Clustering with obstacles for Geographical Data Mining

TITLE:	Clustering with Obstacles for Geographical Data Mining
SHORT TITLE:	Clustering with Obstacles for Geographical Data Mining
AUTHORS:	
AFFILIATION 1:	
AFFILIATION 2:	
TOPICS IN SPECIAL ISSUE	New techniques to derive information and knowledge from geo-spatial data; advanced data mining techniques. New methods and approaches to spatial analysis of vector data, raster data and hybrid data
KEYWORDS	Geographical Data Mining, Clustering, Delaunay Triangulation, Obstacles

Clustering with Obstacles

for

Geographical Data Mining

Abstract

(*100 words*)

Clustering algorithms typically use the Euclidean distance. However, spatial proximity is dependent on obstacles recorded in other layers. We present a clustering algorithm suitable for large spatial databases with obstacles. The algorithm is free of user-supplied arguments and incorporates global and local variations. The algorithm detects clusters in complex scenarios and successfully supports association analyzes between layers. All this within $O(n \log n + [s + t] \log n)$ expected time, where n is the number of points, s is the number of line-segments that determine obstacles and t is the number of Delaunay edges intersecting obstacles. Performance evaluations illustrate the power of our approach.

Clustering with Obstacles for Geographical Data Mining

1 Introduction

Geographic Information Systems (GIS) are advancing from mere repositories of geo-referenced data to powerful analysis tools for real world phenomena. However, advances in the automation of analysis are behind the advances in the technologies for collecting and manipulating data. Fast data collection processes result in massive spatial databases with many geographical layers that far exceed the exploration capabilities of human analysis (33; 37; 38). Hence, the recent interest for automatic or semiautomatic tools to enhance the exploratory and analytical capabilities of GIS under the umbrella of Geographical Data Mining (37) or Spatial Data Mining (Koperski et al.). GIS needs to expand in the direction of a sophisticated pattern spotter, that suggests highly likely plausible hypotheses, and is capable of handling hundreds of different layers that may contain thousands or millions of data points.

Traditionally, spatial statistics have been the core of analytical methods to identify spatial patterns, but these traditional approaches are computationally expensive and essentially confirmatory. In addition, they require prior knowledge and domain knowledge (33). While some have indicated why statistical analysis becomes inappropriate or unsuitable for data-rich environments (37; 38), this very much depends on what can be considered an statistical approach, since data mining overlaps significantly with the aims of statistics for understanding and analyzing data (24).

As an alternative, Geographical Data Mining offers approaches like Spatio-Dominant Generalization, where clustering point data is a frequent and fundamental operation (Koperski et al.). Clustering for large spatial datasets is so central to data mining in GIS that several algorithms have been developed (9; 23; 26; 35; 50; 51; 54). In Spatio-Dominant Generalization, clustering works as a summarization or generalization operation on the spatial component of the data. Later, an agent (computer or human) can explore associations between layers (17; 14; 28; 44; 45; 46) or conduct exploratory data analysis. In fact, other layers are explored to attempt to suggest hypotheses that may explain the concentrations found by the clustering algorithm.

The benefit is that unsuspected relationships between the layers are identified (discovered) as potential candidates for causal relationships (perhaps to be submitted to a further confirmatory analysis). For example, when clusters are found in the point data, we expect to be able to suggest explanations for them from the interaction of the layers.

One crucial interaction between the themes represented in layers is that one theme may work as

an obstacle for the entities represented in another theme. For clustering, in particular, it is not only relevant but necessary to consider and explore obstacles as part of the elements that define spatial proximity and spatial heterogeneity. Clearly, clustering point data in one layer, on the assumption that the Euclidean distance is the optimal evaluation of proximity when another layer creates barriers, is bound to be unsuccessful. This paper extends a spatial clustering algorithm AUTOCLUST+ (13) into a framework of algorithms that can incorporate the information of other layers as obstacles. The result is a generic algorithm that allows the exploration of the effect of the obstacles on the clustering results and allows the regulation of the influence of the obstacles. It also makes computationally efficient such exploration for different levels of effect. Note that a simple but crisp consideration of the effect of the obstacles can directly use an algorithm that clusters disregarding the obstacles, and use the Euclidean distance. This crisp approach later overlays the obstacles to cut (separate) the clusters. While this approach is one extreme of the possibilities of our algorithm, we suggest here that the spatial heterogeneity resulting from measuring distance along paths avoiding obstacles is to be considered in the evaluation (and formation) of clusters. We also make the point that the result will be more informative.

We refer to our algorithms as AUTOCLUST+ because its fundamental methods are based on AUTOCLUST (15), but they incorporates the consideration of obstacles while preserving crucial properties in AUTOCLUST. The user does not have to supply argument values. The values of potential arguments are derived from the analysis of Delaunay Diagrams. For all those directions in which the possibility of exploration exists because there are modeling assumptions, the algorithm still uses what we have found to be very robust defaults (and for which several justifications have been supplied in earlier work). The default form of AUTOCLUST+ suggests quality clusters to further the investigation of relative information. All this is performed within $O(n \log n + [s+t] \log n)$ expected time [1], where n is the number of data points, s is the number of line-segments that determine the obstacles and t is the number of Delaunay edges intersecting at least one obstacle. This is suitable for mining large spatial databases. Moreover, if the set of obstacles changes, the algorithm recomputes the clustering only in $O(n + s_1 \log n)$ expected time where s_1 is the number of line segments defining the new set of obstacles.

The remaining sections of the paper are organized as follows. Section 2 presents the context of clustering within Geographical Data Mining and data-rich environments. Section 3 summarizes the working principle of AUTOCLUST. Section 4 describes its extension for settings with obstacles. In this section, we also justify our claims for the time complexity of the algorithm. Section 5 offers an examination of the performance of AUTOCLUST+ with an image processing application. A series of detailed evaluations with GIS data have been presented for the default form of AUTOCLUST+ (13).

[1] A function $f(n)$ is $O(g(n))$ if $Cg(n)$ bounds from above $f(n)$ except on a finite set for a suitable constant C; that is, there is n_0 such that $f(n) \leq Cg(n)$ for all $n > n_0$. We use the standard notions of algorithmic complexity (1) for expected-case performance.

These evaluations illustrate the power of our approach and confirm the virtues of the proposed method. Section 6 discusses the differences between our approach and COE (47; 48).

2 The clustering problem

Clustering a point dataset $P = \{p_1, p_2, \ldots, p_n\}$ in some study region R can be regarded as finding smaller homogeneous subgroups according to spatial proximity. Alternatively, any result of a clustering algorithm is a hypothesis that explains the data in some way. It is a selection, among a family of models, for that model that best fits the data. All clustering algorithms optimize an induction principle (6) that defines the family of models and the criteria by which a model is to be regarded as the one that best fits the data. Criteria are usually defined in terms of a similarity or dissimilarity measure and if the algorithms proceeds to construct a hierarchical structure that optimizes the criteria or a partition has been repeatedly used to categorize families of clustering methods (22). Clustering is in the eye of the beholder as much as there are inductive principles (11; 52). The algorithms here find boundaries between regions with different density. Thus, a cluster is a region of relatively uniform density regardless of the shape of its boundary.

Clustering algorithms are applied because they reveal two types of information. First, information that we will refer to as *indicative*, which includes the number of clusters that reside in P, the extent, scatter, size of each cluster, location in spatial dimension, relative proximity among clusters, and relative density and perhaps location in a temporal dimension. Second, information that we refer to as *relational*, which consists of causal factors, potential explanations, cluster justifications, and association to other themes.

Naturally there are several dimensions in this, and some algorithms will be more efficient than others mainly because they either restrict the models by which they will explain the data or they will simplify their search space by asking the user to supply some of the indicative or relational information that they are to discover. For example, an algorithm like Expectation Maximization (EM) (7) may restrict the models for fitting the data to multivariate normal distributions of common variance, and request that the user supply the number of clusters (components) in the mixture. In fact, restricting the models that can fit the data is a form of asking for domain knowledge because the user must know (or be aware) that the true structure of the data is among the family of models the algorithm will attempt to fit (if the data does not fit a mixture of multivariate exponential distributions, EM is unsuitable).

Thus, for exploratory analysis, when the agent (human or artificial) is a data miner we would like to impose the least assumptions or artificial constraints on the data. We hope this will maximize the exploration and reduce external biases. This principle is based on the philosophy "to let the data speak for themselves" (36).

Because of users' domain knowledge, the view in the geography research community is that clus-

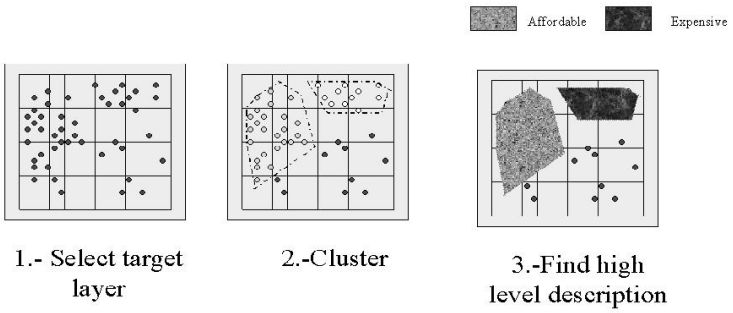


Figure 1: Generic schema of Spatio-Dominant Generalization

ters are, at least in part, in the eye of the beholder. Nevertheless for Geographical Data Mining applications, it is very important that the clustering method detects and suggests as many alternative clusterings as possible. In traditionally human-driven geographical analysis, clustering is very much guided by reference to the background layers. For example, epidemiologists judge high densities of cases in the disease layer as a cluster only with respect to the layer that records the population at risk. However, this approach requires the user to select and incorporate the background layers. In contrast, our clustering methods are for exploration by Geographical Data Mining engines that use the clustering results in Spatio-Dominant Generalization (Koperski et al.). This overcomes issues of background layers and populations at risk. This is specially relevant for data-rich environments where there are many other layers that could be explored as the possible population at risk.

The algorithm here allows this flexibility while maintaining sub-quadratic time complexity (crucial for the massive datasets of Geographical Data Mining applications). It minimizes the need for users to explore arguments of the algorithm in order to obtain good results, but allows for the exploration of scenarios in which the influence of obstacles may not be as crisp (or as transparent as if they were not there).

Clustering is the starting point of many exploratory analyzes in point data (16). The context of our first illustration is Spatio-Dominant Generalization (Koperski et al.). In this setting, the agent (miner) uses clustering to generalize the spatial component of the data (Figure 1 step 1 and step 2). The concentrations or clusters found are then contrasted with the many other layers integrated into a GIS for discovering associations and therefore suggesting relations for further confirmatory analysis (refer to Figure 1, step 3). The exploration phase attempts to find autonomously the unexpected relations between the different layers, and thus, many possible associations are tested. Since clustering is used in this approach to extract point patterns in one layer for association analysis between themes, clustering results must be of high-quality. Otherwise, spatial associations between themes is undetected. In particular, obstacles may be crucial to determining the shape of clusters, and thus, they impact directly the opportunity for discovering associations (since these are revealed by autonomous analysis

with overlays). A motivation of scenarios for clustering analysis with obstacles can be found in (13).

3 The working principle of AUTOCLUST

Two-dimensional point clustering is central to spatial analysis processes. Goodchild (21) emphasized the importance of spatial analysis and modeling for the application of GIS as a spatial information science rather than as a technology. The modeling and structuring of geo-referenced data is strongly related to the analysis capability and functionality of GIS (40). Thus, data modeling, data structuring and spatial analysis are inter-related and play major roles for informative mining of spatial data.

3.1 Background on AUTOCLUST

Modeling topological relations, for instance p_i *is nearby neighbor of* p_j, is central for spatial clusters. We expect that the relation p_i *is in the same spatial cluster as* p_j has some association to the *is_nearby_neighbor* relation. Topological methods, like saying that two objects are neighbors if they intersect or if they share a common boundary, apply to line or area objects but not directly to point data, since points neither intersect nor share boundaries in common unless they coincide. Topological information is derived for point data through point-to-area transformations. Thus, two points are regarded as neighbors if their transformed areas share at least a common boundary. A widely adopted transformation is to assign every location in the study region R to the nearest $p_i \in P$ based on a certain metric. This transformation divides R into non-overlapping regions except for boundaries. The resulting tessellation is the well-known Voronoi Diagram. Thus, two points are neighbors if, and only if, their corresponding Voronoi regions share a common Voronoi edge. By connecting two neighboring points in the Voronoi Diagram, another tessellation, the Delaunay Triangulation, is obtained. Thus, the dual graph explicitly encodes spatial neighborhood information.

Being conscious of the interplay between modeling and analysis capability, AUTOCLUST (15) uses Voronoi Diagrams and their dual Delaunay Diagrams as the data model and data structure, respectively (these remove ambiguities from Delaunay Triangulations when co-circularity[2] occurs). The model, Voronoi Diagram, overcomes the limitations of conventional vector and raster models and guarantees unique modeling of discrete point data (2; 3; 16; 15; 19; 20). The structure, Delaunay Diagram, is succinct and efficient for clustering (3; 8; 25).

Clustering finds sudden global changes in point density. In Delaunay Diagrams points in the border of clusters tend to have greater standard deviation on the length of their incident edges. This is because border points have both short edges and long edges. The short edges connect points within a cluster while the long edges straddle between clusters or between a cluster and noise points. AUTOCLUST accounts for local and global variations to ensure that relatively homogeneous clusters are not broken

[2]Four or more points are co-circular if there exists a circle through them.

into tiny uninteresting subsets, and relatively heterogeneous subsets are split into meaningful clusters. This analysis allows AUTOCLUST to be argument free. In the next subsection, we briefly introduce how AUTOCLUST discriminates intra-cluster edges from inter-cluster edges.

3.2 The Three-Phase clustering in AUTOCLUST

AUTOCLUST proceeds in three phases. Each phase is an edge correction phase. The idea is to start with the Delaunay Diagram and produce a planar graph that synthesizes the discrete relation p_i *and* p_j *are in the same cluster* with the condition that p_i *is connected to* p_j *by a path in the planar graph.* AUTOCLUST first computes a global variation indicator *Mean_St_Dev*(P) that is the average of the standard deviations in the length of incident edges for all points p_i in the Delaunay Diagram. The process to automatically find cluster boundaries uses this global variation as global information in the first two phases. **Phase I** eliminates edges that are evidently too long. In order to detect this, AUTOCLUST computes a local indicator *Local_Mean*(p_i) for each p_i. The value of *Local_Mean*(p_i) is the mean length of edges incident to point p_i. The first phase classifies incident edges for each p_i into three groups.

- *Short_Edges*(p_i) are those edges whose lengths are less than *Local_Mean*(p_i) - *Mean_St_Dev*(P),
- *Long_Edges*(p_i) are those edges whose lengths are greater than *Local_Mean*(p_i) + *Mean_St_Dev*(P) and
- *Other_Edges*(p_i) = (incident edges - (*Short_Edges*(p_i) $\cup$ *Long_Edges*(p_i))).

Criteria for these three groups *Short_Edges*(p_i), *Long_Edges*(p_i) and *Other_Edges*(p_i) are not static, but rather dynamic over the study region, since *Local_Mean*(p_i) varies with p_i. This dynamic nature of AUTOCLUST overcomes the static nature of peak-inference clustering methods (8; 9; 23; 25; 39; 50) and the lack of local considerations in partitioning clustering methods. This constitutes the balance of local and global information. AUTOCLUST obtains rough boundaries of clusters by removing inconsistently long and short edges (*Long_Edges*(p_i) and *Short_Edges*(p_i)) for all points p_i. Because *Long_Edges*(p_i) are too long to join points in the same cluster, they are permanently removed. On the other hand, *Short_Edges*(p_i) may correspond to links between points within a cluster but also may correspond to bridges[3] between clusters, or close-by noise points. The removal of *Short_Edges*(p_i) enables AUTOCLUST to detect clusters linked by multiple bridges. This is a great improvement over traditional graph-based clustering methods (53) that use cut-points to detect bridges.

Phase II recuperates edges in *Short_Edges*(p_i) that are intra-cluster links. It computes connected components and re-connects isolated *singleton connected components* (connected components having only one element) to *non-trivial connected components* (connected components having more than one

[3]Here, bridges is used in the graph theoretic sense, and means edges that connect connected components.

element), if and only if they are very close. The re-connection is performed if and only if (*singleton connected component* p_i is very close to to p_j [specifically, $e = (p_i, p_j) \in Short\_Edges(p_i)$] and $p_j \in$ some *non-trivial connected component*).

Phase III extends the notion of neighborhood of a point p_i to those points reachable by paths of length 2 or less. Then, the indicators of local variation are recalculated. The third phase detects and removes inconsistently long edges for each p_i by the use of a new indicator $Local\_Mean_{2,G}(p_i)$ (the average length of edges in paths of 2 or less edges starting at p_i).

Thus, the three-phase clustering removes all globally long edges and bridges, and reports clusters that are interconnected by intra-cluster edges.

4 AUTOCLUST+

In a proximity graph, points are linked by edges if and only if they seem to be close by some proximity measure (31). In the Euclidean Delaunay Diagramof the point set P (also called ordinary Delaunay Diagram), each Delaunay edge represents a relationship that indicates that two end-points are relative neighbors. In particular, any circle through these two points (of P) is either empty of other points from P or includes some other points from P. Moreover, any three points of P are mutual neighbors if and only if the circumcircle through them is void of any other points from P. Thus if p_i and p_j (both in P) have an edge in the Euclidean Delaunay Diagram, they are as close as possible. So close that you can draw a small circle trough them and exclude any other point point in P. Of course you can draw a large circle and include other points in P. However, when there is no edge between two points in P for the Euclidean Delaunay Diagrams, it means that you can not draw a circle through them that does not include somebody else from P. In such a case, they are not neighbors because there is always some other point between them. Intuitively, the edges are between closest neighbors since there are no other points in between.

Strong interactions (short edges) indicate that two end-points belong to the same cluster while weak ones (long edges) imply that end-points belong to distinct clusters. But, in the presence of obstacles, interactions (Delaunay edges) and paths are blocked by obstacles. An ordinary Delaunay edge which intersects the defining lines of obstacles no longer suffices to link its end-points as members of the same cluster; even if it is very short. The criteria to evaluate now depend also on the length of edges on the detour path (and the transparency associated to the obstacle). We use *detour path* for the shortest path between two points in the Delaunay Diagram where the edges on the path are ordinary Delaunay edges that avoid all obstacles. This is a path in a graph embedded in the plane. In contrast, the *detour distance* is the length of the shortest path in the plane between two points. Figure 2 illustrates the distinction. In Figure 2(a), thick solid lines represent the path whose length constitutes the detour distance. The detour path for the same pair of points is highlighted in Figure 2(b). Dotted edges represent Delaunay edges that traverse an obstacle. Note that, the

lines determining the detour distance are not edges in the Delaunay Diagram. The detour distance is approximated by the length of existing Delaunay edges (detour path). A graph embedded in the plane is a *spanner* if the length of the shorted path *on the graph* between any pair of nodes approximates within a constant the Euclidean distance between those nodes. The length of a detour path in the Delaunay Diagram is a good approximation to the detour distance because the Delaunay Diagram is a spanner (27). Thus, two end-points of a Delaunay edge traversing some obstacles would not belong to the same cluster if Delaunay edges on a detour path of the Delaunay edge are long.

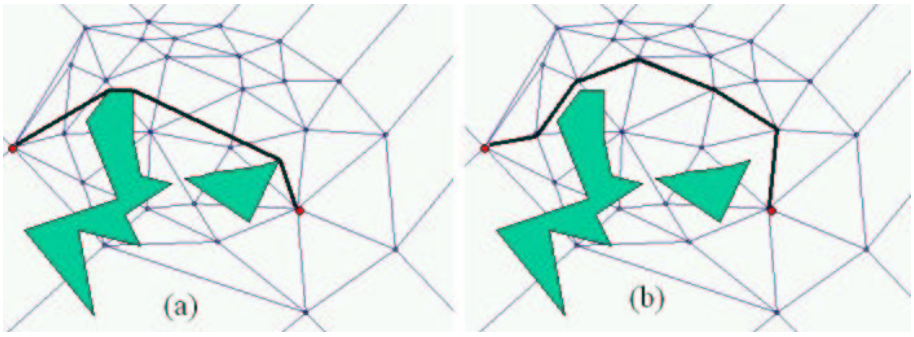


Figure 2: Differentiation between measuring the detour distance and the length of the detour path: (a) Path that determines the detour distance (thick solid lines); (b) Detour path (thick solid lines).

4.1 Algorithms

Here we generalize AUTOCLUST+ (13) into at least four policies to incorporate the effect of obstacles. Two statistics are major components of AUTOCLUST.

1. The local cohesion *Local_Mean*(p_i) evaluates p_i as a spatial median[4] of it and its neighbors.
2. The global degree of variation is measured with *Mean_St_Dev*(P).

The four proposed methods correspond to levels of transparency in the obstacles. The participation of the obstacles into the analysis is implemented as degrees of participation (or removal) into the computation of the two statistics or into the construction of the proximity graph. To show these degrees of inclusion of obstacles we summarize AUTOCLUST+ in pseudo-code next.

Algorithm **AUTOCLUST+**

Input: A set P of points and a set B of obstacles;

[4]Given a set $P = \{p_1, \ldots, p_n\}$ of points, the spatial median (of Fermat-Weber point (29)) of P is the point q that minimizes $FW(q) = \sum_{i=1}^{n} d(q, p_i)$. The smaller the value of $FW(p_i)$, the better nearly p_i approximates the spatial median.

Output: A graph $outG$ with intra-cluster edges;

1) **begin**

2) $\begin{cases} \text{[v1] } G = \texttt{BuildGeneralizedDelaunayDiagram}(P, B); \\ \text{[v2] } G = \texttt{BuildDelaunayDiagram}(P); \end{cases}$

   $\longleftarrow$ `ProcessObstacles`(G, B);

3) $Mean\_St\_Dev(P) = \texttt{ComputeMeanStDev}(G)$;

   $\longleftarrow$ `ProcessObstacles`(G, B);

4) $outG = \texttt{ThreePhaseClustering}(G, Mean\_St\_Dev(P))$;

   $\longleftarrow$ `ProcessObstacles`($outG$, B);

5) **end**

In this template for AUTOCLUST+, we have indicated with a left pointing arrow ($\longleftarrow$) the possible location of an optional step. We have indicated that the second step has two versions, with a version identifier. One or the other version (exclusively) in step 2 is to be used. Now, the step labeled `BuildDelaunayDiagram()` computes an ordinary Delaunay Diagram of the point dataset that it receives as an argument. This step alone simply ignores obstacles. However, `BuildGeneralizedDelaunayDiagram`() returns a generalized Delaunay Diagram derived from a generalized Voronoi Diagram that considers obstacles, and where distances are computed as detour distances to define Voronoi regions. This point-to-area step will be fully aware of crisp obstacles. One or the other (but not both) will define the starting proximity graph G.

We proceed now to describe and discuss four approaches that result from the proposed template. We will refer to Approach 1 as the instantiation of AUTOCLUST that uses step 2.v1 to determine the initial proximity graph. We will refer to Approach 2 as the use of step 2.v2 to build the initial proximity graph. The process labeled `ComputeMeanStDev()` calculates the global variation indicator $Mean\_St\_Dev(P)$. This indicator takes as input a proximity graph and uses only the graph information to determine the points adjacent to each point p_i. The length of these links is the Euclidean distance. From here, the standard deviation at each p_i is derived and their mean (average) is the result of this process.

The process labeled `ThreePhaseClustering()` procedure represents the three-phase clustering of AUTOCLUST, which also only requires a proximity graph and the calculation of the global information value. The process `ProcessObstacles`() incorporates the obstacles into the template by modifying (at that stage) the proximity graph with the removal of all Delaunay edges that traverse an obstacle. The routine `ProcessObstacles`() can be placed in one of three possibilities.

1. Between step 2 and step 3
2. Between step 3 and step 4.

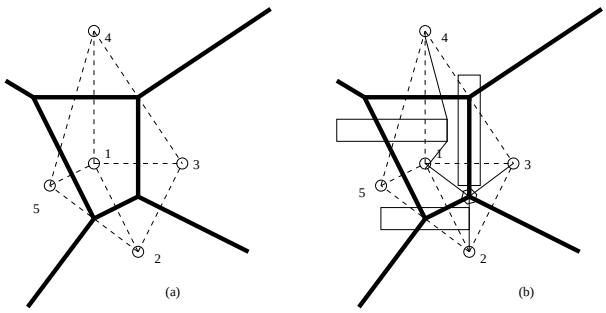


Figure 3: Voronoi regions change in the presence of obstacles, but neighborhood information may not change in the constrained Voronoi Diagram.

3. Between step 4 and step 5.

We will see now that the two options for step 2 and the three insertion points for `ProcessObstacles`() result in essentially four approaches to incorporate obstacles.

1. Constrained Voronoi Diagram based approach (step 2.v1).

   Although this approach constructs a Voronoi Diagram with obstacles and the resulting generalized Voronoi Diagram is typically different from the ordinary Voronoi Diagram without obstacles, the resulting proximity graph may not be informative. In fact, when obstacles are small with respect to Delaunay triangles, even if obstacles block straight lines between two points, the proximity graph may be the same. For the construction of the generalized Voronoi Diagram, the metric used evaluates the distance between two points p_i and p_j as the length of the shortest path in the plane between p_i and p_j that does not cut any obstacle. The geometry is determined by the location, size and every aspect of the obstacles (as well as the metric used to measure the length of segments of the path that does not touch obstacles). Thus, computing the proximity graph is more laborious. We illustrate this in Figure 3. Figure 3 (a) shows 5 data points. The Voronoi regions are indicated by the dark lines, while the Delaunay Diagram is shown in dotted lines. Figure 3 (b) shows the same configuration but with 3 rectangular obstacles. We do not show the constrained Voronoi Diagram but the edges of the proximity graph (the dotted lines) remain although they intersect the obstacles. This is counterintuitive since Point 1 is almost surrounded by obstacles. However, the point equidistant to Points 1, 2 and 3 (highlighted with a $\circ$) remains equidistant to these points despite the obstacles, and thus in the boundary of the constrained Voronoi regions. Similarly, there is still a path to get from Point 1 to Point 4, and any position points along this path have as their nearest point, Point 1 or Point 4. So this path must go over the boundary of the constrained Voronoi regions for 1 and 4. So, Points 1 and 4 remain neighbors.

Thus, because the proximity graph may be very similar to the ordinary Delaunay Triangulation, AUTOCLUST+ with step 2.v1 may reduce to the application of the three-phase of AUTOCLUST to a Delaunay Diagram which may look like Delaunay Diagram without obstacles. The additional complexity in computing the generalized Voronoi Diagram does not pay off. The inclusion of `ProcessObstacles`() is not effective because the obstacles are not reflected in the proximity graph. Moreover, we will see that the algorithms described below are faster for equivalent or better results.

Also, this first approach (Approach 1) considers the neighboring information derived directly from the obstacles. This seems to incorporate the influence of obstacles into the proximity modeling. However, we see that users may be interested in exploring and clustering a target layer with respect to various settings. For instance, they may want to inspect a clustering of the target layer with rivers as obstacles, later with mountains as obstacles, and further, to test the clusters by overlaying waterways, rivers and lakes as obstacles. This very important exploratory spatial analysis would be computationally costly because the entire proximity graph is revised for each layer used as obstacles. For each obstacle layer, we need to compute the constrained Voronoi Diagram. Consequently, this approach is not as efficient for what-if analysis. It will not return results quickly enough for decision making and mining massive spatial databases. Another aspect that is of concern is that the statistic *Mean_St_Dev*(P) is an indication of global and local effects in the dataset P and not on the obstacles. But, under this approach, it will vary significantly with the type of obstacle considered.

2. Edge removal based approach (step 2.v2).

   In this approach, we compute the ordinary Voronoi Diagram without obstacles and derive the ordinary Delaunay Diagram. And then, we remove Delaunay edges that intersect obstacles using `ProcessObstacles`().

   (a) Computing *Mean_St_Dev*(P) after overlaying obstacles.
   (`ProcessObstacles`() between step 2 and step 3.)

   This approach removes Delaunay edges first with the edge removal procedure `ProcessObstacles`() and then calculates the global variation indicator *Mean_St_Dev*(P) after the edge removal. Thus, the obstacles influence the global variation indicator. Finally, we apply the three-phase clustering to the modified Delaunay Diagram.

   Users shall consider some features of this alternative. Here again, the method incorporates an association between layers when computing *Mean_St_Dev*(P). However, this alternative has several drawbacks. First, it is still slightly more expensive in terms of computational efficiency than the two approaches we describe next. This is because here we must compute *Mean_St_Dev*(P) for each new set of obstacles. Second, clustering results are sometimes too

sensitive to the types of obstacles. Typically, this approach cannot separate closely located clusters, because most obstacles are located between clusters or on sparse regions (where obstacles intersect relatively long Delaunay edges). Therefore this approach computes a value for *Mean_St_Dev*(P) that is smaller than what is actually needed. As a consequence, fewer edges belong to *Short_Edges*(p_i) and *Long_Edges*(p_i). Therefore, fewer edges are removed in Phase I and Phase III of the AUTOCLUST procedure. This eventually causes closely located clusters to merge into the same cluster.

(b) Computing *Mean_St_Dev*(P) before overlaying obstacles.
(`ProcessObstacles`() between step 3 and step 4.)

This approach is the same as Approach 2a except it calculates the global indicator *Mean_St_Dev*(P) before removing Delaunay edges intersecting overlaid obstacles. This approach obtains a global indicator of variation in P that is not affected by obstacles, but allows obstacles to change the topological information (incident edges). Thus, the inverse local strength indicator *Local_Mean*(p_i) is affected by obstacles and this new local indicator is used in the three-phase clustering.

This alternative is efficient for clustering with different sets of obstacles. Once the Delaunay Diagram is built, removal of Delaunay edges that traverse obstacles requires $O([s+t]\log n)$, where s is the number of line segments that determine the obstacles and t is the total number of edges removed from the Delaunay Diagram. Typically, s and t are much smaller than n unless the set of obstacles is extremely complex. Once we have the Delaunay Diagram, clustering with respect to new obstacles can be obtained in linear time. Thus, this approach is suitable for exploratory data analysis and what-if analysis. Another good aspect of Approach 2b is that *Mean_St_Dev*(P) is now invariable, regardless of the types of obstacles considered in clustering. Namely, *Mean_St_Dev*(P) is the same whether obstacles are present or absent, so this approach is able to automatically detect globally significant (within the context of R and P) spatial concentrations irrespective of types of obstacles. Thus, we can compare and contrast clustering in the absence of obstacles with clustering in the presence of obstacles with the same global variation indicator.

(c) Removing Delaunay edges after computing spatial clusters.
(`ProcessObstacles`() between step 4 and step 5.)

We compute spatial clusters with total disregard for the obstacles. When the results with obstacles are to be presented, the obstacles are used to cut, divide or separate clusters. Clusters are built with a geometry that is oblivious to obstacles.

This last approach is the most efficient computationally if one considers that it is very likely that the obstacles are actually a different geographical theme and they are stored in a different layer. This approach computes the clusters independently of the obstacles (and

thus, the computational effort for clustering depends only on the size n of the dataset P). It depends on the complexity of the obstacles only later when an overlay of the clusters and the obstacles is being performed; however, this is very fast because the data in P has been summarized to the clusters and the complexity of the obstacles is usually much smaller than n. Thus, exploration with alternative sets of obstacles becomes very fast (the point clustering needs to be computed only once). However, this approach incorporates obstacles only as separators of clusters and not as elements that affect the neighboring information of the point data.

Each approach offers valid modeling for settings with obstacles. The user may have application-specific information in order to prefer one above the other. For example, is housing distributed along a rail track, or does the rail track cut through and separate an apparent cluster of houses? The distinction may be historical. If the rail track was there before many of the houses, because of the presence of a station and a bridge[5] houses may tend to form the cluster located along the track. If the need for a fast train has developed the train line after the cluster of houses, the rail track will effectively split a cluster.

Nevertheless, each decision to model the geometry, and to allow its impact on the topology will result in slightly different results. With AUTOCLUST+, the scenarios of different geometries can all be explored and contrasted. They offer different geometric interpretations and they offer different computational trade-offs.

Table 1 summarizes the differences brought into the two statistics (*Local_Mean*(p_i) and *Mean_St_Dev*(P)) by the four proposed approaches. Approach 1 and Approach 2a have different values of *Local_Mean*(p_i)

Table 1: Comparison of the four approaches (the status of value in the presence of obstacles).

	Approach 1	Approach 2a	Approach 2b	Approach 2c
Local_Mean	changed	changed	changed	not changed
Mean_St_Dev	changed	changed	not changed	not changed

and *Mean_St_Dev*(P) in the presence of obstacles from those values they hold in the absence of obstacles. Different values of *Mean_St_Dev*(P) for different sets of obstacles imply difficulties in exploratory data analysis. Also, the computational cost of Approach 1 makes it less attractive. Approach 2b is seen as a hybrid approach of Approach 2a and Approach 2c. We have explored these alternatives and found Approach 2b the most adequate in terms of the indicative information provided as well as the significantly lower computational cost with respect to Approach 1 and Approach 2a. We believe this approach is the most suitable for exploratory data analysis and what-if analysis. Thus, AUTOCLUST+ uses this approach as a default method for clustering in the presence of obstacles.

[5]Here, a bridge is a physical structure allowing crossing over railtracks.

4.2 Time complexity analysis

In this section, we review in detail the time complexity of Approach 2b. AUTOCLUST+ requires $O(n \log n)$ time to construct Delaunay Diagram (if Approach 1 is used, this is more costly). Next, computing *Mean_St_Dev*(P) is bounded by twice the number of edges. Thus, this takes linear time; that is $O(n)$ time. Removal of edges that traverse obstacles requires $O([s+t] \log n)$, where s is the number of line segments that determine the obstacles and t is the total number of edges removed from the Delaunay Diagram. This is because for each line-segment l defining an obstacle, its end-points can be located within the Delaunay Diagram in $O(\log n)$ expected time. Then, the planar structure of the Delaunay Diagram can be navigated along the line-segment l collecting edges to remove in $O(\log n)$ expected time. Our implementation uses LEDA (32). Typically, s is much smaller than n, resulting in $O(n)$ time unless the set of obstacles is extremely complex. Finally, the three-phase cleaning process works in linear time (15). Thus, AUTOCLUST+ (in default mode) only requires $O(n \log n + [s+t] \log n)$ time. The time complexity of AUTOCLUST+ is dominated by computing Delaunay Diagram, and in default mode, this does not depend on the obstacles. This means that once we have the Delaunay Diagram, it can be stored, and analysis with clustering can be obtained in $O(n)$ time for new sets of obstacles.

5 Performance evaluation

To illustrate the contrast in the alternatives of incorporating obstacles into AUTOCLUST+, we use a real dataset from Sydney, Australia (refer to Figure 4 (a)). The region of study has extensive waterways and is bounded by ocean on the East. Several roads cross water via bridges. The point dataset that is the target of clustering is presented in Figure 4 (b). Clustering without obstacles results in just one cluster as illustrated in Figure 4 (c) (using k-means, k-medoids, CLARANS (35), COE (47; 48) requires users to provide k. Analysis of separation (4) suggest $k = 1$.) Clearly, obstacles must be incorporated for the analysis. Unfortunately, if obstacles are incorporated as Approach 1, the constrained Voronoi Diagram computes distances between the points that are very similar for many cases to the distances that would exist if there were no obstacles. This is due to the pathways over the rivers via roads and bridges. Thus, once again, the result is just one cluster as in Figure 4 (c). The last approach (Approach 2c) that ignores obstacles even after clusters are formed, results in only two clusters, because the streams going East-West cut a North section of the data. Approaches that compute the ordinary Voronoi Diagram but use obstacles to remove edges of the Delaunay Diagram produce much better results. These are shown in Figure 4 (d). Note that the North section is shown as two clusters (one labeled with $\triangle$ and the other with $\heartsuit$) although there is no obstacle between these two sets of points. Nevertheless, these points in isolation from the others (which happens because of the East-West rivers) do seem more like two clusters. Similarly on the region south of the East-West

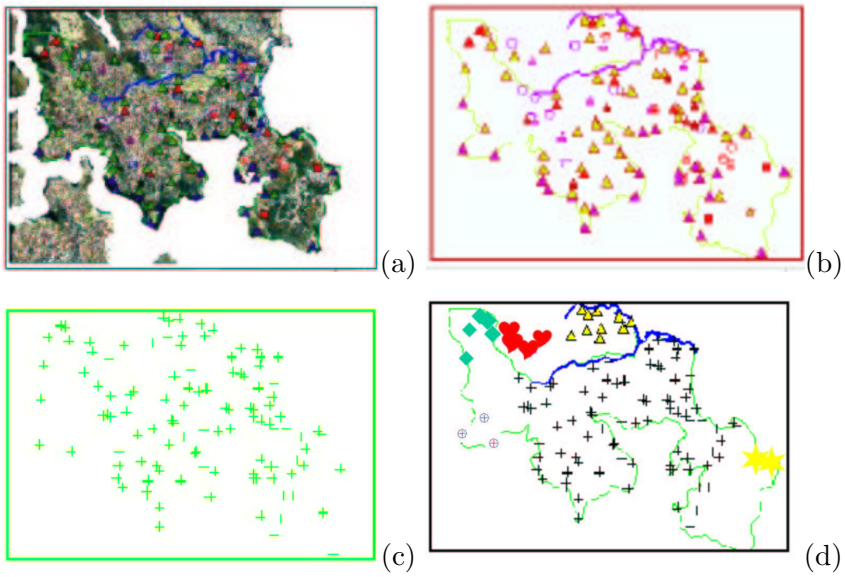


Figure 4: Illustration with real dataset from Sydney, Australia: (a) The region of study has wide waterways and is bounded by the ocean. (b) Clustering is to be performed for point features in the presence of obstacles. (c) If obstacles are ignored, one cluster is obtained involving all points. (d) If obstacles are considered, 6 clusters are found and 4 objects are isolated.

rivers. We actually have four clusters (labeled with $+$, $\star$ $\diamond$ and $\oplus$). The geography of the coastline makes these relevant subgroups. In the light of the obstacles, we see the North-West cluster labeled with $\star$ is indeed relative far from other points, similarly the points in the east peninsula labeled with $\oplus$, as well as the points in the west peninsula labeled with $\diamond$. Note that we have removed points that are placed in a cluster by themselves (singletons). Because Approach 2b is the more efficient computationally, and produces the type of informative results that we observe in this example, it is the default mode for AUTOCLUST+. Experimental results reported below are produced with Approach 2b.

AUTOCLUST+ with its default settings (Approach 2b) produces robust outcomes for complex scenarios with obstacles and noise (13). In particular, AUTOCLUST+ produced remarkable results with the CHAMELEON's benchmark data (26).

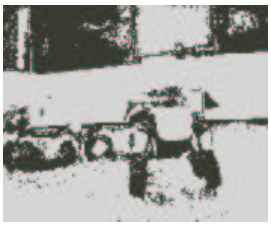
We present here results in the domain of image segmentation that are interesting for remote sensing scenarios where noisy images prevent the use of edge detection. The application of clustering algorithms for large data sets in this context was previously explored in BIRCH (54). We use data from an application of object-recognition based on color in a controlled illumination environment. This is the vision system for the legged league RoboCup competition and associated research symposium (49). Algorithms must be very fast, because of the real-time requirements. Figure 5 illustrates

(a) Original

(b) Edge detection

(c) Color Segmentation

(d) Red pixels



(e) Black pixels

(f) Red clusters with no obstacles

(g) Clusters with Obstacles

Figure 5: Object identification in an image by clustering points of the same color.

the applicability of our algorithm where 25 frames a second must be analyzed. Figure 5(a) shows an image captured by the camera on-board one of these robots. Although in this environment images have little noise with respect to color segmentation, edge detection algorithms have difficulty in finding the patches of the same color that allow the identification of objects. Figure 5(b) shows that edges appear on shadows on the ground and produce some incomplete polygon boundaries on colors. The lack of noise can be confirmed by the color segmentation image (Figure 5(c)). Unfortunately, to convert color classified point data into polygons, clustering algorithms that only produce convex clusters are inapplicable. Moreover, we suggest that the results of clustering say, the red pixels, is much better if we use say, the black pixels, as obstacles. Figure 5(d) shows the red pixels, while Figure 5(e) shows the black pixels. Clustering algorithms without obstacles have difficulty identifying different polygons. Even our powerful AUTOCLUST, when clustering the red pixels, does not distinguish the patches in the heads of the robots from the large body, placing all this in only one cluster (refer to Figure 5(f)). Also, the patches in the legs of the closest red dog are merged with the patch for the body. However, clustering the black pixels first, and using them as obstacles for clustering the red with AUTOCLUST+ results in separate head patches as well as a separate leg path on the right, and two on the left (Figure 5(g)).

6 Discussion

We now review the advantages of the proposed algorithm with respect to other known recent approaches to clustering geo-referenced point data.

First, we believe AUTOCLUST+ better supports the fact that all information is extracted from the data. Other commonly used clustering methods impose a number of assumptions and require a number of user-supplied arguments prior to clustering. The arguments (parameters) required are as follows.

- The number of clusters (typically in partition or representative-based clustering).
- Termination conditions (as in hierarchical clustering methods (23; 26; 54)).
- Values for density thresholds (as in density-based clustering).
- Values for resolution (as in grid-based clustering).
- The form of underlying probability distributions (as in model-based clustering (5; 18; 41)).
- Threshold values for lengths of edges (as in graph-based clustering).

Determining suitable values for these arguments in order for the algorithm to produce its best results is usually a difficult task. The need to find best-fit arguments in these semi-automatic clusterings demands pre-processing and/or several trial and error steps. This is very expensive for massive

spatial datasets. These semi-automatic algorithms inherit their argument-tuning to search best-fit arguments when they are modified for clustering in the presence of obstacles. Minimizing the need to adjust arguments not only reduces the risk of user bias polluting exploratory analysis, but promises higher user friendliness. Our algorithm AUTOCLUST+ is able to find these values from the data automatically, so that novice users can expect quality clusters in the presence of obstacles. It could be argued that the definition of *Short_Edges*(p_i) in Section 3 should introduce parameters α_1 and α_2 to the clustering approach by defining *Short_Edges*(p_i) as those edges whose lengths is less than *Local_Mean*(p_i) $-$ α_1 *Mean_St_Dev*(P)), and *Long_Edges*(p_i) as those edges whose lengths are greater than *Local_Mean*(p_i) $+$ α_2 *Mean_St_Dev*(P)). However, several experiments with even many proximity graphs have suggested that $\alpha_1 = \alpha_2 = 1$ is a very robust starting point (30).

Second, we believe AUTOCLUST+ is more adequate for data mining under schemes that find associations across layers, as in Spatio-Dominant Generalization. Partitioning algorithms (12; 35; 47) provide limited opportunity for advancing the clustering to association analysis. They produce representatives of clusters that can be used to explore associations with other point data layers (17). However, for other themes coded as areas, they are not as effective, since these algorithms produce convex-shaped clusters. Also, these algorithms are very sensitive to large discrepancies in the size or the density. In these scenarios, their results are not as good and big clusters pass undetected since they are split into many fragments. Thus, these clustering methods fail to detect clusters of arbitrary shapes and clusters of different sizes. Similarly, density-based clustering algorithms (9; 39) and grid-based clustering algorithms (43; 50; 51) do not serve association analysis well because they fail to detect clusters of different densities, sparse clusters near high-density clusters and closely located high-density clusters. Graph-based clustering methods (8; 23; 25; 26; 53) are not as robust as AUTOCLUST+ at handling bridges. AUTOCLUST+ is robust to noise. It can detect local density variations. AUTOCLUST+ not only detects clusters of arbitrary shapes, but also clusters of different densities. Further, it successfully finds sparse clusters near to high-density clusters, clusters linked by multiple bridges and closely located high-density clusters.

Third, AUTOCLUST+ has several direct improvements over COE (47; 48); the only method for clustering large geo-referenced point data with obstacles. COE is a modification of a partitioning algorithm named CLARANS (35). This embodies randomized heuristics to solve the p-median problem, under the assumption that the number p of clusters is known (the facility location literature uses p for the number of representatives, but here $p = k$). Searching for p (or k) increases the complexity of the algorithm by a factor of k and care must be taken not to favor larger p since $p = n$ is an absolute minimum. The first advantage is that AUTOCLUST+ uses the Euclidean distance and not the square of the Euclidean distance. Using the square of the Euclidean distance is typically motivated by considerations like speeding the implementation by a constant factor and/or by an inductive principle of least-squares (as is the case with k-means (10)). However, using squares of the Euclidean distance in the modeling makes the approach extremely sensitive to outliers and/or noise (12; 34; 42). A second

advantage of AUTOCLUST+ over COE is that COE needs to experiment with different values for the number of clusters. This is a significant benefit, both in CPU-time and in user-analysis time. As a third advantage, AUTOCLUST+ operates with all the data and does not require the preprocessing of COE to form mini-clusters. This construction of mini-clusters introduces numerical error and biases to the grouping. Thus, the quality of the clustering is reduced and the exploratory capability is limited in COE. Fourth, COE uses a randomized interchange heuristic to approximate the optimum set of k representatives for the clusters. This heuristics is a very poor hill-climber that can not guarantee even local optimality and that sacrifices significantly the quality of the clustering for speed in the optimization. Much better variants of interchange heuristics are well known from the literature of facility location, and they have been applied to medoid-based clustering (17). Fifth, because of the representative-based nature of COE, the algorithm has a bias for convex clusters, and in particular for spherical clusters. This is probably acceptable when statistical justifications allow the assumption that the data has a probabilistic distribution from a mixture model. However, in the presence of obstacles, this assumption will certainly be very difficult to sustain. AUTOCLUST+ is not restricted to convex clusters. Moreover, even without obstacles on the dataset from Figure 4, AUTOCLUST+ detects several edges of the Delaunay Diagram that are too long and although it produces only one cluster, its shape is not convex and the water bays around the dataset are identified.

One thing in which AUTOCLUST+ (especially Approach 1, Approach 2a and Approach 2b) and COE are similar is that they are a "tightly-coupled" approach to including obstacles into the clustering. That is, rather than computing the clustering ignoring the obstacles and then cutting the resulting groups by the obstacles, the obstacles are used to modify parameters (the distance or neighboring information) when considering the cluster. In the case of COE, there is no real explanation to avoiding the "loosely-coupled" approach (Approach 2c), specially since Approach 2c is more efficient and produces similar results to the "tightly-coupled" approach with COE. On the other hand, in AUTOCLUST+, the computational efficiency does not suffer as much across the approaches that involve the obstacles. However, in AUTOCLUST+, we set the default to the "tightly-coupled" Approach 2b because the evaluation of global effects is considered with all Delaunay edges, even those that intersect obstacles, since this is what allows the discovery of density estimates over the region of study. But when analyzing the neighbors of a data point, obstacles are considered. An edge intersecting an obstacle is not an edge that justifies that its end-points are neighbors.

References

[1] Aho, A., Hopcroft, J., and Ullman, J. (1974). *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Co., Reading, MA.

[2] Ahuja, N. (1982). Dot Pattern Processing Using Voronoi Neighborhoods. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, PAMI-4(3):336–343.

[3] Ahuja, N. and Tuckeryan, M. (1989). Extraction of Early Perceptual Structure in Dot Pattenrs: Integrating Region, Boundary, and Component Gestalt. *Computer Vision, Graphics, and Image Processing*, 48:304–356.

[4] Bezdek, J. (1981). *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York.

[5] Celeux, G. and Govaret, G. (1995). Gaussian parsimonious clustering models. *Pattern Recognition*, 28(5):781–793.

[6] Cherkassky, V. and Muller, F. (1998). *Learning from Data — Concept, Theory and Methods*. John Wiley & Sons, NY, USA.

[7] Dempster, A.P., Laird, N.M. and Rubin D.B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.

[8] Eldershaw, C. and Hegland, M. (1997). Cluster analysis using triangulation. In Noye, B., Teibner, M., and Gill, A., editors, *Proceedings of Computational Techniques with Applications CTAC97*, pages 201–208. World Scientific Singapore.

[9] Ester, M., Kriegel, H., Sander, S., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., and Fayyad, U., editors, *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, Menlo Park, CA. AAAI, AAAI Press.

[10] Estivill-Castro, V. (2000). Hybrid genetic algorithms are better for spatial clustering. In Mizoguchi, R. and Slaney, J., editors, *Proceedings Sixth Pacific Rim International Conference on Artificial Intelligence PRICAI 2000*, pages 424–434, Melbourne, Australia. Springer-Verlag Lecture Notes in Artificial Intelligence 1886.

[11] Estivill-Castro, V. (2002). Why so many clustering algorithms – a position paper. *SIGKDD Explorations*, 4(1):65–75.

[12] Estivill-Castro, V. and Houle, M. (1999). Robust clustering of large geo-referenced data sets. In Zhong, N. and Zhow, L., editors, *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, pages 327–337. Springer-Verlag Lecture Notes in Artificial Intelligence 1574.

[13] Estivill-Castro, V. and Lee, I. (2000). AUTOCLUST+: Automatic clustering of point-data sets in the presence of obstacles. In Roddick, J. and Hornsby, K., editors, *Proceedings of the International*

Workshop on Temporal, Spatial and Spatio-Temporal Data Mining - TSDM2000, in conjunction with the 4th European Conference on Principles and Practices of Knowledge Discovery and Databases, pages 131–144, Lyon, France. Springer-Verlag Lecture Notes in Artificial Intelligence 2007.

[14] Estivill-Castro, V. and Lee, I. (2001). Data mining techniques for autonomous exploration of large volumes of geo-referenced crime data. In Pullar, D., editor, *6th International Conference on Geocomputation*, Brisbane. University of Queensland. CD-ROM, ISBN 1864995637.

[15] Estivill-Castro, V. and Lee, I. (2002a). Argument free clustering for large spatial point-data sets. *Computers, Environment and Urban Systems*, 26(4):315–334.

[16] Estivill-Castro, V. and Lee, I. (2002b). Multilevel clustering and its visualization for exploratory data analysis. *GeoInformatica*, 6(2):123–152.

[17] Estivill-Castro, V. and Murray, A. (1998). Discovering associations in spatial data - an efficient medoid based approach. In Wu, X., Kotagiri, R., and Korb, K., editors, *Proceedings of the 2nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-98)*, pages 110–121, Melbourne, Australia. Springer-Verlag Lecture Notes in Artificial Intelligence 1394.

[18] Fraley, C. and Raftery, A. (1998). How many clusters? which clustering method? answers via model-based cluster analysis. *Computer Journal*, 41(8):578–588.

[19] Gold, C. M. (1991). Problems with handling spatial data - The Voronoi approach. *CISM Journal ACSGC*, 45(1):65–80.

[20] Gold, C. M. (1992). The meaning of Neighbour. In Tinhofer, G. and Schmidt, G., editors, *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 220–235, Berlin. Springer-Verlag Lecture Notes in Computer Science 639.

[21] Goodchild, M. (1992). Geographical information science. *International Journal of Geographical Information Systems*, 6:31–45.

[22] Grabmeier, J. and Rudolph, A. (2002). Techniques of Cluster Algorithms in Data Mining, *Data Mining and Knowledge Discovery*, 6(4): 303–360.

[23] Guha, S., Rastogi, R., and Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, volume 27, pages 73–84, New York. ACM, ACM Press.

[24] Hastie, T., Tibshirani, R. and Friedman J.H. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* Springer-Verlag, NY.

[25] Kang, I.-S., Kim, T.-W., and Li, K.-J. (1997). A spatial data mining method by Delaunay triangulation. In *Proceedings of the Fifth ACM Workshop on Geographic Information Systems*, Las Vegas, Nevada.

[26] Karypis, G., Han, E.-H., and Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75. Special Issue on Data Analysis and Mining.

[27] Keil, M. and Gutwin, G. (1989). The Delaunay triangulation closely approximates the complete graph. In Denhe, F., Sack, J.-R., and Snatoro, N., editors, *Proceedings of the First Workshop on Algorithms and Data Structures WADS-89*, pages 47–56. Springer-Verlag Lecture Notes in Computer Science. 382.

[28] Knorr, E., Ng, R., and Shilvock, D. (1997). Finding boundary shape matching relations in spatial data. In School, A. and Voisard, A., editors, *Advances in Spatial Databases, 5th International Symposium, SDD-97*, pages 29–46, Berlin, Germany. Springer-Verlag Lecture Notes in Computer Science 1262.

[Koperski et al.] Koperski, K., Han, J., and Adhikari, J. Mining knowledge in geographical data. *Communications of the ACM.* to appear.

[29] Kuhn, H. (1973). A note on Fermat's problem. *Mathematical Programming*, 4(1):98–107.

[30] Lee, I. and Estivill-Castro, V. (2001). Effective and Efficient Boundary-based Clustering for Three-Dimensional Geoinformation Studies. *Proceedings of the The Third International Symposium on Cooperative Database Systems for Advanced Applications, CODAS* April 23-24, Beijing, China. 87-96. Lu, H. and Spaccapietra, S., editors.

[31] Liotta, G. (1996). Low Degree Algorithm for Computing and Checking Gabriel Graphs. Technical Report 96-28, Department of Computer Science, Brown University.

[32] Mehlhorn, K. and Näher, S. (1999). *LEDA A platform for combinatorial and geometric computing.* Cambridge University Press, UK.

[33] Miller, H. and Han, J., editors (2001). *Geographic Data Mining and Knowledge Discovery.* Research Monographs in Geographic Information Systems. Taylor and Francis.

[34] Murray, A. and Estivill-Castro, V. (1998). Cluster discovery techniques for exploratory spatial data analysis. *International Journal of Geographic Information Systems*, 12(5):431–443.

[35] Ng, R. and Han, J. (1994). Efficient and effective clustering methods for spatial data mining. In Bocca, J., Jarke, M., and Zaniolo, C., editors, *Proceedings of the 20th Conference on Very Large Data Bases (VLDB)*, pages 144–155, San Francisco, CA; Santiago, Chile, Morgan Kaufmann Publishers.

[36] Openshaw, S. (1994). Two exploratory space-time-attribute pattern analysers relevant to GIS. In Fotheringham, S. and Rogerson, P., editors, *Spatial Analysis and GIS*, pages 83–104, London, UK. Taylor and Francis.

[37] Openshaw, S. (1999). Geographical data mining: key design issues. In *Proceedings of 4-th GeoComputation 99*.

[38] Openshaw, S. and Alvanides, S. (1999). Applying geocomputation to the analysis of spatial distributions. In Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W., editors, *Geographical Information Systems: Principles and Technical Issues*, volume 1, pages 267–282. John Wiley & Sons, New York, second edition.

[39] Openshaw, S., Charlton, M., Wymer, C., and Craft, A. (1987). A Mark 1 geographical analysis machine for the automated analysis of point data sets. *International Journal of Geographical Information Systems*, 1(4):335–358.

[40] Paper, J. F. and Maguire, D. J. (1992). Design models and functionality in GIS. *Computers and Geosciences*, 18(4):387–394.

[41] Posse, C. (1999). Hierarchical Model-Based Clustering for Large Datasets. Technical Report 363, Department of Statistics, University of Washington.

[42] Rousseeuw, P. and Leroy, A. (1987). *Robust regression and outlier detection*. John Wiley & Sons, NY, USA.

[43] Schikuta, E. and Erhart, M. (1997). The BANG-clustering system: Grid-based data analysis. In *Proceedings of the Second International Symposium IDA-97*. Springer-Verlag Lecture Notes in Computer Science 1280.

[44] Shekhar, S. and Huang, Y. (2001). Discovering spatial co-location patterns: A summary of results. In Jensen, C., Schneider, M., Seeger, B., and Tsotras, V., editors, *Proceedings of the 7th International Symposium SSTD-2001*, pages 236–256, Redondo beach, CA. Springer-Verlag Lecture Notes in Computer Science 2121.

[45] Son, E.-J., Kang, I.-S., Kim, T.-W., and Li, K.-J. (1998). A spatial data mining method by clustering analysis. In *Proceedings of the sixth International symposium on Advances in Geographical Information Systems*, pages 157–158. ACM-GIS.

[46] Tsoukatos, I. and Gunopulos, D. (2001). Efficient mining of spatiotemporal patterns. In Jensen, C., Schneider, M., Seeger, B., and Tsotras, V., editors, *Proceedings of the 7th International Symposium SSTD-2001*, pages 425–442, Redondo beach, CA. Springer-Verlag Lecture Notes in Computer Science 2121.

[47] Tung, A., Hou, J., and Han, J. (2000). COE: Clustering with obstacles entities: A prelimianry study. In Terano, T., Liu, H., and Chen, A., editors, *Proceedings of the 4th Pacific-Asia conference on Knowledge Discovery and Data Mining*, pages 165–168, Kyoto, Japan. Springer-Verlag Lecture Notes in Computer Science 1805.

[48] Tung, A. K. H., Hou, J., and Han, J. (2001). Spatial Clustering in the Presence of Obstacles. In *Proceedings of the International Conference on Data Engineering*, pages 359–367.

[49] Veloso, M., Uther, W., Fujita, M., Asada, M., and Kitano, H. (1998). Playing soccer with legged robots. In *In Proceedings of IROS-98, Intelligent Robots and Systems Conference*, Victoria, Canada.

[50] Wang, W., Yang, J., and Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining. In Jarke, M., editor, *Proceedings of the 23rd International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece. VLDB, Morgan Kaufmann Publishers.

[51] Wang, W., Yang, J., and Muntz, R. (1999). STING+: An approach to active spatial data mining. In Kitsuregawa, M., Maciaszek, L., Papazoglou, K., and Pu, C., editors, *Proceedings of the Fifteenth International Conference on Data Engineering*, pages 116–125, Los Alamitos, CA. IEEE Computer Society. Sydney, Australia.

[52] Witten, I. and Frank, E. (2000). *Data Mining — Practical Machine Learning Tools and Technologies with JAVA implementations*. Morgan Kaufmann Publishers, San Mateo, CA.

[53] Zahn, C. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20(1):68–86.

[54] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *SIGMOD Record*, 25(2):103–114. Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data.