An Interactive Tour of Voronoi Diagrams on the GPU

Frank Nielsen
Sony Computer Science Laboratories, Inc.
Frank.Nielsen@acm.org

Version 3.1415, August 21st 2007.

1] Introduction

Voronoi diagrams are fundamental geometric structures that have been both deeply and widely investigated since their inception in disguise by René Descartes for analyzing the gravitational influence of stars, in the 17th century. Voronoi diagrams find countless applications in science and engineering beyond graphics as attested by the long representative, and yet non-exhaustive, list of applications reported at the web portal www.voronoi.com (eg., collision detection, path planning, meshing and surface reconstruction, etc.).

The *Voronoi diagram* of a finite set of points $S = \{P_1,....,P_n\}$ partitions the underlying space into elementary structures called *Voronoi cells* that define combinatorial proximity location information. The Voronoi cell of a site P_i is defined as the locii of points closer to site P_i than to any other site. That is mathematically, $\mathrm{Vor}(P_i) = \{P | \mathrm{Distance}(P_i, P) \leq \mathrm{Distance}(P_j, P) \forall j\}$ .The Voronoi diagram is thus an essential combinatorial structure that splits the continuous space into a finite discrete number of bounded and (necessarily) unbounded cells. The *ordinary* Voronoi diagram of a given set of sites $P_1,....,P_n$ is defined as the *cell complex* induced by the Voronoi cells $\mathrm{Vor}(P_1),....,\mathrm{Vor}(P_n)$ for the Euclidean distance [VC05]: $\mathrm{Distance}(P_{i,}, P) = \|P_i P\| = \sqrt{(P_{i,x} - P_x)^2 + (P_{i,y} - P_y)^2}$. Note that since the Voronoi diagram is defined in terms of distance comparisons, the structure does not change if we take any arbitrary monotonously increasing function of the distance, like the square function for example, that allows to bypass unnecessary square root computations.

The graphics processor unit (GPU) provides a nice commodity hardware for either visualizing these partitions interactively by rasterizing the Voronoi cells, or for computing even the exact combinatorial structures per see, as described in papers [JF06][DF06][FG06]. Rasterizing 2D Voronoi diagrams is based on computing the *index function* for each pixel position in screen space (or texture space for GPUs using per-pixel shaders). The index function reports for a current pixel/point position the index of the *closest* site :

$\text{index}(P) = \arg\min_{i\in\{1,\ldots,n\}} d(P,P_i)$. In all our Cg codes below (9 in total), we return both the smallest distance (min function) and its corresponding index (argmin function) in a float2 structure by the Cg function called Winner. We may fill Voronoi cells using predefined colors based on the index function, or raster the borders of Voronoi cells if neighborhood pixels (either in C4 or C8 connectivity) have different indices, ie., different closest points. To emphasize on the role of the distance function, we may also further rasterize distance isolines at predetermined values by prescribing beforehand a modulo offset value. This isoline drawing style emphasizes another interpretation of Voronoi diagrams as dynamic crystal growths whose seeds are anchored at sites, better called generators in that context (see Section 6). The OpenGL® GLUT program **OrdinaryVoronoiDiagram** allows to interactively pick up a site and move it so that one can explore how the overall structure changes globally from the relative point positions. Figure 1 displays different snapshots showing the variety of drawing styles rasterized in real-time by the per-pixel Cg shader. An excerpt of the OrdinaryVoronoi.cg code is given below:

```
// Report the index of the closest point
float2 Winner(float2 p)
{
int i, winner;
float dist,mindist;

mindist=distance(p,position[0]);

for(i=1;i<MAXN;i++)
        {
        dist=distance(p,position[i]);
        if (dist<mindist) {mindist=dist;winner=i;}
    }
return float2(winner,mindist);
}


// Voronoi diagram rasterization
float3   OrdinaryVoronoi(float2 pos: TEXCOORD0) : COLOR0
{
// Index for current and x- y-neighborhood position
float2 w,wx,wy;
int index, indexx,indexy;
// position of the shader pixel
float2 posx,posy;
```

```
float3 color;
float3 bordercolor=float3(0,0,0);
float3 isolinecolor=float3(0.5,0.5,0.5);
float iso, iso2, f;

pos=ToDomain(pos);posx=ToDomain(pos)+float2(s,0);posy=ToDomain(pos)+float2(0,s);

w=Winner(pos);wx=Winner(posx);wy=Winner(posy);

index=w[0];indexx=wx[0];indexy=wy[0];

// number of isoline strips
iso=30.0*frac(w[1]); // period
iso2=frac(iso);

color=ColorCell(index);

// overwrite isoline
  if (    ((iso2>t1)&&(iso2<10.0*t2)) )
                                color=isolinecolor;

// Overwrite
if ((index!=indexx)||(index!=indexy))
            return color=bordercolor;

return color;
}
```

<<
Code 1. The Cg code OrdinaryVoronoi.cg for rasterizing Voronoi cells with different styles.
>>

<< Insert icon code OrdinaryVoronoiDiagram.exe here>>

<< Insert Figure 1 here Figure1a-OrdinaryVoronoiDiagram-Border.tif Figure1b-OrdinaryVoronoiDiagram-CellBorder.tif Figure1c-OrdinaryVoronoiDiagram-IsolineBorder.tif Figure1d-OrdinaryVoronoiDiagram-isofilledborder. >>
<<Caption
Different rendering styles for the ordinary Voronoi diagram: (a) Rasterizing borders, (b) rasterizing color cells with thick borders, (c) rasterizing distance isolines with thick borders, and (d) rasterizing all three types of information at once: colored cells, distance isolines, and cell borders.
>>

Voronoi diagrams have been generalized in many ways [CVD06] by considering arbitrary objects instead of points for sites or various distance functions instead of the Euclidean distance, just to name a few. In computational geometry, there exists also important variations called k-order Voronoi diagrams that choose all subsets of k sites instead of a single site for defining the notion of ``closest/furthest'' proximity cells (many cells are empty).

In this paper, we emphasize on the educational aspects of interactive GPU applications for computing in real-time 2D Voronoi diagrams beyond the Euclidean geometry. Namely, we first give brief account and visually depict affine and curved Voronoi diagrams [CVD06]. Since there are uncountably infinitely many potential distance functions, each defining a proper Voronoi diagram in itself, it does not make really sense to catalog all of them. We rather present the reader in the latter part a neat generalization based on the axiomatization of distances. Namely, we will describe two classes of generic Voronoi diagrams based on the information-theoretic parametric distances [BVD'07]: Bregman and Csiszár divergences. Bregman and Csiszár divergences cover interestingly many familiar distances and yet intersect only for the most fundamental information measure: The Kullback-Leibler divergence, better known as the relative entropy or information discrepancy. This allows one to compute statistical Voronoi diagrams, like for example the Voronoi diagrams of a finite set of normal distributions $N(\mu_i, \sigma_i)$ encoded as 2D parameter information points (μ_i, σ_i). This reveals all the more important for computing Voronoi diagrams under uncertainty (eg., points with individual variance-based noise).

But first, let us quickly examine some fundamental properties of the ordinary Euclidean Voronoi diagrams.

2] Bisectors and dually orthogonal Delaunay triangulations

We previously defined the Voronoi cell of site P as the locii of points closer to P than to any other sites. However, that definition of Voronoi cells also implicitly highlights the notion of territory of a point P with respect to the other sites. So instead of looking up for each point its cell, we rather consider computing the cell boundaries. The notion of territory is decomposable and let intervene only the discrete finite set of input points. For a given pair of sites P and Q, we consider the boundary of their cells as an elementary territory frontier (the mere Voronoi diagram of two points) and called it the *bisector:* $\mathrm{Bisector}(P,Q) = \{X \mid \mathrm{Distance}(X,P) \leq \mathrm{Distance}(X,Q)\}$.

Since the Voronoi cell $\mathrm{Vor}(P_i) = \{P \mid \mathrm{Distance}(P_i,P) \leq \mathrm{Distance}(P_j,P) \forall P \in S\}$ can be rewritten as $\mathrm{Vor}(P_i) = \cap_{j \neq i} \mathrm{Bisector}(P_i,P_j)$. The bisector in the plane is a line (generalizes to

plane in 3D and hyperplane in higher dimensions): its characteristic derived from the distance equality $\text{Distance}(P_i, P) = \text{Distance}(P_j, P)$ yields indeed an affine equation for the locii of points at equidistance of P_i and P_j :

$\text{Bisector}(P_i, P_j) :< P, 2(P_j - P_i) > + < P_i, P_i > - < P_j, P_j >= 0,$ where

$< P, Q >= P_x Q_x + P_y Q_y$ denotes the 2D inner product (dot product).

Thus the Voronoi cell is expressed as the intersection of a set of half-planes yielding a convex polygon, also called Dirichlet cell or Thiessen polygon, eventually open to infinity for unbounded cells.

But there is more and this is obviously a key reason of their success and fame in computer graphics: Voronoi diagrams exhibit a unique dual structure called the Delaunay triangulations which enjoy nice properties (eg., no thin triangles – that is maximizes the smallest angle and minimizes the radius of the smallest enclosing disks of triangles) for points in general position (that is, no four co-circular points). The Delaunay triangulation meshing the input point set is derived from the Voronoi diagram by linking with straight line segments (ie., geodesics) the sites of adjacent cells. The boundary of the mesh is the convex hull of the point set and contains all finite Voronoi cells. Further, since the line segment joining two sites is provably perpendicular to their bisector, it follows that the Delaunay triangulation structure is globally orthogonal to the Voronoi diagram.
Observe that the intersection point of the bisector/line segment joining two sites may not belong to the Voronoi boundaries. Although the Delaunay edges can be derived from the second-order Voronoi diagram [FG06] by checking non-empty cells, we rather proceed by expanding the former ordinary Voronoi code and detect the edges yielding corresponding Delaunay edges by scanlining the frame buffer.

The program DelaunayTriangulation demonstrates that technique (Cg code DelaunayTriangulation.cg). Observe that it is not flawless as sometimes the program will fail to report edges (notably on the convex hull, if not all bounded Voronoi cells are fully rendered in the viewport) but yet is interesting for educational purposes. It can be checked that the maximum number of Delaunay edges is 3n-6. This number may varies according to the configuration of the point set, namely depending on the size of the convex hull.

<< Insert Figure 2 Figure2-DelaunayTriangulation.tif here>>
<<Caption
The dual Delaunay triangulation is orthogonal to the primal Voronoi diagram.
>>

3] **Affine and curved Voronoi diagrams**

Having lines (planes or hyperplanes in higher dimensions) as bisectors is definitively a convenient property since cells can be computed as convex polygons (polyedra and

polytopes in higher dimensions). Another classic Voronoi diagram is the power diagram of a set of disks. The power distance of a point P to a disk (C,r) centered at C with radius is defined as $\text{Power}(P;C,r) = \| P - C \|^2 - r^2$. The power distance is symmetric, and positive if and only if the point P lies outside the disk. Power diagrams generalize ordinary Voronoi diagrams (all radii set to zero) but note that some Voronoi cells of Power diagrams may be empty (and by virtue of the pigeonhole principle some cells may contain (partially) several points). Affine diagrams a;so include interestingly the Voronoi diagram for the generalized quadratic distance $d_Q(P,P_i) = \sqrt{(P-P_i)^T Q (P-P_i)}$ for a positive semi-definite matrix Q (the usual ordinary Euclidean distance is obtained by setting Q to the identity matrix), and k-order diagrams. It is therefore natural to ask whether there exists a common universal methodology for computing these affine diagrams? The striking result is that *any* affine diagram can be computed as the power diagram of a set of disks. (Thus, we can also compute the Delaunay edges from a power diagram representing the second-order point Voronoi diagram since 2nd order diagrams are affine.) Another common distance variation of Voronoi diagrams is to add or multiply the distance by a *weight* anchored at each site (these parameters can be interpreted as a time lag and a speed attribute for each generator). These generalizations yield the so-called additively and multiplicatively weighted Voronoi diagrams. In [CVD06], the most common curved and affine Voronoi diagrams are presented. They can all be computed from the following *generic* distance function $d(P,P_i) = w_i \left(\sqrt{(P-P_i)^T Q_i (P-P_i)} \right)^\alpha - r_i^\alpha$. That is, to each disk (P_i, r_i) we further attach a weight w_i and a positive semi-definite symmetric matrix Q_i (often taken as the inverse of a variance-covariance matrix). The radii of disks can potentially be imaginary, ie. negative. Equipped with that parametric distance function, we get the following diagrams explained in details in [CVD06]:

- Möbius diagrams obtained by the following distance: $\text{Moebius}(P,P_i) = w_i \| P - P_i \|^2 - r_i$. Möbius diagrams have the particularity of having their bisectors as arcs of circles. The Voronoi diagrams with arcs of circle bisectors are called spherical diagrams by analogy to affine diagrams. Similarly, there is a universality theorem for that class of Voronoi diagrams since *any* spherical diagram can be computed as a Möbius diagram too.

- Apollonius diagram of a set of spheres is defined by the following distance function: $\text{Apollonius}(P,P_i) = \| P - P_i \| - r_i$. In a sense, it is conceptually similar to power diagrams except that the Euclidean distance is not squared. The name of that diagram comes from the fact that at each vertex of the diagram there exists a circle tangent to three input circles. Computing such a circle was first raised by Apollonius and is known as Appolonius' Tenth problem. Bisectors are characterized by arcs of hyperbolae, and the

diagram is also called Johnson-Mehl diagram in the physics/chemistry literature.

- Anisotropic Voronoi diagrams are defined by the following weighted distance function $\text{Anisotropic}(P, P_i) = (P - P_i)^T Q_i (P - P_i) - r_i$. They recently gained attention in the computer graphics and computational geometry community for meshing anisotropically CAD objects with sharp ridges. The bisectors of anisotropic Voronoi diagrams are quadratic curves, and again there is a corresponding universality theorem that proves that any quadratic Voronoi diagram can be obtained from an anisotropic diagram.

Figure 3 displays these different diagrams, and the program VorPDMöbiusAppolonius allows to interact real-time with all these diagrams.

<< Insert Figure 3 here>>
<< Caption:
A gallery of affine and curved Voronoi diagrams: (a) Power diagram, (b) Apolonius diagram, (c) Möbius diagram, and (d) Anisotropic diagram. These diagrams can also be rendered using the distance isoline style.>>

<< Insert code icon here VorPDMöbiusAppolonius.exe >>

Next, we will now present and interactively visualize a few Voronoi diagrams on non-Euclidean geometries. The power diagram can be interpreted as one of those for the Laguerre geometry.

4] Spherical and hyperbolic Voronoi diagrams

For a very long time, Euclidean geometry derived from Euclid's five postulates was considered the unique geometry prevailing on Earth. Failing to prove that the more complicated fifth parallel postulate could be derived from the first four axioms, yielded eventually to one of the greatest discovery of mankind: The birth of non-Euclidean geometry in the 17th Century. The hyperbolic and spherical non-Euclidean geometries were thoroughly investigated in the 18th Century. Historically, these abstract geometries were called *imaginary* geometries. To visualize them, we need to map their structures on the Euclidean space. This is always possible by virtue of the Riemann mapping theorem.

The Voronoi diagram of a set of points on the 3D sphere, can be rasterized using the GPU per-pixel shader using the spherical coordinates on a texture map. The distance between any two points on a sphere is taken as the angle formed by any two 3D points of the unit sphere: The arccosine of the inner product of these points, namely

$$\text{Distance}_{Sphere}(P, P_i) = \arccos(P_x P_{i,x} + P_y P_{i,y} + P_z P_{i,z}) .$$

Figure 4 depicts such a Voronoi diagram rasterized on the texture map using the latitude and longitude spherical coordinates, and textured on the 3D unit sphere for visualization.

<<Insert Code icon EllipticalGeometryVoronoi.exe>>

The Cg code follows the same spirit of the ordinary Voronoi diagram except for the computations of the spherical distance:

```
//
// Convert latitude longitude to 3D xyz Cartesian coordinate
// Unit vector
float3 Spherical2Cartesian(float2 tp)
{float3 xyz;

xyz[0]=cos(tp[1])*sin(tp[0]);
xyz[1]=sin(tp[1]);
xyz[2]=cos(tp[1])*cos(tp[0]);

return xyz;
}

float norm(float3 P)
{
return P[0]*P[0]+P[1]*P[1]+P[2]*P[2];
}

float DistanceSphere(float2 tp, float2 tq)
{
float3 P, Q;
float angle;

P=Spherical2Cartesian(tp);
Q=Spherical2Cartesian(tq);
angle=acos(P[0]*Q[0]+P[1]*Q[1]+P[2]*Q[2]);

return abs(angle);
}
```

<< Insert Figure 4 here Figure4a-SphericalVoronoiTexture.tif, Figure4b-SphericalVoronoiSphere.tif and Figure4c-HyperboloicVoronoi.tif>>
<< Caption
Spherical (latitude-longitude map and textured 3D sphere) and hyperbolic Voronoi

diagrams (conformal Poincaré disk in red and non-conformal affine Beltrami-Klein disk in blue).>>

There are several realizations of the hyperbolic geometry. The two most famous ones are the conformal Poincaré disk preserving the angles, and the non-conformal Beltrami-Klein disk. The notion of conformality indicates that the mapping preserves the incidence angles, an important feature first used historically in cartography, and later considered in texture mapping of 3D meshes. For Cartesian point coordinates lying on a unit disk centered at the origin, we have the respective hyperbolic distance functions given as;

$$\text{Distance}_{Klein}(P,Q) = \text{arccosh}\left(\frac{1- < P,Q >}{\sqrt{(1- < P,P >)(1- < Q,Q >)}}\right)$$

$$\text{Distance}_{Poincare}(P,Q) = \text{arccosh}\left(1 + 2\frac{\| PQ \|^2}{(1- \| P \|^2)(1- \| Q \|^2)}\right)$$

where $\text{arccosh}(x) = \log(x + \sqrt{x^2 - 1})$ and $< P,Q >= P_x Q_x + P_y Q_y$.

<<Insert Code icon GPUHyperbolicVoronoi.exe>>

Interestingly, it can be checked that the Beltrami-Klein hyperbolic Voronoi diagram is affine and can thus be computed equivalently as a power diagram. Moreover, there exists simple one-to-one mappings for going from one hyperbolic realization to another.

In the 19th Century, Riemann further proved that there exist infinitely many abstract geometries, generalizing the elliptical and spherical geometries using the notion of Riemann metric, that can eventually further be defined locally using a tensor metric. This formalization is at the heart of the general space-time relativity theory of Einstein.

5] Information-theoretic Voronoi diagrams

Most of the common distance functions we usually meet in practice either belong to the generic class of Csiszár divergences, or to the class of Bregman divergences. These parametric distance functions are not necessarily symmetric nor do they respect the triangle inequality. Their justification is based on the characterization of least square problems as “projections” and the existence of generalized Pythagorean theorems. Details are out of scope of this paper, see [BVD’07]. Again interactive GPU rasterization allows the user to explore and gain intuition of their properties fostering visual thinking and mathematical

intuition.

The Csiszár divergence is defined for a strictly convex generator function f such that f(1)=0 as the following statistical distance:

$$I_f(P \,||\, Q) = \int P(x) f\left(\frac{Q(x)}{P(x)}\right) \mathrm{d}\, x,$$

where P and Q are probability distributions (ie., both $\int P(x)\,\mathrm{d}\, x = 1$ and $\int Q(x)\,\mathrm{d}\, x = 1$). For example, the total variation distance is obtained for $f(x) = |\, x - 1\,|$. Table 1 lists a few usual generators encountered in practice. This diagram is equivalent to the L1 norm Voronoi diagram.
Because the distance measure may not be symmetric, it is also called divergence and the $\mathrm{Distance}(P \,||\, Q)$ notation emphasizes on the non-metric property of these distance functions. For asymmetric divergences, we may further define two-types of cells depending on the position (left/right) of the the site for defining Voronoi cells:

$$\mathrm{Vor}_f\ (P_i) = \{P |\, I_f(P \,||\, P_i) \leq I_f(P \,||\, P_j) \forall j\}$$ (right-type)

$$\mathrm{Vor}_f{}^*(P_i) = \{P |\, I_f(P_i \,||\, P) \leq I_f(P_j \,||\, P) \forall j\}$$ (left-type)

We may associate to a Csiszár generator function a dual *-conjugate function $f^*(x) = x f\left(\frac{1}{x}\right)$ so that we have $I_f(P \,||\, Q) = I_{f^*}(Q \,||\, P)$. Therefore, we deduce that $\mathrm{Vor}_f{}^*(P_i) = \mathrm{Vor}_{f^*}(P_i)$. For example, the *-conjugate of the negative Shannon entropy is $f^*(x) = x f\left(\frac{1}{x}\right) = x \frac{1}{x} \log \frac{1}{x} = -\log x$, the Burg entropy (often used in sound processing).

For symmetric Csiszár divergences, the generator is self-dual and both left-type and right-type Voronoi cells coincide. Figure 5 displays some examples of Csiszár Voronoi diagrams derived from the Cg code CsiszárVoronoi.cg.

<< Insert Figure 5 here>>
<< Caption

Examples of Csiszár Voronoi diagrams: (a) Total variation distance (L1), (b) Perimeter divergence, (c) Kullback-Leibler divergence, and (d) Chi squared distance. Because the divergences may not be symmetric, we define two dual Voronoi diagrams, rasterized here in blue and red colors.
>>

Csiszár divergence	Generator function
Kullback-Leibler divergence	$x \log x$ (negative Shannon entropy)
Chi squared divergence	$\frac{1}{2}(x-1)^2$
Total variation distance	$\lvert x-1 \rvert$
Perimeter divergence	$\left\lvert \sqrt{1+x^2} - \frac{1+x}{\sqrt{2}} \right\rvert$

<< Table 1: Common examples of Csiszár divergences. >>

The other generic family of distortion measures are Bregman divergences. Bregman divergences are informally defined as the tail of a Taylor expansion for a strictly convex and differentiable function F as follows:

$$D_F(P \,||\, Q) = F(P) - F(Q) - < P - Q, \nabla F(Q) > .$$

Bregman divergences include among many others the squared Euclidean distance and the statistical Kullback-Leibler divergence. Table 2 summarizes the usual ones.

Bregman divergence	Bregman generator
Squared Euclidean distance	x^2
Kullback-Leibler divergence	$x \log x - x$ (Ext. neg. Shannon entropy)
Itakura-Saito divergence	$-\log x$ (Burg entropy)
Exponential divergence	$\exp x$

<<Table 2: Common Bregman divergences.>>

Since there are not necessarily symmetric, we define again two types of Voronoi cells:

$$\mathrm{Vor}_F(P_i) = \{P | D_F(P \,||\, P_i) \leq D_F(P \,||\, P_j) \forall j\} \quad \text{(right-type)}$$

$$\mathrm{Vor}_F^*(P_i) = \{P | D_F(P_i \,||\, P) \leq D_F(P_j \,||\, P) \forall j\} \quad \text{(left-type)}$$

Similarly, a dual divergence may be defined using the Legendre transformation that associates to a convex function a unique dual convex function as follows:

$$F^*(P') = \sup_P \{< P', P > - F(P)\}.$$

It can be shown that the supremum is reached at the unique point $P' = \nabla F(P)$.
The gradient of the primal and dual Legendre functions are inverse of each others.
Further, the primal and dual divergences are related by the following equation:

$$D_F(P \,||\, Q) = F(P) + F^*(Q') - < P, Q' >= D_{F^*}(Q' \,||\, P').$$

For example the Legendre transform of the extended Shannon entropy $x \log x - x$ is the exponential entropy $\exp x$, as it can be easily checked that their gradient functions are inverse of each other.

Again, as in the case of Csiszár divergences, the arguments swap but observe that this time the space/gradient spaces swap too. This property is at the core of the dually flat shape geometry of information geometry [BVD'07].

Figure 6 displays the snapshot of the OpenGL® GLUT program that manages simulatenously two windows to display the primal and dual Voronoi diagrams. The user can interactively pick up a point either in the primal space or dual gradient space, and observe how the structures change and are related to each other. Since the first-type Bregman Voronoi diagram is affine, it can be conveniently computed as a special power diagram.(with all non-empty cells). The second-type curved Bregman Voronoi diagram amounts after space/gradient space mapping to compute the dual affine Bregman Voronoi diagram from the Legendre convex conjugate.

<<Icon: DualVoronoiGPU.exe code>>

<< Insert Figure 6 here Figure6a-ExponentialLoss.tif and Figure6b-DualShannon.tif>>
<< Caption:
Primal and dual Bregman Voronoi diagrams for the exponential and Shannon entropies.>>

6] Voronoi diagrams as minimization diagrams

As mentioned in the introduction, the Voronoi diagram can be computed from the minimization diagram of a set of distance functions anchored at sites using the index function.

Namely, let $D_i(P) = \text{Distance}(P, P_i)$ be the function attached to the site P_i. The Voronoi diagram can be obtained from the minimization diagram $D(P) = \min_{i \in \{1,\ldots,n\}} D_i(P)$, the lower envelope of the functions. For the (squared) Euclidean distance function, this amounts to consider the lower envelope of a set of paraboloids anchored at each respective site, as shown in Figure 7.
For the affine Bregman Voronoi diagrams, we have $D_i(P) = D_F(P \,||\, P_i) = F(P) - F(P_i) - < P - P_i, \nabla F(P_i) >$. We can remove the common terms $F(P)$ appearing in all distance functions and get equivalently a set of planes (or hyperplanes in higher dimensions) thus showing that the right-type Bregman Voronoi diagram is indeed affine.

<< Insert Figure 7 here: Figure7-MinimizationDiagrams.tif>>
<< Caption:
Voronoi diagrams as minimization diagrams: Visualizing lower envelopes. Here for the case of ordinary Voronoi diagrams, we visualize the lower envelope of corresponding anchored parabolae shifted upward to improve visibility.
>>

<< Code icon: VoronoiByMinimizationDiagram.exe>>

7] Conclusion

Interactive visualization of Voronoi diagrams for various symmetric/non-symmetric distance functions allows one to gain better understanding of their fundamental properties and intrinsic dualities. We encourage the reader to look at the set of 20 videos provided on the accompanying DVD that recorded interactive sessions. As such, the GPU provides a fabulous tool and framework to gain intuition for discovering mathematical structural invariants. The GPU, a powerful visualscope, allows one to foster ``visual thinking'' and thus potentially hint at further major mathematical discoveries in the future. We refer the reader to the paper [BVD'07] for further theoretical insights of information-theoretic diagrams including a neat extension of the space of spheres from which many computational geometric algorithms such as the smallest enclosing balls rely on.

<< Icon DVD videos here >>

References

[VC05] Frank Nielsen: Visual Computing: Geometry, Graphics and Vision. Charles River Media, ISBN: 1-58450-427-7, 2005.

[CVD06] Jean-Daniel Boissonnat, Camille Wormser, Mariette Yvinec: Curved Voronoi Diagrams. In Effective Computational Geometry for Curves and Surfaces. Springer-Verlag 2006.

[JF06] Guodong Rong, Tiow-Seng Tan: Jump flooding in GPU with applications to Voronoi diagram and distance transform. ACM Symposium on Interactive 3D graphics and games, 109-116, 2006.

[DF06] Avneesh Sud, Naga K. Govindaraju, Russell Gayle, Dinesh Manocha: Interactive 3D distance field computation using linear factorization. ACM Symposium on Interactive 3D graphics and games, 117-124. 2006.

[FG06] Ian Fischer and Craig Gotsman, Fast Approximation of High-Order Voronoi Diagrams and Distance Transforms on the GPU, Journal of graphics tools, volume 11, number 4, pages 39-60, 2006. http://jgt.akpeters.com/papers/FischerGotsman06/

[BVD07] Frank Nielsen, Jean-Daniel Boissonnat and Richard Nock, Bregman Voronoi Diagrams: Properties, Algorithms and Applications, INRIA Research Report No 6154. (online at hal.inria.fr), 2007.