Visualization of Generalized Voronoi Diagrams

Alexandru Telea, Jarke J. van Wijk

Eindhoven University of Technology,
Den Dolech 2,Eindhoven 5600 MB, The Netherlands,
{alext,vanwijk}@win.tue.nl

Abstract. Voronoi diagrams are an important data structure in computer science. However well studied mathematically, understanding such diagrams for different metrics, orders, and site shapes is a complex task. We propose a new method to visualize k-order diagrams and give an efficient adaptive implementation for this method. The algorithm is easy to customize for different metrics and site shapes. Its real-time performance makes it suitable for interactive planning and analysis of complex Voronoi configurations in 2D. We illustrate the method for different combinations of metrics and site shapes.

1 Introduction

Voronoi diagrams are a fundamental data structure in computer science. Mostly used in computational geometry, Voronoi diagrams have found their way in many application areas, such as computer graphics (collision detection, motion planning), optimization theory (associative file searching, clustering, scheduling), and physics (crystal and cell growth studies).

Voronoi diagrams based on Euclidean distance are the best known. Such diagrams partition a 2D plane in regions such that all points within a region are closest to one site from a given site set. Visualizing such diagrams is straightforwardly done by drawing the set of disjoint, adjacent planar polygons that represent the diagram.

We present a new visualization method for two generalizations of the Voronoi diagrams. The first generalization regards *k-order diagrams*, which partition the plane in cells such that all points in a given cell have the same k closest sites. Although many studies cover the mathematics of k-order Voronoi diagrams [2, 1], getting an intuitive understanding of such diagrams is a difficult task. Specifically, we would like to answer questions such as 'which are the k sites that influence a given partition' and 'which are all points under the k-order influence of a given site' in a simple, visual manner. The second generalization concerns using different metrics besides the Euclidean distance and different site shapes besides points. Diagrams for higher orders, different site shapes and metrics lead to complex shape-site relationships that require a more elaborate visualization than a straightforward polygonal drawing.

In Section 2, we give a mathematical overview of Voronoi diagrams and outline the difficulties inherent to their visualization. In Section 3, we introduce our method for visualizing generalized Voronoi diagrams and illustrate it with several examples. Section 4 presents an efficient implementation of the method. We conclude in Section 5 with future research directions.

2 Background

We begin with a description of the elementary properties of 2D Voronoi diagrams. More on the mathematical aspects is available from several surveys [2, 4, 5].

Let $P = p_1, \ldots, p_n$ be a set of n distinct points in the plane, called sites, and $d(p, q)$ the Euclidean distance between points p and q. The first order Voronoi diagram of P is a subdivision of the plane in n cells, one for each site in P, such that a point q lies in the cell of a site p_i if and only if $d(q, p_i) < d(q, p_j)$ for all sites $p_j \in P$ with $j \neq i$. The cell boundaries lie thus on the perpendicular bisectors of the line segments $p_i p_j$ (Fig. 1 a). First order Voronoi diagrams are used, for example, to partition a city map into regions (cells), given a set of fire station positions (sites), such that any city location is assigned to the closest fire station.

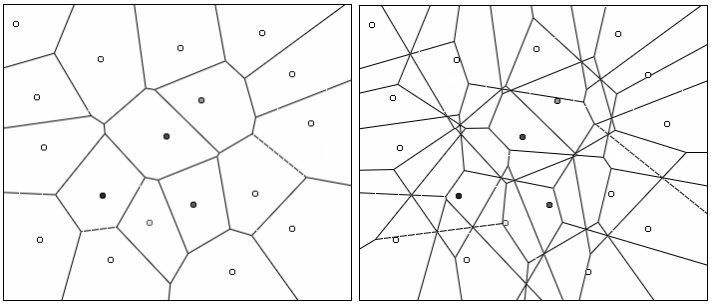


Fig. 1. Order-1 (a) and order-2 Voronoi diagram of a point set

A *k-order* Voronoi diagram subdivides the plane in cells such that all points in a cell have the same ordered set of k closest sites from the site set P. In our example, a k-order diagram would indicate which are the second, third, and k-th fire stations that serve a given city location if the closest (first order) fire station is unavailable for some reason. Conversely, one can visualize the regions served by a given fire station when some neighboring stations fail to work. In an interactive city planning setup, one could add, delete, or move the locations of the fire stations to optimize the area coverage and redundancy in case of fire station failure.

However, visualizing k-order diagrams by simply drawing the involved cells produces hardly readable drawings (Fig. 1 b). It is hard to tell from such a drawing which is the area served by a fire station if other stations fail or, conversely, which are the k stations serving a given city location. These questions could be answered by interactively selecting a site and highlighting its influence region or, conversely, by highlighting all stations that serve a given city location. However, such a method is not capable of producing an overview of the influence of *all* fire stations on *all* city locations simultaneously.

A second generalization of the Voronoi diagrams involves the distance function used. The L_1 (Manhattan distance) metric $d(p, q) = |x_p - x_q| + |y_p - y_q|$ is used to model access times to strategic locations in a city where the streets form an orthogonal grid, or the access time to given records in mass storage systems where the read/write head can only move in orthogonal directions [3]. All edges in such a Voronoi diagram are vertical,

horizontal, or diagonal at 45 degree angles. Weighted Voronoi diagrams assign a weight w_i to each site p_i and define the distance $\delta(q, p_i)$ by adding or multiplying the Euclidean distance with the weight w_i. Additive weights were used by Johnson and Mehl [2, 4] to model the growth of crystals from a given seed set. Multiplicative weights lead to the so-called Apollonius model that describes the growth of plant cells, coverage areas of trees, or areas of best received transmitters [6]. Finally, distances can be computed from other site shapes than points, such as lines and curves.

So far, visualizing Voronoi diagrams for higher orders and/or different metrics has been limited to drawing the cell boundaries, such as in Fig. 1. As mentioned, such drawings communicate little or no insight in the k-level hierarchy of subregions generated by the sites. In the following, we shall exploit other graphical dimensions, such as color and shading, to communicate more insight into the complexity of Voronoi diagrams.

3 Shaded Voronoi Diagrams

Our basic idea is to use the natural ability of the human visual system to interpret shade and color cues as boundaries of illuminated objects. We construct two types of such graphical objects, i.e. cushions and bevels, as follows.

3.1 Cushions

Suppose first we have a first order Voronoi diagram. For each site $p_i = (x_i, y_i)$ of the diagram, consider a curved surface $z(x, y)$ given by:

$$z(x, y) = h_1 \min_{i=1}^{n} [(x - x_i)^2 + (y - y_i)^2], \tag{1}$$

where h_1 is a (usually negative) height scale factor. In other words, we build a parabolic cushion under each site p_i. To shade the cushion, we use a diffuse shading model. The surface normal $\mathbf{n}$ is given by:

$$\begin{aligned} \mathbf{n} &= [-\partial z/\partial x, -\partial z/\partial y, 1] \\ &= [2h_1(x_i - x), 2h_1(y_i - y), 1] \end{aligned} \tag{2}$$

The final pixel intensity I is given by:

$$I = I_a + I_l \max(0, \frac{\mathbf{n} \cdot \mathbf{l}}{|\mathbf{n}||\mathbf{l}|}) \tag{3}$$

where I_a is the ambient light intensity and $\mathbf{l}$ and I_l are the direction and the intensity of a directional light source. This shading scheme (Fig. 2 a) is similar with the rectangular treemap cushions described by Van Wijk et al [9]. The shading of the cushions gives a visual cue for the distance of any point to the closest site, as any pixel is ultimately shaded in function of the closest site p_i.

For a k-order Voronoi diagram, we superimpose extra cushions on top of the k cushions generated by the k closest sites $p_1, \ldots, p_k$ for every pixel, to visualize the higher order structure. The center of the order-1 cushion is p_1, the center of the order-2 cushion is between p_1 and p_2, and the center of the order-i cushion is $c_i = \frac{1}{i} \sum_1^i p_i$. To emphasize the

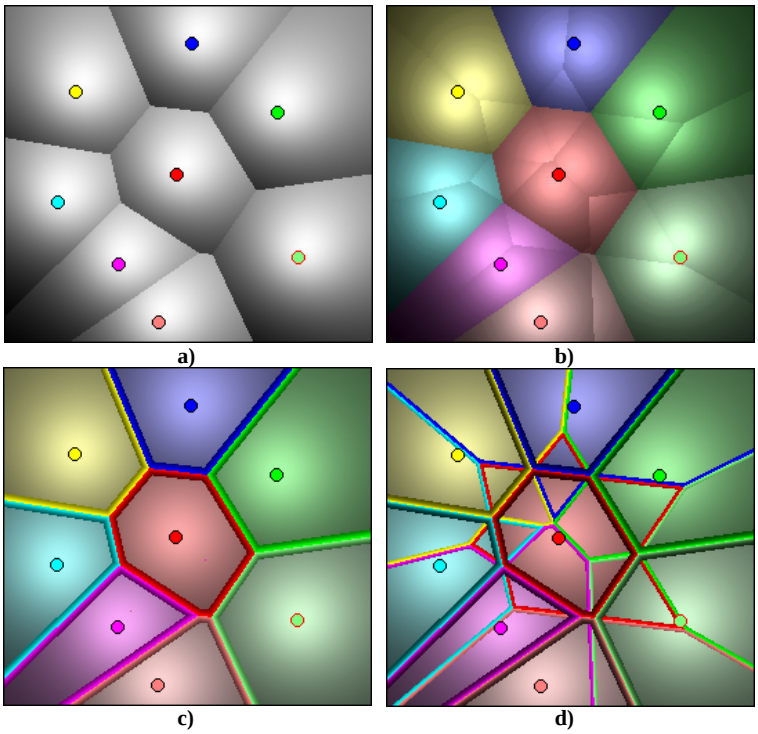


Fig. 2. Cushions (a,b) and cushions and bevels (c,d) visualization

cell nesting in the diagram, the height h_i of the i th level cushion is scaled to $h_i = s^{i-1}h_1$, where the scale factor s lies between 0 and 1. In this way, closer sites have a stronger effect on the cushions than farther ones. Figure 2 b shows the 2-order diagram rendering for the same sites as in Fig. 2 a. To enhance the nesting effect, we assign a hue to every site p_i and color all cushions influenced by that site with that hue (see Appendix). If desired, one could easily automate the assignment of hues to sites such that no two sites with neighboring first order diagram cells have the same hue.

3.2 Bevels

The shading described so far emphasizes the discontinuities between the cells of a Voronoi diagram as well as the nesting of higher order cells within the lower order ones. The shade of a pixel gives a cue to its distance to the sites that influence it. Although better than drawing cell contours only, shaded cushions still cannot indicate clearly the sites that influence a given pixel and the influence region of a given site. To emphasize these aspects, we augment our visualization by drawing *colored bevels* along the cell edges. The basic bevel idea is similar to the concept introduced by Bruls et al. for visualizing squarified treemaps [10].

A bevel is defined here as a parabolic surface of given height h_{bevel} and width w_{bevel} whose medial axis follows the Voronoi cell boundaries. Just as cushions, bevels are scaled

to reflect the hierarchy level. For a bevel of level i we have:

$$w_{bevel\,i} = t^{i-1} w_{bevel\,0} \quad \text{and} \quad h_{bevel\,i} = t^{i-1} h_{bevel\,0} \tag{4}$$

where the bevel scaling factor t is between 0 and 1. Figures 2 c and d show the order-1 and order-2 bevels corresponding to the cushion diagrams in Figs. 2 a and b respectively.

While the bevel size depends on the order of the Voronoi cells that meet at that edge, its color indicates the two sites that are at equal distance from that edge. For two sites p_i and p_j, we color the two halves of the bevel equidistant to p_i and p_j with the hues of p_i and p_j.

The size and color coding of bevels is a non-ambiguous, intuitive way to interpret a k-order Voronoi diagram. To identify the k sites closest to a point p in the plane, we first look at the cushion containing p (see Appendix). The cushion's hue indicates the closest site, i.e. the site that has the strongest influence on p. The cell edge's bevels describe the other (at most $k-1$) sites influencing the cell as follows:

- the *inner bevel hues* identify the sites
- the *bevel widths* are proportional with the distances to the sites

Conversely, to identify the regions influenced by a given site p_i, we first look at the cell in which p_i is found, i.e. the cell bearing p_i's hue. This is the region to which p_i is the closest site. To identify the regions to which p_i is the second,third,...k th closest site, we look at the regions bordered by bevels of p_i's hue of decreasing bevel width. These regions are nested (order $k+1$ region contains order k region). Since the bevels of a region *never* overlap with the ones of a region of different order of the same site, it is rather easy to visually follow a bevel of a given width and color.

3.3 Interactive visualization

We have built an interactive application in which the user can control several aspects of the Voronoi diagram visualization orthogonally. First, the diagram order k, saturation, hue, height h, and scale factors s, t of both cushions and bevels can be chosen independently. This allows viewing a k-order Voronoi diagram in a multitude of ways, e.g. cushions or bevels only, or a gradual blend of the two, depending on the height and saturation factors. An effective combination is for example displaying only the first order cushions with low hue saturation and height values, together with the 1 to k order bevels with high hue and height values. In this way, the background is split into a tiling of non distracting cushions, whereas the bevels form the visualization foreground. A small bevel scale factor $t \leq 0.4$ produces an effective display of the diagram hierarchy: thicker bevels show the first order cells whereas thinner bevels depict the higher order cells.

Secondly, one can interactively add, remove, or drag the sites with the mouse. This gives a direct insight in the way the 2D plane is subjected to the influence of the sites and assists the user effectively in finding an optimal site distribution to a given situation. Problem-specific configurations are very easy to implement e.g. by constraining the sites to stay within or outside prescribed areas on a city or geographical map, such as markets, urban areas, lakes, forests, etc. Following this idea, we have constructed a user interface for a musical synthesizer in which the sites represent musical instruments and the Voronoi cells their influence areas (Fig. 4). To create a new sound, one drags a marker and/or the

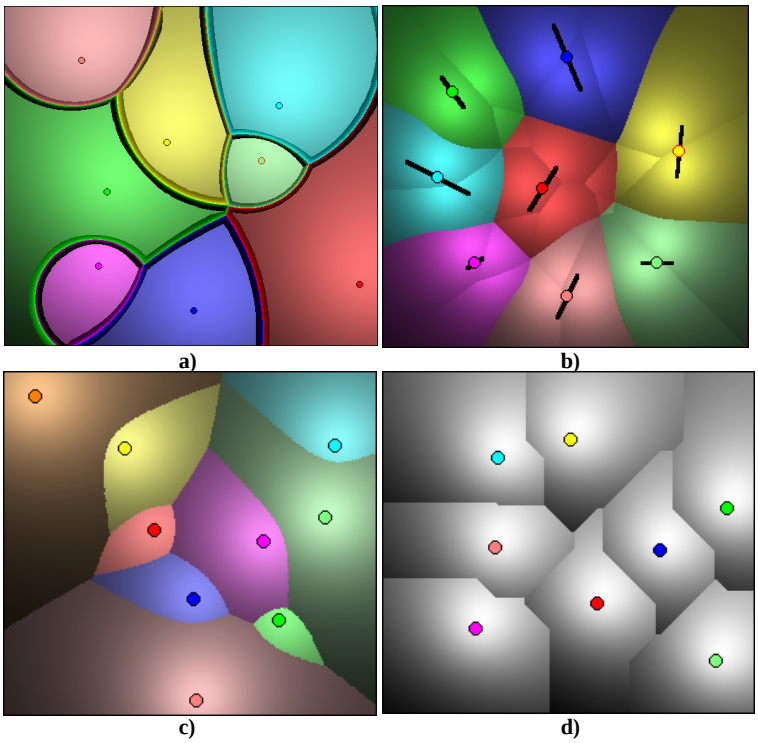


Fig. 3. Voronoi diagrams for different metrics

sites interactively on the Voronoi diagram. The sound is synthesized based on the marker-instrument sites distances.

Thirdly, the distance metric and site shape can be specified by the user. Figure 3 shows renderings for the multiplicative-weighted L_2 (Apollonius model) metric (a), an order-2 diagram for line sites (b), the additive-weighted L_2 (Johnson-Mehl model) metric (c), the L_1 (Manhattan distance) metric (d) (see also Appendix). In Fig. 5 a, a diagram defined by a discrete distance metric on a hexagonal board is shown. The map and cushion rendering are combined into a single image that shows the influence areas of several strategic sites in a computer military simulation [11] (Fig. 5 b).

4 Algorithm

A brute force implementation of the diagram rendering would compute $O(kNS)$ distances for a k-order diagram rendering of N pixels and S sites. Dragging a site with the mouse stops being interactive for $N > 200^2$ pixels and $S > 3$ sites. However, we can exploit the fact that all pixels of a given Voronoi cell share the same k closest sites. One idea would be to compute the k-order Voronoi diagram geometrically, which is $O(Slog^3S+k(S-k))$ complex [1], and then to use it as a spatial search structure to locate the k closest sites for each pixel. However, implementing a robust k-order analytic Voronoi diagram algorithm for any distance metric is a very complex task.

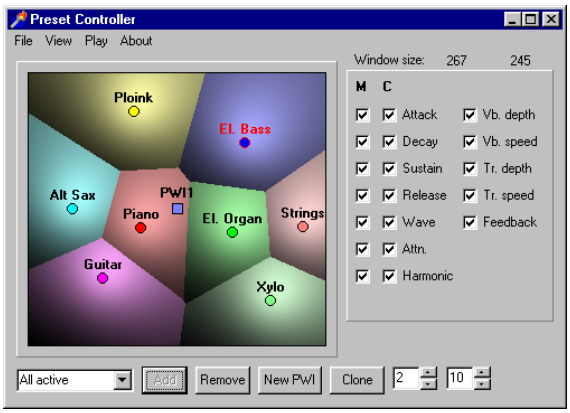
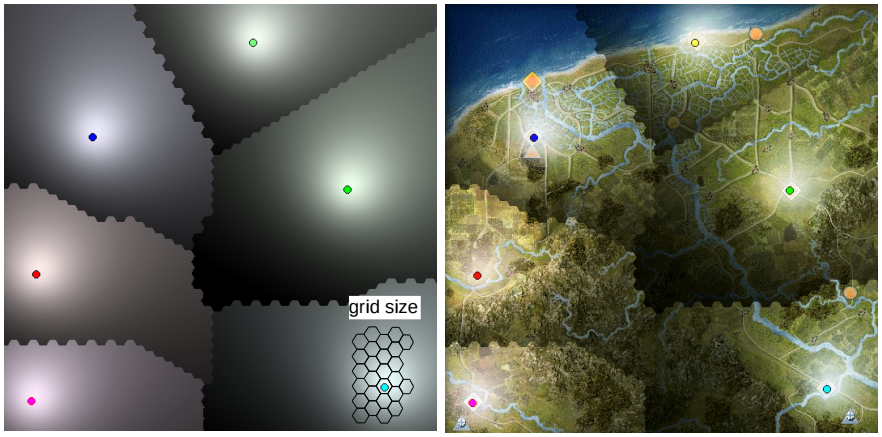


Fig. 4. Voronoi diagram used as input interface for a musical synthesizer

We present a simpler, yet very efficient pixel-based rendering approach that accepts different distance metrics and site shapes in a generic fashion for a k-order diagram.



a) **b)**

Fig. 5. a) Cushions for discrete distance metric b) Cushions layed over geographical map

4.1 Rendering first order diagrams

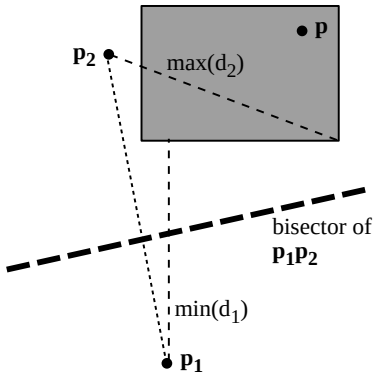
We use a recursive quadtree approach to reduce the number of sites considered per pixel. The recursion termination is based on the following observation. Consider an axis-aligned pixel rectangle to be shaded under the influence of two sites p_1 and p_2 (Fig. 6 b). Denote the minima and maxima of the distances of the sites i to the rectangle by $\min(d_i), \max(d_i)$. If $\max(d_2) < \min(d_1)$, then all pixels p in the rectangle are influenced only by the site p_2.

```
    ShadeRect(Rectangle r, Site sites[])
    {
1       for i in sites
2         compute min(d(i,r)) and max(d(i,r));

3       dlim := min(max(d(i,r))) for all i in sites;

4       if (s = {s1}) or (r is of 1 pixel size)
5           ShadeRectSingle(r,s1);
6       else if (s = {s1,s2})
7           ShadeRectDouble(r,s1,s2);
8       else   // s contains at least 3 sites
        {
9           split r in r1,r2,r3,r4;
10          ShadeRect(r1,sites); ShadeRect(r2,sites);
11          ShadeRect(r3,sites); ShadeRect(r4,sites);
        }
    }
```

a)



b)

Fig. 6. a) Recursive algorithm b) Algorithm principle

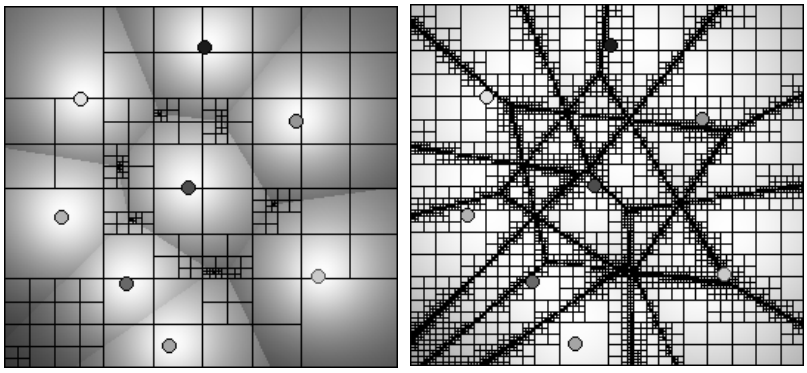
We start with the complete pixel rectangle to shade r and a list of all sites $sites$. The recursive algorithm (Fig. 6 a) proceeds as follows. First, we determine $\min(d_i)$ and $\max(d_i)$ for all sites i to the rectangle r (distance of site i to rectangle r is denoted as `d(i,r)` in Fig 6 a). Next, sites j for which $\min(d_j) > \min(\max(d_i))$ are discarded since they don't influence the rectangle, according to the previous observation. If one or two sites remain, the rectangle is scan converted by the `ShadeRectSingle`, respectively `ShadeRectDouble` optimized routines, else the rectangle is split in four and processed recursively. As a result, the recursion arrives at pixel level only near points that are at equal distances from at least three sites. For S sites, there are less than $2S - 4$ such points [5, 2]. Figure 7 a shows the subdivision for an order-1 diagram of 8 sites. To estimate the algorithm's complexity, assume S sites randomly distributed over an area of N pixels. Each of the $2S - 4$ points where subdivision reaches pixel level influences thus $\frac{N}{S}$ pixels on the average. On the other hand, a region of n pixels needs $\log_2\sqrt{n}$ steps to be subdivided until pixel level. Therefore we need $O(S\log_2\sqrt{N/S})$ `ShadeRectSingle` and `ShadeRectDouble` calls to render the whole N pixels covered by S sites. These two routines use only fixed-point (DDA-like) arithmetic for pixel-to-site distance computations. Lighting (3) is accelerated via table lookup. For each discrete gray level $\frac{I_l}{|\mathbf{n}||\mathbf{l}|}$, we store the corresponding value of d^2 in a table. As starting point for looking up shade as function of distance, the gray level of the previous pixel is used. Overall, we achieve 16 frames per second in software rendering for a 400x400 image with 20 sites on a Pentium II 350 MHz processor.

4.2 Rendering k-order diagrams, bevels, and different metrics

To render k-order diagrams, we modify the line 3 of the algorithm such that $dlim$ is the k^{th} smallest $\max(d)$ instead of the first. To render bevels of a given image-size width w, we adapt the `ShadeRectSingle` and `ShadeRectDouble` routines to compute the *Euclidian* distance from any pixel p to the closest Voronoi cell edge. This edge is found by searching for the closest pixel q where $f(q) = d(s_i, q) - d(s_j, q) = 0$, for any sites

s_i, s_j. The search is implemented independently on the user-chosen Voronoi metric d by following f's steepest descent (gradient) from the current pixel p.

An important result is that the subdivision algorithm is independent on the distance metric and site shape. The only operations needed are the distance between a site and a point and the minimum and maximum distances between a site and an axis-aligned rectangle. Overall, rendering Voronoi diagrams with circular or hyperbolic cell edges (the Apollonius, respectively Johnson-Mehl models) and diagrams for line sites (Fig. 3) is as fast as rendering the L_2 norm- and point-based diagrams.



a) **b)**

Fig. 7. Subdivision for order-1 (a) and order-2 (b) diagram rendering

For $k > 1$, rendering k-order diagrams is slower than for first-order diagrams, since we haven't implemented a k-order `ShadeRectDouble` procedure. Pixel-level subdivision calling the `ShadeRectSingle` routine occurs now on all pixels along the cell edges. Figure 7 b shows the subdivision for the 2-order diagram of the sites in Fig. 7 a. To estimate the complexity, consider first an order-1 diagram of S sites spread over N pixels. In the worst case, a site causes an edge of $n = O(\sqrt{N})$ pixels length. To subdivide such an edge to pixel level, $O(n)$ steps are needed. For S sites we perform thus at most $O(S\sqrt{N})$ `ShadeRectSingle` calls. An order-k diagram has a total edge length at most k times larger than the order-1 diagram, so its rendering complexity is $O(S\sqrt{kN})$, which is still better than the $O(kNS)$ brute-force approach. The method still performs in near real-time for 2-order diagrams on the above metioned platform.

Other pixel-level Voronoi diagram algorithms exist. A similar algorithm presented by Vleugels [7] uses a top-down rectangle subdivision based on the distances between the rectangle corners and the sites. However, this algorithm proceeds till pixel level along *all* Voronoi boundaries even for order-1 diagrams, and computes distances to *all* sites at any subdivision level. The authors reported around 5 seconds per frame for configurations similar to the ones we compute in real time. Other fast pixel-level algorithms use OpenGL hardware to compute Voronoi diagrams by rendering polygonal approximations of the distance functions [8]. Although impressive as performance, it is not clear how easy is to extend such algorithms to arbitrary distance functions, k-order diagrams, and maintain pixel-level distance computing accuracy.

5 Discussion and Future Work

We have presented a new method to visualize k-order Voronoi diagrams generated by different metrics and site shapes. A simple graphical element - a shaded, colored parabolic cushion - is used to visualize both edge and surface information. Spatial nesting is exploited in two different ways to depict the *k* levels of a k-order diagram in a single image. This visualization answers two questions in a compact way, namely: "which are the *k* sites influencing any given point?" and "which are the *k* hierarchical influence areas of a given site?". By varying the cushions' hue, height, and saturation one can intuitively emphasize the different aspects of the Voronoi diagrams. The above are best illustrated by Appendix b, where the first two levels of a 3-order Voronoi diagram are rendered with bevels whereas level 3 is rendered with cushions only. A new adaptive algorithm is presented for fast rendering Voronoi diagrams for different metrics and shape sites, allowing interactive exploration of optimal site placement sites. Using different site shapes and distance metrics involves only implementing three simple point-to-site and rectangle-to-site distance functions.

One limitation of the method is that renderings for *S*-site, k-order diagrams with higher *k* and *S* values become quickly cluttered. This limitation is inherent to the many-to-many relationship between points in the plane and sites. We plan to explore how rendering such higly-dimensional relationships can be improved by using different shading models and cushion profiles. Secondly, we plan to use this method for more complex site shapes than lines, for visualizing several mathematical abstractions related to Voronoi diagrams such as medial axes and skeletons [2].

References

1. P. AGARWAL, M. DE BERG, J. MATOUSEK, AND O. SCHWARZKOPF, *Constructing levels in arrangements and higher order Voronoi diagrams,* SIAM J. Comp., no. 27, 1998, pp. 654-667.
2. F. AURENHAMMER, *Voronoi Diagrams: A Survey of a Fundamental Geometric Data Structure,* ACM Computing Surveys, no. 23, 1991, pp. 345-405.
3. D. LEE AND C. WONG, *Voronoi diagrams in the L_1 and $L_{\inf}$ metrics with 2-dimensional storage applications,* SIAM J. Computing, no. 9, 1980, pp. 200-211.
4. A. OKABE, B. BOOTS, AND K. SUGIHARA *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams,* John Wiley & Sons, Chichester, UK, 1992.
5. M. DE BERG, M. VAN KREFELD, M. OVERMARS, O. SCHWARZKOPF, *Computational Geometry - Algorithms and Applications,* Springer, 1998.
6. M. SAKAMOTO AND M. TAKAGI, *Patterns of weighted Voronoi tessellations, Sci. Forum,* no. 3, 1988, pp. 103-111.
7. J. VLEUGELS AND M. OVERMARS *Approximating Generalized Voronoi Diagrams in Any Dimension,* Technical Report UU-CS-1995-14, Utrecht University, 1995.
8. K. HOFF, T. CULVER, J. KEYSER, M, LIN, D. MANOCHA, *Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware,* Proc. SIGGRAPH '99, ACM Press, 1999, pp. 277-286.
9. J. J. VAN WIJK, H. VAN DE WETERING, *Cushion Treemaps: Visualization of Hierarchical Information,* Proc. Information Visualization '99, IEEE Press, pp. 73-78.
10. M. BRULS, J. J. VAN WIJK, K. HUIZING, *Squarified Treemaps,* Proc. IEEE VisSym 2000, Springer, 2000, pp. 33-42
11. STRATEGIC SIMULATIONS, INC., *Panzer General 3D,* `http://www.ssionline.com/`, 1999.

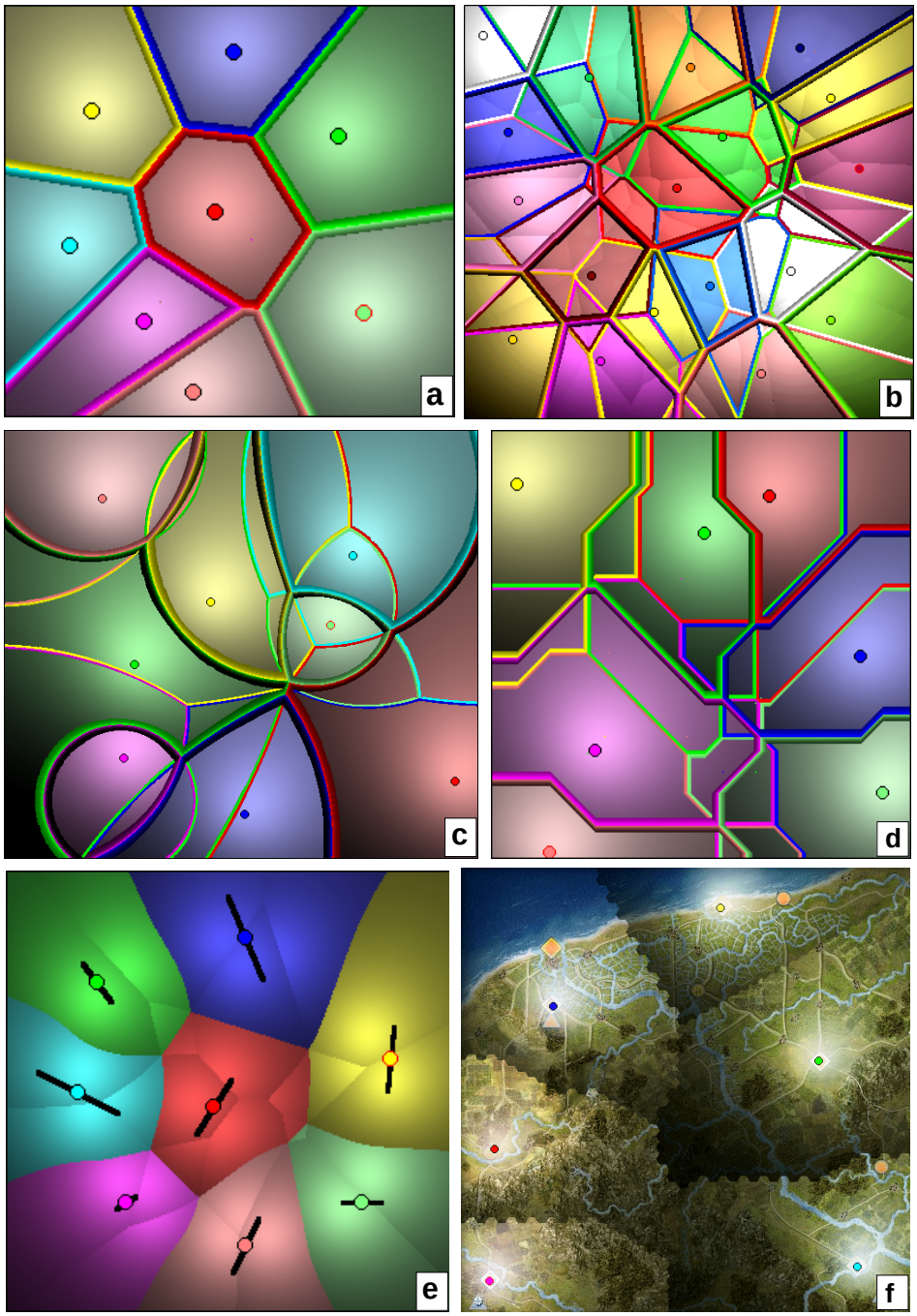


Fig. 8. Voronoi diagram renderings for Euclidian distance, order-1 and order-3 (a,b), multiplicative-weighted distance, order-2 (c), and Manhattan (d) distance order-2, line sites order-2 (e), and hexagonal grid-based discrete distance (f)