# FINDING THE nD VORONOI DIAGRAM BY GEOMETRIC ALGEBRA MEANS

M. D. ZAHARIA*

*This paper describes two methods for solving the nD Voronoi problem. Both methods use geometric algebra to accomplish the basic geometric computations. First method does the computations directly in the n dimensional soace; it finds a Voronoi cell through repeated clipping operations of a bounding hypercube of the sites collection. The second method determines the Voronoi diagram following a techique proposed by Edelsbrunner which finds the diagram by projecting the upperhull of a hyperplane arrangement from a n+1 dimensional data space. This last method does the computations in a n+2 dimensional space using $G_{n+2}$ geometric algebra and the homogeneous model of $R^{n+1}$.*

*Lucrarea prezintă două modalităţi de rezolvare a problemei determinării unei diagrame Voronoi în spaţiul n-dimensional. Ambele metode utilizează algebra geometrică pentru efectuarea calculelor de natură geometrică. Prima metodă determină o celulă a diagramei Voronoi prin decupări repetate ale unui paralelipiped încadrator al mulţimii de puncte caracteristice ale diagramei. Metoda a doua este bazată pe tehnica propusă de Edelsbrunner care determină diagrama Voronoi ca proiecţia unei înfăşurători a unui aranjament de hiperplane dintr-un spaţiu n+1 dimensional. Această ultimă metodă efectuează calculele într-un spaţiu n+2 dimensional utilizând algebra $G_{n+2}$ şi modelul omogen al spaţiului $R^{n+1}$.*

**Key Words**: computational geometry, geometric algebra, Voronoi diagram, homogeneous model, flag

## Introduction

Given a collection of m characteristic data points in a n-dimensional Euclidean data space ($E^n$): $S=\{S_i|\ 0\le i<m,\ S_i\in E^n\}$ and considering an arbitrary point of this collection ($S_k$), the *Voronoï cell* associated to $S_k$ represents the locus of points from $E^n$ that are placed closer to $S_k$ than to any other point from S. i.e.

$$VCell(S_k)=\{x\in E^n \mid d_E(x, S_k)\le d_E(x, S_j),\ \forall\ 0\le j<m,\ j\ne k\},$$

where $d_E:E^n\times E^n\rightarrow R^+$ is the Euclidean distance.

* Associate Professor, Computer Science and Engineering Department, University POLITEHNICA Bucharest, Romania

The *Voronoï diagram* associated to S is formed by m Voronoï cells corresponding to every point from S. The points $S_k$ ($0 \le k < m$) are called the *sites* of the diagram.
Each Voronoï cell is a convex set; it results as the intersection of some infinite half-spaces (themselves convex regions) bounded by perpendicular bisector hyperplanes associated to some properly selected site pairs. Indeed, denoting by $\Pi_{ij}=\{x \in E^n \mid d_E(x, S_i) \le d_E(x, S_j)\}$, the half-space delimited by the hyperplane perpendicular bisector between $S_i$ and $S_j$, (i.e. the hyperplane $\{x \in E^n \mid d_E(x, S_i)=d_E(x, S_j)\}$), and containing the points placed on the same side as $S_i$ with respect to the above mentioned hyperplane; we may write:

$$VCell(S_k) = \bigcap_{\substack{i=0 \\ i \neq k}}^{m-1} \Pi_{ki} \qquad (1)$$

Obviously, some Voronoï cells are not finite and, to avoid the separate treatment of these cases, most algorithms start the computations from a bounding polytope associated to the site collection. Usually this bounding polytope is a hypercube whose edges are parallel with the coordinate axes of the data space. The simplest algorithm based on (1) that determines the Voronoi diagram of a given point collection S is:

```
for * p in S do
  *initialize VCell(p) with the bounding hypercube of S
  // the algorithm will produce only finite sized Voronoi cells
  for *q in S and q≠p do
    VCell(p) ← ClipandIntersect(VCell(p), Π_pq)
```

**Algorithm 1 – determines a Voronoi cell by successive clipping operations of a hypercube shaped bounding box**

This is an $O(\|S\|^2)$ algorithm but, the elementary operation ClipandIntersect() corresponds to the intersection of a n-dimensional polytope with the half-space $\Pi_{pq}$ and this has in the worst case an exponential behavior with respect to the dimensionality of the embedding space. The intersection of a collection of hyperplanes in a n-dimensional data space induces a spatial subdivision containing faces of different dimensions. This forms the so-called *arrangement* induced by the hyperplanes collection. As it was previously noticed, the combinatorial complexity of an arrangement (i.e. its total number of vertices, edges or faces included in subspaces of dimension greater than 1) is, in the worst case, exponential.

*Note*: The algorithm 1 has an intrinsic parallelism, the Voronoi cells are independently computed and this may eventually be done on independent processors.
Other more efficient algorithms that solve the same problem were presented in the computational geometry literature ([1], [2], [3]). One of the most popular is the algorithm invented by Fortune ([4]). It is a O(mlog(m)) algorithm ($m=\|S\|$), based on the sweep-line paradigm. Another bright idea to determine the Voronoï diagram was given by Edelsbrunner and Seidel ([5]). They emphasized the link between a Voronoï diagram in a n-dimensional space and an upper-envelope of a collection of hyperplanes from a n+1-dimensional space. In fact, projecting the above-mentioned envelope onto the initial n-dimensional data space produces the Voronoï diagram we are looking for.
Let us denote by $S_i \in E^n$, $0 \le i < m$, the initially given sites. Edelsbrunner's method does the following steps:

1. Extend the initial data space to $E^{n+1}$ (subsequently called Edelsbrunner's space) and denote by $\mathbf{e}_\infty$ (orthogonal to $E^n$) the unit vector characterizing the new dimension
2. Build the paraboloid (Par) specified explicitly by
$$x_n = \sum_0^{n-1} x_i^2 \qquad (2)$$
in $E^{n+1}$
3. For every point $Q_i \in E^{n+1}$ obtained by intersecting (Par) with a line passing through $S_i$ and parallel to $\mathbf{e}_\infty$ ($Q_i$ has therefore the coordinates $(s_{i0}, s_{i1}, \ldots, s_{in-1}, \sum_{j=0}^{n-1} s_{ij})$ ) construct the plane ($\tau_i$) tangent to the paraboloid in $Q_i$.
4. The upper-envelope of planes ($\tau_i$) $0 \le i < m$, projected on $E^n$ is the Voronoi diagram of sites $S_i$

**Algorithm 2 -- Edelsbrunner's method to solve the nD Voronoi problem**

The abstract operations necessary to follow Edelsbrunner's recipe find particularly simple expressions using the formalism of geometric algebra and $G_{n+2,0}$ model of the Euclidean space. This corresponds in fact to the homogeneous model of Edelsbrunner's space. The two extra dimensions, (relative to the Euclidean space $E^n$) whose associated basis vectors will be denoted $\mathbf{e}_\infty$ and $\mathbf{e}_0$ serve to encode the metric of the Euclidean space $E^n$ and respectively to represent projectively (removing therefore the origin singularity) Edelsbrunner's data space. The metric of the Euclidean space $R^n$ is encoded in Edelsbrunner's space through a parabolic hypersurface represented in its explicit form . This paper

emphasizes the strong relationship between the type of solution proposed by Edelsbrunner to the nD Voronoi problem and the reasoning capabilities (forms to structure the thinking) offered by the geometric algebra language. Section 1 describes briefly some useful notions of geometric algebra and one important model of the Euclidean space, the homogeneous model. Section 2 specifies a first type of geometric algebra based solution for the nD Voronoi problem; here the reasoning is accomplished in the lower dimensional Euclidean space $E^n$. Section 3 gives another type of geometric algebra based solution. This follows the Edelbrunner's method and the reasoning is accomplished in the higher dimension space $R^{n+2}$, using the geometric algebra $G_{n+2}$. This solution uses the concept of *flag* that is a hierarchy of subspaces and is represented by a special class of multivectors from $G_{n+2}$. The most significant algebraic derivations and details of concrete implementation are also provided.

## 1. Basic Notions of Geometric Algebra

Geometric Algebra is a formalism capable to provide a unified framework to solve geometric problems with broad areas of applicability as: robotics, astronomy, computer graphics or quantum mechanics.

Geometric algebra is in fact a Clifford algebra (i.e. an algebra generated by the real scalars, the elements of a vector space $V^n$ and having an associated metric) whose main computational elements have appropriate geometric semantic.

Given a n-dimensional vector space $V^n$ over the field R, its associated geometric algebra $G_n$ is generated by defining the *geometric product*. This is an associative and distributive product (when operating on vector operands) and supplementary satisfies the conditions:

$\lambda\mathbf{a}=\mathbf{a}\lambda$, $\forall\lambda\in R$, $\mathbf{a}\in V^n$

$\mathbf{a}^2=\varepsilon_a|\mathbf{a}|^2$, $\forall\mathbf{a}\in V^n$ where $\varepsilon_a\in\{-1, 0, 1\}$ is the signature of $\mathbf{a}$ and $|\mathbf{a}|\in R^+$ is the magnitude of the vector $\mathbf{a}$, determined by the metric.

If the vector space $V^n$ has subspaces with basis vectors having positive, negative and null signatures, we denote by p, q and r the dimensions of the largest dimensionality subspaces of $V^n$ having respectively (basis vectors with) positive, negative and null signatures (p+q+r=n). In this case, the vector space $V^n$ will be denoted $V^{p,q,r}$ and its associated geometric algebra is $G_{p,q,r}$. An analogous notation $G_{p,q}$ is used when the basis contains only p vectors of positive signature and q vectors of negative signature.

Geometric algebra has algebraic operators that are interpretable as geometric operations like spanning or projection of subspaces. Apart the geometric product, important products characteristic to a geometric algebra environment (see [6]) are:

- the *outer product* (denoted ^) a grade increasing, linear operator related to the spanning concept,
- the *inner product* (denoted •) a grade decreasing operator related to the orthogonality, metric and projection concepts

Another operators as meet, join and contractions (denoted respectively $\cap$, $\cup$ and $\rfloor$, $\lfloor$) have useful associated geometric semantics; for example the meet of two subspaces $\mathbf{A}\cap\mathbf{B}$ denotes the largest possible (in dimension) common subspace of **A** and **B**. The algebraic manipulation rules are detailed in books like [7] and will not be presented here.
The main computational elements of Clifford algebra are the *multivectors*. These are non-homogeneous elements containing components of different grades. For example, in $G_3$ geometric algebra a "complete" multivector could, have one scalar, one vector, one bivector (specifies a plane component) and one trivector (specifies a volume element) component. These are the homogeneous components of a $G_3$ multivector.
The totality of multivectors from $G_n$ form together a $2^n$ dimensional vector space denoted $\mathcal{G}(V^n)$. In general all the r-grade homogeneous multivectors of $G_n$ form together a $\binom{n}{r}$ dimensional subspace.
A basic computational element of geometric algebra is the *blade*. A blade of grade r is a homogeneous multivector that can be written as an outer-product of r vectors (or as a geometric product of r mutually orthogonal vectors)
$\mathbf{A_r}=\mathbf{v_1}\wedge\mathbf{v_2}\wedge\ldots\wedge\mathbf{v_r}=\mathbf{a_1}\mathbf{a_2}\ldots\mathbf{a_r}$, where $\mathbf{a_i}\bullet\mathbf{a_j}=0$, $\forall 1\le i,j\le r$, $i\ne j$
The blades represent subspaces. Indeed, the blade $\mathbf{A_r}$ from $G_n$ can be associated to the subspace $A=\{\mathbf{x}\in R^n|\mathbf{x}\wedge\mathbf{A_r}=0\}$. An interesting property of a blade is that its inverse represents the same unoriented subspace (see [8]). Given a non-zero blade **A** representing a certain subspace $A$ the orthogonal projection of an arbitrary vector **x** onto the subspace $A$ is computed by $P_{\mathbf{A}}(\mathbf{x})=(\mathbf{x}\rfloor\mathbf{A})\mathbf{A}^{-1}$ were the left contraction was used as inner-product due to its superior computational properties as was stated in [9].
All the geometric algebra specific computations necessary to solve the Voronoi problem were accomplished using the GAP package ([6]).

### 1.1 The homogeneous model of Euclidean space

The classical geometric algebra based model of the Euclidean space, that is mainly based on the geometric semantic of algebraic operators, has an important shortcoming. The separate treatment required by the origin

comparatively to the other points of the space. The origin corresponds to a 0-blade of an arbitrary grade. This deficiency was resolved by introducing the homogeneous model of the Euclidean space.

If the basic Euclidean space (denoted $E^n$) is n-dimensional, then the corresponding homogeneous space (denoted $H^{n+1}$) is n+1 dimensional and any line passing through the origin of $H^{n+1}$ projects to a line of $E^n$. Figure 1 shows the case when n=2, in that case the homogeneous space is the 3D space. Denoting the Euclidean vectors by attached arrows and the homogeneous vectors with bold face characters, from the figure 1 we can deduce that:

$$\vec{p} = \frac{\mathbf{p} \wedge \mathbf{e}_0}{\mathbf{p} \bullet \mathbf{e}_0} \mathbf{e}_0^{-1} = \frac{\mathbf{p} \wedge \mathbf{e}_0}{\mathbf{p} \bullet \mathbf{e}_0} \mathbf{e}_0 = -\mathbf{e}_0 + \frac{\mathbf{p}}{\mathbf{p} \bullet \mathbf{e}_0}$$

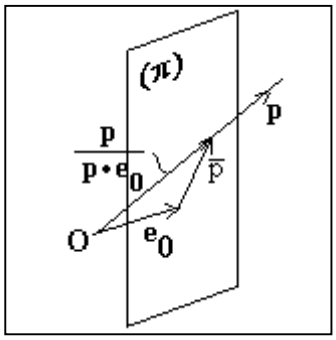


Figure 1 -- The projective mapping specific to a 3D homogeneous space

In the general case when $E^n$ ($R^n$) is a n-dimensional space, the homogeneous model establish a mapping between $R^{n+1}$ and the vectors and scalars from $G_n$ as follows:

- All the points (labeled by vectors **p**) from a line passing through a homogeneous origin, are considered forming an equivalence class whose representative element is labeled by the homogeneous vector $\frac{\mathbf{p}}{\mathbf{p} \bullet \mathbf{e}_0}$ this representative element is called the normalized **p**.
- Every vector $\mathbf{x} \in R^{n+1}$ has a corresponding vector from $R^n$, which is the rejection with respect to $\mathbf{e_0}$ of the normalized **x**. This is the projective map characteristic to the homogeneous model: $\vec{x} = \frac{\mathbf{x} \wedge \mathbf{e}_0}{\mathbf{x} \bullet \mathbf{e}_0} \mathbf{e}_0 \Leftrightarrow \vec{x} = \frac{\mathbf{x}}{\mathbf{x} \bullet \mathbf{e}_0} - \mathbf{e}_0$
- The points from the ray determined by **x** are all equivalent homogeneous representations of the Euclidean point labeled $\vec{x}$ where $\vec{x} = \frac{\mathbf{x} \wedge \mathbf{e}_0}{\mathbf{x} \bullet \mathbf{e}_0} \mathbf{e}_0$

Practically, the rejection of normalized vectors $\{\frac{\mathbf{x}}{\mathbf{x} \bullet \mathbf{e_0}} \mid \mathbf{x} \in \mathbf{R}^{\mathbf{n+1}}\}$ with respect to $\mathbf{e_0}$ form a space dual to $\mathbf{e_0}$. Therefore the homogeneous representation of an Euclidean space $R^n$ is obtained by embedding it into an n+1 homogeneous space, fixing the origin of $R^n$ with the unit vector $\mathbf{e_0}$, and considering $R^n$ as a hyperplane of $R^{n+1}$, normal to $\mathbf{e_0}$. This hyperplane is the translation by vector $\mathbf{e_0}$ of the blade dual to $\mathbf{e_0}$ from the homogeneous space.
*Note*: A Euclidean line that passes through two points (labeled by two Euclidean vectors $\vec{p}$ and $\vec{q}$) is represented projectively by the homogeneous bivector $\mathbf{p}\wedge\mathbf{q}$.

## 2. First type of geometric algebra based solution

In this type of solution the reasoning is made directly in the Euclidean space; it uses geometric algebra to specify (implement) the necessary abstract operations directly in the Euclidean space in which it is desired to determine the Voronoï diagram.

For example, if we use the simplest method to determine a Voronoï cell associated to a given site i.e. the technique given in Algorithm 1, the Voronoï cell will be represented by a node of a multiway tree. Every cell embedded in a space of dimension g has an associated list of pointers to its subcells (or faces) embedded in g-1 dimensional spaces. The cells of grade 1 (i.e. edges) are represented by two finite points. Finally a NULL pointer represents a void cell.

```
typedef struct VC {
  int grad; // grade of the cell embedding space
  union{
    LIST_VCELL lcels;
    // if (grad>1) list of pointers to subcells
    HMV pextr[2];  // if(grad==1) the cell is a finite segment encoded by
    // the two conformal representants of its extremities
    int tk[2]; //array used in construction of a normalized hypercube
  } subcell;
} *VCELL;
```

A Voronoï diagram is an array of cells and their associated sites:

```
typedef struct dv{  // Voronoi diagram
  HMV site;      // conformal representant of the cell site
```

```
  VCELL cell;  // the convex polytope corresponding to a Voronoi cell
} *D_VORONOI;
```

The type LIST_VCELL is simply a generic single linked list:

```
typedef struct lvc{
  void *cell;
  struct lvc *urm;
} *LIST_VCELL;
```

This solution used a n-dimensional hypercube bounding box having its hyperfaces embedded in the n-1 elementary hyperplanes (wedge products of (n-1) basis vectors).
A dictionary (actually implemented as an AVL tree) is used to store only once every point (cell vertex) together with its number of occurrences in the entire Voronoï diagram.
The algorithm implementing the clipping of the Voronoi cell by a hyperplane, (ClipCell() abstract operation from Algorithm 1) is essentially a Depth First Traversal of the tree encoding the cell. A backtracking algorithmic pattern is followed.

```
VCELL ClipCell(VCELL c, HMV cs, HMV os, void **rlid)
  // c=cell to be clipped by the hyperplane perpendicular bisector
  // between cs and os (where cs=cell site, os=other site)
  // rlid=a cell of c->grade-1 reconstructed  from c->grade-2 cells
  // resulted by clipping the sons of c with the (cs, os) hyperplane
  // (recursive return path)
  *treat the NULL cell case
  *treat the 1-grade cell case
   for *each subcell sc of c
     if *BoundingCondition holds then
       *clip and update the subcell i.e. sc←ClipCell(sc, cs, os, l);

     *add the non-null lid "l" to a list of cells "list_lid"
  endfor
  *build a new cell (pointed by rlid) of grade c->grade-1 through assembling the
    cells from "list_lid" and add rlid in the kist of  subcells of c
  return c;
end
```

The bounding condition can be implemented by verifying that the bounding box of sc intersects the hyperplane determined as perpendicular bisector of cs and os (and denoted further PB(cs, os)). If the intersection between c and the plane PB(cs, os) is non empty, this new c->grade-1 convex polytope is added to the list of subcells of c. This polytope is built assembling the c->grade-2 polytopes resulted by intersecting the subcells of c with PB(cs, os). This recombination process is done during the return from recursivity path of function ClipCell().

## 3. Second type of geometric algebra based solution

This type of solution uses Edelsbrunner's idea (see algorithm 2) and does all the computations in the higher dimensionality space. After computing the edges of the arrangement of hyperplanes tangent to the paraboloid ($Par$), the result is projected onto the Euclidean space were we want to determine the Voronoi diagram. In this section a n-dimensional Euclidean space will be considered and the reasoning will be done using the $G_{n+2,0}$ geometric algebra. The Euclidean vectors are denoted using attached arrows, the vectors from the homogeneous space $R^{n+2}$ will be denoted using bold face characters. The case from Figure 2 corresponds to n=2.

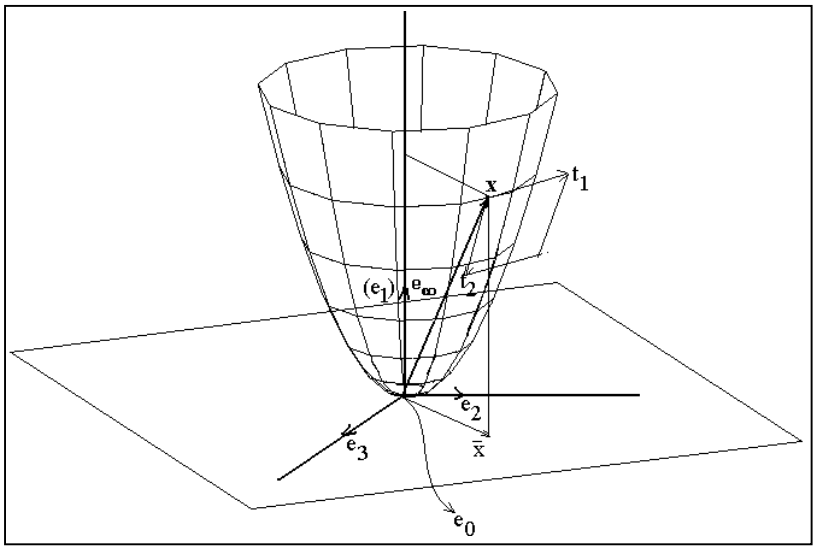


Figure 2 – The principle of Edelsbrunner's method

Be $\vec{x}$ the Euclidean vector labeling an arbitrary site of the diagram. First of all let us find the blade corresponding to the tangent space of the paraboloid in the point **x**. The computation is restrained to the n+1-dimensional space obtained by extending the Euclidean space with a new (orthogonal) dimension associated to $\mathbf{e}_\infty$. The subspace with the same orientation as the tangent to the paraboloid in the point **x** (where $\mathbf{x}=\vec{x}+\vec{x}^2\mathbf{e}_\infty$) has two components:

$\mathbf{t_1}$, the component in the radial space (Euclidean space):

$$\mathbf{t_1}=\vec{\mathrm{x}}^*=\vec{\mathrm{x}}\bullet\mathbf{I_E}^{-1}=\vec{\mathrm{x}}\bullet\tilde{\mathbf{I}}_\mathbf{E}=(-1)^{\frac{n(n-1)}{2}}\vec{\mathrm{x}}\bullet\mathbf{I_E}$$

where $\mathbf{I_E}$ denotes the Euclidean pseudoscalar, n the dimension of the Euclidean space and ~ is the *reverse* operator.

$\mathbf{t_2}$ is the tangent component in the axial space (i.e. $\vec{x}\wedge\mathbf{e}_\infty$)

$$\mathbf{t}_2=\lim_{x1\to x}\frac{\mathbf{x1}-\mathbf{x}}{|\mathbf{x1}-\mathbf{x}|} \tag{3}$$

But $\mathbf{x1}-\mathbf{x}=(\vec{x1}-\vec{x})+(\vec{x1}^2-\vec{x}^2)\mathbf{e}_\infty$ and, under the assumption above, $\vec{x1}-\vec{x}=\beta\vec{x}$ where $\beta\in\mathrm{R}$. The limit (3) becomes then:

$$\mathbf{t}_2=\lim_{\beta\to0}\frac{\beta\vec{x}+\beta(\beta+2)\vec{x}^2\mathbf{e}_\infty}{\sqrt{\beta^2\vec{x}^2+\beta^2(\beta+2)^2\vec{x}^4\mathbf{e}_\infty^2}}=\frac{\vec{x}+2\vec{x}^2\mathbf{e}_\infty}{|\vec{\mathbf{x}}|\sqrt{1+4\vec{x}^2}}$$

let us denote $\mathbf{HT}=\mathbf{t_1}\wedge\mathbf{t_2}$. It is a n-dimensional subspace of $\mathrm{R}^{n+1}$ that has the same orientation as the tangent in **x** to the Edelsbrunner paraboloid. In order to express this tangent (in an arbitrary point) as a blade, we must add a new dimension to the space following the construction method specific to the homogeneous model. Let $\mathbf{e_0}$ be the unit basis vector associated to this new dimension. In this n+2-dimensional space (using its related geometric algebra $G_{n+2,0}$) we obtain the blade specifying the tangent in **x** as:

$$\mathbf{T}=\mathbf{x}\wedge\mathbf{t}_1\wedge\mathbf{t}_2=(-1)^{\frac{n(n-1)}{2}}(\vec{\mathrm{x}}+\mathbf{e_0}+\vec{\mathrm{x}}^2\mathbf{e}_\infty)\wedge(\vec{\mathrm{x}}\bullet\mathrm{I_E})\wedge\frac{\vec{\mathrm{x}}+2\vec{\mathrm{x}}^2\mathbf{e}_\infty}{|\vec{\mathbf{x}}|\sqrt{1+4\vec{\mathrm{x}}^2}} \tag{4}$$

where **x** is the $\mathrm{R}^{n+2}$ vector labeling P.

The implementation of this type of solution uses the concept of *flag*. A flag is a hierarchy of half-spaces having at its lowest level a pair of points (see [13]**)**. These points can be considered as "finite" or "with infinite extension" and consequently, the line determined by this pair of points can be infinite, half-infinite or a simple line segment (both extremities are finite points). Using geometric algebra, the flags are represented by multivectors; they are a special class of multivectors whose homogeneous components are blades and have the

lowest grade homogeneous component of grade greater or equal than 2 (in $G_{n+2,0}$). If we denote the flag
$\mathbf{F}=<\mathbf{F}>_{n+2} + <\mathbf{F}>_{n+1} + \ldots + <\mathbf{F}>_{p}$, $p \geq 2$
our application will maintain the meet: $<\mathbf{F}>_{n+2} \cap <\mathbf{F}>_{n+1} \cap \ldots \cap <\mathbf{F}>_{p}$.
The representation of a flag is given by:

```
typedef struct { //Double Point
  HMV p0, p1;
  int pinf0, pinf1; // have value 1 if the associated point is an infinity point and 0 otherwise
} DPOINT;
typedef struct fl{
  MULTIVECTOR ssh; //subspace hierarchy
  DPOINT edge; // the point pair corresponding to the bivector (lowest grade) component of ssh
} *FLAG;
```

For example, if the dimension of the Euclidean space is n=2, following the model from Algorithm 1 (detailed in section 2) where the Voronoi cell is initialized with the hypercube bounding box., F will be initialized with the $\mathbf{I}_{n+2}$ pseudoscalar $\mathbf{F} \leftarrow \mathbf{I}_{n+2}$. The "intersection" between **F** and one tangent hyperplane $\mathbf{T}_i$ (its form was specified by (4)) is defined as the sum between **F** and the homogeneous component LowestGrade(**F**) $\cap \mathbf{T}_i$, this definition is valid if LowestGrade(**F**)>2. In this case, when the resulted meet reaches for the first time the grade 2, (the meet is a bivector denoted further **B**), the coresponding line (resulted through the meet) induces a pair of infinity points resulting through the factorization of bivector **B**. (the description of the factorization algorithm used here is given in [6]). If LowestGrade(**F**) equals 2 (i.e. the lowest grade component can be factorized as a pair of points) the intersection process can change the characteristics of the points (from infinite to finite) and their position. The function CutFlag(), presented below, implements the operation of intersection between a flag and a hyperplane (that is tangent to Edelsbrunner's paraboloid).

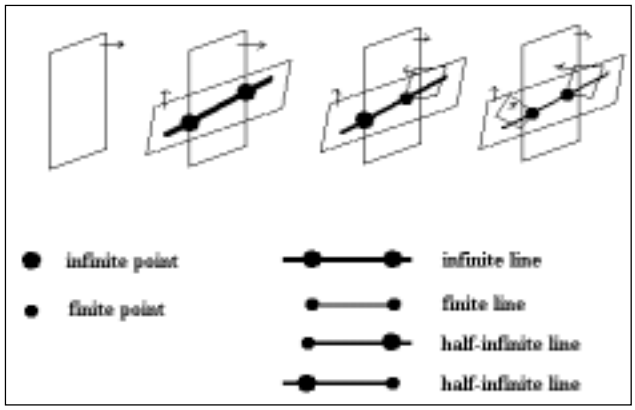


**Figure 3 – the outline of CutFlag() operation, in a 3D space**

```
function CutFlag(FLAG f, HMV c) returns FLAG
  * treat the case when "f is a NULL flag"
  flg←*GetLowestGrade(f->ssh);
  if(flg->grade==2) then
   * treat the case of a intersection between a homogeneous line and hyperplane c.
  else begin
    t←meet(flg, c)
    if(t->grade==2) {
      * factorize t as two vectors v0 and v1
      f->edge.p0←v0; f->edge.pinf0=0;
      f->edge.p1←v1; f->edge.pinf1=0;
    }
  * add homogeneous component t to flag f->ssh
end
```

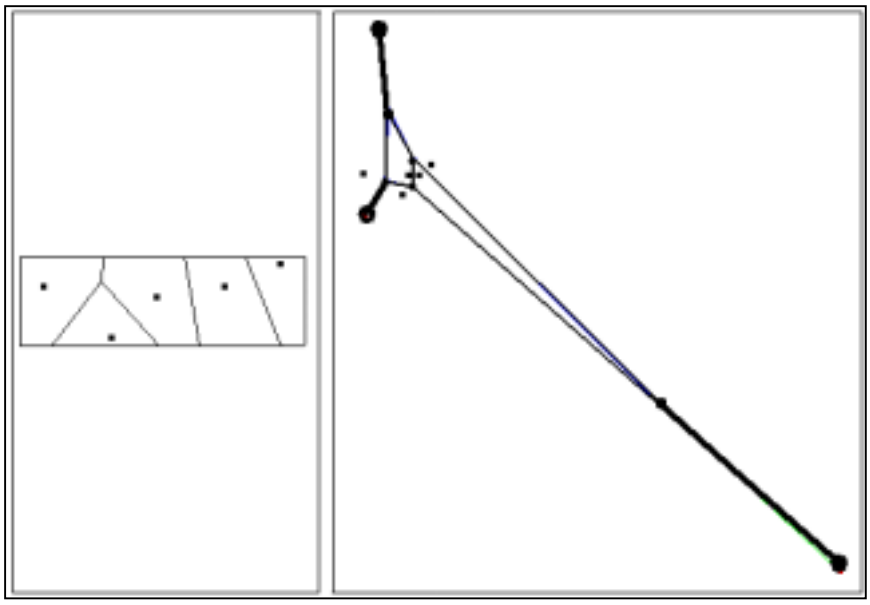

**Figure 2 – a 2D Voronoi diagram determined by the two methods mentioned above**

The enumeration procedure that gives the ordered combination of the tangent hyperplanes that are intersecting the current flag follows a backtracking programming paradigm. (see [14]).

## Conclusions

The language of geometric algebra helps the designer of an application to think in higher dimensionality spaces. It emphasizes at the language level the elementary constructive elements that are the subspaces (blades). The hierarchies of subspaces, corresponding to the mathematical concept of flag, can model any linear subdivision. Obviously, any application can be solved through "conventional" methods. Using geometric algebra does not improve the lower bounds of the algorithms, it simply help to develop a new way of thinking structured in subspaces. The hierarchy of subspaces composing a flag can be used to improve performances of geometric search/intersection processes. A standard way to solve a problem by geometric algebra is to use the fundamental idea: "think in higher dimensionality spaces and then project to obtain the desired results in lower dimensionality spaces". This fundamental idea that stays behind Edelsbrunner's method has been previously used to proof major results in different fields of research.

## Acknowledgements

My grateful acknowledgements to Dr. Leo Dorst, the discussions with him during the postdoctoral stage I had at the University of Amsterdam, were of great help to draw up this paper.

## R E F E R E N C E S

1. *F. Preparata, M. Shamos*, *Computational Geometry: An Introduction,* Springer Verlag, New York, 1988.
2. *J. O'Rourke*, Computational Geometry in C, ISBN 0-521-64010-5, Cambridge University Press, 1999.
3. *M. de Berg, M. van Krefeld, M. Overmars, O. Schwarzkopf*, Computational Geometry Algorithms and Applications, ISBN 3-540-65620-0, Springer Verlag, Berlin-Heidelbergm 2000.
4. *S. Fortune,* A sweepline algorithm for Voronoi diagrams, Algorithmica **2**, pp. 153-174, 1987.
5. *H. Edelsbrunner, R. Seidel,* Voronoi diagrams and arrangements, Discrete Computational Geometry **1**, pp. 25-44, 1986.
6. *M. D. Zaharia, L. Dorst,* Interface Specification and Implementation Internals of a Program Module for Geometric Algebra, submitted to publication to: The Journal of Logic and Algebraic Programming (An Elsevier Publication), February, 2003. available at http://www.science.uva.nl/research/ias.
7. *D. Hestenes, G. Sobczyk,* Clifford Algebra to Geometric Calculus, D. Reidel Publishing Company, Dordrecht, Holland, 1984.

8. *T. Bouma, L. Dorst, H. Pijls*, Geometric Algebra for Subspace Operations, *Acta Applicandae Mathematicae*, **73**, pp285-300, 2002.
9. L. Dorst, The inner products of geometric algebra, in: Applications of Geometric Algebra in Computer Science and Engineering (L. Dorst, C. Doran, J. Lasenby (ed.)) Birkhäuser, Boston, 2002.
10. *J. Poso, G. Sobczyk,* Realizations of the Conformal Group, in: Geometric Algebra with Applications in Science and Engineering, (E. Corrochano, G. Sobczyk (eds.)), Birkhauser, Boston 2001.
11. *D. Hestenes*, Old wine in new bottles: A new algebraic framework for computational geometry, in: Geometric Algebra with Applications in Science and Engineering, (E. B. Corrochano, G. Sobczyk (eds.)), Birkhäuser, Boston, pp. 3-17, 2001
12. *H. Li, D. Hestenes, A. Rockwood,* Generalized Homogeneous Coordinates for Computational Geometry, in: Geometric Computing with Clifford Algebras (G. Sommer (ed.)), Springer Verlag, Berlin, Heidelberg, 2001.
13. *W. Frenchel*, Elementary Geometry in Hyperbolic Space, ISBN 0-89925-493-4, Walter de Gruyter, Berlin, New York, 1989.
14. *E. Horowitz, S. Sahni*, Computer Algorithms/C++, Computer Science Press, 1996.