MULTIDIMENSIONAL DATA SETS REDUCTION THROUGH SPATIAL DECOMPOSITION

Marius Dorian Zaharia
POLITEHNICA University of Bucharest
Computers Department
Splaiul Independentei, 313, Bucharest - 77206, Romania
zaharia@cs.pub.ro

Abstract. *The paper describes a way to find representative points of a multidimensional point data collection. This method uses Voronoï diagrams. The algorithm is based on geometric computations as those used in classical clipping algorithms. A parallel algorithm to find a Voronoï diagram associated to a multidimensional point data set is also presented.*

Keywords: multidimensional point data, Voronoï diagram, parallel algorithm, geometric clipping.

Introduction

A lot of application classes are managing complex structured data sets. This raises problems of correct interpretation of the information represented by the data. Some data sets are usually produced by numerical simulation processes (e.g. programs for determination of the thermic functioning (heat transfer) of different systems, or applications in the field of computational fluid dynamics). Other multidimensional point data sets are produced by experimental measurement processes for example in domains as geology, seismology, teledetection or medicine (MRI, CAT). Sometimes, modeling applications (molecular systems modeling) could generate also massive data sets.

One of the main research areas related to these categories of data is scientific visualization which uses the computer graphics methods to create images which could help to understand the nature or structure of these data. The multidimensional point data sets could be assimilated with sets of tuples in a relational database or could be used in applications as geographic information systems, or CAD systems. Due to the complexity of such data sets a usual problem is their reduction. This could be achieved by grouping similar data points and assigning to each group a representative point.

For example an application for scientific visualization of large data sets could have the following main stages; (one of them consists of data reduction):

- Data generation (acquisition) and validation (i.e. producing numerical data through measurement processes or numeric simulations)
- Improvement of the "data information content" using methods to modify the data values or to select representative subsets (data points) to be presented to the end user
- Data *analysis* and *reduction* consists in determining the structure of the data set, extracting data (quantitative or statistical) features, identifying some restrictions the data must satisfy or applying algorithms to group data values in collections which could more easily be manipulated. The algorithms described below could be considered as included in this stage of the visualization process.
- The data *graphic representation* uses various modeling techniques, for example surface or volume models, multi-resolution models, or scattered data models (the data values are provided by sampling the data space across a non-regular spatial grid).

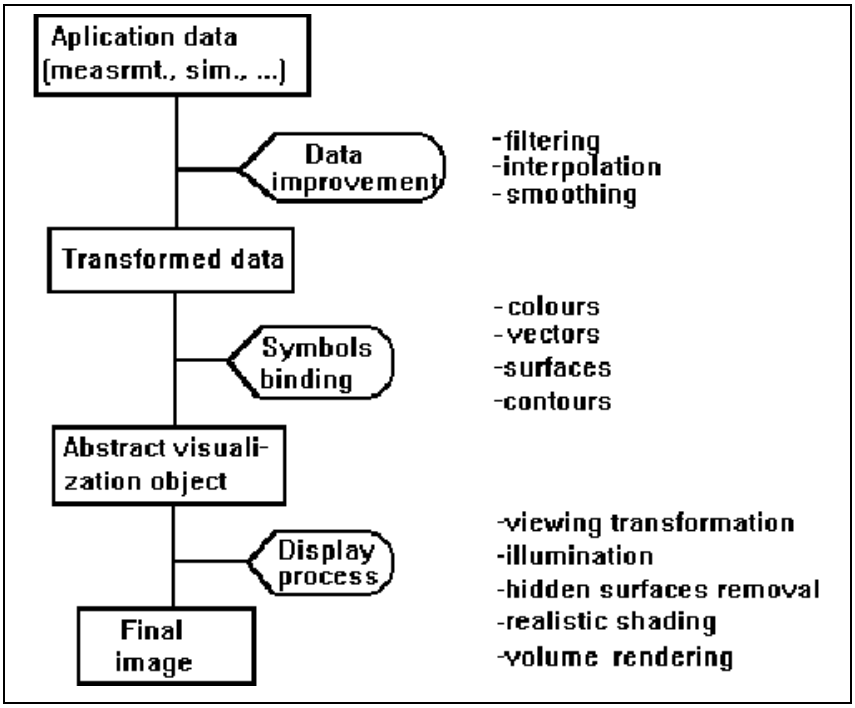

Figure 1 - Stages of a Scientific Visualization Application

Determination of representative data points

One of the fundamental methods to reduce the cardinality of the multidimensional point data sets consists of grouping (clustering) data with similar characteristics. The similarity is achieved by optimising an objective function which is determined by some features (properties) of the data. This technique is used in applications in the field of digital image processing, pattern recognition or image analysis. If the data space is Euclidean, the spatial region corresponding to a data class could be a hyperpolyhedral cell of a Voronoï diagram.

Let (P)={p_1, p_2, ..., p_n} be a point data set, (the points p_i are distinct) and w_1, w_2, ..., w_n positive integer weights associated to each data point. (p_i has w_i instances in the data set). w_i could be considered as the degree of multiplicity of p_i. The algorithm described below will find the points q_i, $1 \le i \le m$, which are centers of regions in which the data points p_i $1 \le i \le n$ will be grouped. The representative points are selected so that the functional:

$$Er = \sum_{j=1}^{m} e_j = \sum_{j=1}^{m} \sum_{i \in I_j} w_i \left\| p_i - q_j \right\|^2 \qquad (1)$$

is minimized.

The "grouping sets" I_j $(1 \le j \le m)$ are defined by:

$$I_j = \left\{ i \;\middle|\; \left\| p_i - q_j \right\| < \left\| p_i - q_k \right\|, \forall k \neq j \right\}$$

The centers of those groups are the best m sample points which are representing the

points of (P).

Finding the global minimum of (1) is a NP-complete problem but a local minimum could be determined by using the greedy algorithm below. This algorithm uses a Voronoï diagram to find the grouping sets I_j.

1. Initially all the data points are grouped in only one hyperrectangular region (the entire data space) whose center is computed by:

$$\frac{\sum w_i p_i}{\sum w_i} \qquad (2)$$

2. At an arbitrary step (moment) during the execution of the algorithm the region (Reg) with the largest associated error term e_{Reg} is selected. Let q_1, ...q_{Reg}, ... q_r be the representative points already computed. The region Reg is splitted across the dimension d of the data space for which the dispersion of the data points from Reg is the largest. The splitting hyperplane is normal to (the hyperaxis) d and the two resulted subdivision regions R_A and R_B have equal sizes.

3. The centers q_A, q_B of R_A and R_B are computed using relation (2)

4. A new Voronoï diagram for points q_1, ..., q_{Reg-1},q_A,q_B,q_{Reg+1}, ...q_r is computed (q_{Reg} is substituted by q_A and q_B). All the original data points from (P) which are interior to a hyperpolyhedral region of the Voronoï diagram will be grouped in the same collection G_i and the representative point (centroid) of G_i will be computed according to (2). In this way a new set of r+1 representative points is obtained.

5. Steps 2,3,4 will be repeated until a predefined number (m) of subdivision regions is attained (or, better, the medium distance between a representative point and its associated data points does not overflow a predefined threshold value).

In figure 1 a data set of 82 equally weighted (two-dimensional) data points was considered. The optimal grouping (whose medium distance per cluster does not overflow 20 units) is achieved for N=12 clusters. In that case (Figure 2b) the clustering time per data point was 68 msec (on an INTEL 486SX/33 processor).

Determination of a Voronoï diagram

The algorithm designed for computing a Voronoï diagram corresponding to r representative points q_1, q_2, ...,q_r finds at one step one hyperpolyhedral cell (C_i) associated to a point q_i ($1 \le i \le r$). For that purpose the perpendicular bisectors of q_iq_j ($1 \le j \le r$, $j \ne i$) are sequentially determined.

Let med_{ij} be one perpendicular bisector found at an arbitrary step of the algorithm. It will clip the "partial cell" c_i of the Voronoï diagram whose boundary is formed by the previously determined perpendicular bisectors of line segments q_iq_k ($1 \le k < j$) and eventually some pieces of the boundary of the entire data space (considered hyperrectangular).

The algorithm can be easily parallelised. The determination of the cell associated to one point q_i can independently be done (by a separate process) with respect to the other cells of the diagram.

Here is a formal description of the parallel algorithm in UNITY ([1]):

$$\langle\| i{:}1 \le i \le r :: * find\_the\_cell\_C_i\_associated\_with\_q_i \rangle$$

** find the cell C_i associated with q_i **is**

* initialize "polyline" with the boundary of the hyperrectangular data space

<j : 1≤j≤r ∧ j≠i :: med[i,j]:= *perpendicular bisector of q_iq_j |

< k : 1≤k<j :: * clipping of "polyline" with edges med[i,k], by med[i,j] >>

The complexity of the algorithm is $O(r^2/P)$ where r is the number of clusters (centroid points) and P the number of processors. The elementary operation is clipping a (hyper)polygon by a (hyper)line.

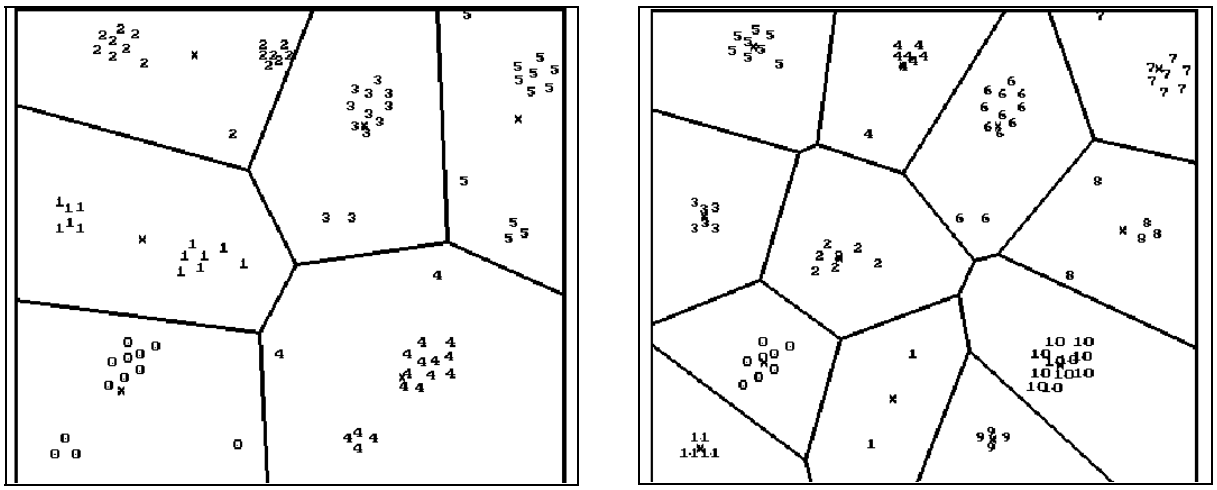

Figure 2 – Two-dimensional Voronoï Diagrams

An analogous clustering method the mean-cut quantization method is used in [5] for finding representative colors of all the colors of the pixels in a digital image. For that type of problem the cubic (R,G,B) space is the data space. In the case of this previously mentioned method the splitting process is analogous but the clusters (grouping cells) have hyperrectangular shapes whose boundary planes are parallel with coordinate planes. The precision of the method based on Voronoï diagrams is greater, the grouping cells could be arbitrary convex hyperpolygons, not only simple hyperrectangles.

Implementation Considerations

The main geometric computations necessary to determine a multidimensional Voronoï diagram are: the decision process if a point is (or not) interior to a hyperpolyhedron and the geometric split (clipping) process of a multidimensional polyhedron.

These problems were solved in case of a two-dimensional and of a three-dimensional data space. The data structures used to represent a polyhedron and a plane were specially selected to allow easy generalization of splitting computations in case of N-dimensional (N>3) data spaces.

For example a plane (π) was represented by two N-dimensional points the first representing the normal versor and the second an arbitrary point in ( ). A hyperline (l) was also represented by two N-dimensional points the first representing the direction versor and the second an arbitrary point on (l). The cells of a Voronoï diagram are always convex hyperpolyhedrons (a hyperrectangular data space was considered). The 3D polyhedron was modelled using a boundary representation with an explicit list of vertices and a list of faces. Each face is a list of vertices (i.e. references to the items of the polyhedron vertex list). The C specification of these data structures follows:

```
typedef double punct[DIM];    /* a DIM-dimensional point in cartesian
                              coordinates */

typedef struct lv {           /* a vertex list */
        punct varf;
        struct lv *urm;
      } *lista_varf, *polig, *lista_puncte;

typedef struct ft {     /* one face is a list of pointers to elements */
        lista_varf vf;  /* in the vertices list; the last vertex is */
        struct ft *urm; /* identical to the first one*/
      } *fata;          /* vertices are traversed in counterclockwise */
                        /* order */

typedef struct lf {     /* faces list */
        fata f;
        struct lf *urm;
      } *lista_fete;

typedef struct pl{
        lista_fete listf;     /* faces list of a polyhedron*/
        lista_varf listv;     /* vertices list of a polyhedron*/
      } *poliedru;

typedef struct dv {     /* a Voronoï diagram is a list of polyhedrons */
        poliedru pcomp; /* a polyhedral cell */
        punct preprez;  /* the associated representative point */
        struct dv *urm;
      } *diagvor;

typedef struct pln {          /* a hyperplane is represented by */
        punct norm;           /* his normal vector */
        punct pplan;          /* and the coordinates of a point in plane */
      } *plan;

typedef struct ln {           /* a hyperline is represented by */
        punct dir;            /* its direction versor dir */
        punct plin;           /* and the coordinates of a point on line */
      } *linie;
```

The first of the above mentioned problems (deciding if a point p is interior to a hyperpolyhedron P) could be resolved in two ways: by counting the intersections between a straight line which starts from p and the boundary of the hyperpolyhedron or by computing the total signed angle around p determined by each face of the hyperpolyhedron (as in [6], [7]). In case of an exterior point the total signed angle is 0. Both methods were tested and the first one was less time consuming.

The second problem (geometric split of a hyperpolyhedron by a hyperplane) is solved by treating each polyhedron face separately. The boundary of each face is processed edge by edge. For each edge (E) the positions of the two associated vertices (E_A, E_B) with respect to

the splitting plane will be determined. If E_A and E_B are placed on opposite sides of the plane the edge (E) will be splitted. Because the faces are convex polygons the splitting process of a face will generate at most two other faces. If the splitting plane (π) intersects the polyhedron (P) each of the two resulted (sub)polyhedrons (P1) and (P2) will have a new face placed in (π); all the other faces of P1 and P2 will be included in correspondent faces of (P). The C specification of the splitting function is given below

```
int split_face    (fata p, plan q, fata *pminus, fata *pplus,
             lista_varf *lvpminus, lista_varf *lvpplus);
/*    tests the position of polygon "p" with respect to plane "q"
      if q intersects p two polygons (faces) pplus and pminus will be
      produced.
      The function returns:
      1  iff p is placed entirely on the positive side of q
      -1 iff p is placed entirely on the negative side of q
      0  iff q splits p */

int split_poliedru(plan pl, poliedru p, poliedru *pminus, poliedru *pplus)
/* split a 3D polyhedron p by a plane pl
   returns  1 iff all p's vertices are on the positive side of pl
   returns -1 iff all p's vertices are on the negative side of pl
   returns  0 iff p's vertices are placed on both sides of pl */
{ int i, k, k1, codret;
  double plc[DIM+1];
  fata f, fm, fp, fnou;
  lista_varf lvfm, lvfp;
  lista_fete lsf;
  /* determining plane equation coefficients */
  for(i=0; i<DIM; i++)
    plc[i]=pl->norm[i];
  for(i=0, plc[DIM]=0; i<DIM; i++)
    plc[DIM] -= pl->norm[i]*pl->pplan[i];
  *pminus=(poliedru)farmalloc(sizeof(struct pl));
  *pplus =(poliedru)farmalloc(sizeof(struct pl));
  (*pminus)->listf=NULL; (*pplus)->listf=NULL;
  (*pminus)->listv=NULL; (*pplus)->listv=NULL;
  vf_fata_noua=NULL;
  /* traversing the polyhedron face list */
  for(lsf=p->listf, codret=2; lsf!=NULL; lsf=lsf->urm) {
    k=semn_fata(lsf->f, plc);
    if(codret!=0) codret=k;
    switch(k) {
      case 1:
        (*pplus)->listf=addf(lsf->f, (*pplus)->listf);
        (*pplus)->listv=add_vf_inex(lsf->f, (*pplus)->listv);
        break;
      case -1:
        (*pminus)->listf=addf(lsf->f, (*pminus)->listf);
        (*pminus)->listv=add_vf_inex(lsf->f, (*pminus)->listv);
        break;
      case 0:
        k1=split_face(lsf->f, pl, &fm, &fp, &(*pminus)->listv,
                  &(*pplus)->listv);
        if(k1<=0) {
          (*pminus)->listf=addf(fm, (*pminus)->listf);
        }
        if(k1>=0){
          (*pplus)->listf=addf(fp, (*pplus)->listf);
        }
        break;
    }
  } /* for */
  /* assembling the new face included in the splitting plane */
```

```
  if(codret>=0) {
    fnou=asambl_fata(&(*pplus)->listv);
    (*pplus)->listf =addf(fnou, (*pplus)->listf);
  }
  if(codret<=0) {
    fnou=asambl_fata(&(*pminus)->listv);
    fnou=inv_list_f(fnou);
    (*pminus)->listf=addf(fnou, (*pminus)->listf);
  }
  farfree(vf_fata_noua);
  return codret;
}
```

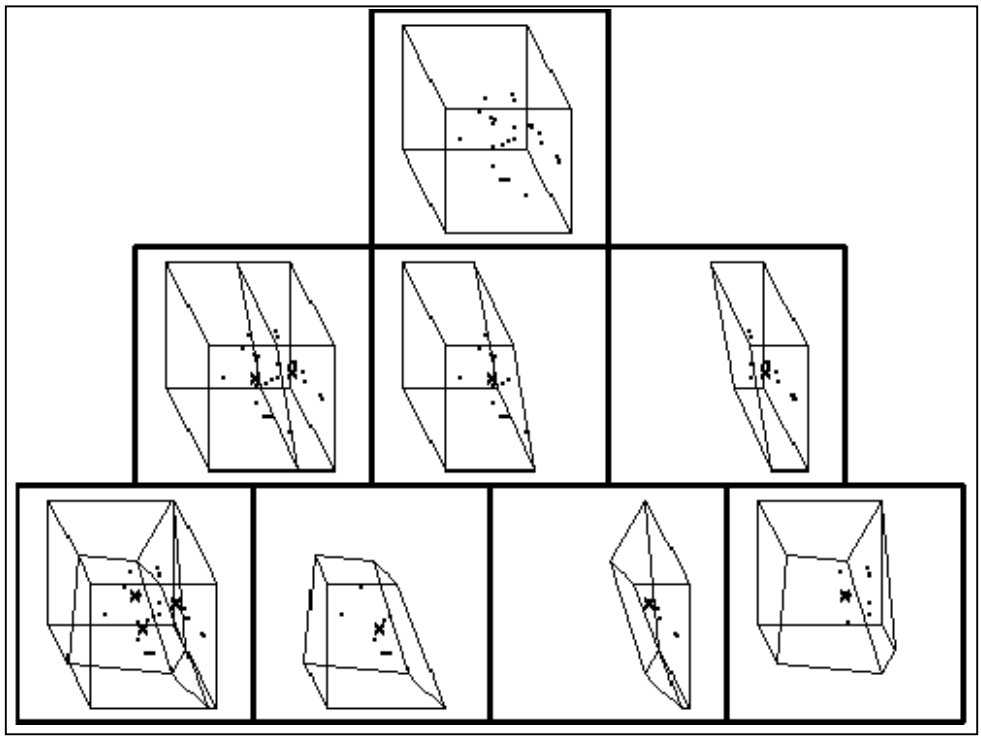

Figure 3 – Three-dimensional Voronoï Diagrams

The test case presented in Figure 3 corresponds to a set of 25 data points. The data space is a cube. Two Voronoï diagrams with 2 respectively 3 cells are visualized. Each cell corresponds to a cluster of data points and has an associated representative point visualized as an "x-shaped" cross.

Acknowledgements

I should like to thank to Mr. Prof. Dr. Eng. Mircea Petrescu for his kind encouragement to participate at DAS '98.

References:

[1] Chandy, K. M. and Misra, J. (1988) *Parallel Program Design*, Addison-Wesley Publishing Company.
[2] Gershon, N. and Brown, J. (1996) *Computer Graphics and Visualization in the Global Information Infrastructure*, IEEE Computer Graphics & Applications, No. 5.
[3] Hagen, H. (1994) *Visualization of large data sets*, Scientific Visualization, Advances and

Challenges, Academic Press.
[4] Lavender, D., Boyer, A., Davenport, J., Wallis, A. and Woodwark, J. (1992) *Voronoï Diagrams of Set-Theoretic Solid Models*, IEEE Computer Graphics & Applications, No. 9.
[5] Lindley, C. A. (1992) *Practical Ray Tracing in C*, John Wiley & Sons Inc.
[6] Miller, R. D. (1994) *Computing Area of a Spherical Polygon*, in Graphics Gems, vol. 4, (coordinator Heckbert, P.), Academic Press Inc.
[7] Pintho-Carvalho, P. C. and Cavalcanti, P.R. (1995) *Point in Polyhedron Testing using Spherical Polygons*, in Graphics Gems, vol. 5, (coordinator Paeth, A.), Academic Press Inc.
[8] Preparata, F. P. and Shamos, M. I. (1985) *Computational Geometry: An Introduction*, Springer Verlag, New York.