Eurographics/ IEEE-VGTC Symposium on Visualization 2009
H.-C. Hege, I. Hotz, and T. Munzner
(Guest Editors)

Volume 28 (2009), Number 3

Enclosing Surfaces for Point Clusters Using 3D Discrete Voronoi Diagrams

Paul Rosenthal† and Lars Linsen‡

School of Engineering and Science
Jacobs University
Bremen, Germany

Abstract

Point clusters occur in both spatial and non-spatial data. In the former context they may represent segmented particle data, in the latter context they may represent clusters in scatterplots. In order to visualize such point clusters, enclosing surfaces lead to much better comprehension than pure point renderings.
We propose a flexible system for the generation of enclosing surfaces for 3D point clusters. We developed a GPU-based 3D discrete Voronoi diagram computation that supports all surface extractions. Our system provides three different types of enclosing surfaces. By generating a discrete distance field to the point cluster and extracting an isosurface from the field, an enclosing surface with any distance to the point cluster can be generated. As a second type of enclosing surfaces, a hull of the point cluster is extracted. The generation of the hull uses a projection of the discrete Voronoi diagram of the point cluster to an isosurface to generate a polygonal surface. Generated hulls of non-convex clusters are also non-convex. The third type of enclosing surfaces can be created by computing a distance field to the hull and extracting an isosurface from the distance field. This method exhibits reduced bumpiness and can extract surfaces arbitrarily close to the point cluster without losing connectedness.
We apply our methods to the visualization of multidimensional spatial and non-spatial data. Multidimensional clusters are extracted and projected into a 3D visual space, where the point clusters are visualized. The respective clusters can also be visualized in object space when dealing with multidimensional particle data.

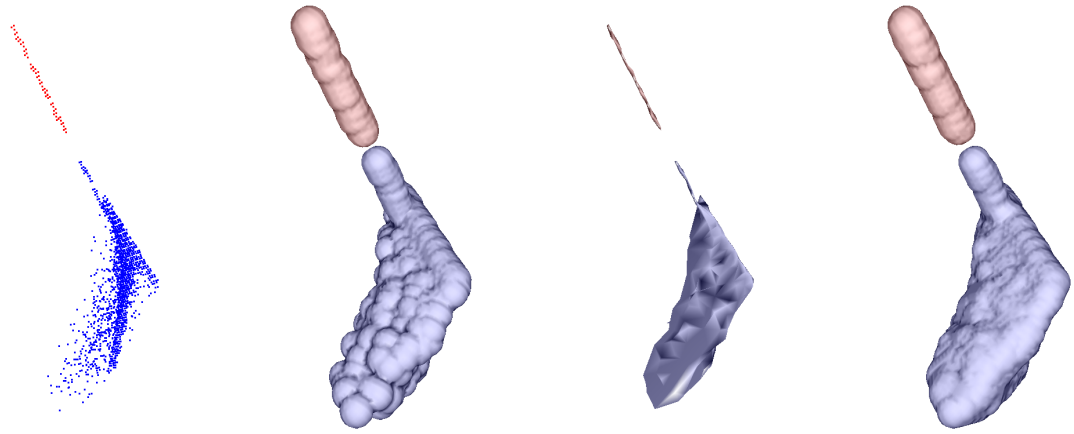
Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—

1. Introduction

Point clusters are sets of data points with a certain similarity in a given space. Typically, clusters are computed by defining a similarity metric in the given space and grouping points that are closest with respect to the chosen metric. Many clustering methods exist; some are automatic, others interactive or user-guided; some generate a partitioning, others generate hierarchies. They all have in common that they output several sets of data points (i. e., the point clusters) in the given space, which then have to be visualized in a visual space. If the given space has a low dimension (2D or 3D), the visual space is commonly equal to the given space. Otherwise, one typically projects the given space to a lower-dimensional visual space. Choosing a 3D visual space instead of a 2D visual space provides a higher flexibility and supports keeping detected clusters separated. Again, many projection methods exist. In the end, the visualization task is to render point clusters in a visual space.

A common choice when rendering point clusters is to use color coding to identify clusters and to render the individual points in the respective colors. Such a visualization is shown in Figure 1 a). However, color coding is problematic for hierarchical clusters and, more importantly, point renderings have severe depth perception problems. Instead, it would be desirable to render surfaces to represent point clusters. When properly shaded the surfaces allow for a good depth percep-

† p.rosenthal@jacobs-university.de
‡ l.linsen@jacobs-university.de



a) b) c) d)

Figure 1: *Different visualizations of two point clusters (colored red and blue) from the 2008 IEEE Visualization Design Contest data. The clusters were found using density-based clustering of the multidimensional feature space and were projected to a 3D visual space using a linear projection. Additionally to the cluster points a), three types of enclosing surfaces are shown. b) Isosurface extraction from distance field computed using a 3D discrete Voronoi diagram of resolution* $256 \times 256 \times 256$*. c) Hull of the cluster computed from the isosurface of the distance field. d) Isosurface extraction from the distance field to the hull in c).*

tion and they represent the boundary of a continuous subspace of the visual space. Such a surface has to adhere to several desired properties:

- The surface must be a manifold with an unambiguous inside/outside property.
- The surface must enclose all points of the point cluster.
- The surface must stay close to the points of the point cluster.
- The surface must be connected, unless the clustering metric does not apply to the visual space.
- The surface must be sufficiently smooth.

It is worth mentioning that a convex hull computation violates the third postulation in case of non-convex point clusters. Indeed, in the case of two interlaced, but separated non-convex clusters, convex hull visualization would lead to overlapping surfaces, which is undesirable. In general, it should be avoided that an enclosing surface of a cluster includes any points of the complement of the cluster. We do not list this as a postulate, as projected multidimensional clusters may actually overlap in visual space, if the projection cannot keep the clusters separated.

We propose a flexible system for generating enclosing surfaces for point clusters based on 3D discrete Voronoi diagram computations. Our system supports three type of enclosing surfaces.

The first type of enclosing surfaces is illustrated in Figure 1 b) and is obtained by exploiting the discrete distance field that is computed along with the Voronoi tesselation. The distance field describes the distance to the points of the point cluster. Extracting isosurfaces from the distance field leads to an enclosing surface that adheres to the listed properties. Such a surface is equivalent to a surface one would obtain by convoluting each point with a radial basis function, summing all convoluted basis functions, and extracting surfaces from the resulting field. We can interactively extract any isosurface from the distance field such that any distance to the point cluster is supported.

The second type of enclosing surfaces is obtained by exploiting the natural neighborhood property induced by the Voronoi tesselation with respect to an appropriate isosurface of the distance field. Where three Voronoi regions come together, we generate a triangle that connects the points of the respective Voronoi regions. This approach leads to a surface that describes a hull of the point cluster, i. e., a triangular mesh whose vertices are a subset of the points of the point cluster, as shown in Figure 1 c). Note that the hull actually is non-convex in the case of non-convex point clusters. This approach is related to the α-shape approach [EM94] for surface reconstruction. The difference is that the α-shape approach is applied for surface reconstruction from point clouds, i. e., from points that all are supposed to lie on the surface, while our approach is applied to point clusters, where the majority of the points is supposed to lie inside the generated surface. The equivalent of the α-value in our approach would be the isovalue that is chosen to select the isosurface of the distance field.

The third type of enclosing surfaces addresses the issue

that the surfaces of the first type typically lead to bumpy surfaces. Hence, we propose to generate surfaces that are not isodistant to the point cluster, but isodistant to the hull, i. e., to the enclosing surface of the second type. An illustration of such an enclosing surface is shown in Figure 1 d). We describe how the 3D discrete Voronoi diagram computation can be adjusted accordingly to produce a distance field to the hull. Then, we can interactively generate enclosing surfaces with any distance to the hull by extracting isosurfaces from the new distance field. The resulting enclosing surfaces are much less bumpy than the ones of the first type and they still stick close to the point cluster. In fact, the isovalue can be chosen arbitrarily small without violating the connectedness postulate.

Our methods for computing enclosing surfaces can be applied to different tasks in both spatial and non-spatial data visualization. In terms of non-spatial data, our methods apply to any 3D scatterplot with a given classification. Classification can be obtained with respect to the given 3D space, but also with respect to a multidimensional space, to subspaces thereof, or to individual dimensions. One obvious example would be the visualization of categorical data. In terms of spatial data, our methods apply to any particle data set with a given segmentation. Segmentation can be obtained with respect to a given data field, but also with respect to a derived field or to multidimensional feature space clustering. One example would be the visualization of a derived characteristic property like positive, negative, and zero divergence of a vector field.

We evaluate our system by applying it to two tasks from multidimensional data visualization and discuss the three types of enclosing surfaces. The first task is the common information visualization task of visualizing multidimensional non-spatial data. We apply a clustering algorithm to the data points in the given multidimensional space and project the data into a 3D visual space. We apply our enclosing surface generation methods to the projected point clusters in 3D visual space.

The second task is the visualization of multi-field particle data. In this application, we are dealing with a 3D object space and a multidimensional feature space. We apply a clustering algorithm to the multidimensional feature space. When projecting the feature space into a 3D visual space, we produce the analogon to the first task. However, the clustering in feature space also induce a segmentation or partitioning in object space. Hence, we can use the 3D object space as a visual space and apply our enclosing surface generation methods to the particles, which are segmented with respect to the feature space clustering. For the enclosing surface generation in object space, the resulting surface must not be connected anymore, as the clustering was performed in feature space. The feature space cluster do not need to belong to one region in object space.

2. Related Work

The first approach feasible for visualizing point clusters was the blobs method presented by Blinn [Bli82]. Blinn associates each point with a radial basis function. Some parameters control the "blobbiness". Many modifications and variations to this method exist [BR01, KV06, RB08]. Still, the generated enclosing surfaces all follow the original blob model. For some algorithms an adaptive adjustment of the radius of influence is necessary. For long thin clusters, all these approaches lead to bumpy enclosing surfaces.

To overcome this problem, more geometry information has to be considered for the point cluster. Voronoi diagrams [Vor08] have become a valuable and widely used tool in this context. Since the exact computation of a Voronoi diagram is not applicable to large sets of points, many approximation algorithms exist, like (t,ε)-approximate Voronoi diagrams [AMM02] or Voronoi Octrees [BCMS05].

For the generation of distance fields, the approximation by discrete Voronoi diagrams is favorable, which has already been constructed in many different ways [MRH00, TT97]. With the rise of general purpose computations on graphics cards, several algorithms using the features of GPUs were introduced [HT05, SGM05, SPG03]. We follow the ideas of Hoff et al. [HKL*99, HCK*99], improve the proposed method, and generate 3D discrete Voronoi diagrams. A discussion can be found in Section 4. With the help of these 3D discrete Voronoi diagrams, we are able to generate hulls and enclosing surfaces for point clusters.

3. Enclosing Surfaces

Let $M \subset D$ be a set of unstructured sample points in the domain $D \subset \mathbb{R}^3$. A point cluster C is a non-empty subset of M with points having a certain common property. Intuitively, clusters consist of points lying close together, although this may not be true, if the regarded property is not reflected in the distribution of M within D.

The goal is to find a closed orientable surface Γ surrounding point cluster C, i. e., all points of C should lie on one side of Γ. Additionally, the surface Γ should be smooth but also close to the points of C to exhibit the structure and distribution of the point cluster in the visual space D.

To generate a surface fulfilling these requirements, we developed algorithms based on Voronoi tesselations. For a given geometric structure such as point cluster C, Voronoi tesselations compute for each position $\mathbf{x} \in D$ the closest point of the geometry. They also deliver the distance of $\mathbf{x}$ to that closest point. In addition, they provide neighborhood information for the geometrical primitives; primitives with neighboring Voronoi cells are natural neighbors.

As 3D Voronoi diagrams are expensive to compute, we use a discrete version, which can be supported by a GPU

implementation. For the discretization, we generate a uniform regular hexahedral grid (voxel grid) and sample the point cluster to that grid. Multiple points falling in one grid cell can be treated as a single entry for our purposes. After discretization, all operations are applied to the discretized point cluster. Of course, discretization introduces error, but errors in the subvoxel range of the chosen discretization grid are not an issue for our application. Details on the 3D discrete Voronoi diagram computation are given in Section 4. Exploiting the above-mentioned information provided by Voronoi diagrams, we generate enclosing surfaces for point clusters. We support three approaches:

Surfaces Isodistant to Point Cluster. To generate enclosing surfaces that are isodistant to the point cluster, we generate a discrete Voronoi diagram with the points of the point cluster as geometric primitives. This computation is described in the first part of Section 4. We take the distance field delivered by the discrete Voronoi computation and extract isosurfaces from them as described in Section 6. The isovalue can be changed interactively. As a default isovalue we choose $\frac{d}{2}+\varepsilon$, where d is the length of the longest edge in a minimum spanning tree of the point cluster and ε is a small positive constant. This default isovalue is the smallest value that ensures that the isosurface is connected.

Hull of Point Cluster. A surface isodistant to the point cluster consists of surface patches that are induced by the nearest point. Hence, the neighborhood relation of the surface patches also creates a neighborhood information on the points. This neighborhood information can directly be obtained from the discrete Voronoi diagram by investigating natural neighborhoods in the isosurface region. When three neighborhoods come together, the respective points of the point cluster can be connected with a triangle. Generating all those triangles leads to a hull in form of a closed surface, see Section 5.

Surfaces Isodistant to Hull. The bumpiness of the surface isodistant to the point cluster can be reduced by computing a surface isodistant to the hull of Section 5. In order to compute a distance field to the hull, we extend our discrete Voronoi diagram computation from point clusters to polygonal models as described in the second part of Section 4. Then, we can apply the isosurface extraction of Section 6 to the distance field to obtain the enclosing surface. Again, the isovalue can be changed interactively.

The results of the three types of enclosing surface generation are discussed in Section 7.

4. 3D Discrete Voronoi Diagrams

For the generation of discrete Voronoi diagrams, Hoff et al. [HKL*99] proposed a method making use of modern GPU capabilities. They developed an algorithm using cones to generate 2D discrete Voronoi diagrams for different geometric primitives and proposed ideas on how to generalize them to 3D discrete Voronoi diagrams. We have pursued their ideas for point primitives and propose a simple and fast generalization for generating 3D discrete Voronoi diagrams for scenes with points, lines and polygons.

4.1. Points

A direct generalization of the ideas of Hoff et al. to 3D discrete Voronoi diagrams would require us to render a 3D cone for each cluster point, i. e. a 3D manifold in a 4D space, which is then projected into a 3D output buffer. Such a mechanism is, of course, not supported by graphics cards. Hence, we generate the 3D discrete Voronoi diagram as a stack of 2D layers.

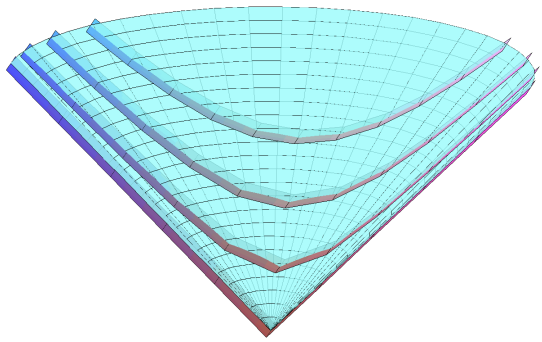


Figure 2: *Approximating one sheet of a 2D cone by layers of 1D conic sections.*

When doing so, the 3D cones have to be layered, as well. We obtain layers of 2D conic sections parallel to the axis of the cone. Figure 2 depict these layers by illustrating everything one dimension lower. Each conic section describes one sheet of a two-sheeted parabolic hyperboloid. They represent objects in a 3D space that can be rendered to the respective 2D layer on the GPU.

For each 2D layer of the 3D discrete Voronoi diagram and each cluster point, the appropriate hyperboloid, i. e., the conic section associated with the orthogonal distance of the point to the current layer, is rendered using depth buffering of the graphics card. Figure 3 illustrates the depth buffering idea, again, depicting everything one dimension lower. Iterating over all 2D layers completes the 3D discrete Voronoi diagram.

From the illustration in Figure 2, one can deduce that we only have to generate a discrete set of conic sections, as we are only dealing with discrete distances between layers. The orthogonal distances of the conic sections to the current layer are given by $d = 0, 1, \ldots, d_{\max}$, where $d_{\max}$ depends on the grid resolution in the third dimension. Hence, the required

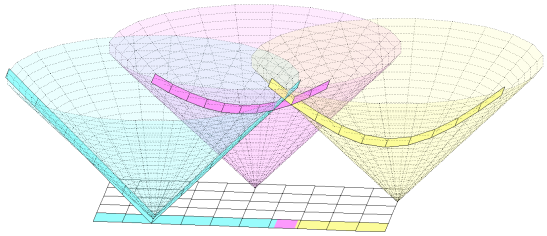


Figure 3: *Generation of a one-dimensional layer of a two-dimensional discrete Voronoi diagram. For each sample point one conic section is rendered above the layer with respect to the distance of the point to the layer. Using the depth buffer and orthogonal projection results in an approximation of the discrete Voronoi diagram in the current layer.*

set of hyperboloids can be precomputed and stored as a triangular mesh. No other geometry needs to be rendered in the entire process.

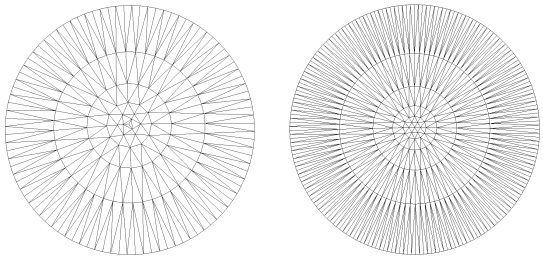


Figure 4: *Projections of different triangulations for a conic section. The triangulation assures rotational symmetry of the conic section. All points lie on circles with growing radii. The number of triangles is determined by the number of triangles in the innermost ring: a) 3, b) 6.*

To generate the triangular mesh, we parameterize the hyperboloids using the canonical parameterization in cylindrical coordinates. We obtain rings of triangles with growing radii. The radii double from one ring to the subsequent one. To assure a close-to-constant triangle-per-diameter ratio, the amount of triangles are also doubled from one ring to the subsequent one. This concept also ensures that the approximation error is bounded by the error in the innermost ring. Two triangulations with different number of triangles in the innermost ring are shown in Figure 4. For all examples we tested, it turned out that choosing six triangles and radius 2 for the innermost ring was a good compromise between quality and speed. An illustration of triangulated conic sections with different distances d to the current layer is shown in Figure 5.

To establish a correspondence between points and conic sections, we use color. Each point gets assigned a unique color, which is used for rendering the respective conic section. Hence, the resulting discrete Voronoi regions are color-coded. We use the alpha channel to store the depth buffer values. Hence, a distance field to the points is generated without extra costs.

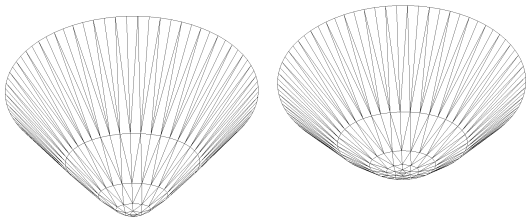


Figure 5: *Illustration of triangulated conic sections with different distances d to the current layer.*

4.2. Lines and Polygons

Hoff et al. [HKL*99] describe a generalization of their 2D approach to 3D scenes. For handling triangles, they suggest to use methods for its vertices and lines combined with a rendering of a triangle. Their description is very vague. Rendering the triangle itself would actually lead to incorrect results, unless the triangle is parallel to the layers. This is illustrated in Figure 6 (in a lower dimension). The black line indicates the given triangle, whereas the red line indicates the triangle we would have to draw in order to generate a correct distance field. However, there is no obvious way on how one could compute this triangle without complicated geometric considerations. There is no evidence that Hoff at al. actually implemented the 3D case. It seems that they only used a brute-force method to compute 3D discrete Voronoi diagrams as stated in their technical report [HCK*99].

We propose to use a discrete approach for including lines and polygons into our 3D discrete Voronoi diagram framework. We sample the lines and polygons using points and reduce the problem to the problem solved in the previous subsection. This is a simple and effective solution.

For sampling line segments or triangles, we utilize the graphics card's rasterization ability to generate a discrete representation of the geometric primitives over the underlying regular grid. This 3D rasterization is done in 2D layers using orthogonal projection. For each layer represented as a slab of the underlying regular grid, all line segments and triangles are clipped to the domain of that layer and orthogonally projected to the layer. For each grid cell that is marked as containing geometry during the rasterization step, we generate a sample point. Analogously, one could proceed with any geometric primitive. Any polygon could be handled by triangulating it in a preprocessing step, although only for planar simple polygons the distance is well-defined.

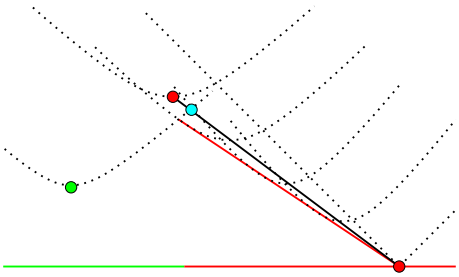


Figure 6: *Correct 1D discrete Voronoi diagram generation for a line and a point. The intersection of the black line with the hyperbola of the green point, gives a wrong border (cyan point) of the discrete Voronoi diagram. The correct line to render would be the red one. We approximate the red line by discretizing the black line with point samples (indicated as hyperbolae).*

5. Hull Generation

A hull of a point cluster is a connected enclosing surface whose polygonal representation uses a subset of the given points as vertices. A hull is the connected enclosing surface with minimum volume. Hence, for certain applications surfaces may be preferable over isosurfaces of the distance field.

Exploiting the discrete Voronoi diagram of the point cluster, we first use the distance field to extract a surface isodistant to the point cluster. Connectedness of this surface can be achieved by considering the length of the longest edge in the minimum spanning tree of the point cluster for the choice of the isovalue. However, a larger isovalue is typically desirable.

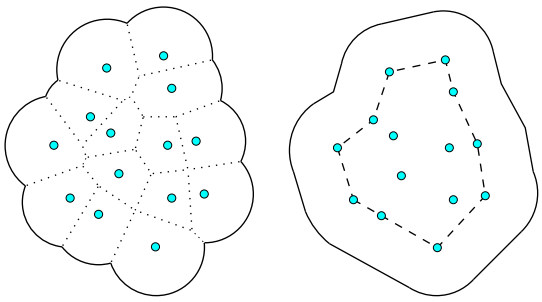


Figure 7: *Hull generation for a point cluster: a) Extracting an isosurface from the distance field to the point cluster. Voronoi regions on the isosurface induce neighborhoods. b) Neighbors are connected to form a hull. The image also shows an isosurface extracted from the distance field to the hull.*

The intersection of the discrete Voronoi diagram with the extracted isosurface results in a two-dimensional Voronoi diagram on the isosurface. Now, we can exploit the natural neighborhood property of the discrete Voronoi diagram to produce the hull. For each three Voronoi regions that come together in a point on the isosurface, a triangle of the hull is generated that connects the three Voronoi points of the respective Voronoi regions. A lower-dimensional illustration of this process is shown in Figure 7. Generating triangles for all triplets of adjacent Voronoi regions results in a triangular mesh describing the hull of the point cluster. This hull can be used for displaying the cluster directly or for the generation of an isosurface to the distance field to the hull, see Figure 7 b).

6. Isosurface Extraction and Rendering

As we are operating on a uniform regular hexahedral grid, isosurfaces can be extracted from the distance field using a standard marching cubes technique [LC87]. Once we have computed the distance field, isosurfaces with any distance can be extracted. For the rendering of the of the surfaces we use standard local illumination and Gouraud shading. For future work, we plan on using higher-quality shading techniques and we plan on investigating whether improved isosurface extraction techniques can reduce the typical marching cubes artifacts.

7. Results and Discussion

We have implemented our system using C++ and OpenGL together with the OpenGL shading language for the GPU programming part. All computations were done on a 2.66 GHz Intel Xeon processor with an NVIDIA Quadro FX 4500 graphics card. For the generation of the meshes of the hyperboloids a radius of $r = 2$ and an initial number of 6 triangles was used for the first ring.

The algorithms were applied to point cluster data sets of different types and origins. The first data set we used was a time step of the 2008 IEEE Visualization Design Contest data [WN08]. This is spatial data with a multidimensional feature space. For each sample point, temperature, density, mass abundances, and turbulence is given. The data set simulates the propagation of an ionization front through a gas. Although the spatial data is sampled on a volumetric grid, data points are unstructured in multidimensional feature space. The feature space was analyzed using a hierarchical density-based clustering algorithm. The resulting multidimensional point clusters were linearly projected to a 3D visual space using a 3D star coordinates layout [LLRR08]. The used projection is contractive, i. e., it does not break the clusters. We selected two clusters with very similar characteristic in parallel coordinates and show a scatterplot of the cluster points as well as renderings of all three types of enclosing surfaces in Figure 1.

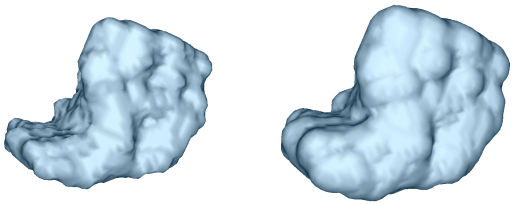
By just looking at the scatterplot in Figure 1 a) nearly no

information about the spatial structure of the clusters is visible. The enclosing surface isodistant to the points in Figure 1 b) gives a very good perception of the distribution of the points and the shape of the clusters. The direction in which the red cluster stretches exhibits high turbulence. From the figure it also gets immediately clear, why both clusters have nearly the same properties but were separated by the clustering algorithm: they differ in terms of turbulence. The hulls of the cluster points in Figure 1 c) show the real positions of the cluster points especially interesting in cases were clusters interlace each other, as shown later. In this particular case the surface isodistant to the hull in Figure 1 d) shows the most desired visualization of the relation between both clusters. The smooth surfaces emphasize the global shape of both clusters and their relation to each other.

It is an important feature of our approach that the isovalue for extracting the surface isodistant to the hull in Figure 1 d) can be chosen arbitrarily small without violating the connectedness of the surface, whereas the surface in Figure 1 b) is constrained to the minimum spanning tree criterion and would be unconnected for any smaller isovalue.

The second example is a multidimensional data set of unstructured points in 3D space coming from a smoothed particle hydrodynamics simulation by Stephan Rosswog, Jacobs University, Bremen. The data set represents a time step of the simulation of the disruption of a white dwarf star by a black hole. For each particle the density, temperature, and mass abundances are given.

The data points were clustered in feature space using the same clustering method as above. The resulting three clusters induce a segmentation in object space. The multidimensional feature space clusters also form clusters in object space. We compute the hull for each cluster on a $64 \times 64 \times 37$-grid and show the relations between them in Figure 8. The resulting surfaces are connected. The clusters from Figures 8 a) and b) are interlaced, which could not have been observed when using, e. g., convex hulls of the cluster points.



$f_{\text{iso}} = 0.02$ $f_{\text{iso}} = 0.03$

Figure 9: *Enclosing surface of a cluster from the "out5d" data set. The hull of the cluster was generated as well as a distance field to it. We show isosurfaces extracted from that distance field with two different isovalues* f_{iso}.

The third data set we used is the 5D non-spatial data set "out5d" (Courtesy of Peter Ketelaar.), which describes several physical properties measured by sensors. We apply the same clustering and projection methods as before. We generated the hull and a distance field to it. Afterwards, we extracted isosurfaces from the distance field with different isovalues. The development of the enclosing surface is shown in Figure 9.

The computation times for all steps of the distance field computations via 3D discrete Voronoi diagrams are presented in Table 1. Note that these steps are performed only once in a precomputation. The subsequent visualization step is a mere hull rendering or an isosurface extraction and rendering. All types of generated surfaces stay close to the cluster points and capture all important features also for non-convex clusters. They also adhere to all the other formulated postulates.

8. Conclusion

We have presented a system of three methods to compute enclosing surfaces for point clusters. They all rely on a GPU-based 3D discrete Voronoi diagram computation, which we developed to handle points as well as polygonal scenes. The enclosing surface can be isodistant to the point cluster, a hull, or isodistant to the hull. All surfaces are well-behaved with respect to the formulated postulates. The last approach produces surfaces that are less bumpy than the ones generated by the first approach. Moreover, the last approach allows for the generation of enclosing surface that are arbitrarily close to the hull. Hence, separated clusters that are rendered with this method will also appear as separated.

We applied our approaches to different scenarios. They can be used for displaying clusters in a 3D scatterplot. In our examples, the scatterplot was a projection of a multidimensional space. They can also be used for displaying segmented particle data. In our examples, the segmentation was obtained by clustering the multidimensional feature space of the particle data. For multidimensional particle data, our methods can be used for both visualization in object space and in the projected feature space.

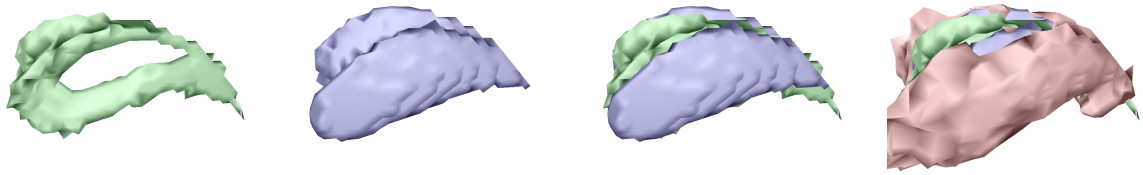
Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under project grant LI-1530/6-1.

References

[AMM02] ARYA S., MALAMATOS T., MOUNT D. M.: Space-efficient approximate voronoi diagrams. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing* (New York, NY, USA, 2002), ACM, pp. 721–730.

[BCMS05] BOADA I., COLL N., MADERN N., SELLARES J.: Approximations of 3d generalized voronoi diagrams. In *Proceedings of the European Workshop on Computational Geometry* (2005), pp. 163âĂŞ–166.



a) b) c) d)

Figure 8: *Enclosing surfaces of three different clusters (colored green, blue, and red) of a time step of the white dwarf data set. Using a grid of resolution* $64 \times 64 \times 37$*, the hulls to the clusters were extracted. a) Hull of the first cluster exhibiting a hole. b) Hull of the second cluster. c) Combination of blue and green cluster, showing how the blue cluster fills green cluster's hole. d) Rendering of all three cluster hulls. The enclosed volumes of the three hulls are interwoven but non-overlapping.*

point cluster	#points	grid size	#filled cells	dist. field	hull dist. field
VisContest 08	107k	$256 \times 256 \times 256$	1,553	133.0s	274.7s
White Dwarf	260k	$64 \times 64 \times 37$	12,323	40.1s	44.7s
out5d	16k	$64 \times 64 \times 48$	4,833	20.1s	13.0s

Table 1: *Computation times for generating the distance field to the points and for generating a distance field to the hull of the points including hull computation. For each point cluster, the number of points is given together with the dimensions of the used grid and the number of cells the cluster occupies in the grid.*

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256.

[BR01] BARANOSKI G. V. G., ROKNE J.: An efficient and controllable blob function. *journal of graphics tools 6*, 4 (2001), 41–54.

[EM94] EDELSBRUNNER H., MÜCKE E. P.: Three-dimensional alpha shapes. *ACM Trans. Graph. 13*, 1 (1994), 43–72.

[HCK*99] HOFF III K. E., CULVER T., KEYSER J., LIN M., MANOCHA D.: *Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware*. Tech. rep., University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1999.

[HKL*99] HOFF III K. E., KEYSER J., LIN M., MANOCHA D., CULVER T.: Fast computation of generalized voronoi diagrams using graphics hardware. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 277–286.

[HT05] HSIEH H.-H., TAI W.-K.: A simple GPU-based approach for 3d voronoi diagram construction and visualization. *Simulation Modelling Practice and Theory 13*, 8 (2005), 681–692.

[KV06] KRUITHOF N., VEGTER G.: Envelope surfaces. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry* (New York, NY, USA, 2006), ACM, pp. 411–420.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH 1987* (1987), ACM Press, pp. 163–169.

[LLRR08] LINSEN L., LONG T. V., ROSENTHAL P., ROSSWOG S.: Surface extraction from multi-field particle volume data using multi-dimensional cluster visualization. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (Nov.-Dec. 2008), 1483–1490.

[MRH00] MEIJSTER A., ROERDINK J., HESSELINK W. H.: A general algorithm for computing distance transforms in linear time. In *Mathematical Morphology and its Applications to Image and Signal Processing* (2000), Kluwer, pp. 331–340.

[RB08] ROSENBERG I. D., BIRDWELL K.: Real-time particle isosurface extraction. In *SI3D '08: Proceedings of the 2008 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2008), ACM, pp. 35–43.

[SGM05] SUD A., GOVINDARAJU N., MANOCHA D.: Interactive computation of discrete generalized voronoi diagrams using range culling. In *In Proc. International Symposium on Voronoi Diagrams in Science and Engineering* (2005).

[SPG03] SIGG C., PEIKERT R., GROSS M.: Signed distance transform using graphics hardware. In *Proceedings of IEEE Visualization '03* (2003), IEEE Computer Society Press, pp. 83–90.

[TT97] TEICHMANN M., TELLER S.: *Polygonal approximation of Voronoi diagrams of a set of triangles in three dimensions*. Tech. rep., Tech Rep 766, Lab of Comp. Sci., MIT, 1997.

[Vor08] VORONOI G.: Nouvelles applications des parametres continus a la theorie des formes quadratiques. *J. Reine Angew. Math. 134* (1908), 198–287.

[WN08] WHALEN D., NORMAN M. L.: Competition data set and description. In *2008 IEEE Visualization Design Contest* (2008), http://vis.computer.org/VisWeek2008/vis/contests.html.