GPU-Accelerated 2D Point Cloud Visualization using Smooth Splines for Visual Analytics Applications

Thomas Kalbe* TU Darmstadt Germany

Tatiana Tekušová† Fraunhofer IGD Germany

Tobias Schreck‡ TU Darmstadt Germany

Frank Zeilfelder§ TU Darmstadt Germany

Abstract

We develop an efficient point cloud visualization framework. For efficient navigation in the visualization, we introduce a spline-based technique for the smooth approximation of discrete distance field data. Implemented on the GPU, the approximation technique allows for efficient visualizations and smooth zooming in and out of the distance field data. Combined with a template set of predefined, automatically or interactively adjustable transfer functions, the smooth distance field representation allows for an effective visualization of point cloud data at random abstraction levels. Using the presented technique, sets of point clouds can be effectively analyzed for intra- and inter-point cloud distribution characteristics. The effectiveness and usefulness of our approach is demonstrated by application on various point cloud visualization problems.

Keywords: spline approximation; point cloud visualization; hardware acceleration; transfer function

1 Introduction

The visualization of 2D point clouds is one of the most basic yet one of the most important tasks in many data analysis applications. Point clouds are an ubiquitous type of data arising in many data analysis tasks. For example, point clouds may be obtained by plotting pairs of selected attributes of a multivariate data set against each other, obtaining scatter plots which are useful for analysis of correlations, clusters, and outliers. As another example, high-dimensional data sets can be visually inspected by obtaining projections to low-dimensional display space, e.g., by using Principal Component Analysis or Multidimensional Scaling techniques. Addtionally, the visualization of geospatial data often has to deal with sets of points representing certain locations. Typical analysis tasks on point cloud data include assessing the overall structure and distribution of the data, assessing spatial relationships between data elements, and identification of clusters and outliers.

Standard point-based visualization methods do not scale well with respect to the data set size. More specifically, as the number of data points and data classes increases, the display usually gets crowded quickly. Then, it is very difficult for the user to distinguish different point clouds from each other, or to correctly perceive their shape. Also, effective point cloud visualizations should give appropriate visual clues on the density of the point cloud throughout the point cloud domain. *Distance Fields* are an appropriate point cloud representation to this end, as they allow visualizations indicating both the notion of proximity to points, and the distribution of points. While different definitions are possible, a distance field defines for each location in the display plane a scalar value, usually the Euclidean distance to its nearest neighbor point.

We first develop a scheme for smooth approximation of distance fields based on discretely sampled distance data. Relying on a GPU implementation, our scheme of low total degree allows the smooth and efficient approximation of distance field data at arbitrary resolutions and zooming levels. Based on this approximation approach, we describe our point cloud visualization system, discussing important aspects of generating visualizations from distance fields. We demonstrate the suitability of the technique for the effective visualization of point cloud data. Specifically, in conjunction with appropriately designed transfer functions, important point cloud characteristics can be visually analyzed on different abstraction levels. Also, we show that multiple distance fields can be effectively blended, by application of the technique on a data set from the geo-analytics domain.

The remainder of this paper is structured as follows. In Section 2, we briefly introduce related work. Sect. 3 gives the details on our spline-based approximation scheme, while our GPU-based implementation thereof is described in Sect. 4. After sketching options for interactive and data-dependent specification of transfer functions in Sect. 5, in Sect. 6 we apply our solution on different data sets, where the effectiveness of our technique is demonstrated. Finally, Sect. 7 concludes and outlines future work in the area.

2 Related Work

We discuss visualization and representation of the point cloud data type. We also briefly recall visualization of higher order primitives.

2.1 Point Cloud Visualization and Representation

In many important application areas, point clouds need to be visualized for analysis. Point-based data is obtained, for example, by projecting high-dimensional input data to display space for visual analysis. Principal Components Analysis (PCA) [Jolliffe 2002] or Multidimensional Scaling (MDS) [Cox and Cox 2001] are prominent techniques for mapping high-dimensional vector data to display space. Projection-based point cloud visualization has been applied in many fields, examples including financial data [Dwyer and Gallagher 2004], and database exploration [Pu et al. 2007]. Other applications often relying on point cloud data visualization include the analysis of geo-spatial [Panse et al. 2006; Herrmann and Keim 1998] and bivariate data [Rousseeuw et al. 1999]. Multivariate data can be visualized by so-called scatter plot matrices [Wilkinson et al. 2005]. On the efficiency side, the visualization of massive point

*e-mail: thomas.kalbe@gris.informatik.tu-darmstadt.de
†e-mail: tatiana.tekusova@igd.fraunhofer.de
‡e-mail: tschreck@gris.informatik.tu-darmstadt.de
§e-mail: frank.zeilfelder@gris.informatik.tu-darmstadt.de

cloud data sets may be accelerated by appropriate data structures as presented in [Hopf and Ertl 2003]. Point clouds may be visualized by representing the point clouds with solid shapes, using various geometric constructs. In [Schreck and Panse 2007], the use of minimum bounding discs, boxes, and convex hulls for visual analysis of many point clouds simultaneously was explored. In [Schreck et al. 2008], an algorithm for construction of compact, enclosing shapes for effective point cloud visualization is introduced.

Distance fields are an important data structure with many interesting theoretical and practical properties. According to practical definition, a distance field is a field of scalars representing at each field position the distance between the respective field position and a given surface, or the nearest point of a cloud of points. Distance fields allow representation of surfaces and point sets, and often are the key to elegant and intelligent analysis and transformation approaches [Jones et al. 2006]. For example, distance fields are prominently used for the extraction of skeletons from 2D and 3D shapes, which in turn are useful for shape manipulation, analysis or compression [Cornea et al. 2005]. In our work, we rely on distance field representations, as it elegantly captures the notion of proximity to clouds of points. In conjunction with appropriate visualization, in our approach the distance field representation allows the smooth formation of visual areas from points. Specifically, we will rely on appropriately chosen transfer functions [Preim and Bartz 2007] for visualization of sets of distance fields.

2.2 Visualization of Higher Order Primitives

Our approach is based on bivariate splines defined w.r.t. triangulations of a planar domain. Splines of this type have been studied in the approximation theory literature of the recent years [Chui 1989; Nürnberger et al. 2004; Lai and Schumaker 2007] and rely on the *Bernstein-Bézier-form (BB-form)* [Farin 1986; de Boor 1987] of the polynomial pieces. GPU-based visualizations of bivariate polynomials in BB-form via raycasting have been described in the work of Reis [Reis 2005]. This approach was later extended to terrain rendering with bivariate cubic and quartic splines [Reis et al.].

GPU-implementations of visualizations with piecewise quadratic polynomials in three variables have been described in the work of [Sigg et al. 2006] and [Stoll et al. 2006]. While [Loop and Blinn 2006] use the BB-form of the polynomial pieces, [Kalbe and Zeilfelder 2008] extend this approach to the visualization of trivariate splines, first taking their complex structure into account.

3 Piecewise Quadratic Approximation in Bernstein-Bézier–Form

Our visualization scheme for signed distance field data is based on a quasi-interpolating quadratic C^1 scheme first described by Sorokina & Zeilfelder [Sorokina and Zeilfelder 2005]. In this section, we present the relevant concepts of the technique. We consider bivariate splines on *type-2 triangulations* $\triangle$, also referred to as four-directional meshes. Given is a set of $(n+1)\times(m+1)$ discrete points $V := \{\mathbf{v}_{ij} = (ih, jh) \in \mathbb{R}^2, i = 0,\dots,n, j = 0,\dots,m\}$ in the rectangular domain $\Omega = [0, nh] \times [0, mh]$ with $h > 0$ and associated values $f_{ij} \in \mathbb{R}$. The collection of squares $Q_{ij} = [ih,(i+1)h] \times [jh,(j+1)h]$, where $i = 0,\dots,n-1, j = 0,\dots,m-1$, forms a rectangular partition $\Diamond$ of Ω. The triangulation $\triangle$ is then obtained by subdividing each Q_{ij} into eight triangles, where we use the two diagonals connecting the points $\mathbf{v}_{ij}, \mathbf{v}_{i+1,j+1}$ and $\mathbf{v}_{i,j+1}, \mathbf{v}_{i+1,j}$ (dotted lines in Fig. 1, left), the horizontal edges formed by $(\mathbf{v}_{ij} + \mathbf{v}_{i,j+1})/2$ and $(\mathbf{v}_{i+1,j} + \mathbf{v}_{i+1,j+1})/2$, as well as the vertical edges given by $(\mathbf{v}_{ij} + \mathbf{v}_{i+1,j})/2$ and $(\mathbf{v}_{i,j+1} + \mathbf{v}_{i+1,j+1})/2$ (dashed lines).

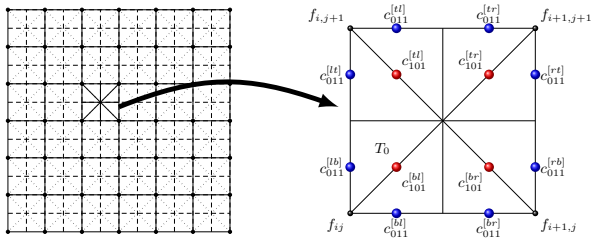


Figure 1: Left: type-2 triangulation defined on Ω. The black dots correspond to vertices $\mathbf{v}_{ij}$ along with their associated values f_{ij}. Right: detail view of a square Q_{ij}. The colored dots correspond to the coefficients from the determining set.

Bivariate splines s defined on $\triangle$ are piecewise polynomials in two variables x, y of (total) degree q, and should be at least continuous, i.e. for all triangles $T \in \triangle$, the spline is defined as the polynomial $p_T := s|_T \in \mathscr{P}_q := \mathrm{span}\{x^\alpha y^\beta : \alpha,\beta \geq 0, \alpha+\beta \leq q\}$, and for any two triangles T_1, T_2 sharing a common edge $E = T_1 \cap T_2 \neq \emptyset$, we assume that $p_{T_1}|_E = p_{T_2}|_E$.

For every triangle $T = [\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2]$, the spline s is given in its piecewise Bernstein-Bézier-form (BB-form) [Farin 1986]

$$s|_T = \sum_{\alpha+\beta+\gamma=q} b_{\alpha\beta\gamma} B_{\alpha\beta\gamma},\ T \in \triangle,\ \alpha,\beta,\gamma \in \mathbb{N}^+,$$

with the BB-coefficients $b_{\alpha\beta\gamma} \in \mathbb{R}$ and the bivariate *Bernstein polynomials* $B_{\alpha\beta\gamma} = q!/(\alpha!\beta!\gamma!)\lambda_0^\alpha \lambda_1^\beta \lambda_2^\gamma \in \mathscr{P}_q$. In case of quadratic polynomials used in this paper, $\alpha+\beta+\gamma = 2$. The *barycentric coordinates* $\lambda_\nu \in \mathscr{P}_1$ are determined by $\lambda_\nu(\mathbf{v}_\mu) = \delta_{\nu,\mu}$, where $\delta_{\nu,\mu}$ is the Kronecker symbol. The BB-coefficients $b_{\alpha\beta\gamma}$ are associated with the 6 domain points $\xi_{\alpha\beta\gamma} = (\alpha\mathbf{v}_0, \beta\mathbf{v}_1, \gamma\mathbf{v}_2)/2$ at the vertices and the center points on each edge of T.

A characteristic property of this scheme is that the quasi-interpolating spline pieces are given in BB-form and can be obtained directly from the data by simple formulae. A subset of the BB-coefficients is associated with the *determining set* [Lai and Schumaker 2007] of each Q_{ij}. Only the coefficients from this set need to be computed from the values f_{ij}. Considering the values f_{ij} on a regular lattice, and expanding terms of the corresponding scheme from [Sorokina and Zeilfelder 2005], we arrive at the following formulae for the coefficients of the determining set on Q_{ij},

$$\begin{aligned}
c^{[bl]}_{101} &= f_{ij} + \frac{1}{8}(f_{i+1,j} - f_{i-1,j} + f_{i,j+1} - f_{i,j-1}) \\
c^{[br]}_{101} &= \frac{1}{8}(f_{i+1,j+1} - f_{i-1,j} - f_{i+1,j-1} + 4f_{ij} + 5f_{i+1,j}) \\
c^{[tr]}_{101} &= \frac{1}{8}(-f_{i+1,j-1} - f_{i-1,j+1} + 2f_{i+1,j+1} + 4(f_{i+1,j} + f_{i,j+1})) \\
c^{[tl]}_{101} &= \frac{1}{8}(f_{i+1,j+1} - f_{i-1,j+1} - f_{i,j-1} + 4f_{ij} + 5f_{i,j+1})
\end{aligned} \tag{1}$$

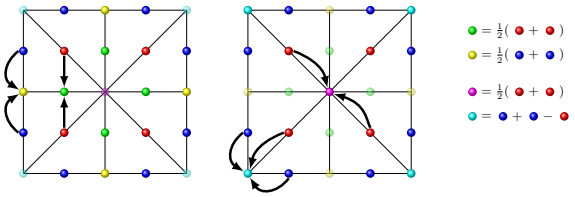


Figure 2: The remaining BB-coefficients for each Q are given by simple averaging formulae following from smoothness conditions.

(red dots in Fig. 1, right), and

$$
\begin{aligned}
c_{011}^{[bl]} &= \frac{1}{16}(f_{i,j+1} - f_{i,j-2} + 2(f_{i+1,j} - f_{i-1,j}) + 3f_{i,j-1} + 13f_{ij}) \\
c_{011}^{[br]} &= \frac{1}{16}(f_{i+1,j+1} - f_{i+1,j-2} - 2f_{i-1,j} + 3f_{i+1,j-1} + 7f_{i+1,j} + 8f_{ij}) \\
c_{011}^{[rb]} &= \frac{1}{16}(f_{i+2,j} - f_{i-1,j} + 2(f_{i+1,j+1} - f_{i+1,j-1}) + 3f_{ij} + 13f_{i+1,j}) \\
c_{011}^{[rt]} &= \frac{1}{16}(f_{i+2,j+1} - f_{i-1,j+1} - 2f_{i+1,j-1} + 3f_{i,j+1} + 7f_{i+1,j+1} + 8f_{i+1,j}) \\
c_{011}^{[tr]} &= \frac{1}{16}(f_{i+1,j+2} - f_{i+1,j-1} - 2f_{i-1,j+1} + 3f_{i+1,j} + 7f_{i+1,j+1} + 8f_{i,j+1}) \\
c_{011}^{[tl]} &= \frac{1}{16}(f_{i,j+2} - f_{i,j-1} + 2(f_{i+1,j+1} - f_{i-1,j+1}) + 3f_{ij} + 13f_{i,j+1}) \\
c_{011}^{[lt]} &= \frac{1}{16}(f_{i+1,j+1} - f_{i-2,j+1} - 2f_{i,j-1} + 3f_{i-1,j+1} + 7f_{i,j+1} + 8f_{ij}) \\
c_{011}^{[lb]} &= \frac{1}{16}(f_{i+1,j} - f_{i-2,j} + 2(f_{i,j+1} - f_{i,j-1}) + 3f_{i-1,j} + 13f_{ij})
\end{aligned}
\tag{2}
$$

(blue dots in Fig. 1, right), where the superscript l stands for left, r for right, b means bottom and t is top. The remaining BB-coefficients are then determined by the C^1 smoothness conditions on two neighboring triangles T_1 and T_2, $T_1 \cap T_2 \neq \emptyset$, and are given by simple averaging formulae (see Fig. 2, right).

We set $\mathbf{v}_0 = (\mathbf{v}_{ij} + \mathbf{v}_{i+1,j+1})/2$ for each of the triangles $T_\sigma \in Q_{ij}, \sigma = 0,1,\ldots,7$. The remaining vertices for each T_σ are then determined in an anti-clockwise fashion, starting from $\mathbf{v}_0$. From this it follows that for each T_σ, there is a one-to-one correspondence between the coefficients $c_{\alpha\beta\gamma}$ from the determining set and the BB-coefficients $b_{\alpha\beta\gamma}$ on T_σ. For example, with $T_0 = (\mathbf{v}_0, (\mathbf{v}_{ij} + \mathbf{v}_{i,j+1})/2, \mathbf{v}_{ij})$, we set $b_{011} = c_{011}^{[lb]}$ and $b_{101} = c_{101}^{[bl]}$. b_{110} is given by the "green" coefficient associated with $(\mathbf{v}_0 + \mathbf{v}_1)/2$ and $b_{200}, b_{020}, b_{002}$ correspond to the "magenta", "yellow" and "cyan" coefficients at the vertices $\mathbf{v}_0, \mathbf{v}_1$ and $\mathbf{v}_2$, respectively (see Fig. 2). The BB-coefficients on the remaining triangles follow from symmetry and rotation.

4 Hardware-Accelerated Visualization

In this section, we describe the implementation details on our real-time visualization algorithm with smooth quadratic splines defined on triangulations Δ. We make use of a combination of textures and vertex buffers to represent the data plus the geometry on the GPU. The actual visualization is finally handled in the graphics pipeline by a set of vertex and fragment programs.

4.1 Organization of Data

Prior to the actual visualization, the data need to be organized in a way that allows for efficient access on the GPU. To do this, the data values f_{ij} are loaded into a two-dimensional, single-channeled floating-point texture of size $(n+1) \times (m+1)$ and transferred to the GPU. The geometry is encoded as a quadrangular mesh of size $n \times m$, where the squares correspond to the Q_{ij}. On each vertex of a square, additional information is given as texture coordinates: For the evaluation of the polynomial pieces s on the GPU (see Sect. 4.3), we need the appropriate barycentric coordinates on s. We can arrive at those if we store the values ν, μ for each of the vertices $\mathbf{v}_{i+\nu,j+\mu}$, $\nu, \mu \in \{0,1\}$ on Q_{ij}. For efficiency, this mesh is stored as a vertex buffer object and also transferred to the GPU (once).

For the later determination of the fragment's color, we store a discretized version of the transfer function (see Sect. 5), in addition. The transfer function is represented as a one-dimensional *RGBA*-texture and is also transferred to the GPU.

4.2 Vertex Shader Details

The purpose of our vertex program is to provide all the information which is necessary to evaluate the polynomial pieces $s|_{T_\alpha}$ on each Q_{ij} in the fragment shader. To do this, we draw the mesh as described in Subsect. 4.1. For each resulting vertex of Q_{ij}, the corresponding values have to be read from the texture storing the data values f_{ij}. Since texture coordinates are usually normalized and centered, the two-dimensional coordinates for a value f_{ij} within the texture are given as $((i+0.5)/(n+1), (j+0.5)/(m+1))$. Note that, according to the formulae (1) and (2), for each Q_{ij} the value f_{ij} along with 15 additional neighboring values need to be known. Reasonable values at the borders, i.e. when $i < 0$ or $i > n+1$ (respectively $j < 0$ or $j > m+1$), are obtained by setting the texture wrapping mode to `CLAMP`. This means that the first and last values on each line (column) are repeated beyond the borders.

The coefficients from the determining set can now be calculated using (1) and (2), and the remaining coefficients are obtained by applying the averaging formulae from Fig. 2. We make the resulting 25 coefficients available in the fragment shader by assigning them to *varying* slots, which are usually used for (automatic) linear interpolation across the primitive. Since the coefficients are the same for each of the four vertices of Q_{ij}, no linear interpolation takes place at this point.

The values ν, μ are also assigned to varying slots. Since ν, μ differ on each vertex, they are linearly interpolated across the quad in the rasterization. Finally, the screen-space coordinates for each vertex are determined by transforming with the current modelview and projection matrices.

4.3 Fragment Shader Details

The rasterizer provides the fragment shader with all the necessary information to determine the final color of the current fragment. To do this, we first have to identify in which of the triangles T_σ, $\sigma = 0,1,\ldots,7$, the fragment is situated. The fragment's quadrant can be easily determined by a simple comparison of coordinates: if $\nu < 0.5$, we are in one of the left quadrants, else we are on the right side. The same holds true for μ and the lower and upper quadrants, respectively. A third comparison gives us the proper triangle within a quadrant. For instance, if we are in the lower left quadrant and $\nu < \mu$, the fragment belongs to the triangle T_0 (see Sect. 3). Otherwise, it has to be the next triangle, which is T_1 in this case.

For each T_σ, there exists a unique mapping from ν, μ into the barycentric coordinates $\lambda_0, \lambda_1, \lambda_2$. As an example, the barycentric coordinates for T_0 are given as $\lambda_0 = 2\nu$, $\lambda_1 = 2(\mu - \nu)$, and $\lambda_2 = 1 - \lambda_0 - \lambda_1 = 1 - 2\mu$. We proceed by collecting the corresponding BB-coefficients $b_{\alpha\beta\gamma}$ for T_σ from the data provided by

the vertex shader in order to evaluate the polynomial $s|_{T_\sigma}$ at the current fragment. The value of s at the current fragment is calculated from the bilinear product

$$(\lambda_0,\lambda_1,\lambda_2)\begin{pmatrix} b_{200} & b_{110} & b_{101} \\ b_{110} & b_{020} & b_{011} \\ b_{101} & b_{011} & b_{002} \end{pmatrix}\begin{pmatrix} \lambda_0 \\ \lambda_1 \\ \lambda_2 \end{pmatrix}.$$

This accounts for a vector-matrix product, followed by a scalar product, both being native instructions on the GPU. Finally, we need to determine the color of the fragment. To do this, the value of s is normalized and used as an index into our current transfer function *RGBA*-texture (see Subsect. 4.1).

4.4 Discussion and Analysis

Alternatively to the approach described above, we can think of different strategies for the spline visualizations. For example, the mesh geometry can be represented directly as triangle meshes. As a consequence, the fragment program complexity is significantly reduced since the determination of the triangle the fragment belongs to can be ommited. At the same time, the costs for the calculations and texture accesses in the vertex shader increase, since we have to process at least 10 vertices (using triangle fans) for each Q_{ij} instead of four when we use quads. Geometry processing, which is often a bottleneck in GPU algorithms, also increases. Using *geometry shaders*, which are introduced as a new stage in the graphics pipeline, we can represent the geometry as a set of points $\mathbf{v}_{ij}$. For each point, the BB-coefficients are calculated in the vertex shader and in the following, the geometry stage generates the respective triangles. A significant disadvantage of this approach is that processing units for geometry programs compete with the other stages (vertex and fragment), since the total number of processing units on the graphics board is limited. This suggests that our first approach takes the best advantage of the graphics pipeline in most of the common cases.

For comparison reasons, we also implemented a simple bilinear interpolation technique. Here, we draw a single quad of size $(n+1)\times(m+1)$, textured with the data values representing the distance field. At the intermediate positions, we rely on simple bilinear interpolation, which is automatically performed by the texture unit. Again, we use the given value as an index into our transfer function texture. See Subsect. 6.3 for a visual comparison of the two approaches.

5 Visualization

In this section, we describe our point-cloud visualization approach. For any given point cloud, we start by calculating a discrete distance field. By relying on the approximation scheme and implementation described in Sections 3 and 4, we are able to efficiently and smoothly zoom in or out at arbitrary scales into the distance field. We approach the visualization of the distance field representation by applying the concept of *transfer functions* known prominently from volume visualization [Preim and Bartz 2007]. We consider here a set of transfer functions for visualization of distance fields, by mapping distance field values to the *Red*, *Green*, *Blue* and *Alpha* (or opacity) channels in *RGBA* color space. The role of the *R*, *G*, and *B* color channels is to assign a color gradient to each field of distances. The alpha (or opacity) channel is important for blending multiple distance field visualizations. Based on a varying set of transfer functions applied on a given set of distance fields, a plethora of different effects may be achieved. We allow two modes of operation for visualization. In *interactive mode*, the user may either choose a transfer function from a set of predefined template functions or, if necessary, interactively adjust the template functions to specific needs. The user is also allowed to freely draw suitable transfer functions over the cumulative histogram calculated from a given distance field. Once a transfer function is updated or newly created, it can be stored for future use. Fig. 3 shows the user interface we provide to this end. We also provide an *automatic mode* for transfer function specification. In this mode, the distance field data is automatically analyzed, and a transfer function, heuristically based on the data to be visualized, is assigned. We next detail on the interactive and automatic generation of transfer functions.

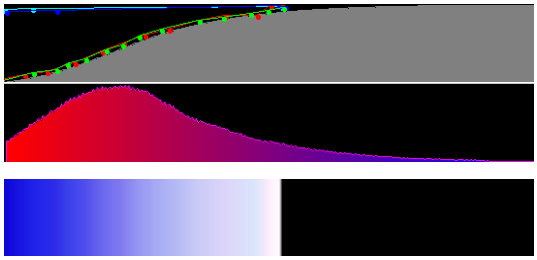


Figure 3: Display and editing of transfer functions. Top: cumulated histogram of distances of a given distance field. It serves also as the background for the interactive transfer function editor. Middle: non-cumulated distance histogram. Bottom: preview of the resulting color gradient, giving immediate feedback to the user when selecting or editing the transfer functions to apply.

5.1 Interactive Transfer Function Assignment

We identify different types of transfer functions useful for different analysis tasks; we provide a number of transfer function templates from which the user may chose in interactive mode. For example, simple unichromatic transfer functions may assign decreasing values of a given color channel (or combination thereof) to increasing distance values. The typical result is a radially decaying visualization, like shown in Fig. 5 (b) (respective transfer function sketched in Fig. 4, top).

Visual analysis of point clouds at different levels of abstraction is possible by setting the transfer functions to constant values only within certain distance bands. Depending on the distance band in which the functions are positive, the notions of *near*, *middle*, and *far* proximity areas covered by the points are supported. Fig. 5 (c) illustrates three proximity areas assigned by three different short-banded transfer functions (sketched in Fig. 4, bottom).

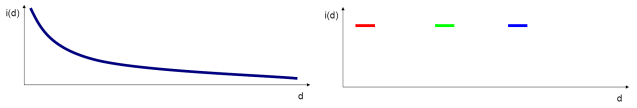
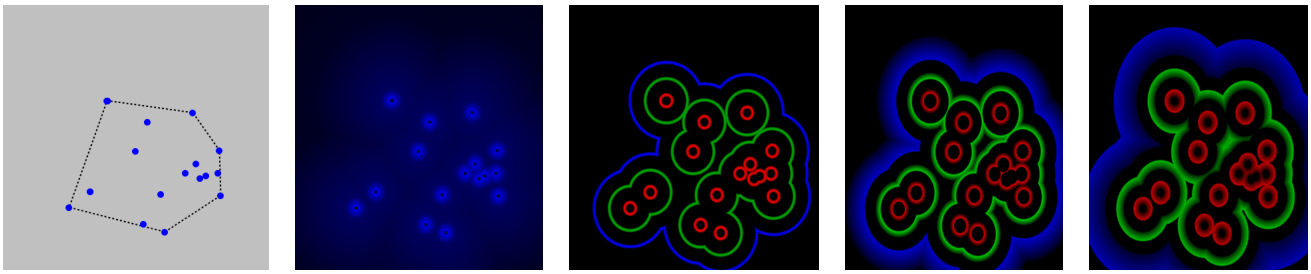


Figure 4: Simple mono- (left) and multi-chromatic (right) transfer functions applied in Fig. 5 (b) and (c).

Different effects may be achieved by appropriate setting of the transfer functions, and these can be exploited for effective visualization construction. For example, by considering the opacity channel, it is possible to discretely or continuously hide certain distance regions, achieving explicit transparency effects. Also, by adjusting the slope of the transfer function, certain shading effects are possible and can be selectively employed to enhance the perception of background-foreground boundaries, increasing the perceived contrast between areas in the distance field as Fig. 5 (d and e) illustrates. Further example transfer functions will be discussed in Sect. 6.



(a) (b) (c) (d) (e)

Figure 5: Point cloud with associated convex hull (a) and monochromatic distance field visualization (b). The blending of individual transfer functions applied on the RGB color channels allows the construction of different areas around the point clouds (c). Outward and inwards oriented shading effects may be achieved by employing transfer functions of positive or negative slopes, respectively (d and e).

5.2 Automatic Transfer Function Assignment

Based on the data analysis requirements at hand, it is possible to derive transfer functions in a data-dependent way, highlighting certain data features deemed important. This usually requires knowledge of the underlying application - typically, a data analysis function detects interesting features, and appropriately configures the transfer functions. Basically, two different types of analysis functions are possible. *Histogram-based* analysis functions analyze the overall distribution of distances present in the distance field image, disregarding spatial information on the localization of individual distance field values. In contrast, *spatial analysis* functions also consider spatial properties of the distance field, in forming the overall transfer function.

In our visualization, we currently implement a simple histogram-based analysis function, which yields histogram-equalizing transfer functions for given distance fields. The basis for our analysis function is the standard histogram equalization image processing technique [Gonzalez and Woods 2007]. We alter the characteristic of a selected channel of the transfer function (R, G, B, or opacity), adjusting certain properties of the function (e.g., its slope, amplitude, or maximum width) in accordance with the underlying data distribution. For example, if the data points are dominantly distributed closely to each other, then higher color and/or opacity contrast is dedicated to the small distances interval, to emphasize smaller values in the distance field (see Fig. 6 for an illustration).

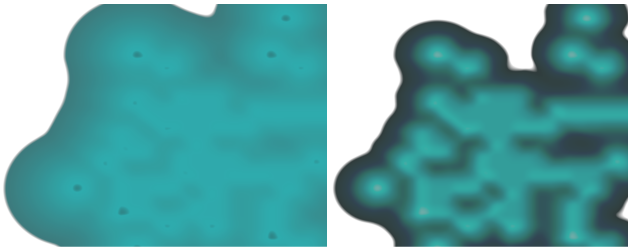


Figure 6: Basic (left) and equalized (right) transfer functions.

Algorithmically, we calculate the cumulative point distance distribution and in accordance with the standard histogram equalization approach [Gonzalez and Woods 2007], recalculate the transfer function values. The selection of the basis function is task and application specific (see Sect. 6). For example, in geographic applications, a maximum distance from a city can be used as a parameter for setting the size of the transfer function for visualizing the distance field around that city. Or in cluster analysis, point clusters can be identified by using automatic maximum within cluster distance.

5.3 Visualization of Multiple Distance Field Layers

In extension of the visualization of a single point cloud, we also support visualizing *combinations* of distance fields. For (layered) combinations of distance fields, not only the distance field-specific distance distribution properties are relevant, but also, the 'vertical' aspects of the layered distance fields. The latter allows the inclusion of e.g., interference effects which the different point clouds may exercise upon each other (see Fig. 7 for an example).

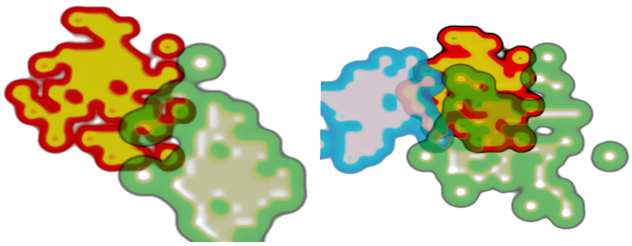


Figure 7: Blending two (left) and three (right) distance field layers.

6 Applications

In this Section, we apply our technique on different data sets, demonstrating its versatility and effectiveness and compare our results with the standard bilinear interpolation approach.

6.1 Visual Cluster Analysis

Fig. 8 shows an example of a point cloud, which we would like to analyze for potential clusters. We apply transfer functions for the R, G, and B color channels which are defined around three distinct intervals on the distance histogram. While the red band pins down individual points, the green band is defined around a larger interval of distances, consequently outlining a region more apart of the individual points. Finally, the blue band is defined at the largest of the three intervals, visualizing the coarse outlines of the cloud. In the visualization, we can easily observe *point groupings on various distance levels*. The visualization is useful for interactive cluster analysis of objects (points) by interactively varying the transfer function. The display, being updated in real time, communicates the clustering structure at the respective abstraction levels. The analyst can thereby generate and validate hypotheses regarding plausible clustering structures given in a given data set.

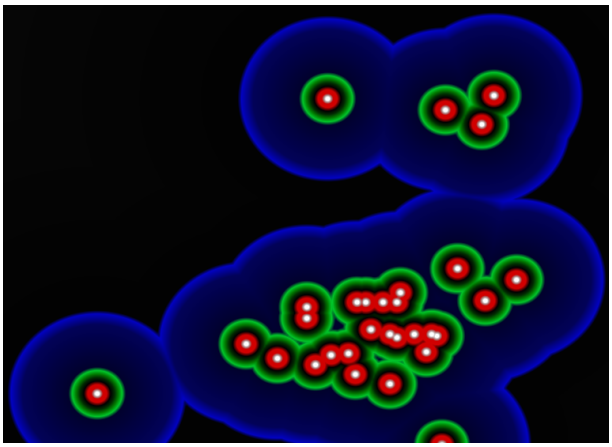


Figure 8: Visual analysis for radial clusters on varying levels of abstraction.

6.2 Geo-spatial Analysis

Geo-spatial data analysis includes techniques studying entities using their geographic properties. Modern spatial analysis examines large amounts of data using statistical and computational modeling. Spatial analysis models are broadly used in urban planning applications such as road planning, migration analysis, or other socio-economic analysis scenarios. We here focus on an analysis of commuting flows. The aim of the analysis is to find out variables that determine the amount of people commuting to and from towns in a selected area. Commuting flows are often modeled using *gravity models* which are based on physical Newton's law. For a given location the attraction to a city is set proportional to the mass of the city (e.g. population, number of employed persons, etc.) and inversely proportional to its distance to the city [Torrens 2000]. Although modern spatial analysis offers effective analytical tools for modeling commuting flows (measured by statistical indicators), by now it has been very difficult to visually present results of the analysis or to visualize and analyze the resulting data interactively. In this paper, we threrefore use spatial locations and socio-economic data for visual analysis of commuting attraction forces in the western part of Slovakia (see Fig. 9, left). We have collected data for 27 district capitals from Slovak Statistical Office. We use population size as a measure of the cities' respective mass (attraction), and their geographic position for approximation of the distances between cities.

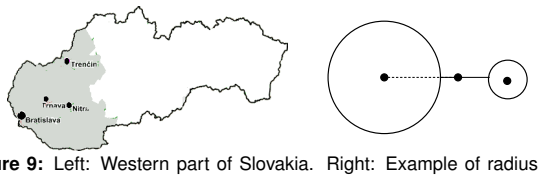


Figure 9: Left: Western part of Slovakia. Right: Example of radius dependency on city population.

We apply the presented point cloud visualization to visually explore the attraction forces of the cities. We represent the cities as points, and the attraction forces of the cities as distance fields. More specifically, we consider the location of each city as the center of a disc with radius proportional to the population of the corresponding city (see Fig. 9, right).

In Fig. 11, upper right, we show a model of commuting flow attraction for the analyzed Slovak cities. In this view, the analyst can easily recognize the distribution of attraction forces around all examined cities. The shape of the attraction areas as well as the decaying magnitude of the attraction forces around the cities is clearly visible. On demand, the analyst may zoom into a specific part of the view to examine the distribution of attraction forces in more detail. The spline-based distance field approximation enables a detailed view on the distribution of forces as it better reflects the non-linearities of the gravity model than the bilinear interpolation approach (see Subsect. 6.3 for a comparison of the techniques).

As can be seen from Fig. 11, upper right, the shapes of attraction areas are strongly dependent on the spatial locations of the cities. An isolated city attracts a radially distributed, radially decreasing number of commuters (see Fig. 10, left). However, when two or more entities are located close to each other, their commuting attraction forces interfere. The resulting model *shows the areas of maximum attractions of commuters to each city* (see Fig. 10, right). The areas are here no longer circular. The point cloud visualization approach introduced in this paper allows addressing interesting spatio-analytical questions. Using interactive adjustment of the transfer function and blending of multiple point clouds, the users are able to efficiently examine the underlying data. We next sketch interesting exemplary analysis cases.

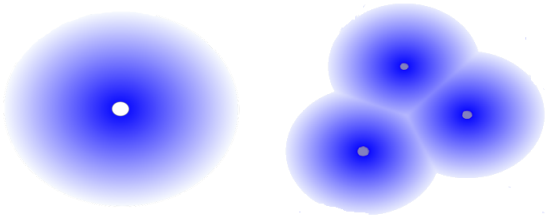


Figure 10: Visualization of the shape of the attraction forces, according to the gravity model for one city (left) and for multiple neighboring cities (right).

A transfer function with stepwise coloring - using piecewise constant red and green coloring and constant blue color (see Fig. 11, bottom left) allows showing *different attraction zones around the cities* (see Fig. 11, upper left). For analysis of the *model-based dependency of the attraction forces on the distance*, a transfer function with linear increasing green and red colors and constant blue color (see Fig. 11, bottom right) is suitable. Using this function, we can see how the commuting flow attraction of the cities is distributed. Both views also allow the user to view the borders of the commuting areas around the cities (see Fig. 11, upper right).

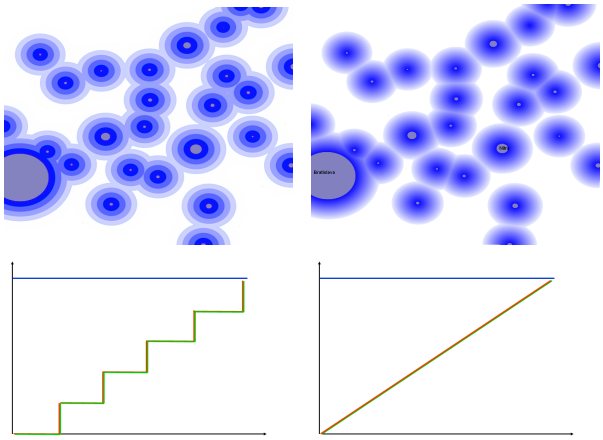


Figure 11: Visualization of commuting patterns for cities in Western Slovakia. Left: Commuting zones. Right: model-based commuting pattern. Top: Distance field visualization. Bottom: Transfer functions applied.

In general, the magnitude of the attraction forces of region capitals differs significantly from those of the smaller cities. This is obvious from Fig. 12, which shows the attraction fields of the largest three, and all the remaining cities separately. To effectively analyze the interaction between such groupings, *blending of multiple point cloud visualizations* is applied. This better supports the perception of different data groups, as opposed to visualizing a combined distance field representation. Also, this approach allows for much more flexibility in designing the visualization with respect to the areas of interference. We illustrate several possible application scenarios. The

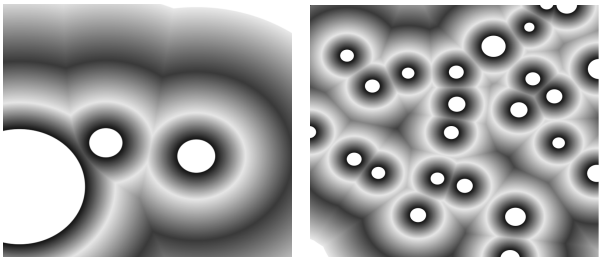


Figure 12: Large metropolitan areas (left) and smaller cities (right).

top row in Fig. 13 shows the *attraction area of the region capitals when analyzed separately* (the foreground picture) and its commuting pattern when considering the influence of the other analyzed cities. It clearly shows that the commuting agglomerative area of region capitals is covering small cities situated close to the capitals. In the overlaid areas both forces influence the commuting behavior of people living there. Using various transfer functions for the foreground layer we can analyze the effect of region capital attraction from different view points. For example, by looking at the inferred view, these forces are lower than the attraction of these cities in smaller areas around them.

The bottom row of Fig. 13 shows several rather *illustrative visualizations* of the same point sets. The transfer functions were set so to give a kind of 'cellular' or 'organic' style to the visualization. In Figures 13 (d) and (e), the large cities form a rather closed plane of attraction, clearly dominating the smaller cities. This might help to better communicate the role of the larger cities. Fig. 13 (f) finally shows the most abstract of our transfer functions. We transfer shades of blue to the smaller cities layer, while the larger cities are visually aggregated by a green veil. Note that we set opacity to a low value for the center areas of the latter layer, allowing a look-through effect onto the layer of smaller cities. While such visualizations might introduce a rather strong element of abstraction, we argue that such visualizations might be well suited for usage in popular media information design, an application domain for Information Visualization often overlooked [Ericson 2007].

6.3 Evaluation

In order to evaluate the spline-based approximation, we compare it with a standard bilinear interpolation approach. In order to show the difference between the two approaches, we present visualization of two point clouds of lower resolution with different transfer functions using spline quasi- and bilinear interpolation, respectively (see Fig. 14). The pictures on the left-hand side were rendered using bilinear interpolation and those on the right-hand side using spline-based approximation. As can be seen from the figure, in case of lower resolution distance fields, the spline-based interpolation nicely reconstructs the true distance field distribution around point clouds. In contrast, the bilinear approach causes squarified (instead of circular) distance field visualization. The difference between the two techniques is dependent on the resolution of the underlying distance field data. The lower the resolution the larger the advantage of using spline-based approximation.

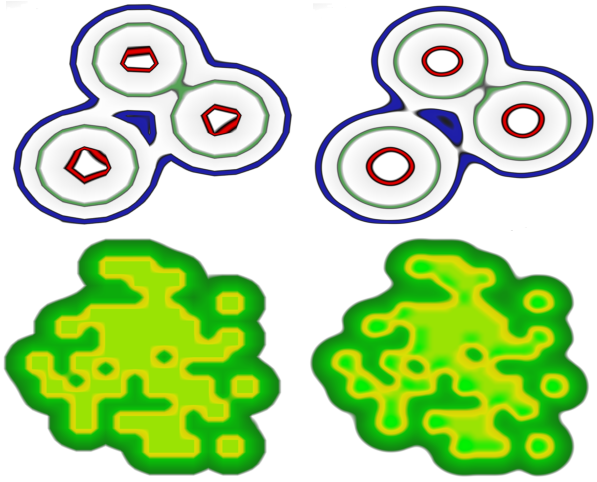


Figure 14: Visual comparison (50×50 data points). Left: Bilinear interpolation. Right: Quasi-interpolating splines.

7 Conclusions

We described a novel scheme for smooth approximation of distance fields using a GPU implementation. Our spline-based visualization scheme for distance field data provides smooth and efficient approximation of distance field data at arbitrary resolutions. In spite of low total degree approximation, we demonstrated the advantages of this approach in comparison to the bilinear interpolation scheme. We visualize the distance field representation by mapping distance values to channels in *RGBA* color space. Interactive or automatic variation of the transfer functions allows the user to achieve a plethora of different effects useful for data analysis in various application areas. In extension of the visualization of a single point cloud, we also support combinations of distance fields.

We have shown that our approach is suitable for application in point cloud analysis (cluster analysis) and in the broad subject of geo-spatial analysis models. The spline-based distance field approximation enables a realistic detailed view on the distribution of the spatial forces as it better reflects the non-linearities of the spatial analysis models then the bilinear approach. Furthermore, using various shapes of transfer functions the analyst can easily create appropriate views allowing her to effectively examine the data. The blending of multiple point clouds enhances the analysis of the spatial models by showing various model scenarios at the same time.

To address additional analytical scenarios, we would like to implement two variations of semantic zooming. The shape of the transfer function will be automatically adjusted to the given zooming level, to support application-based detail or context analysis. In addition, according to the zoom level, we could show varying levels of details by adding or removing smaller cities in the view. This approach should be especially suited to support the visual exploration of very large geo-spatial data sets. In addition, automatic generation of transfer functions can be extended for (layered) combinations of distance fields. In this case not only the distance field-specific distance distribution properties are relevant, but also, the 'vertical' aspects of the layered distance fields.

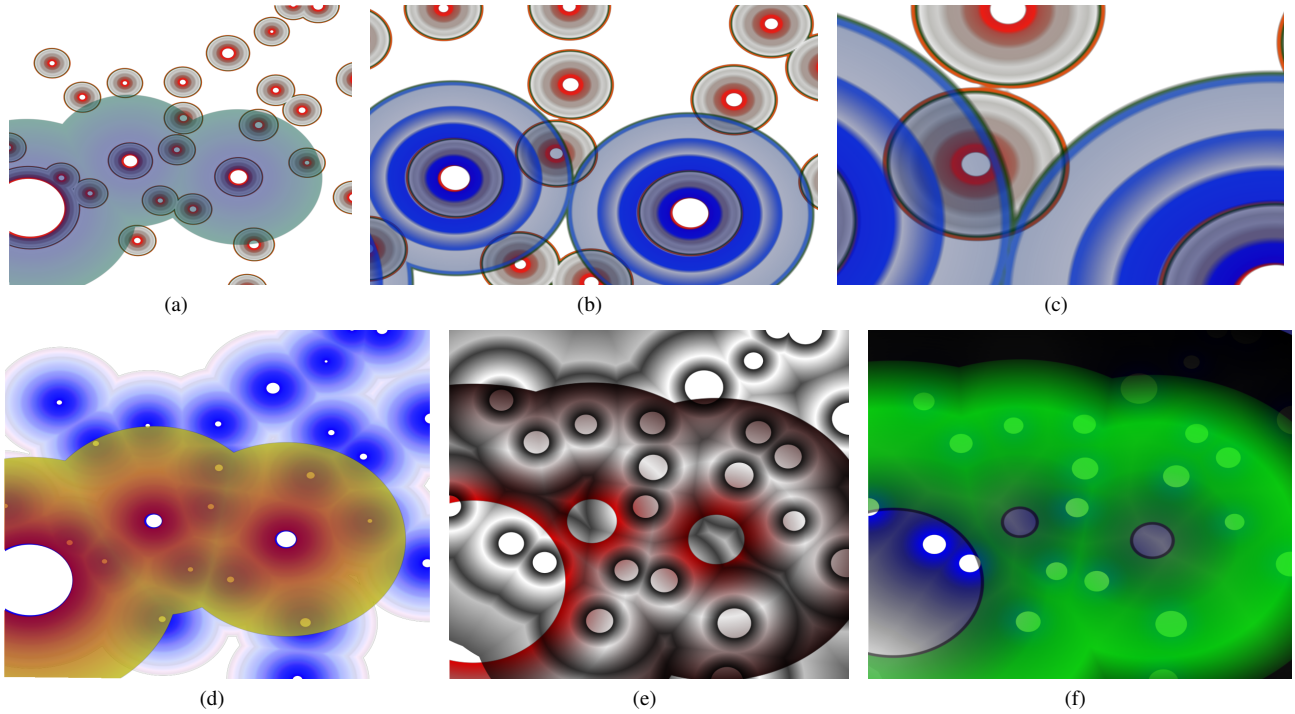


Figure 13: Blending of two layers of distance fields, representing the groups of large and smaller cities. The top row shows the interfering commuter attraction forces excerted by the large cities, overlaid over those of the smaller cities. Zooming allows for detail analysis (b,c). We defined several transfer functions supporting also the *illustrative* visualization of such diagrams, e.g., proposed for usage in popular media.

References

CHUI, C. 1989. *Multivariate Splines*. CBMS 54, SIAM.

CORNEA, N. D., SILVER, D., AND MIN, P. 2005. Curve-skeleton applications. *vis 00*, 13.

COX, M., AND COX, M. 2001. *Multidimensional Scaling*. Chapman and Hall.

DE BOOR, C. 1987. B-form basics. In *Geometric Modelling*, SIAM, G. Farin, Ed., 131–148.

DWYER, T., AND GALLAGHER, D. 2004. Visualising changes in fund manager holdings in two and a half-dimensions. *Information Visualization 3*, 4, 227–244.

ERICSON, M. 2007. Visualizing data for the masses. In *Proc. IEEE Symposium on Information Visualization*. Keynote.

FARIN, G. 1986. Triangular Bernstein-Bézier patches. *CAGD 3*, 2, 83–127.

GONZALEZ, R., AND WOODS, R. 2007. *Digital Image Processing*, 3rd ed. Prentice Hall.

HERRMANN, A., AND KEIM, D. 1998. The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data. In *Proc. IEEE Visualization*, 181 – 188.

HOPF, M., AND ERTL, T. 2003. Hierarchical splatting of scattered data. In *Proc. IEEE Visualization*.

JOLLIFFE, I. 2002. *Principal Components Analysis*, 3rd ed. Springer.

JONES, M. W., BAERENTZEN, J. A., AND SRAMEK, M. 2006. 3d distance fields: A survey of techniques and applications. *IEEE Trans. on Visualization and Computer Graphics 12*, 4, 581–599.

KALBE, T., AND ZEILFELDER, F. 2008. Hardware-Accelerated, High-Quality Rendering Based on Trivariate Splines Approximating Volume Data. *Computer Graphics Forum 27*, 2.

LAI, M.-J., AND SCHUMAKER, L. 2007. *Spline functions on Triangulations*. Cambridge Universiy Press.

LOOP, C., AND BLINN, J. 2006. Real-time GPU rendering of piecewise algebraic surfaces. *ACM Trans. on Graphics 25*, 3 (July), 664–670.

NÜRNBERGER, G., SCHUMAKER, L. L., AND ZEILFELDER, F. 2004. Lagrange interpolation by c^1 cubic splines on triangulated quadrangulations. *Adv. Comput. Math. 21*, 3-4, 357–380.

PANSE, C., SIPS, M., KEIM, D., AND NORTH, S. 2006. Visualization of geo-spatial point sets via global shape transformation and local pixel placement. *IEEE Trans. on Visualization and Computer Graphics 12* (September-October), 749–756.

PREIM, B., AND BARTZ, D. 2007. *Visualization in Medicine*. Morgan Kaufmann.

PU, J., KALYANARAMAN, Y., JAYANTI, S., RAMANI, K., AND PIZLO, Z. 2007. Navigation and discovery in 3d cad repositories. *IEEE Computer Graphics and Applications 27*, 4, 38–47.

REIS, G., ZEILFELDER, F., HERING-BERTRAM, M., FARIN, G., AND HAGEN, H. GPU-based High-Quality Rendering of Quartic Splines, submitted.

REIS, G. 2005. Hardware based Bézier patch renderer. In *Proceedings of IASTED Visualization, Imaging, and Image Processing (VIIP) 2005*, 622–627.

ROUSSEEUW, P., RUTS, I., AND TUKEY, J. 1999. The bagplot: A bivariate boxplot. *The American Statistician 53*, 4, 382–387.

SCHRECK, T., AND PANSE, C. 2007. A new metaphor for projection-based visual analysis and data exploration. In *IS&T/SPIE Conf. on Vis. and Data Analysis*.

SCHRECK, T., SCHUESSLER, M., ZEILFELDER, F., AND WORM, K. 2008. Butterfly plots for visual analysis of large point cloud data. In *Proc. WSCG 2008*.

SIGG, C., WEYRICH, T., BOTSCH, M., AND GROSS, M. 2006. GPU-based ray-casting of quadratic surfaces. In *Proceedings Eurographics*, 59–65.

SOROKINA, T., AND ZEILFELDER, F. 2005. Optimal quasi-interpolation by quadratic C^1 splines on four-directional meshes. In *C. K. Chui, M. Neamtu, and L. L. Schumaker, (Eds.), Approximation Theory XI: Gatlinburg 2004*, 423–438.

STOLL, C., GUMHOLD, S., AND SEIDEL, H.-P. 2006. Incremental raycasting of piecewise quadratic surfaces on the GPU. In *Proc. IEEE Symp. Inter. Raytr.*, 141–150.

TORRENS, P. M. 2000. *How Land-Use Transportation Models Work*. No. 20. Centre for Advanced Spatial Analysis.

WILKINSON, L., ANAND, A., AND GROSSMAN, R. 2005. Graph-theoretic scagnostics. In *Proc. IEEE Symposium on Information Visualization*, 157 – 164.