of LOS applied by using all or fewer of the criteria based on the need and computational costs. Finally, the third criteria may be tightened or relaxed based on the limit placed on the summed variance of the L1 distance. As each LOS criteria becomes more restrictive, the number of partitions considered can be reduced at each step. After criteria #1, only those partitions within the $L2_T$ convex hull are considered for further paths. After criteria #2, only those partitions which survived the corner condition are then considered for the criteria #3. For this reason, the first criteria runs the fastest, being stepwise, while the second criteria requires the most computational effort, and the third criteria may take as long as the second, but generally will be faster as fewer partitions are considered for LOS.

References

Wesam Ashour Barbakh, Ying Wu, and Colin Fyfe. Review of clustering algorithms. In *Non-Standard Parameter Adaptation for Exploratory Data Analysis*, pages 7–28. Springer, 2009.

D. A. Binder. Bayesian cluster analysis. *Biometrika*, 65(1):31–38, 1978. ISSN 00063444. URL `http://www.jstor.org/stable/2335273`.

F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

E.W. Djikstra. A note of two problems in connexion with graphs. *Numerische Matematik*, 1: 269–271, 1959.

A. L. Dulmage and N. S. Mendelsohn. Coverings of bipartite graphs. *Canad. J. Math.*, 10:517–534, 1958.

James A. Hansen. Accounting for model error in ensemble-based state estimation and forecasting. *Monthly Weather Review*, 130(10):2373–2391, 2002.

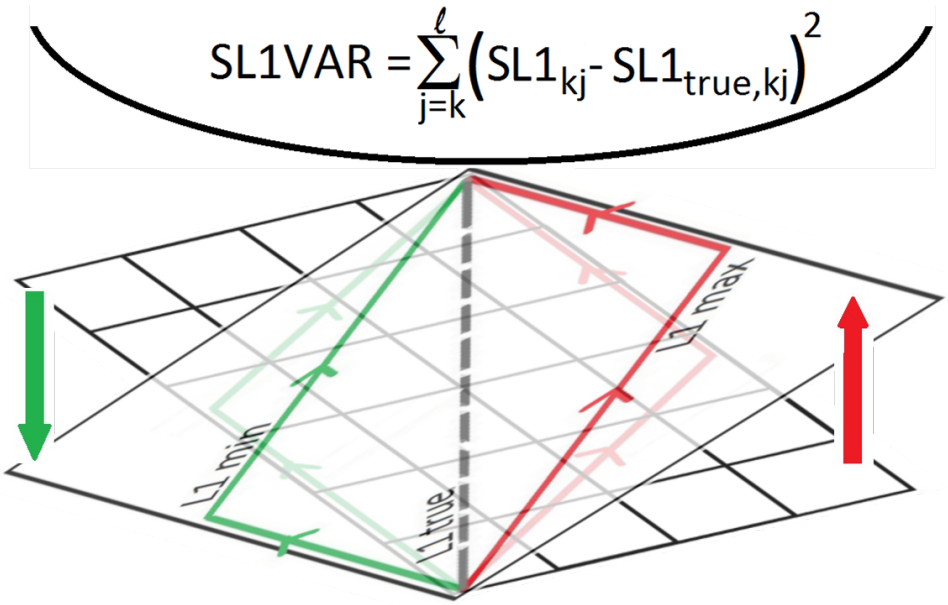


Figure 19: Figure illustrates the two paths of $\mathbf{\Delta SL1}_{min}$ and $\mathbf{\Delta SL1}_{max}$, which in 2D is a plane. The average between them gives the straight line "true" value, $\mathbf{\Delta SL1}_T$. When all paths within the $L2$ convex hull are available, Dijkstra's algorithm will seek the $\mathbf{\Delta SL1}_{min}$ path when going from $[k, \ell]$ and will seek the $\mathbf{\Delta SL1}_{max}$ path going from $[\ell, k]$. This asymmetric behavior is exploited in order to find the straight line path by applying Dijkstra on a third pass, where the pathwise value applied is the squared difference between the $\mathbf{\Delta SL1}$ and $\mathbf{\Delta SL1}_T$, as is shown above the plane as a parabolic bowl.

Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.

Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

Alexander Strehl and Joydeep Ghosh. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, 3:583–617, March 2003. ISSN 1532-4435.

Claudia Tebaldi and Reto Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 365(1857):2053–2075, 2007.

Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007. ISSN 1573-1375. doi: 10.1007/s11222-007-9033-z. URL `http://dx.doi.org/10.1007/s11222-007-9033-z`.