Expert Systems with Applications 39 (2012) 11135–11148

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Map segmentation for geospatial data mining through generalized higher-order Voronoi diagrams with sequential scan algorithms

Ickjai Lee *, Christopher Torpelund-Bruin, Kyungmi Lee

School of Business (IT), James Cook University, Cairns, QLD 4870, Australia

ARTICLE INFO

Keywords:
Map segmentation
Higher order Voronoi diagrams
Geospatial data mining
Sequential-scan algorithms

ABSTRACT

Segmentation is one popular method for geospatial data mining. We propose efficient and effective sequential-scan algorithms for higher-order Voronoi diagram districting. We extend the distance transform algorithm to include complex primitives (point, line, and area), Minkowski metrics, different weights and obstacles for higher-order Voronoi diagrams. The algorithm implementation is explained along with efficiencies and error. Finally, a case study based on trade area modeling is described to demonstrate the advantages of our proposed algorithms.

Crown Copyright © 2012 Published by Elsevier Ltd. All rights reserved.

1. Introduction

Segmentation is one approach to build topology (Chen, Wang, & Feng, 2010; Hanafizadeh & Mirzazadeh, 2011; Lee, Qu, & Lee, 2012; Seng & Lai, 2010) and it is one popular method for geospatial data mining. Generalized Voronoi Diagrams (GVDs) are generalizations of the ordinary Voronoi diagram to various metrics, different weights, in the presence of obstacles, complex data types (point, line and area), and higher order. Recently, Lee and Torpelund-Bruin (2012) proposed an efficient and effective GVD algorithms for use in geospatial data mining. However, the model is limited to the first order and is not able to capture higher order scenarios. In many real business settings, people are more interested in the second or third nearest object. For instance, patients are interested in the second nearest hospital when the first nearest hospital is fully occupied or closed. This article extends (Lee & Torpelund-Bruin, 2012) to implement truly flexible higher-order Voronoi diagrams (HOVD) for map segmentation and geospatial data mining.

HOVD have been studied by many researchers and found useful for a variety of applications when k number of points are considered for partitioning (Boots & South, 1997; Lee & Gahegan, 2002; Lee & Lee, 2007; Lin & Kung, 2001; Xie, Wang, & Cao, 2007). Recently, interest has been growing into exploiting HOVD for the partitioning of geospatial information for GIS applications (Pinliang, 2008). However, current HOVD modeling techniques are limited because of the complexity of the traditional vector-based algorithms used (Lee & Lee, 2009). This encompasses generators being typically limited to points in the absence of obstacles, the underlying metric limited to the Euclidean metric and weights of generators assumed to be invariant. For accurate geospatial analysis, a robust and versatile method is required to model the diverse and various components associated with real world geospatial analysis. This has lead to the exploration of efficient raster-based methods for GVD.

Due to advances in Web 2.0 technologies and the prevalence of Web Map Service (WMS) such as Google Map (http://maps.google.com), Google Earth (http://earth.google.com), NASA World Wind (http://worldwind.arc.nasa.gov), and Open Street Map (http://www.openstreetmap.org), districting Web maps through raster-based GVD is of emergence. In this article, we propose efficient and effective sequential-scan algorithms for HOVD districting. We extend the distance transform algorithm (Shih & Wu, 2004) to include complex primitives (point, line, and area), Minkowski metrics, different weights and obstacles for HOVD. The algorithm implementation is explained along with efficiencies and error. These new algorithms can be used to enhance accuracy for order-k and ordered order-k queries in various geospatial applications. To demonstrate the advantages of our proposed algorithms within an application, a case study based on trade area modeling is described.

Section 2 begins with a brief explanation of the properties of Voronoi diagrams. Section 3 describes current raster-based techniques for distance transforms and their application to Voronoi diagrams. Section 4 describes the reasoning of the sequential-scan algorithm for higher-order analysis. Sections 4.2.2 and 4.3.2 describe the implementation of the generalized higher-order algorithms for no obstacles and obstacles in the plane respectively. Sections 6 and 7 reflect the efficiency and error of the described algorithms. Section 8 describes a case study demonstrating how the algorithms can be applied for various applications and studies. Finally, Section 9 reflects on the study and future work.

* Corresponding author.
E-mail addresses: Ickjai.Lee@jcu.edu.au (I. Lee), christopher.torpelund@my.jcu.edu.au (C. Torpelund-Bruin), Joanne.Lee@jcu.edu.au (K. Lee).

0957-4174/$ - see front matter Crown Copyright © 2012 Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.03.042

2. Generalized Voronoi Diagrams

Let $G = \{g_1, g_2, \ldots, g_n\}$ be a set of *generator* points of interest in the plane P in R^m space. Every location p in P can be assigned to the closest generator $g \in G$ with a certain distance metric defined by $dist(p,g)$. The assignment of generators G over the plane gives the set of Voronoi regions $\mathcal{V} = \{V(g_1), V(g_2), \ldots, V(g_n)\}$. If a generator is equally close to two points in G then the location becomes part of a Voronoi boundary, whilst if it is equally close to more than two points then the location becomes a Voronoi vertex. The Voronoi boundaries between g_i and g_j can be defined as $e(g_i,g_j) = V(g_i) \cap V(g_j)$. The complete boundary of g_i over g_j gives the dominance region $Dom(g_i,g_j)$. The dominance region defined by the subsequent Voronoi regions is generalized by the type of generator, such as points, lines or polygons, various weights, plane constraints, and the metric space used. The result of the combination of generalizations generates a districted plane polygons and arcs made up of lines and Bezier curves. The way GVD can be represented is virtually limitless. For a further detailed explanation of the properties of Voronoi diagrams readers should consider (Okabe, Boots, Sugihara, & Chiu, 2000). This article focuses on higher-order generalizations of Voronoi diagrams produced with the Minkowski metric so these properties are briefly examined in the next sections.

2.1. Minkowski metric

The Minkowski metric, also called the Minkowski tensor or pseudo-Riemannian metric, defines the setting for space–time. The three ordinary dimensions of space are combined with a single dimension of time to form a four-dimensional manifold. The Minkowski metric is defined as:

$$d_{L_p}(g,g_i) = \left[\sum_{j=1}^{m} |x_j - x_{ij}|^p\right]^{1/p} \quad (1 \leqslant p \leqslant \infty) \tag{1}$$

where $(x_1,x_2,\ldots,x_m)$ and $(x_{i1},x_{i2},\ldots,x_{im})$ are the Cartesian coordinates of g and g_i, respectively. The parameter p can be in the range of $1 \leqslant p \leqslant \infty$. When $p = 1$, then $d_{L_1}(g,g_i) = \sum_{j=1}^{m} |x_j - x_{ij}|$ is the Manhattan metric. The Minkowski metric becomes the Euclidean metric when $p = 2$. If $p = \infty$, then the Minkowski metric becomes $d_{L_\infty}(g,g_i) = max_j|x_j - x_{ij}|$, which is called the chessboard metric. The most popular distance metric d is the Euclidean metric. Being rotationally invariant makes it most desirable for depicting two- and three-dimensional characteristics of the Earth's surface, subsurface, and atmosphere. However, in urban geography the Manhattan distance metric better approximates real world situations because of its constraints on diagonal movement (Krause, 1975). Due to this attribute, the Manhattan distance can be derived generally in a very efficient linear time $O(n)$ (Kolountzakis & Kutulakos, 1992). However, a drawback is that diagonal distances are over-estimated because a diagonal connection counts as 2 steps rather than $\sqrt{2}$ for the Euclidean distance.

2.2. Higher-order Voronoi diagrams

HOVD are natural and useful generalizations of Voronoi diagrams for more than one generator (Okabe et al., 2000). They provide tessellations where each region has the same unordered k closest sites for a given k. Let us consider the set of generators G in a plane. Considering the higher-order Voronoi diagram of the k closest unordered (order-k) points can be defined as $\mathcal{V}^k = \left\{V\left(G_1^k\right), \ldots, V\left(G_l^k\right)\right\}$, where the order-$k$ Voronoi region $V\left(G_i^k\right)$ for a random subset G_i^k consists of k generators from the set G and G_l^k represents all of the possible subsets. The set of points in the plane assigned to the order-k Voronoi region $V\left(G_i^k\right)$ can be defined as:

$$V\left(G_i^k\right) = \left\{p | \max_{g_h}\left\{d(p,g_h) | g_h \in G_i^k\right\} \leqslant \min_{g_j}\left\{d(p,g_j) | g_j \in G/G_i^k\right\}\right\} \tag{2}$$

Because HOVD are a set of k Generalized Voronoi Diagrams, the same generalizations apply. Order-k Voronoi diagrams have been combined with weights to generate *Order-k Multiplicatively Weighted Voronoi Diagram* (OKMWVD). If the order of k generators is considered then this generates the *Ordered, Order-k, Multiplicatively Weighted Voronoi diagram* (OOKMWVD) (Boots & South, 1997).

3. Raster based algorithms and distance transforms

The general idea of the raster based method is to expand the Voronoi diagram incrementally by adding a point at a time for all points in the image by considering the *number of neighbors* or *direction of connection* (Fortune, 1987; Lee & Drysdale, 1981; Li, Chen, & Li, 1999). The Voronoi diagram is represented by a discrete grid lattice of size $N \times N$ which gives N^2 points in the plane. Unlike vector districting methods, which generally have efficient time complexities at the expense of a lack of diversity, raster based methods are robust and versatile and are able to support accurate modeling of the diverse and various components associated with real world geospatial analysis. Despite the potential advantages, there is little literature based purely on this method (Li et al., 1999). The methods of generating raster based Voronoi diagrams are based heavily on the body of work related to *distance transforms* (DTs). DTs are characterized by the mappings of each image pixel (foreground) into its smallest distance to regions of interest (background) (Rosenfeld & Pfaltz, 1966). Generally, these algorithms can be divided into *iterative algorithms* and *sequential algorithms* based on the order used to scan the pixels (Cuisenaire & Macq, 1999).

Iterative algorithms use *wave* scans over points in the background image B which are seen as sources from which distance values are calculated for the foreground image F. The waves propagate out from the sources like how a grass fire would burn from the source of ignition outwards. Time complexity for such parallel processing is typically $O(N)$. However, the computing architecture required for this parallel processing requires a separate processor for each generator in the plane. This results in a *time-processor complexity* of $O(N^3)$ (Danielsson & Tanimoto, 1983). The same propagation wave scan is possible with a single processor with a time-complexity of $O(N^2)$ (Rosenfeld & Pfaltz, 1966). However, most iterative algorithms based on this single processor architecture are inefficient as many pixels are often unnecessarily updated. Alternatively, sequential algorithms have better time-processor complexities for single processors.

Sequential-scan algorithms are characterized by the *number of neighbors* and *plane-sweeps* that are considered. The number of neighbors range from four-neighborhoods for the Manhattan metric, to eight or more arbitrarily large neighborhoods. The shape of neighbors being considered also differs between algorithms. There is a trade-off between the size and shape of the neighborhood with the number of iterations needed to achieve a solution and the quality of the approximation (Breu, Gil, Kirkpatrick, & Werman, 1995). Chamfer DTs include individual weights in various sized masks to minimize the deviation from the Euclidean DT (EDT) (Borgefors, 1984). The downside with this method is that the usage of weighted masks can only allow approximate distance values. Even large Chamfer masks can only approximate to within 2% error (Cuisenaire, 1999). Recent methods to reduce the error include

the use of *vector propagation* which propagates the absolute value of the relative coordinates to the nearest background image pixel. Often, in order for sequential algorithms to calculate the exact EDT some form of post-processing is performed to correct eventual errors caused by the problems of DTs on the grid lattice (Fabbri, Costa, Torelli, & Bruno, 2008). To date, sequential-scan algorithms seem to be the fastest. However, like with the iterative scan algorithms, many pixels might be processed more than once.

One of the fastest methods sequential-scan methods was pioneered by Cuisenaire and Macq (1999), who proposed propagation through multiple neighborhoods and bucket sorting. However, complex multiple neighborhoods and time consuming sorting are considerable overheads of their approach. This was further improved by Shih and Wu (2004) who employed a 3×3 neighborhood method with vector propagation. They proved the algorithm to be extremely efficient with $O(N)$ time-complexity. However, their approach is limited to simple primitives (point type) and unweighted primitives. We further develop their approach and propose a three scan algorithm to deal with multiple metrics, weighted primitives, complex primitives and obstacles for HOVD.

4. The sequential-scan higher-order Voronoi diagram algorithms

In order to develop the sequential-scan HOVD method we must first look into the details of generating the 1st-order, or more commonly known as the ordinary, Voronoi diagram. The following section examines the 1st-order method, with the proceeding section explaining how this method can be used to generate the HOVD.

4.1. The weighted ordinary Voronoi diagram

4.1.1. Background

The non-weighted ordinary Voronoi diagram can be generated with the original 3×3 neighbor scan algorithm with three sweep scans. However, for correct weighted distance transforms, distance values cannot be simply propagated because the addition of weight factors creates non-convex sets of dominance regions. Instead, distance values must be calculated by the propagated relative distance values which takes into account the direction of connection for each propagation. We consider the background binary image of features B and the foreground image F over the background image. Let $q1, q2, \ldots, q8$ be the eight neighbors of p. The mask $q1$, $q2$, $q3$, $q4$ is defined by $N_1(p)$ and $q5$, $q6$, $q7$, $q8$ is defined by $N_2(p)$, as described in Fig. 1. Let $R(q)$ be the relative vector coordinates of the neighboring pixel q. The relative distances keep track of the direction of connection by being negative for negative axis movements and positive for positive axis movements. Let $h(p,q)$ be the difference of the squared Euclidean distance between p and q. It is important to note that we take the absolute value of the difference otherwise we will get negative distances from p to q. Let $G(p,q)$ be the difference of the vector distance used to calculate the distance values of the pixels preceding p given by the masks. The $h(p,q)$ and $G(p,q)$ can be computed in the following way:

$$h(p,q) = \begin{cases} (2|R_x(q)|+1) & \text{if } q \in q1, q5 \\ (2|R_y(q)|+1) & \text{if } q \in q3, q7 \\ (2(|R_x(q)|+|R_y(q)|+1)) & \text{if } q \in q2, q4, q6, q8 \end{cases} \quad (3)$$

$$G(p,q) = \begin{cases} (1,0) & \text{if } q \in q1 \\ (1,-1) & \text{if } q \in q2 \\ (0,-1) & \text{if } q \in q3 \\ (-1,-1) & \text{if } q \in q4 \\ (-1,0) & \text{if } q \in q5 \\ (-1,1) & \text{if } q \in q6 \\ (0,1) & \text{if } q \in q7 \\ (1,1) & \text{if } q \in q8 \end{cases} \quad (4)$$

Let us define $dis(R_x(q),R_y(q))$ as the process of finding the weighted distance in Manhattan, chessboard and Euclidean metrics from the neighboring pixel q to the nearest generator g_i. The algorithm for generating weighted Voronoi diagrams is described below:

4.1.2. Algorithms

Update Foreground F with Relative Distances $R_x(q)$, $R_y(q)$ Function

1. $f_{new}(p) = (dis(R_x(q),R_y(q)) + h(p,q))/w_q$
2. $f(p) = min(f(p),f_{new}(p))$
3. Modify: $R(p) = R(q) + G(p,q)$

Weighted Voronoi diagram algorithm

1. Process generators to Cartesian coordinates (Rasterize)
2. (Forward Scan) For each pixel $p \in F$
   (a) set all $p \in F = \infty$
   (b) for each neighborhood $q \in N_1(p)$
   - *Update Foreground F with Relative Distances $R_x(q),R_y(q)$ Function*
3. (Reverse Scan) For each pixel $p \in F$
   (a) for each neighborhood $q \in N_2(p)$
   - *Update Foreground F with Relative Distances $R_x(q)$, $R_y(q)$ Function*
4. Repeated Step 2 Forward Scan

The new distance $f_{new}(p)$ is calculated by adding the weight factor to the total distance calculated by adding the distance $dis(R_x(q),R_y(q))$ to the nearest background generator primitive g_i from q and the distance $h(p,q)$. The ordinary Voronoi diagram in the presence of obstacles and with non-weighted complex primitives can be generated by the same algorithm in $O(F)$ time with a few modifications to allow the obstacle and generator primitives to be rasterized into the plane. This rasterization can be computed with a plane-sweep algorithm over the line and polygon primitives testing union intersections to convert them into Cartesian coordinates. However, a different approach is need when both weighted primitives and obstacles are in the plane.

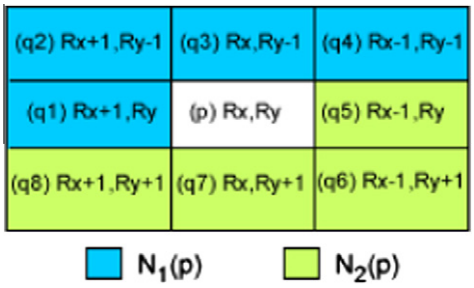


Fig. 1. 3×3 Relative distance propagation neighborhood.

4.2. The weighted higher-order Voronoi diagram

4.2.1. Background

HOVD offer intricately detailed geospatial analysis over a plane. However, the formulation is not overly complex if higher-order logic is considered with predicates. A higher-order predicate is a kth order predicate that takes one or more other $(k-1)$th-order predicates as arguments, where $k > 1$. This formulation can be implemented with the set of Voronoi diagrams $F^k = \{F^1, \ldots, F^i, \ldots, F^k\}$ as arguments to generate. This means in order to achieve an order-k

Voronoi diagram, F^k Voronoi diagram images in total must be generated. Each image F^i requires the distance transforms of F^{i-1}, where i represents an intermediate level such that $1 < i < k$. An example of this is to generate 3rd-order Voronoi diagram. The 3rd-order Voronoi diagram requires the 2nd-order Voronoi diagram, which in turn requires the ordinary (1st-order) Voronoi diagram. This example is shown in Fig. 2.

Initially we attempted to generate F^k levels of the order-k Voronoi diagram by propagating distance values to each $i+1$ layer. However this method proved to be flawed and generated inconsistent diagrams. It was suggested that a method of providing a look-up table of distance values for each generator primitive g_i that would be available to any point p in each image F^i. However, this approach was not favorable because it would requires $O(F * G)$ computation time, where F represents the plane with all pixels and G represents the set of generators, just for generating the propagated distances for each generator primitive $g_i \in G$. What was needed was a more efficient method that provided linear time computation with regard to the size of F and the number of k only.

It was eventually decided that a method comprising of generator information propagation and vector distances should be explored, thus resulting in the generator index propagation approach. During any distance transform calculations that eventually form the Voronoi diagram, there exists a list of generator primitives that define their coordinates and weight within the image F. Instead of attempting the time-consuming task of propagating relative distance values describing the location of generator primitives over all instances of the planes F^i, it makes sense to propagate references of generator primitives whereby the correct distance from the pixel p in F^i can be calculated to g_i directly from the coordinate information of g_i within the list of generator primitives. The reference can simply be an index or a pointer to the location within the list of generator primitives. This concept is described in Fig. 3, where the current pixel p (blue)[1] is being determined with the generator indexes within the mask $N_1(p)$ (red). The generator index is used to access a hashset of generator references which is used to determine the location of the given generator (green). This location information is used to determine the nearest generator to p from the neighborhood. The index of the nearest generator is then given to p and the process repeated over the plane F^1. Subsequent planes greater than F^1 must determine if the generator index already exists at any of the lower levels before assignment to the current pixel p being processed. An example of three planes giving the order-3 Voronoi diagram is described in Fig. 4. Using the list of generator indexes for a given coordinate over all of the generated planes F^k gives the order-k for sorted lists, and ordered order-k Voronoi diagram for unsorted lists. The technique for implementing this method is described in the following section.

4.2.2. Generator index propagation approach

At the heart of the order-k ordinary Voronoi diagram is the order-1 Voronoi diagram, or quite simply the ordinary Voronoi diagram. Next, the order-2 Voronoi diagram is generated by extending the Voronoi regions generated by the order-$(i-1)$, in this case the 1st-order, Voronoi diagram. This process continues until all k degrees have been generated. The general method encompasses the idea of iteratively generating F^i from F^{i-1}. The nearest k generator primitives g_i to p are determined by finding the relative distances using the generator index propagation method. Let the pixels in the planes $f(p)^i$ now refer to the generator index of the nearest generator, instead of the propagate distance values. Let us reuse the previously denoted $h(p,q)$ to define the distance values from p to q for $k = 1$. On top of this, let $h_g(p,q)$ define the distance values for the step from q to p obtained from the relative distances to the referenced nearest generator g. Let g_x and g_y denote the x and y coordinate for the referenced generator determined from the generator index at the neighboring pixels $q1$ to $q8$ from the current pixel being processed p. The relative distances from neighboring pixel q to the nearest generator g_i is denoted as $g_x - q_x$ and $g_y - q_y$ for the relative x and y distances, respectively. The following algorithm defines the generator index propagation approach:

$$h_g(p,q) = \begin{cases} (2|(g_x - q_x)| + 1) & \text{if } q \in q_1, q_5 \\ (2|(g_y - q_y)| + 1) & \text{if } q \in q_3, q_7 \\ (2(|(g_x - q_x)| + |(g_y - q_y)| + 1)) & \text{if } q \in q_2, q_4, q_6, q_8 \end{cases} \tag{5}$$

4.2.3. Algorithms

Update F^1 Function

1. $f_{new}(p) = (dist(R_x(q), R_y(q)) + h(p,q))/g_{weight}$
2. $f(p)^1 = min(f(p)^1, f_{new}(p))$
3. Modify: $R(p) = R(q) + G(p,q)$

Calculate Distance and Update Reference $f(p)^i$ Function

1. $f_{new}(p) = (dist(g_x - q_x), (g_y - q_y)) + h_g(p,q))/g_{weight}$
2. $f(p)^i = min(f(p)^i, f_{new}(p))$

Update F^i Function

1. If generator index at $f(q)^{i-1} \notin \{f(p)^1, \ldots, f(p)^i\}$
   - let g = the referenced generator at $f(q)^{i-1}$
   - *Calculate Distance and Update Reference $f(p)^i$ Function*
2. Else if generator index at $f(q)^i \notin \{f(p)^1, \ldots, f(p)^i\}$
   - let g = the referenced generator at $f(q)^i$
   - *Calculate Distance and Update Reference $f(p)^i$ Function*

The $3 + 3(k-1)$ Scan with Generator Index Propagation Algorithm

1. Process generators to Cartesian coordinates
2. (Forward Scan) For each pixel $p \in F^1$
   (a) set all $p \in F^1 = \infty$
   (b) for each neighborhood $q \in N_1(p)$
   - *Update F^1 Function*
3. (Reverse Scan) For each pixel $p \in F^1$
   (a) for each neighborhood $q \in N_2(p)$
   - *Update F^1 Function*
4. Repeated Step 2 Forward Scan
5. For each degree i: $1 < i < k$
   (a) (Forward Scan) For each pixel $p \in F^i$
   - for each neighborhood $q \in N_1(p)$
     i. *Update F^i Function*

   (b) (Reverse Scan) For each pixel $p \in F^i$
   - for each neighborhood $q \in N_2(p)$
     i. *Update F^i Function*

   (c) Repeat Step 5(a) Forward Scan

The metric can be any of Euclidean, Manhattan and chessboard with the function $dist(R_x(q), R_y(q))$. The order-1, or ordinary Voronoi diagram, is denoted by F^1. Step 5 (Section 4.2.3) onwards details the incremental generation of the order-k Voronoi diagram from $i = 2$ until k degrees by refining the Voronoi regions and distance values at $i-1$. The k degree must be greater than 1 otherwise the Voronoi diagram will simply be the ordinary Voronoi diagram. Unique Voronoi region edges are found at F^{i-1} in Step 1 (Section 4.2.3), which tests if the generator referenced at $f(q)^{i-1}$ is not in the set of references from $f(p)^1$ to the current plane $f(p)^i$. From

[1] For interpretation of color in Fig. 3, the reader is referred to the web version of this article.

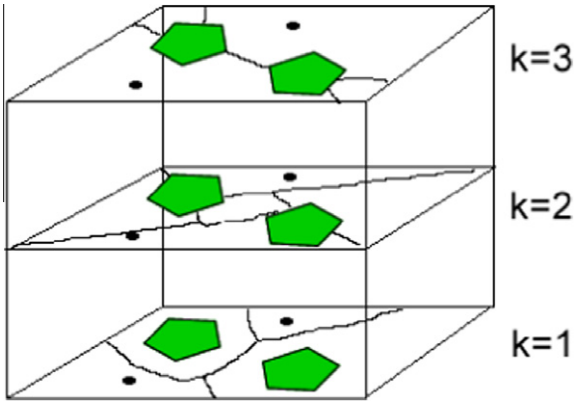


Fig. 2. The individual k layers of HOVD.

the edges, distance values continue and propagation over the new region at $f(p)^i$ until a new edge is formed against another propagation in Step 2 (Section 4.2.3). This process can be mentally visualized by considering the Voronoi regions defined in the plane F^{i-1} being the generators for the plane F^i. The regions expand and propagate from their original regions at F^{i-1} over the new raster plane in F^i until equi-distance is found with another expanding region which forms the new ith-order edges and Voronoi regions. The test at 1 (Section 4.2.3) can therefore be thought of as a *step-up* to the next unique region in F^i. The final results are the raster planes F^k where each pixel $f(p)$ within each of the planes gives the index to the nearest generator, as described in Fig. 4.

4.3. Weighted higher-order Voronoi diagram in the presence of obstacles

4.3.1. Propagated distance map approach

HOVD with obstacles can be generated with the combination of the generator index propagation approach with *propagated distance maps*. The propagated distance map works by *mapping* distance values to all coordinates in the raster image for each separate generator in the set G. The propagated distance map is different from a distance transform because each of the propagated distance maps are focused on one generator primitive g_i individually, as opposed to generating distances to all generator primitives G as with distance transforms. An example of the propagated distance maps generated for a plain consisting of complex primitives and obstacles is shown in Fig. 5.

The general concept is in fact quite similar to the A* algorithm, where each pixel p in the image F takes its turn being the target node that determines its distance to the initial node (generator) from its neighbors (Hart, Nilsson, & Raphael, 1968). The difference is that no heuristic distance is needed because the neighbors of p that have had their turn being the target node are guaranteed to already have the shortest distance to the initial node. Once the maps have been generated for each generator primitive g_i then the weighted Voronoi diagram can be easily generated. The downside to having a propagated distance map for each generator primitive

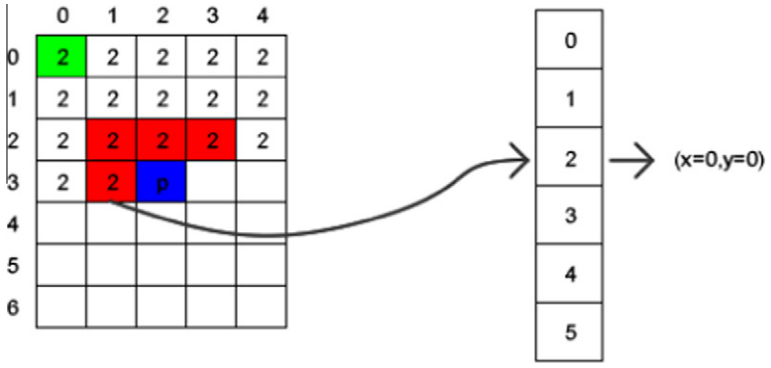


Fig. 3. Generator index propagation over the plane F (left) and hashset lookup for generator location (right).

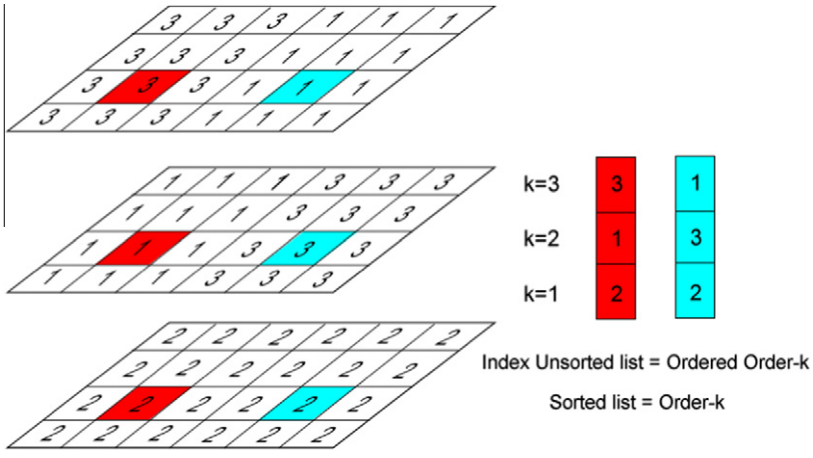


Fig. 4. The order-3 Voronoi diagram with three levels of planes with generator indexes.

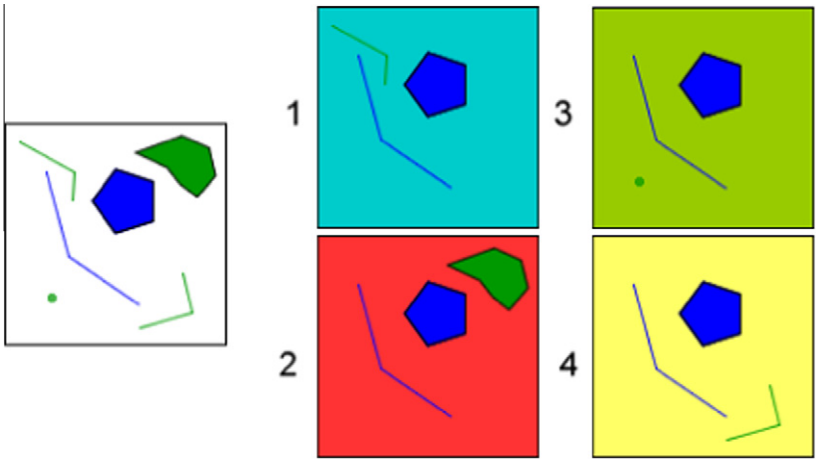


Fig. 5. Four propagated distance maps generated for each of the four generator primitives.

g_i is the linear time needed to generate each map for the set of generators G during pre-processing. If there exists many generator primitives in the plane then this pre-processing could be computationally expensive. However, if the algorithm used for generating the propagated distance maps was optimized and very efficient then its effect would be negligible on the overall processing time. Another additional drawback is if g_n number of propagated distance maps are required, then the same number of relative distance maps are also required to determine distance values for each distance map. Having g_n number of relative distance maps could require a large amount of memory if the set of generator primitives G and the plane F is large.

4.3.2. Propagated distance map approach for obstacles in the plane

The propagated distance map can take the place of the generator index propagation approach because the map already references all distance values within the plane F with obstacles associated with the generator primitive g_i. For any k degrees, only g_n number of propagated distance maps are needed. The relative distance values from q to its nearest referenced generator primitive g_i can be obtained from the propagated distance map regardless if the distance value is for F^i or F^{i-1}. Using the propagated distance map, let $h_m(p,q)$ now define the distance values from p to q with the relative distance obtained by the referenced generator g_i at q. The relative distance to the nearest generator can be found via the propagated generator index at q determining which propagated map to retrieve the relative x and y distances of. This relative distance extraction is demonstrated in Fig. 6. Either the referenced relative distances x, y or a combination of the two are retrieved depending on which neighbor q is currently being processed. Let $M_{(g_i)}$ denote the propagated distance map for the generator primitive g_i. Each distance map $M_{(g_i)}$ is generated with the propagated relative distance approach as described in Section 4.1. Once the propagated distance maps are generated, the higher order Voronoi planes $F^1,\ldots,F^k$ are generated with the generator index propagation approach as described in Section 4.2.2 with the exception that now the generator indexes at $f(p)^i$ are used to determine which distance map should be used to obtain the distance values from p to that particular generator g_i. Using the distance maps to obtain the distance values directly for p means we do not need to take into account the distance from q to p generated by the functions h and h_g. Let $M_{(g_i)}(p)$ represent the distance value to generator g_i from p. The following algorithm describes this process to achieve the HOVD with obstacles.

4.3.3. Algorithms

Update F^1 Function Using Distance Map

1. Select distance map $M_{(g_i)}$ based on generator index at q
2. $f_{new}(p) = M_{(g_i)}(p)$/ weight of g_i
3. $f(p)^1 = min(f(p)^1, f_{new}(p))$

Update F^i Function Using Distance Map

1. If generator index at $f(q)^{i-1} \notin \{f(p)^1,\ldots,f(p)^i\}$
   - Select distance map $M_{(g_i)}$ based on generator index at q^{i-1}
   - $f_{new}(p) = M_{(g_i)}(p)$/weight of g_i
   - $f(p)^i = min(f(p)^i, f_{new}(p))$
2. Else if generator index at $f(q)^i \notin \{f(p)^1,\ldots,f(p)^i\}$
   - Select distance map $M_{(g_i)}$ based on generator index at q^i
   - $f_{new}(p) = M_{(g_i)}(p^i)$/ weight of g_i
   - $f(p)^i = min(f(p)^i, f_{new}(p))$

3 + 3(k − 1) Scan with Propagated Distance Map Algorithm

1. Process generators to Cartesian coordinates
2. Process obstacles to Cartesian coordinates (Rasterize)
3. Generate $M_{(g_i)}$ for the set of generator primitives G function
4. (Forward Scan) For each pixel $p \in F^1$
   (a) set all $p \in F^1 = \infty$
   (b) for each neighborhood $q \in N_1(p)$
      - Update F^1 Function Using Distance Map
5. (Reverse Scan) For each pixel $p \in F^1$
   (a) for each neighborhood $q \in N_2(p)$
      - Update F^1 Function Using Distance Map
6. Repeated Step 4 Forward Scan
7. For each degree i until k degrees: $k > 1$
   (a) (Forward Scan) For each pixel $p \in F^i$
      - for each neighborhood $q \in N_1(p)$
         i. Update F^i Function Using Distance Map
   (b) (Reverse Scan) For each pixel $p \in F^k$
      - for each neighborhood $q \in N_2(p)$
         i. Update F^i Function Using Distance Map
   (c) Repeat Step 7(a) Forward Scan

After rasterization of the primitives, the next step is to generate each distance map $M_{(g_i)}$ for the set of generator primitives G. After the distance maps have been generated, the generation of the Voronoi planes F^1 until F^k can be constructed in essentially the same way as the generator index propagation method except now the generator indexes at $f(p)$ indicate the distance map to obtain the distance values. The figures shown

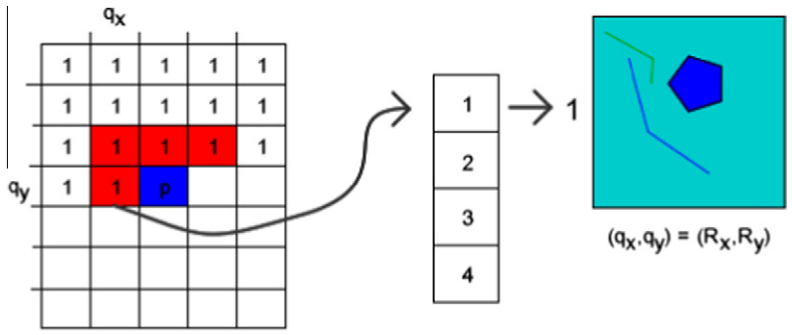


Fig. 6. How relative distances of the nearest generator can be extracted from propagated distance maps determined by propagated generator indexes.

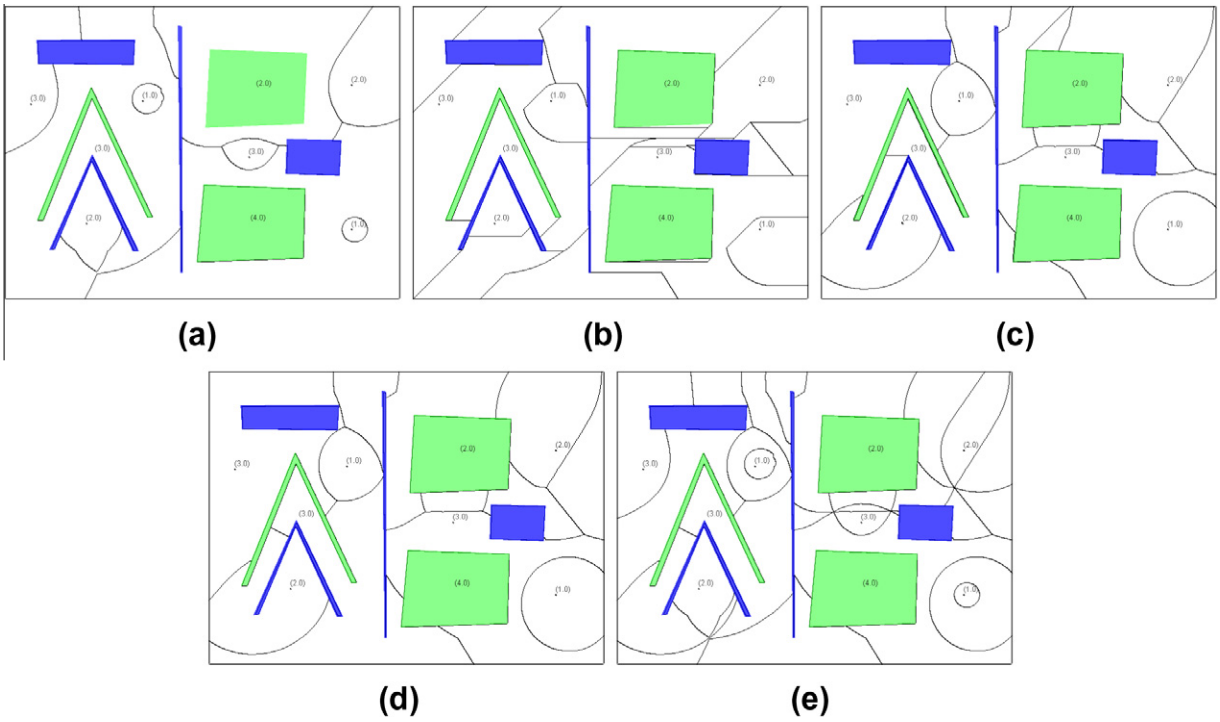


Fig. 7. The incremental scans for generating the 2nd-order Voronoi diagram in the Manhattan metric with weighted generator primitives and obstacles.

in Fig. 7 demonstrate the incremental scans in the Manhattan metric for generating the 2nd-order Voronoi diagram with weighted generator primitives and obstacles. Fig. 7(a) reveals the results of the first three scans in Steps 4, 5, and 6 (Section 4.3.3) to generate the 1st-order Voronoi diagram. To generate the 2nd-order Voronoi diagram, one iteration through the three scans in Step 7 is required. The first scan of the 2nd-order Voronoi diagram in Step 7a produces the results in Fig. 7(b). Step 5 produces Fig. 7(c). The repeat scan in Step 7c produces Fig. 7(d). Fig. 7(e) demonstrates the inclusive 2nd-order Voronoi diagram with the same dataset.

5. The proof of the equations for higher order Voronoi diagrams

For the proof of the equations for HOVD, we must first consider the proof of the weighted 3×3 neighborhood method. The weighted 3×3 neighborhood method is used for each level to achieve the kth order Voronoi diagrams. Once this proof has been established, then the $3 + 3(k-1)$ method can be proved by induction.

5.1. Weighted 3×3 neighborhood method with propagated relative distances

For the proof of the weighted 3×3 neighborhood method, let us consider the weighted distance from p to g_i being propagated from $q1$ and $q2$.

5.1.1. Case 1: Euclidean distance for q1

The direction of connection from $q1$ is diagonally right to p. This gives a relative Euclidean distance to p from $q5$ of $((R_x + 1), R_y)$. The difference between the Euclidean distances (D_e) from g_i to p and $q1$ with weight w_1 is:

$$\begin{aligned} D_e(x_1, x_2) &= (x_1 - x_2)/w_1 \\ D_e(q1, p) &= (((R_x+1)^2 + (R_y)^2) - ((R_x)^2 + (R_y)^2))/w_1 \\ &= (R_x^2 + 2(R_x+1) + 1^2 - (R_x)^2)/w_1 \\ &= (2(R_x+1) + 1^2)/w_1 \\ &= (2(R_x+1) + 1)/w_1 \\ &= (2R_x(q5) + 1)/w_1 \end{aligned} \tag{6}$$

Hence, by expanding the Pythagoras theorem with R_x and R_y and using the binomial theorem to solve the equation, we find the weighted Euclidean distance from $q1$ to p can be defined as $2(q1)+1/w_1$ along with the relative distances from $q1$ to p and the closest generator primitive $min(g_i)$ to p being:

$$G(q1,p) = (+1,0) \tag{7}$$
$$R(min(g_i),p) = (R_x+1,R_y) \tag{8}$$

5.1.2. Case 2: Euclidean distance for q2

The direction of connection from $q2$ is diagonally right and down to p. This gives a relative Euclidean distance to p from $q2$ of $((R_x+1),R_y-1)$. The difference between the weighted Euclidean distances (D_e) from g_i to p and $q2$ is:

$$\begin{aligned} D_e(x_1,x_2) &= (x_1^2+x_2^2)/w_1 \\ D_e(q2,p) &= (((R_x+1)^2+(R_y-1)^2)-(R_x^2+R_y^2))/w_1 \\ &= ((R_x^2+2(R_x+1)+1^2+R_y^2 \\ &\quad +2(R_y-1)-1^2)-(R_x^2+R_y^2))/w_1 \\ &= (2(R_x+1)+2(R_y-1)+2)/w_1 \\ &= (2((R_x+1)+(R_y-1)+1))/w_1 \\ &= (2(R_x(q2)+R_y(q2)+1))/w_1 \end{aligned} \tag{9}$$

Again, as with Case 1, we find the difference of $q2$ from p using the Pythagoras theorem with R_x and R_y and solve with the binomial theorem. Unlike Case 1, both $(R_x+1)^2$ and $(R_y-1)^2$ require expanding because of the diagonal direction of connection. The relative distances from $q2$ to p and the closest generator primitive $min(g_i)$ to p are defined as:

$$G(q2,p) = (1,-1) \tag{10}$$
$$R(min(g_i),p) = (R_x+1,R_y-1) \tag{11}$$

5.2. The 3 + 3(k − 1) scan proof by induction

For the proof by induction of the $3+3(k-1)$ scan method, we must show proof of the higher order predicate. Let us consider the test set of generators $G=\{g_1,g_2,g_3\}$ with weights $w_1=3$, $w_2=2$, $w_3=1$, as described in Fig. 8. Pixel p is located in the center of the generators and is in a perfect position to prove the higher or-

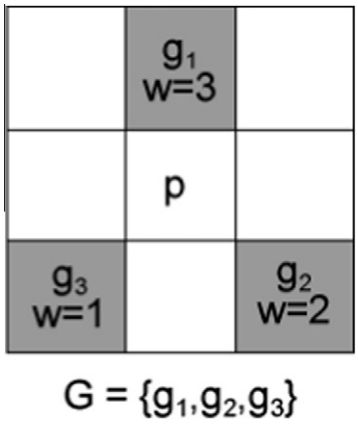


Fig. 8. The individual k layers of HOVD.

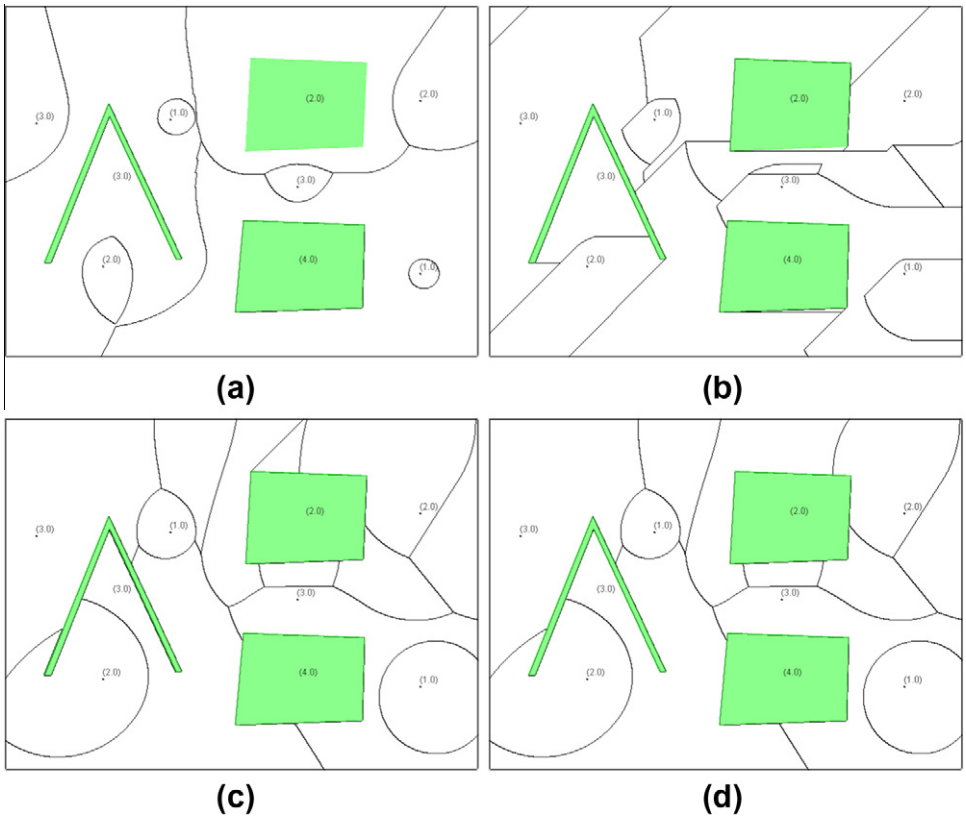


Fig. 9. The propagated distance map generated for each individual generator primitive g_i.

der predicate with weights. For the base case, let us consider the 1st-order Voronoi diagram in the Euclidean metric. To find the nearest generator to p requires a forward scan with $N_1(p)$ and reverse scan with $N_2(p)$. However, with the generators G directly neighboring p, we only need to consider $q3(g_1)$, $q6(g_2)$, $q8(g_3)$. To determine the distances from the generators G to p we use the F^1 Function described in Section 4.2.3

$$D_c(q3,p) = (dist(R_x(q),R_y(q)) + h(p,q3))/w_{q3} \quad (12)$$
$$= (0 + (2|R_y(q3)| + 1))/3 \quad (13)$$
$$= 2|1| + 1/3 \quad (14)$$
$$= 1 \quad (15)$$

$$D_c(q6,p) = (dist(R_x(q),R_y(q)) + h(p,q6))/w_{q6} \quad (16)$$
$$= (0 + (2(|R_x(q6)| + |R_y(q6)| + 1)))/2 \quad (17)$$
$$= 2(|1| + |1| + 1)/2 \quad (18)$$
$$= 3 \quad (19)$$

$$D_c(q8,p) = (dist(R_x(q),R_y(q)) + h(p,q8))/w_{q8} \quad (20)$$
$$= (0 + (2(|R_x(q8)| + |R_y(q8)| + 1)))/1 \quad (21)$$
$$= 2(|1| + |1| + 1)/1 \quad (22)$$
$$= 6 \quad (23)$$

After determining the distances from p to the neighboring generators G, the nearest generator has been determined as g_1 at $q3$. This is because of the non-diagonal connection and high weight associated with the generator. For the inductive step, the maximum HOVD possible with generators G is the 3rd-order Voronoi diagram which includes all generators. We can therefor achieve the 3rd-order Voronoi diagram by combining the last generator with the 2nd-order Voronoi diagram. To achieve the 2nd-order Voronoi diagram, we must determine the next closest Voronoi diagram to p from the set of remaining generators $\{g_2,g_3\}$. From the previous distance calculations, we can determine that the 2nd-order Voronoi diagram is made up of the set of generators $\{g_1,g_2\}$. The set of 2nd-order generators $\{g_1,g_2\}$ with the addition of g_3 gives the 3rd-order Voronoi diagram. Fig. 9 demonstrates the generation of the 2nd-order Voronoi diagram from the ordinary Voronoi diagram with the three sweep scans.

6. Algorithms analysis

Fig. 10 describes the generator pixels used for algorithm analysis. For comparing processing time of the generator index propagation algorithm the naive algorithm is used. The naive algorithm is the simplest form of distance transform which finds the nearest generator g_i to p by calculating and finding the smallest distance out of all generators G. The naive method is not used for comparing the propagated distance map because of the complexities of determining distance values with obstacles in the plane. The two algorithms are tested together for three types of generalizations. This includes few and many weighted generators, few and many complex generators, and few and many generators with obstacles in the plane. Lastly, the effects of k on computation times are considered.

6.1. Weighted higher-order generator pixels

Fig. 10(a) describes the generator pixels used for generating the 2nd-order Voronoi diagrams with low and high computation times. The number of generator pixels in the plane F does not affect the computation time of the nearest-neighbor algorithm, as opposed to the naive algorithm. This is because the nearest neighbor algorithm computation time for higher-order Voronoi diagrams is linear with the total number of pixels in the plane F. Fig. 11(a) depicts experimental results.

6.2. Weighted higher-order complex generator primitives

The primitives described in Fig. 10(b) and (c) are used to benchmark the performance of the algorithms for generating the HOVD with complex generator primitives. The optimal set of three complex generator primitives shows the naive algorithm to be more efficient. However, the generator index propagation with the nearest-neighbor algorithm results in linear time computation with regard to the number of pixels in the plane F. Because the complex generator primitives reduce the total number of distance value calculations for pixels p, an increased amount can actually decrease the computation time required. However, it must be considered that there exists an increased overhead associated with complex primitives because each distance value calculation at pixels p require the closest pixel of the complex generator primitive G to p to be found. However, with optimization this overhead is minimized which results in similar computation times of Voronoi diagrams with generator pixels. Fig. 11(b) shows experimental outcomes.

6.3. Weighted higher-order generator primitives with obstacles

The primitives described in Fig. 10(d) and (e) are used to benchmark the performance of the HOVD with obstacles. The effects of increasing the number of weighted generator primitives is an increase in total computation time. This means the total computation time is $O(F \times G + k \times F)$, where $F \times G$ denotes the propagated distance maps generation and $k \times F$ denotes the individual scans for generating the HOVD. Results are shown in Fig. 12(a).

6.4. Effects of k on weighted higher-order computation time

The results describe the effects of k on computation times for generating the kth-order weighted Voronoi diagram of 4 and 100 weighted generator pixels in a 500×500 pixel plane F. The results prove that the number of weighted generator pixels does not

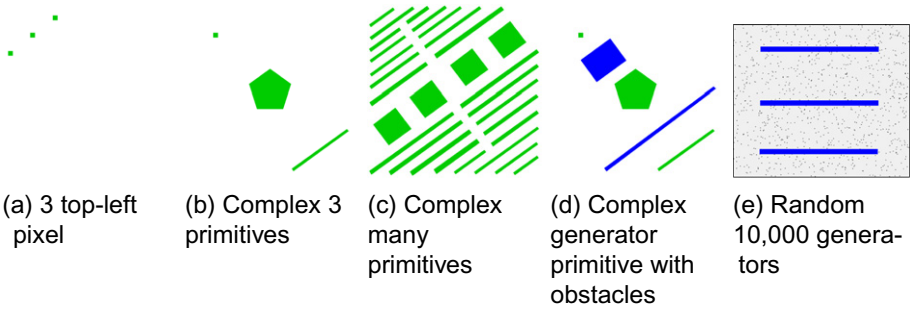
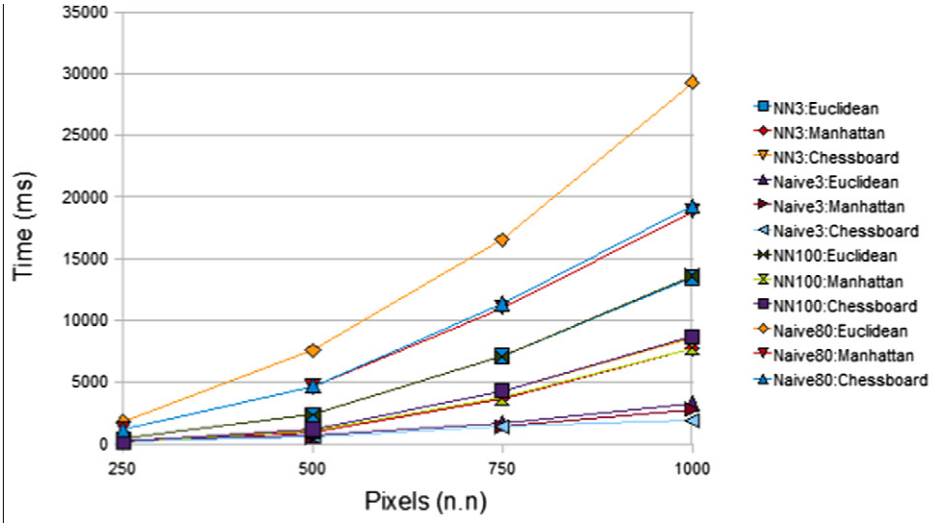
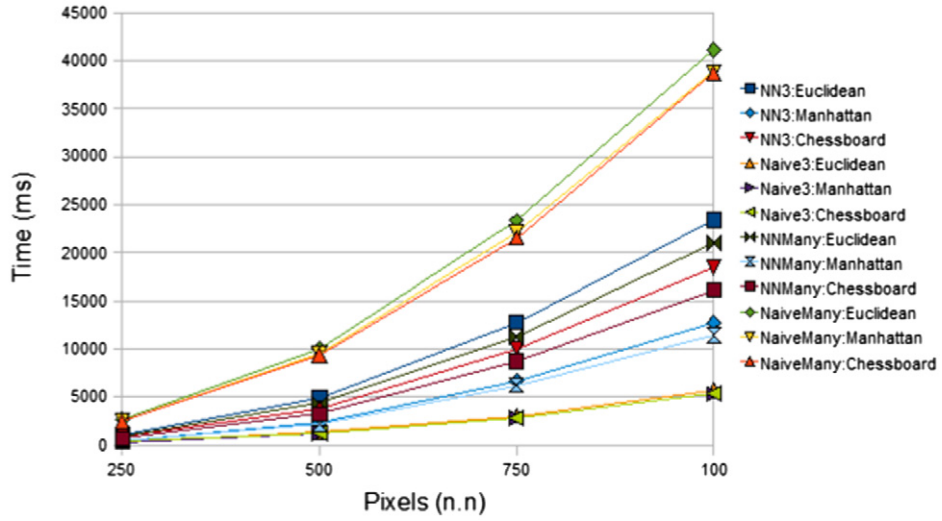


Fig. 10. The test planes and primitives used for analyzing HOVD.



(a) NN and Naive comparison with weighted pixels 3-10000 for NN and 3-80 for Naive



(b) NN and Naive comparison with 3+ weighted complex primitives

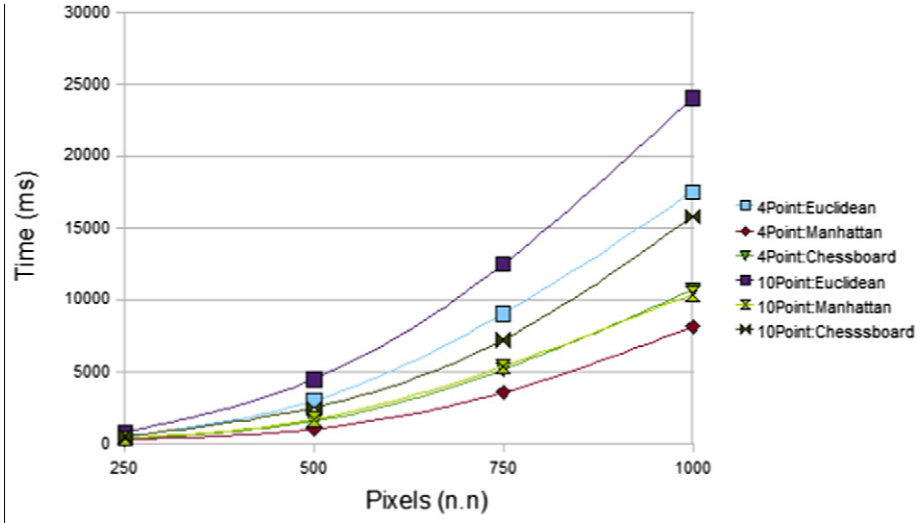
Fig. 11. The computation times of weighted higher-order algorithms for generating: (a) pixels and (b) complex primitives.

significantly affect the computation time of the nearest-neighbor algorithm. The total computation time for the weighted HOVD is linear with regard to the size of the plane F and k-th degree. Fig. 12(b) illustrates experimental outcomes.

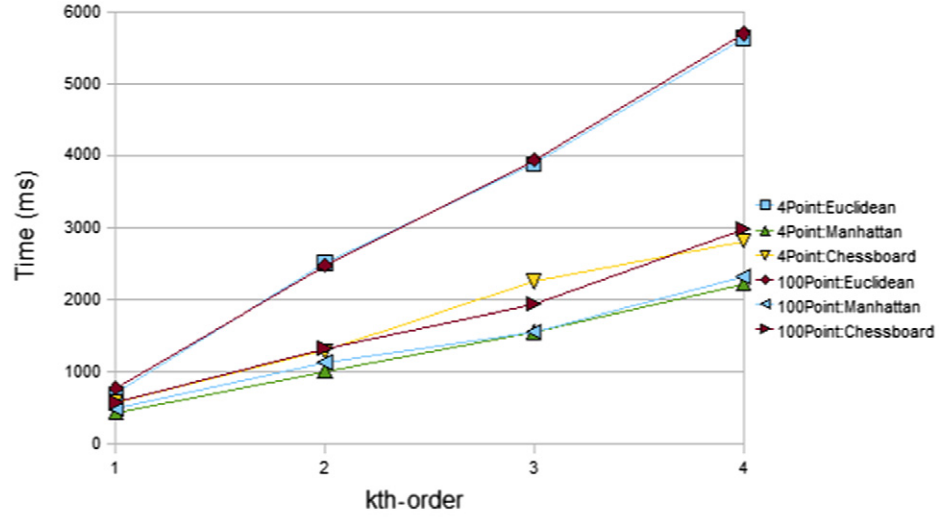
7. Algorithm error

The analysis of the nearest-neighbor algorithms computation times reveal that it is very efficient for pixel and complex primitive generators. However, the efficiency does not matter if the error rate of the generated Voronoi diagrams is found to be high. This section compares the results of the nearest-neighbor algorithm with the naive algorithm for generating the Voronoi diagrams with the primitives described in Fig. 13(a) and (b) for a 640 × 480 pixel plane F. The Generalized Voronoi Diagrams are generated by both algorithms and the produced images are compared to determine the percentage of difference. The naive algorithm calculates the distances values for each pixel p to all generators in the plane to obtain correct nearest generator. The closeness of the nearest-neighbor algorithm to the naive algorithm determines the algorithm error.

The results shown in the table and graph in Fig. 15 give accuracies of 98.83% for complex generator primitives, 97.17% for the series of generator pixels, and a combined total accuracy of 98%. A small percentage of the error is due to the differences of drawing the Voronoi regions with the two algorithms. The remaining error is attributable to the lack of dis-continuous regions associated with the nearest-neighbor algorithm. Because distance values and relative distance values are propagated via nearest-neighbor pixels, if propagation of values from a generator g_i are blocked by propagations from other generators g_j then regions that might have been associated with g_i behind g_j are instead assigned to g_j. This property usually has effect with weighted generator primitives, although it can have effects with Manhattan and chessboard metrics also. This property can be mentally visualized with the analogy of crystal expansion. Crystal cells expand from their original positions at different rates until they are blocked in all directions by competing cells. The results this can have on Voronoi diagrams is shown in the example with Fig. 14(a) and (b). While this attribute of the nearest-neighbor algorithm is considered an error as defined by the properties of Voronoi diagrams, it can still be useful for

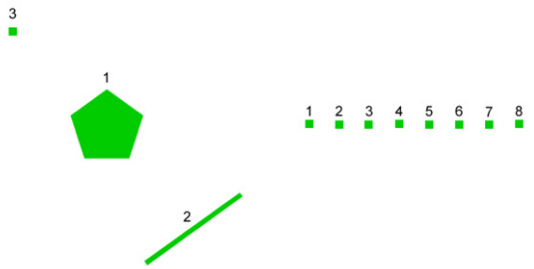


(a) NN computation times with 3 weighted complex primitives and 10 weighted generator pixels with obstacles



(b) NN computation times for weighted generator primitives in a 500×500 pixel plane with regard to k

Fig. 12. The computation times of higher-order algorithms for generating: (a) generator primitives with obstacles and (b) ascending metrics of k dimensions with weighted primitives.



(a) Complex 3 primitives (b) Pixel series

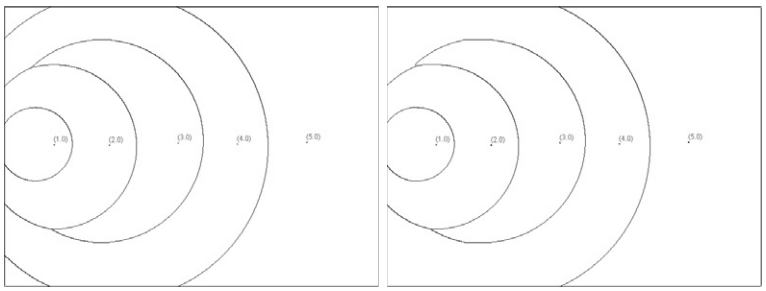
Fig. 13. The test planes and primitives used for comparing the nearest-neighbor algorithm with the naive algorithm.

geospatial analysis in scenarios where having dis-continuous regions is not appropriate.

8. Trade area modeling case-study

Here we demonstrate the usefulness of the HOVD described in the previous sections in terms of modeling trade areas. We focus our study on a similar research done by Boots and South (1997) for determining a customer's preference to k sites of interest. Like their study, we limit our interpretation to the case where $k = 2$. We also use the following:

1. a number of facilities represented by the set of generators G of the same type (e.g., supermarkets), which are located in a finite planar region S;
2. customers patronize one or more of the facilities;
3. an individual facility, j, is assigned a weight, w where $(1 < w)$, on the basis of its attractiveness to customers in terms of one or more attributes (e.g., price, size, parking, age, etc.);



(a) Complex 3 primitives (b) Pixel series

Fig. 14. The test planes and primitives used for comparing the nearest-neighbor algorithm with the naive algorithm.

	Complex Primitives			Series		
	P:Euclidean	P:Manhattan	P:Chessboard	S:Euclidean	S:Manhattan	S:Chessboard
HO	99.00%	98.00%	99.00%	100.00%	100.00%	92.00%
Weighted HO	99.00%	99.00%	99.00%	95.00%	96.00%	100.00%
	Average: 98.83%			Average: 97.17%		
	Total Average: 98.00%					

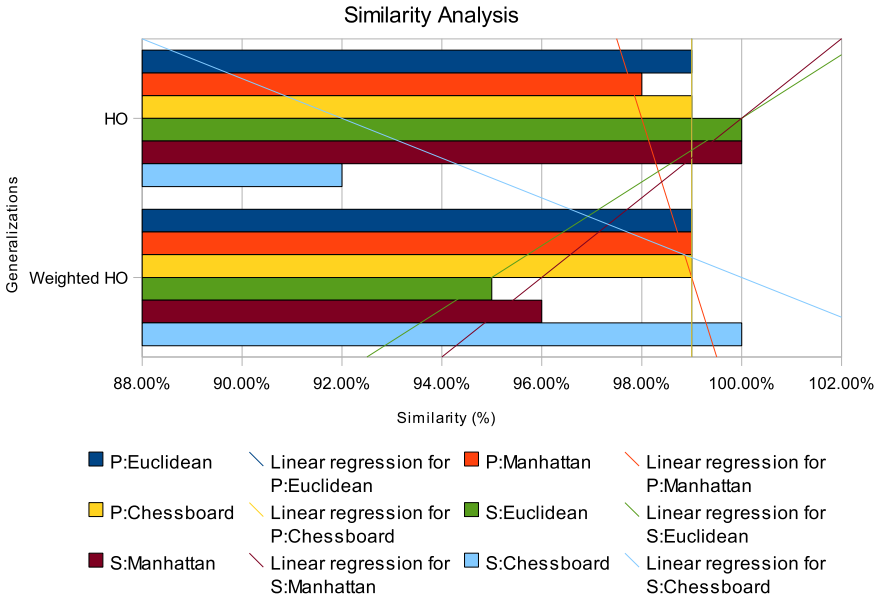


Fig. 15. The comparison of the nearest-neighbor algorithm with the naive algorithm for generating GVD with the test set of primitives in Fig. 13(a) and (b).

4. customers limit their purchases to the two facilities with the highest attractiveness;
5. the customer is indifferent between the two facilities (i.e., the customer is equally likely to select either of the facilities).

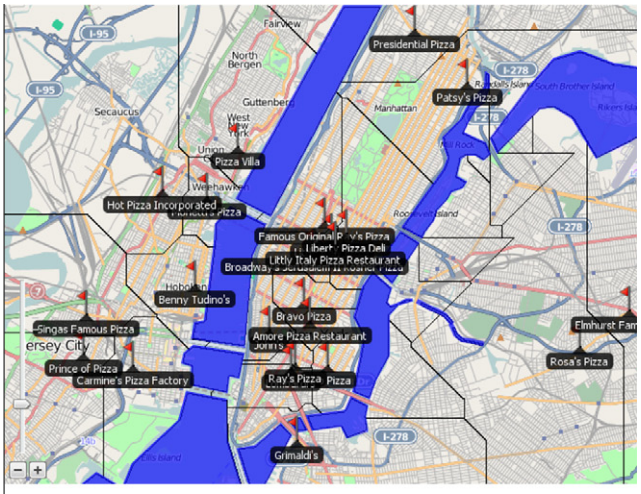
We add a modern features to their study with complex generator primitives and complex obstacle primitives. We also consider the study region in the Manhattan metric for best describing distance values in the city environment. The attractiveness values of facilities are also determined as the average ratings determined from the Yahoo Local (http://local.yahoo.com/) business and services search engine.

8.1. Restaurant location optimization with HOVD

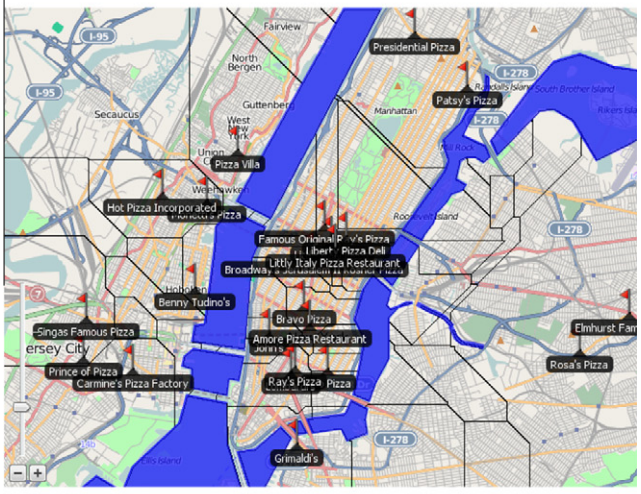
The facilities modeled consist of pizza restaurants located around Manhattan, New York. Manhattan is naturally divided by rivers on both sides of the island. This natural constricting of the plan can influence dominance regions by affecting the total distance required to travel to reach the nearest restaurant. Fig. 16(a) describes the locations of the restaurants and river obstacles. Fig. 16(b) describes the order-2 Voronoi diagram in the Manhattan metric as travel in a city environment is often grid-wise. The size of the dominance areas can be used for generating sales estimates for the restaurants. The order-k Voronoi diagram allows a customer's indifference to k restaurants to be modeled, which can be useful to determine locations best suited to take advantage of the popularity of neighboring restaurants. Fig. 16(c) gives the weighted order-2 Voronoi diagram simulating a customers indifference to the two nearest restaurants with an added attractiveness modifier. A restaurant with a higher rating is more likely to attract a customer, in turn increasing its dominance region. Therefore, restaurants of high rating and in close proximity are likely to be more favorable to attract customers.

8.2. Restaurant location optimization with k-medoid based HOVD

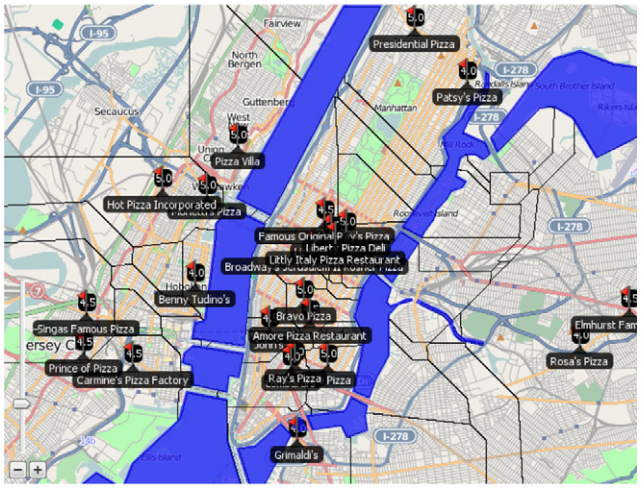
An advanced example of the uses of order-k Voronoi diagrams with retail trade modeling can be shown with the use of k-medoid clustering (Kaufman & Rousseuw, 1990). The k-medoid clustering partitions the dataset into groups and attempts to minimize the



(a) Pizza restaurants located in Manhattan, New York

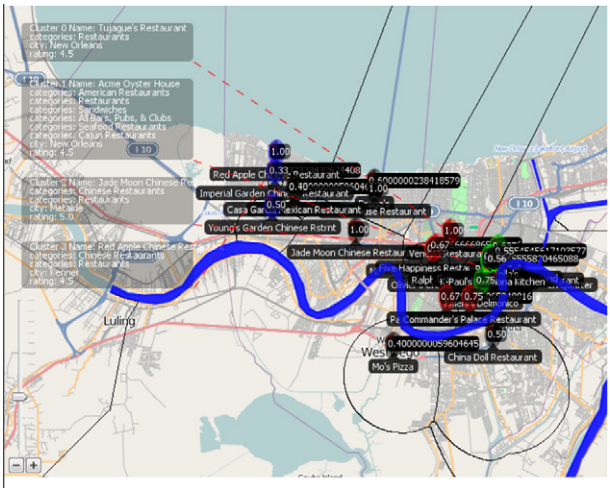


(b) Order-2 Voronoi diagram in the Manhattan metric

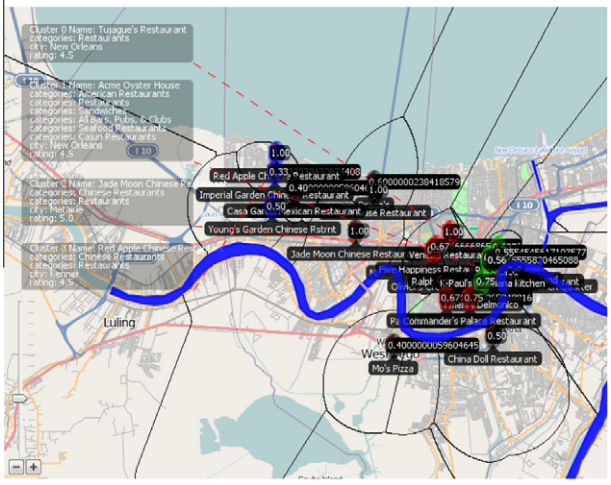


(c) Weighted order-2 Voronoi diagram in the Manhattan metric

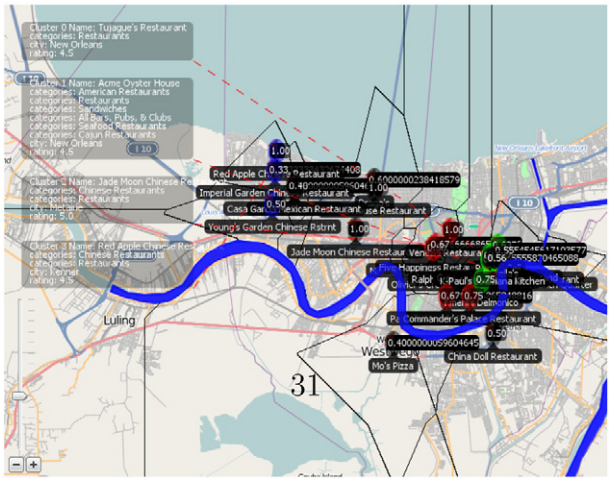
Fig. 16. Trade area modeling with pizza restaurants case-study.



(a) Clustered restaurants located in New Orleans with weighted dominance regions in Euclidean metric



(b) Clustered restaurants in the order-2 Voronoi diagram in the Euclidean metric



(c) Clustered restaurants in the order-2 Voronoi diagram in the Manhattan metric

Fig. 17. Trade area modeling with pizza restaurants case-study.

squared error, the distance between points assigned to a cluster and a medoid point designated as the center of that cluster. When k-medoid clustering is partitioning objects, distance can be defined as the cosine distance similarity between attribute vectors u and v of the objects. The medoid object found defines the most general instance of the object represented by the cluster based upon its common attributes. In the case of retail trade modeling, the medoid represents the most general facility of that cluster. It can be assumed that the most general facility has a high chance of attracting the majority of customers who are in search of the range of attributes defined within the cluster. In the following examples, the facilities of the same cluster are assigned a weight based on the degree of similarity to the medoid facility. The medoid being most general receives a weight of 1 and will produce the largest dominance region. Subsequent facilities are assigned a weight $0 < w < 1$. The reasoning behind this weight association is based on a customer's preference to a facility based on the distance and similarity to a medoid facility. If a customer wishes the attributes defined by the cluster and the customer is in the dominance region of the medoid facility, then the customer will most likely travel there. However, if a customer is in the dominance region of a non-medoid facility, then the customer is assumed to travel to that facility even though it represents a subset of the general attributes. The principle goes "if a facility is close and good enough, then it will do". If we assume that a customer's preference will be a choice out of possible defined clusters, then we can combine k-medoid clustering with order-k Voronoi diagrams to determine a customers preference to k facilities defined in various clusters. The following diagrams describe this functionality.

Fig. 17(a) describes a set of four clustered restaurants located in New Orleans with their dominance regions represented by the weighted Voronoi diagram in the Euclidean metric. The weights assigned describe the degree of similarity to the medoid restaurants. Fig. 17(b) describes the clustered restaurants in the order-2 Voronoi diagram in the Euclidean metric. These dominance regions describe the customer's preference to the nearest two restaurants based on the customer's preference out of the four clustered restaurants and the similarity of the individual restaurants to the medoid restaurants. Fig. 17(c) describes the same model but instead in the Manhattan metric.

9. Summary and future work

In this paper, we described novel sequential-scan districting algorithms. We revealed how the algorithms are capable of producing HOVD with complex primitives, weighted primitives, Minkowski metrics and the inclusion of obstacles. HOVD are generated in $O(k \times F)$ and the addition of obstacles in $O(F \times G + k \times F)$, with k being the kth order, F the total number of pixels and G being the set of generator primitives. The paper also described practical uses of the algorithms for trade area modeling, although the algorithms could potentially be used for a wide variety of applications. The next stage of the project is the inclusion of network Voronoi diagrams and discriminatory HOVD, where discriminatory focuses on generating Voronoi regions with k nearest primitives of interest. This reasoning differs from traditional HOVD properties which seek the first order-k or ordered order-k generators. With the introduction of generator primitives representing real world features, it could prove useful when finding patterns of certain combinations of primitives of interest. The new computation models are intended to be used within a geographic knowledge discovery from GeoWeb framework for detailed geospatial analysis and predictive modeling from Web 2.0 technologies.

References

Boots, B., & South, R. (1997). Modelling retail trade areas using higher-order, multiplicatively weighted Voronoi diagrams. *Journal of Retailing, 73*(4), 519–536.

Borgefors, G. (1984). Distance transformations in arbitrary dimensions. *Computer Vision, Graphics and Image Processing, 17*, 321–345.

Breu, H., Gil, J., Kirkpatrick, D., & Werman, M. (1995). Linear time Euclidean distance transform algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 17*(5), 529–533.

Chen, Y.-K., Wang, C.-Y., & Feng, Y.-Y. (2010). Application of a 3NN + 1 based CBR system to segmentation of the notebook computers market. *Expert Systems with Applications, 37*, 276–281.

Cuisenaire, O. (1999). Distance transformations: Fast algorithms and applications to medical image processing. Ph.D. Thesis. Belgique: Universite Catholique de Louvain.

Cuisenaire, O., & Macq, B. (1999). Fast Euclidean distance transformation by propagation using multiple neighborhoods. *Computer Vision and Image Understanding, 76*(2), 163–172.

Danielsson, P. E, & Tanimoto, S. (1983). Time complexity for serial and parallel propagation in images. In P. E. Danielsson & A. J. Oosterlinck (Eds.). *Proceedings of the SPIE conference on architecture and algorithms for digital image processing* (Vol. 435, pp. 60–67). Society for Photo-Optical Instrumentation Engineers.

Fabbri, R., Costa, L. D. F., Torelli, J. C., & Bruno, O. M. (2008). 2D Euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys, 40*(1), 1–44.

Fortune, S. (1987). A sweepline algorithm for Voronoi diagrams. *Algorithmica, 2*, 153–174.

Hanafizadeh, P., & Mirzazadeh, M. (2011). Visualizing market segmentation using self-organizing maps and fuzzy Delphi method – ADSL market of a telecommunication company. *Expert Systems with Applications, 38*, 198–205.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100–107.

Kaufman, L., & Rousseuw, P. J. (1990). *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley & Sons.

Kolountzakis, M. N., & Kutulakos, K. N. (1992). Fast computation of the Euclidian distance maps for binary images. *Information Processing Letters, 43*(4), 181–184.

Krause, E. F. (1975). *Taxicab geometry*. California: Addison-Wesley.

Lee, I., & Lee, K. (2007). Higher order Voronoi diagrams for concept boundaries and tessellations. In *Proceedings of the 6th IEEE/ACIS international conference on computer and information science ICIS 2007* (pp. 513–518). http://dx.doi.org/10.1109/ICIS.2007.106.

Lee, D.-T., & Drysdale, R. L. (1981). Generalization of Voronoi diagrams in the plane. *SIAM Journal of Computing, 10*(1), 73–87.

Lee, I., & Gahegan, M. (2002). Interactive analysis using Voronoi diagrams: Algorithms to support dynamic update from a generic triangle-based data structure. *Transactions in GIS, 6*(2), 89–114.

Lee, I., & Lee, K. (2009). A generic triangle-based data structure of the complete higher order Voronoi diagrams for distributed disaster and emergency management. *Computers, Environments and Urban Systems, 33*(2), 90–99.

Lee, I., Qu, Y., & Lee, K. (2012). Mining qualitative patterns in spatial cluster analysis. *Expert Systems with Applications, 39*, 1753–1762.

Lee, I., & Torpelund-Bruin, C. (2012). Geographic knowledge discovery from web map segmentation through generalized Voronoi diagrams. *Expert Systems with Applications, 39*, 9376–9388.

Li, C., Chen, J., & Li, Z. (1999). A raster-based method for computing Voronoi diagrams of spatial objects using dynamic distance transformation. *International Journal of Geographical Information Science, 13*(3), 209–225.

Lin, I.-J., & Kung, S.-Y. (2001). Extraction of video objects via surface optimization and Voronoi order. *VLSI Signal Processing, 29*(1–2), 23–39.

Okabe, A., Boots, B. N., Sugihara, K., & Chiu, S. N. (2000). *Spatial tessellations: Concepts and applications of Voronoi diagrams* (2nd ed.). West Sussex: John Wiley & Sons.

Pinliang, D. (2008). Generating and updating multiplicatively weighted Voronoi diagrams for point, line and polygon features in GIS. *Computers & Geosciences, 34*(4), 411–421.

Rosenfeld, A., & Pfaltz, J. L. (1966). Sequential operations in digital picture processing. *Journal of ACM, 13*(4), 471–494.

Seng, J.-L., & Lai, J. T. (2010). An intelligent information segmentation approach to extract financial data for business valuation. *Expert Systems with Applications, 37*, 6515–6530.

Shih, F. Y., & Wu, Y.-T. (2004). Fast Euclidean distance transformation in two scans using a 3×3 neighborhood. *Computer Vision and Image Understanding, 93*(2), 195–205.

Xie, W., Wang, R., & Cao, W. (2007). Analysis of higher order Voronoi diagram for fuzzy information coverage. In H. Zhang, S. Olariu, J. Cao, & D. B. Johnson (Eds.), *Proceedings of the International Conference on Mobile Ad-Hoc and Sensor Networks. Lecture Notes in Computer Science* (Vol. 4864, pp. 616–622). Springer.