Check for updates

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

Volume 28 (2009), Number 3

Efficient and Adaptive Rendering of 2-D Continuous Scatterplots

Sven Bachthaler and Daniel Weiskopf

VISUS, Universität Stuttgart

Abstract

We extend the rendering technique for continuous scatterplots to allow for a broad class of interpolation methods within the spatial grid instead of only linear interpolation. To do this, we propose an approach that projects the image of a cell from the spatial domain to the scatterplot domain. We approximate this image using either the convex hull or an axis-aligned rectangle that forms a tight fit of the projected points. In both cases, the approach relies on subdivision in the spatial domain to control the approximation error introduced in the scatterplot domain. Acceleration of this algorithm in homogeneous regions of the spatial domain is achieved using an octree hierarchy. The algorithm is scalable and adaptive since it allows us to balance computation time and scatterplot quality. We evaluate and discuss the results with respect to accuracy and computational speed. Our methods are applied to examples of 2-D transfer function design.

Categories and Subject Descriptors (according to ACM CCS):
Computer Graphics [I.3.3]: Picture/Image Generation - Display algorithms—

1. Introduction

Data sets for scientific visualization are commonly defined continuously, e.g. by applying interpolation or reconstruction techniques to the data sampled on a grid. A helpful means for analyzing such scientific data is the well-known scatterplot. However, scatterplots only make use of the discrete data samples — they ignore the spatial relationship of neighboring data points and also the interpolated data between samples. This drawback was overcome by our recently published concept of continuous scatterplots [BW08]. Continuous scatterplots make use of continuously defined data by drawing the scatterplot in a dense way — instead of rendering discrete glyphs, the density of the data samples is drawn in the scatterplot domain.

However, the practical implementation of 2-D continuous scatterplots is subject to a few limitations. The first limitation is related to the interpolation method used for computing the density in the scatterplot domain. The original implementation of continuous scatterplots follows the idea of projected tetrahedra [ST90]. Within a tetrahedron, barycentric interpolation is applied; this linear interpolation leads to a simplified computation of scatterplot density. For regular grids, the hexahedral cells are decomposed into five tetrahedra to compute a continuous scatterplot. The drawback of this approach, however, is that native trilinear interpolation of regular grids is not supported. The second limitation of the original continuous scatterplot approach is related to the time needed for computing the overall density. Splitting a regular grid into tetrahedra introduces additional overhead; finding the density for average-sized data sets (e.g. in the size range of 128^3) is very time-consuming and may require up to several minutes. Since the original scatterplot algorithm is based on projected tetrahedra, its run-time behavior is similar and does not scale well for large data sets given on regular grids.

The goal of this paper is to overcome the above drawbacks of the tetrahedra-based scatterplot computation. Our new approach makes use of ideas of adaptive grid subdivision and hierarchical octree structures to efficiently approximate the scatterplot image. In this way, continuous scatterplots can be computed for arbitrary interpolation or reconstruction functions applied to the data set on the spatial grid. Compared to the original algorithm for computing a continuous scatterplot, the new approach is simpler and easier to implement.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.
Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

Our method natively supports regular grids, i.e. triangulation is no longer necessary. This is one of the reasons why the time needed to compute the density for a continuous scatterplot is greatly reduced. Furthermore, the new approach is adaptive, and the approximation error introduced when estimating the density contribution can be controlled by a single parameter. This parameter enables the user to balance computation time and scatterplot quality. The direct support for regular grids, which are most popular in scientific visualization, and the high rendering speed allows one to seamlessly integrate continuous scatterplots into typical interactive visualization systems.

2. Related Work

Using a scatterplot or histogram to visualize statistical data is a popular and widely accepted approach. An extensive overview of scatterplots, histograms, frequency diagrams, and similar plotting techniques can be found in the books [CCT83, Utt04]. The conventional way of drawing a scatterplot is to use discrete glyphs that indicate the location of data samples in scatterplot space. Here, continuously defined data is treated the same way as discrete data — the additional information implicitly contained in the continuous reconstruction is not used for the visualization. This limitation was removed in our previous work [BW08], where we showed that scatterplots can be drawn continuously by finding the density of the data samples in scatterplot space. The basis of that work is a mathematical model that can be used for any dimensionality of the input data and the corresponding generalized scatterplot. A related mathematical approach was independently used by Scheidegger et al. [SSD*08] to compute isosurface statistics, i.e. density histograms that represent the isosurface distribution in 3-D scalar fields.

As mentioned before, this paper uses the mathematical model of continuous scatterplots [BW08]. Our extension concerns the efficient computation of the special, yet most relevant case of 2-D scatterplots from data sets given on 3-D regular grids. In contrast to the original implementation, we directly support regular grids, trilinear interpolation, and fast, adaptive rendering. Our approach utilizes the idea of subdivision or refinement in order to approximate scatterplot rendering in a controlled way. The general idea of subdivision is common in visualization and computer graphics (see, e.g., its popularity in subdivision surfaces [WW01]), but the specific application to scatterplot rendering requires modifications of the subdivision criteria and rules. Our algorithm additionally uses an octree to accelerate the processing of the input data. We apply a concept related to the branch-on-need-octree by Wilhelms and van Gelder [WG92]. Similar to the branch-on-need-octree and the loosely related span-space methods [CMPS96, SHLJ96], extremal values of a cell (minima and maxima) are used to quickly identify the properties of the cell with respect to scatterplot rendering. The spatial hierarchy of the octree is exploited by propagating the min-max values up the octree during its construction and using the min-max values of inner octree nodes during rendering.

3. Mathematical Approximation Model

This section first summarizes the mathematical description of continuous scatterplots [BW08] and then adds the new approximation model for a fast computation of scatterplots. We use the same notation as in the original article [BW08] to support the readability of this paper.

Continuous scatterplots need two different domains: the n-dimensional domain of the input field and the m-dimensional domain of the scatterplot. The first one is denoted *spatial domain* and, in our case, can be assumed to be three-dimensional (i.e. $n = 3$). The second kind of domain is called *data domain*, representing the multi-attribute data values of the input data set. In the case of this paper, $m = 2$, which yields a 2-D scatterplot. The relationship between spatial domain and data domain is described by the map $\tau : \mathbb{R}^n \to \mathbb{R}^m$. The map τ represents the data-set values.

A continuous scatterplot needs to render a density function σ defined on the data domain:

$$\sigma : \mathbb{R}^m \longrightarrow \mathbb{R}, \quad \xi \longmapsto \sigma(\xi) \,. \tag{1}$$

As shown in Section 3.4 of [BW08], the overall density at location $\xi_0 = (\xi_1, \xi_2)$ in the data domain is computed according to

$$\sigma(\xi_0) = \int_{\tau^{-1}((\xi_1,\xi_2))} \frac{s(x)}{|\mathrm{Vol}(D\tau)(x)|} \,\mathrm{d}x \,, \tag{2}$$

where $s(x)$ represents the (given) density in the data domain. Typically, $s(x)$ is chosen constant. The 2-D area $|\mathrm{Vol}(D\tau)|$ in (2) is spanned by the gradients $\partial\xi_1/\partial x$ and $\partial\xi_2/\partial x$, and can be expressed as the cross product of the two gradients:

$$|\mathrm{Vol}(D\tau)| = \left|\left| \frac{\partial\xi_1}{\partial x} \times \frac{\partial\xi_2}{\partial x} \right|\right| \,. \tag{3}$$

The problem is that (2) and (3) require a complicated integration of a potentially varying integrand $s(x)/|\mathrm{Vol}(D\tau)(x)|$. Moreover, the integration domain $\tau^{-1}((\xi_1,\xi_2))$ is the intersection of two isosurfaces within the 3-D spatial domain (corresponding to isovalues ξ_1 and ξ_2). Both issues can be directly resolved for the special case of barycentric interpolation in tetrahedral cells, as exploited in [BW08]. However, for general interpolation or reconstruction functions τ, the computation of (2) and (3) is non-trivial and might not even have an analytic solution.

Therefore, we apply the following approximation strategy. This approximation starts with the observation that generic continuous scatterplots can be derived from an abstract version of mass conservation (see Section 3.1 of [BW08]):

$$M = \int_V s(x) \,\mathrm{d}^n x = \int_{\Phi=\tau(V)} \sigma(\xi) \,\mathrm{d}^m \xi \,. \tag{4}$$

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

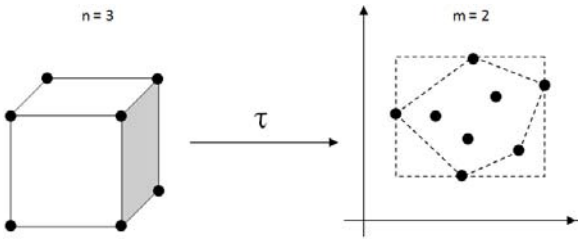

Figure 1: *Illustration of the approximation of* Φ. *Projecting a hexahedron from the spatial domain to the data domain results in eight points located in the data domain. The stippled lines indicate the shape that is constructed to represent* Φ*: an axis-aligned rectangle or the convex hull of the eight points.*

Here, M describes the "mass" of virtual material in either the spatial domain (left integral) or the data domain (right integral). The term V describes any volume in the spatial domain, and $\Phi = \tau(V)$ is the corresponding volume in the data domain. For the special case of 2-D scatterplots of 3-D data sets, $n = 3$ and $m = 2$.

Equation (4) is used to approximate the density σ by assuming constant distributions of densities s and σ inside those volumes:

$$M \approx sV \approx \sigma\Phi\,. \tag{5}$$

Then, we obtain

$$\sigma \approx \frac{sV}{\Phi}. \tag{6}$$

Typically, s is constant for the whole data set and, thus, can be assumed to be 1. Then, (4) is reduced to

$$\sigma \approx \frac{V}{\Phi}. \tag{7}$$

With this equation, we can directly compute the density in the data domain. The problem is reduced to determining the volumes V and Φ. The approach is to consider a volume V in the spatial domain (i.e. some kind of discretization of 3-space) and apply the transformation τ to the volume V, which yields the volume Φ in the data domain. An illustration of this projection step is shown in Fig. 1. Please note that we use a simplified notation in which V and Φ denote the actual geometry of a region or its corresponding volume or area, depending on the context.

This approach lends itself to a control of the approximation error. Depending on the extent of Φ in the data domain, we have a direct measure for the maximum error in the data domain (with respect to the error of the projected footprint). If the extent of Φ exceeds a given error threshold, then V needs to be reduced to keep the approximation error within the specified error bounds. Once the approximation-error requirement is met, we can approximate the density contribution of the data values from V by calculating the ratio V/Φ.

The only remaining issue is to compute the size of V and Φ. A solution to this problem is presented in the following section.

4. Algorithm

We propose two variants of an algorithm that approximates the density in the data domain. Both variants are based on the idea expressed in the previous section: subdivision in the spatial domain controls the approximation error of the density in the data domain. This is done in two steps:

1. the volume V is projected to the data domain,
2. the size of the corresponding volume Φ is estimated.

These two steps are repeated until the extents of Φ are below a user-specified threshold. In contrast to the original continuous scatterplot approach, both algorithmic variants do not tetrahedralize the spatial domain. Instead, the regular grid of the data set is used to form hexahedra, each of them storing eight multi-variate data samples at the corners. As in the original continuous scatterplot approach, the overall density σ in the data domain can be found by linear superposition of all cell contributions. Therefore, our algorithm can construct σ by considering one cell after another.

The two versions of the algorithm differ in the way how the size of the volume Φ in the data domain is computed. The first version uses a convex-hull approach to calculate the volume of Φ accurately, whereas the second version approximates Φ by an axis-aligned rectangle that encompasses the exact shape of Φ.

4.1. Subdivision

Following the idea described in Section 3, the first step is to project a small volume V to the data domain. The spatial volume V is constructed by creating hexahedra within the regular grid, attaching eight multi-variate input data samples to the corners of each hexahedron. Projecting the eight data samples to the data domain results in eight point locations that determine the shape of Φ. Now, the size of Φ is calculated by finding the convex hull of the eight points. The extents of this convex hull can be computed easily.

The subdivision process is triggered when the extent of the convex hull exceeds a user-given limit in the data domain. Possible criteria to measure the extent of Φ include the area of Φ or the maximum extent of Φ in the ξ_1 and ξ_2 dimensions. In our implementation, we use the latter option in order to guarantee that the maximum length of Φ is bounded. When the subdivision criterion triggers a subdivision step, the current hexahedron is split into eight new hexahedra in a regular fashion. Regular subdivision allows us to determine the size of the subdivided volumes V easily: V covers a relative volume of $2^{-n}2^{-n}2^{-n}$ if n is the subdivision level. For each of the new hexahedra, we recompute the attached data values using trilinear interpolation. Please note

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

```
// main loop:
for each Cell in data set
{
  Process (Cell);
}

// ---------------------------------------
function Process (IN: CurrentCell)
{
  project CurrentCell to data domain;

  if (Size (ProjectedCell) > threshold)
  {
    // split into eight new cells
    // generic interpolation possible!
    Split CurrentCell;

    for each NewCell do // recursion
      Process (NewCell);
  }
  else // size of Phi small enough
  {
    // draw either convex hull
    // or axis-aligned rectangle
    create triangles for CurrentCell;
  }
}
```

Figure 2: *Pseudo code for the subdivision approach. For the subdivision step, we use trilinear interpolation. The area Φ can be represented by the convex hull of the projected points or by an axis-aligned rectangle.*

```
// precomputation step:
BuildOctree;

// Invoke rendering:
TraverseOctree (OctreeRoot);

// ---------------------------------------
function TraverseOctree (IN: OctreeNode)
{
  if (OctreeNode.SizeOfPhi > threshold)
  {
    if (OctreeNode has children)
    {
      for all children  // recursion
        TraverseOctree (child)
    }
    else // reached a leaf
    {
      // perform subdivison
      Process (OctreeNode.Cell)
    }
  else // size of Phi small enough
  {
    create triangles for OctreeNode.Cell;
  }
}
```

Figure 3: *Pseudo code that traverses the octree. Once a leaf is reached, the same subdivision is used as in Fig. 2.*

that generic interpolation or reconstruction methods can be applied to find those data values, replacing the trilinear reconstruction filter. The process of projecting the data values of the new hexahedra to the data domain is repeated recursively until the threshold is no longer exceeded. In this case, the resulting convex hull is rendered as a filled polygon with constant density V/Φ, using additive blending. This algorithm is outlined in Fig. 2 as pseudo code.

An even faster approach to approximate the size of Φ uses an axis-aligned rectangle in the data domain that forms a tight fit around the eight projected points. Since only the lower-leftmost and upper-rightmost points have to be found, the computational effort is reduced when compared to finding the convex hull and computing its extents. In addition, the rendering is based on simple rectangles instead of more complex filled polygons. Otherwise, this algorithmic variant is identical to the first one.

4.2. Octree Hierarchy

Until now, all cells of the original data set are taken into account equally, regardless of their contribution to the final density in the data domain. However, many data sets contain large homogeneous regions. To reduce the amount of cells that have to be considered, we apply an additional octree hierarchy that quickly processes those homogeneous regions in the input data set. Since the computational overhead of the octree should be as small as possible, subdivision is done in a regular fashion only (i.e. cells of the same level in the octree all have the same size).

For each node of the octree, the smallest and largest data values of each data dimension are stored. With these values, the extents of the previously mentioned axis-aligned rectangle are known. While traversing the octree, we simply have to compare these extents with the user-given threshold. If the extents exceed the threshold, we descend further down along the octree. Once the octree is traversed to the lowest level, each leaf contains a single hexahedron of the original data set. If the size of Φ of that hexahedron still exceeds the threshold, the hexahedron is subdivided as described in Section 4.1. For this approach, we render filled rectangles with constant density as a representation of Φ. A summary of this algorithm is presented as pseudo code in Fig. 3.

4.3. Generalization

The basic idea of our approach can be extended to other kinds of grids than regular grids. Alternative grids may lead to modifications of the computation of V or Φ, while the ba-

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping University, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

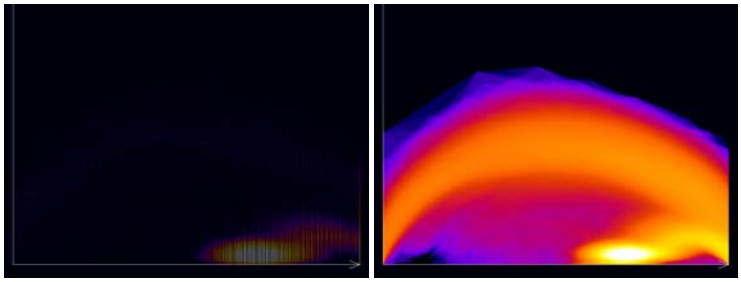

Figure 4: *As a reference, the conventional scatterplot of the "Tooth" data set is shown (left image). The continuous version (right image) is created with the original continuous scatterplot algorithm [BW08].*

sic idea of subdivision remains. For example, generic structured grids, such as curvilinear grids, have the same topology as regular grids. Therefore, the structure and shape of Φ are not affected by this kind of generalization. Only the computation of V needs to incorporate a more complicated volume formula for deformed hexahedral cells. For other kinds of primitive cells — such as tetrahedra or prisms — the shape of Φ might differ from the hexahedral case. However, the concept of axis-aligned bounding volumes or convex bounding cells in the data domain is not affected. In other words, the main issue with more complex grid structures is not the subdivision process, but to construct an adequate replacement of the octree hierarchy that is put on top of the original data set.

Another kind of generalization replaces the trilinear interpolant. The only required change is that, during subdivision, the more general reconstruction function is invoked to compute the new data values at newly inserted grid points. If the generic reconstruction function is convex, then the convex-hull approach still computes an accurate volume Φ. We denote a function as convex when the function values stay within the min-max interval of the input values (i.e. the data values at grid points). This is the case for trilinear interpolation and for the example of on-the-fly gradients used for 2-D transfer functions (see example in Section 6). Even if the generic reconstruction function is not convex, our approach may still provide a good approximation as long as the function values do not reach too far outside the min-max interval of the input values.

5. Implementation

Our implementation to compute the density in the data domain is entirely written in C++ and executed on the CPU. In order to keep the implementation simple, the code is running single-threaded.

The approach that uses the convex hull to compute the size of Φ employs the algorithm described in [Jar73]. The octree needed for acceleration of our proposed algorithm is built in a preprocessing step and kept in main memory for all following traversal steps.

All generated triangles are rendered with OpenGL using triangle strips. The final result is stored in a floating point texture — by doing this, we only have to recompute the continuous scatterplot if the user changes the viewing parameters. For other user inputs, e.g. brushing areas of interest, we can simply draw the texture that stores the continuous scatterplot.

Our implementation is available as open source software on our website (`www.vis.uni-stuttgart.de/scatterplot`).

6. Results

There are two aspects that are of main interest: scatterplot quality and computational speed. The quality of a scatterplot created by our proposed approximation algorithms directly depends on the user-specified threshold. Raising the threshold increases the approximation error, but decreases the time necessary to compute a continuous scatterplot and vice versa. Please note that this threshold is intuitively specified in terms of pixels in the scatterplot — it defines the maximum extents of a projected cell in the data domain.

The scatterplots for the following analysis were created on a personal computer equipped with an Intel CPU (2.4GHz). The CPU has access to 4 GB of RAM. The computer's GPU is an NVIDIA GeForce 8800 GTX with 768 MB of texture memory.

First, we analyze a data set of a human tooth that was created with a CT scan. The spatial extent of this data set is $128 \times 128 \times 160$. A second data dimension is generated during run-time based on the concept of multi-dimensional transfer functions [KKH01]. Following this idea, the additional dimension contains the magnitude of the gradient of the scalar field. The resulting scatterplot allows the user to identify material boundaries as arc-like structures, which can be used to specify transfer functions. Computing those gradients on-the-fly is an example of a generic non-linear reconstruction function that can be used with our approach.

As a reference, we show a conventional scatterplot and a continuous scatterplot of this data set in Fig. 4. The continuous version of the scatterplot was generated with the algorithm described in our original continuous scatterplot paper [BW08].

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

First, we show a series of continuous scatterplots created with the subdivision approach using a convex hull to represent Φ in the data domain. To examine the effect of the user-specified threshold, we decrease the maximum approximation error step by step. All scatterplot images have a resolution of 1024×768.

Figure 6 (upper row) shows this series of scatterplots. Despite the significant differences in the threshold, the continuous scatterplots differ only marginally. As it turns out, density values within a cell do not vary strongly, therefore our approximation that uses constant density does not deviate too far from the true values. Differences to the original continuous scatterplot can be explained with our improved interpolation method.

The same subdivision approach can be used in combination with axis-aligned rectangles to approximate Φ. In Fig. 6, this series in shown in the lower row. In contrast to the previous example, the effect of the threshold is clearly visible. Using a high threshold and therefore allowing a high approximation error results in a coarse scatterplot. When compared with the subdivision approach that uses the convex hull, this approach yields scatterplots of lower quality since the approximation of Φ tends to deviate much more from the correct solution. On the other hand, scatterplots computed with this approach are created faster, due to simpler computations.

We analyze an additional data set which was created with a CT scan of an engine block. The spatial extent of this data set is $256 \times 256 \times 110$. As for the first data set, the second data dimension is generated by computing the magnitude of the gradient of the data samples.

With this "Engine" data set, a third series of continuous scatterplots was created. Here, an octree is used in combination with axis-aligned rectangles to approximate the density contribution of a cell in the data domain. The resulting pictures are shown in Fig. 7. Due to the octree hierarchy, a speed-up of up to two orders of magnitude is achieved compared to the original continuous scatterplot approach. The octree is created in a precomputation step which needs less than one second to prepare for the "Engine" data set. Since memory consumption of the octree is very low, the overhead introduced by the tree structure can be neglected. Depending on the given error threshold, this approach allows one to draw very coarse representations of the scatterplot, since an arbitrary number of cells can be combined in higher levels of the hierarchy. Therefore, this approach is completely independent from the input resolution of the data set. The computational speed directly depends on the error threshold, however, faster computation leads to lower scatterplot quality.

In order to quantify the effects of the threshold with regard to scatterplot quality, we show error plots in Fig. 5. We use the L_2 norm to measure the error between scatterplots. In order to get an L_2 norm that can be interpreted easily, we

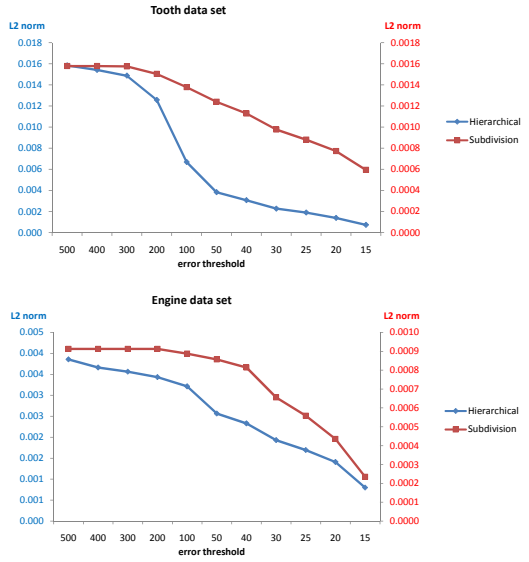

Figure 5: *The upper plot shows the L_2 norm for the "Tooth" data set, the lower plot for the "Engine" data set. For both data sets, the hierarchical octree approach ("Hierarchical") and the subdivion approach that uses the convex hull ("Subdivision") are analyzed. Please note that the scale of x-axis is not uniform. Also, the vertical axis use different scalings for the two different approaches.*

norm the density values of the scatterplots. By doing this, the scale of the scatterplot changes in such a way, that a density value of one corresponds to the arithmetic mean of all density values. Since an analytic solution of a continuous scatterplot is not available, we analyze the convergence behavior by comparison to a numeric solution with a low error threshold of only 10 pixels. Both error plots for the subdivision approach show a similar behavior: first, the L_2 norm stays on a constant plateau before it drops with decreasing error threshold. High error thresholds do not trigger the subdivision process, therefore the same values are returned by the L_2 norm. For lower error thresholds, the subdivision approach converges in an expected way. Please note that the error values are at a very low level at all times (below 0.2 percent ("Tooth") and 0.1 percent ("Engine") respectively), since the convex hull provides a good approximation. The error plots for the hierarchical octree approach do not have an upper bound as the error plots for the subdivision approach. This is explained with the fact that the octree hierarchy allows very coarse representations for the size of Φ. However, we can observe the same behavior as for the subdivision approach for lower thresholds: the error converges similarly to zero for low error thresholds. However, the absolute scale of the L_2 values is higher due to the coarser approximation by axis-aligned rectangles.

Continuous scatterplots are designed to be included in existing or future interactive visualization systems. Therefore,

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

Tooth		200	100	50
Discrete	0.04	-	-	-
Cont.	39.8	-	-	-
Convex Hull	-	17.17	17.68	23.88
Octree	-	0.25	2.92	23.19

Engine		200	100	50
Discrete	0.14	-	-	-
Cont.	106	-	-	-
Convex Hull	-	46.1	48	54.5
Octree	-	7.1	22.3	166

Table 1: *Measurements of the time in seconds needed to compute a scatterplot depending on the chosen approach and error threshold. "Discrete" stands for conventional scatterplots, "Cont." for the original continuous scatterplot approach. For those approaches, there is no threshold that can be set, therefore, only one result is recorded. "Convex Hull" is our subdivision approach using the convex hull to represent Φ. "Octree" is our hierarchical approach that uses axis-aligned rectangles in combination with an octree for speed-up. Different thresholds were used for the performance measurements; these thresholds are listed in the top row.*

their run-time behavior with regard to computation time is of interest as well. Table 1 lists time measurements of different methods that compute a continuous scatterplot of the "Tooth" data set.

There are some interesting conclusions that we can draw from these measurements. First of all, the original continuous scatterplot approach is not very user-friendly, since it needs a very long time to compute. This is overcome by using the approaches presented in the previous sections. However, the user has to take care that the approximation-error threshold is appropriate. If this is not the case, our presented methods may even need more time to compute a continuous scatterplot than in the original approach. On the other hand, if a good trade-off between accuracy and speed is found, our algorithms compute a continuous scatterplot much faster while the scatterplot is still approximated fairly well.

7. Conclusion and Future Work

We have presented methods that generalize continuous scatterplots and substantially increase their performance. Generalization is achieved by supporting generic interpolation or reconstruction methods as opposed to linear interpolation only. Coupled to that aspect is the support of hexahedral cells — a triangulation as in the original continuous scatterplot approach is no longer necessary.

Continuous scatterplots can only be integrated in visualization systems if the response times are within acceptable bounds. The original approach to compute a continuous scatterplot is often beyond that time limit. We have proposed two algorithms that greatly reduce the time necessary to compute a continuous scatterplot. This is achieved by approximating the density contribution of a small volume in the spatial domain. The degree of approximation controls both the quality of the scatterplot as well as the time needed to compute it. The trade-off between speed and quality is adjustable using a single parameter that is specified by the user. In this way, interactive speed is achieved for previews of the continuous scatterplot. Once the user has specified all view parameters, more time can be spent for a high-quality version of the continuous scatterplot.

In future work, our approaches could be extended to higher-dimensional spatial domains, such as time-dependent 3-D data sets. In addition, computational performance could be further improved by porting our approaches to the GPU in order to exploit its superior computation power.

Acknowledgements

The "Engine" data set is courtesy of General Electric. In parts, this work was supported by DFG within SFB 716 / D.5.

References

[BW08] BACHTHALER S., WEISKOPF D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1428–1435.

[CCT83] CHAMBERS J. M., CLEVELAND W. S., TUKEY P. A.: *Graphical Methods for Data Analysis*. Duxbury Press, 1983.

[CMPS96] CIGNONI P., MONTANI C., PUPPO E., SCOPIGNO R.: Optimal isosurface extraction from irregular volume data. In *Proc. Symposium on Volume Visualization* (1996), pp. 31–38.

[Jar73] JARVIS R.: On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2 (1973), 18–22.

[KKH01] KNISS J., KINDLMANN G., HANSEN C. D.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In *Proc. IEEE Visualization* (2001), pp. 255–262.

[SHLJ96] SHEN H.-W., HANSEN C. D., LIVNAT Y., JOHNSON C. R.: Isosurfacing in span space with utmost efficiency (ISSUE). In *Proc. IEEE Visualization* (1996), pp. 287–ff.

[SSD*08] SCHEIDEGGER C. E., SCHREINER J., DUFFY B., CARR H., SILVA C. T.: Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1659–1666.

[ST90] SHIRLEY P., TUCHMAN A.: A polygonal approximation to direct scalar volume rendering. *Computer Graphics 24*, 5 (1990), 63–70.

[Utt04] UTTS J. M.: *Seeing Through Statistics*, 3rd ed. Duxbury Press, Belmont, 2004.

[WG92] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation. *ACM Transactions on Graphics 11*, 3 (1992), 201–227.

[WW01] WARREN J., WEIMER H.: *Subdivision Methods for Geometric Design: A Constructive Approach*. Morgan Kaufmann, 2001.

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License

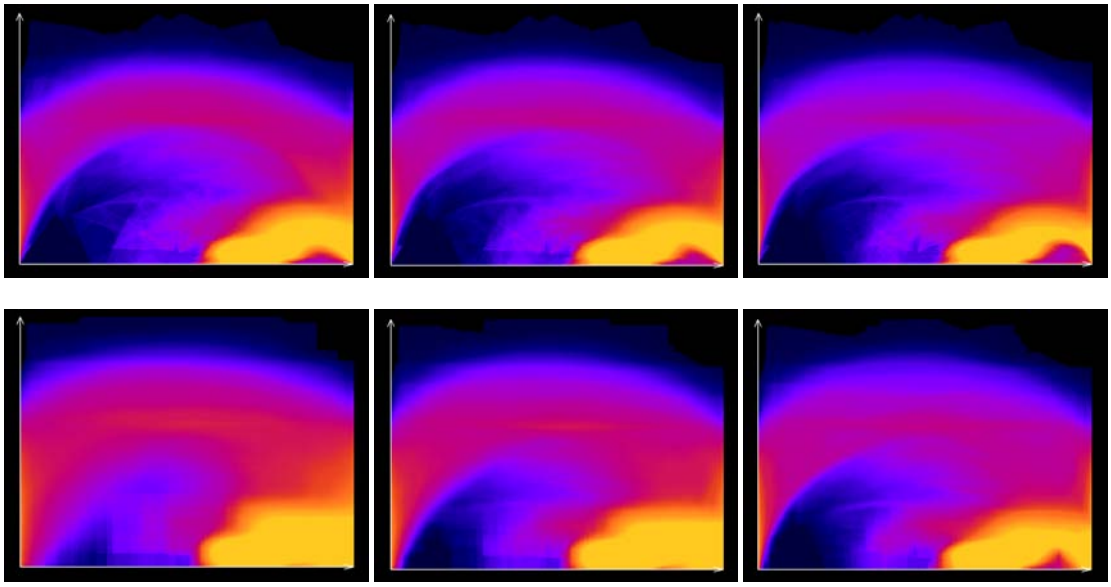

Figure 6: *These continuous scatterplots were created with the subdivision approach for the "Tooth" data set. The pictures in the upper row show results of the subdivision approach using a convex hull to represent* Φ *in the data domain. On the left side,* Φ *is allowed to span up to 200 pixels in each dimension. The picture in the middle is created with a threshold of 100 pixels, whereas the right-most picture uses a threshold of 50 pixels. The lower row shows the same data set, but this time the subdivision approach uses axis-aligned rectangles to represent* Φ*. The same thresholds as for the upper row are applied.*

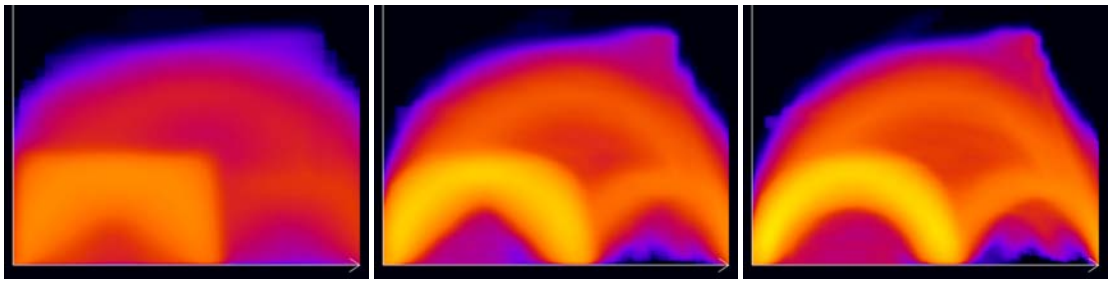

Figure 7: *These continuous scatterplots were created with the octree approach for the "Engine" data set. The effect of different thresholds is shown. The left picture shows a coarse approximation with a threshold that allows* Φ *to extend up to 200 pixels in each dimension. The picture in the middle uses a decreased threshold of 100 pixels. The right-most picture shows a good approximation since the threshold is lowered to 50 pixels.*

© 2009 The Author(s)
Journal compilation © 2009 The Eurographics Association and Blackwell Publishing Ltd.

14678659, 2009, 3, Downloaded from https://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2009.01478.x by Linkoping Universitet, Wiley Online Library on [20/11/2024]. See the Terms and Conditions (https://onlinelibrary.wiley.com/terms-and-conditions) on Wiley Online Library for rules of use; OA articles are governed by the applicable Creative Commons License