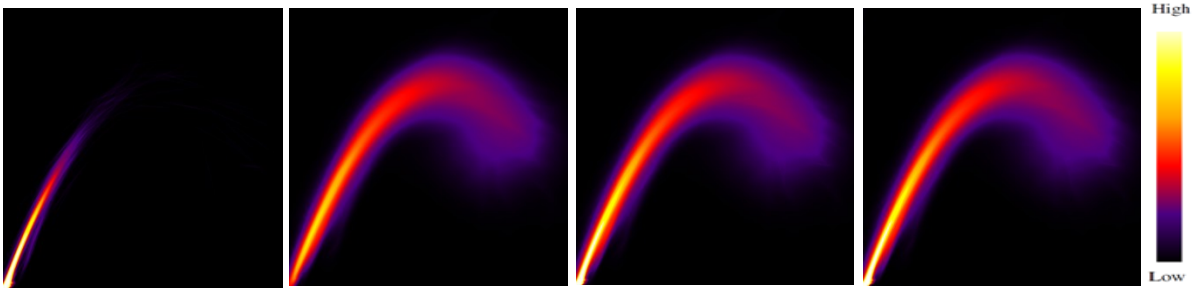
# Frequency-based Progressive Rendering of Continuous Scatterplots

**Vladimir Molchanov**
v.molchanov@jacobs-university.de

**Alexey Fofonov**
a.fofonov@jacobs-niversity.de

**Lars Linsen**
l.linsen@jacobs-university.de

Jacobs University, Campus Ring 1, 28759 Bremen, Germany




a) b) c) d) e)

Figure 1: Hybrid approach for continuous scatterplot computation ("Bucky Ball" dataset): Small splats are rendered directly (a), large splats are computed via their spectral representation (b), and both textures are added to produce the final result (c). Result is compared to direct rendering of all splats (d). Color scheme is shown (e).

## ABSTRACT

Continuous scatterplots are a consistent tool for the visual representation and exploration of continuous multivariate data defined on a continuous domain. Due to the complexity of the construction algorithm, application of continuous scatterplots is limited in terms of data size and screen resolution when interactive frame rates are desired. Progressive rendering is a paradigm of displaying an approximative visual outcome early on, which iteratively and incrementally gets improved until convergence to the final result is reached. This approach maintains the interactivity of the system and allows the user to make decisions immediately, i.e., much earlier than the end of the computation process. We propose a method for progressive rendering of continuous scatterplots based on a Fourier representation. By iteratively advancing from low to high frequencies and inverting the spectrum representation after each iteration, a series of scatterplots converging to the final result is generated and rendered. We demonstrate that this convergence is monotonic and that the proposed approach is more efficient than state-of-the-art methods, i.e., we can faster produce high-quality approximations. We propose to embed this idea into a hybrid approach which allows balancing the trade-off between quality of the image appearing first and its computation time. The proposed algorithms were implemented on the GPU.

## Keywords

Scatterplot, Fourier transform, progressive rendering, unstructured volumetric data



## 1. INTRODUCTION

Most volumetric scalar fields, no matter whether they have been measured or simulated, are assumed to be continuous or, at least, piecewise continuous. Examples include fields of temperature, density, pressure, salinity, etc. Seldom, these scalar fields can be described analytically. In most cases, data values are acquired by sampling the scalar field at discrete locations in space. In order to reproduce their continuous nature, an interpolation method is involved.

Statistical properties of scalar fields can be explored via histograms. Pair wise relations of two such properties are usually represented in form of scatterplots. Multiple scatterplots can be investigated inform of a scatterplot matrix. All these techniques are well known and have been used for decades by operating on the values at the discrete sample locations. However, such an important property as continuity is being lost at this step.

Recently, new methods appeared which address the problem above by creating continuous representations of attribute spaces. They propose continuous histograms [CBB06, SSD*08], continuous parallel coordinates [HW09, HBW11] and continuous scatterplots [BW08, BW09]. The idea behind these approaches is to perform a mapping of parts of the volume rather than just discrete samples when projecting to the visual domain. The distribution of attributes in the volume parts is reconstructed by means of interpolation.

Obviously, much more computational efforts are required to produce a continuous scatterplot when compared to its discrete analog. Therefore, there is a clear need for efficient algorithms showing high performance when applied to large datasets and high-resolution outputs. One approach to design an interactive tool is to exploit the idea of progressive rendering. This idea is based on an iterative generation of visualizations with increasing quality which converge to the final result. The gain is to provide the user with preliminary but still informative outcomes at interactive rates. Based on the preliminary result, the user can decide whether it is worth to wait for the final result or whether the interactive query shall be modified.

The essential properties of progressive rendering algorithm which we target in our research can be summarized as follows:

1. The key features of the result shall appear first.

2. The change of intermediate results when stepping from iteration to iteration shall vanish. In other words, the new contributions to the final image decrease in each iteration.

3. The result obtained at the previous step shall be effectively used when performing the next iteration, i.e., computations already made shall not be repeated.

Our work was inspired by the continuous scatterplots technique proposed by Heinrich et al. [HBW11]. In this algorithm, the physical volumetric domain is represented as a collection of Gaussian kernels centered at data samples. When mapping to the attribute domain (i.e., the domain spanned by the two dimensions of the scatterplot), each kernel appears as an elliptical splat called a *footprint*. The geometry of a footprint is entirely defined by the parameters of a locally linearized mapping, which allows for the pre-computation of footprints in form of splat textures. To handle large datasets, Heinrich et al. propose to split the data into chunks, produce an individual plot for each portion of the data, and then iteratively combine the plots over several frames using alpha blending. The result is a progressively updated image which is based on the amount of data already processed. So, the work by Heinrich et al. uses the general idea of a progressive rendering. However, using their approach, it is not easy to fulfill the first two requirements formulated above. The result may be highly affected by the way, how the data are split into chunks. Sophisticated methods may help to make a wise decision, but they may also break the interactivity of the application.

Our main contribution is a novel approach for progressive rendering of continuous scatterplots based on their frequency representation (*spectrum*). This kind of representation is obtained by applying a Discrete Fourier Transform (DFT) to the scatterplot's density distribution. The key idea is as follows: Small-sized footprints have bad (slowly decaying) spectra but can be efficiently plotted directly into the attribute space density plot (continuous scatterplot). Large-sized footprints slow down the computations when blended in the attribute space, but they have nice frequency representations. Hence, we propose to split the physical space kernels into two classes: Those having small footprints (few pixels support) and those whose footprints have good Fourier representations. The former are plotted directly while the latter are processed iteratively, progressing from the lowest most meaningful modes to the highest frequencies with vanishing contributions.

Besides that, we extend the existing method to unstructured volumetric data and explore the performance and error dependence on most key parameters. We implemented the method by Heinrich et al. [HBW11] and our own algorithm entirely on the GPU (using CUDA) for better evaluation of their applicability and efficiency.

## 2. RALATED WORK

The idea of progressive image generation has been known for decades [BFGS86]. Progressive methods find their application in many visualization and computer graphics problems, e.g., ray tracing [PS89], global illumination [FP04], or volume rendering [CBPS06]. One of the most prominent examples is surface renderings where progressive meshes [Hop96] can be used in case of irregular meshes and subdivision surfaces in case of regular patches or semi-regular meshes. Also, isosurface extraction from progressively refined tetrahedral or pyramidal meshes (e.g., [LPD*04]) and isosurface smoothing

[PB00] are common examples in volume visualization.

Spectral analysis is used to improve sampling of the rendering integral in volume rendering [BMW*06]. A wide range of progressive spectral methods [SDS96], especially based on the Fast Fourier Transform (FFT), are very common in image processing, e.g., for image registration [RC96] or for characterization of brain fibers' shape [PEPM12].

Scatterplots are a well-known tool for multidimensional data visualization and analysis [EDF08]. Recently, an idea of continuous scatterplots was developed in a series of works by Bachthaler, Heinrich, and Weiskopf. Initially applied for mapping tetrahedra by using a linear interpolation of attributes within them [BW08], the approach was generalized to regular rectangular grids with an arbitrary interpolation method [BW09], where recursive subdivision of cells was exploited. Later, isotropic density functions were used to decompose the whole volume into a system of overlapping spheres [HBW11]. Splatting is performed for the generation of progressively sampled intermediate images which are then combined to produce the final continuous scatterplot. The progressive rendering component of the algorithm is based on partitioning the volumetric data into small chunks which are fast to operate. Intermediate images are produced for down-sampled or even freely re-sampled data. A generalization to continuous representations of projected spaces was recently proposed in [MFL13].

The idea of image space footprint computation of volumes has been developed by Westover [Wes90]. Feng et al. [FKLT10] use Gaussian footprints to visualize data samples uncertainty. Lehmann and Theisel [LT10] developed a method to find and highlight discontinuities in continuous scatterplots.

## 3. BACKGROUND

Before we describe our approach in detail, we would like to provide the respective background. For the construction of the continuous projections, we look into the settings of the involved spaces (a volumetric physical domain and a 2D attribute domain which is visualized). We discuss some technical aspects of the method by Heinrich et al. [HBW11] and, in particular, provide useful generalization by allowing the lengths of spatial kernels to vary from sample to sample.

### 3.1 Basic Terms and Notations

Let $\tau(\mathbf{x})$ be a multivariate multidimensional function with $m$ variables defined over $n$ dimensions, i.e., $\tau:\mathbb{R}^n \mapsto \mathbb{R}^m$. It is sampled at position $\{\mathbf{x}_i\} \subset X \subset \mathbb{R}^n$ mapping them to the set

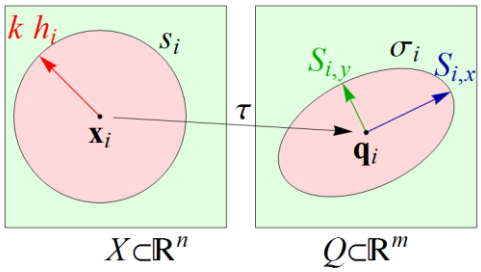



Figure 2: Physical volume $X$ is mapped by $\tau$ to the attribute domain $Q$. A spatial density $s_i$ is defined in a spherical neighborhood of sample $\mathbf{x}_i$, which has an elliptical footprint in $Q$. Scalar function $\sigma_i$ denotes density on the footprint.

$\{\mathbf{q}_i\} \subset Q \subset \mathbb{R}^m$, $i = 1,\ldots,N$. Figure 2 illustrates our notations. There exists a scalar non-negative function $s_{\text{overall}}(\mathbf{x})$ defined on $X$ which is called the *spatial density function*. It represents the importance of data in the volume and is usually set to be constant. For later consideration it is useful to approximate the density function by a weighted sum

$$s_{\text{overall}}(\mathbf{x}) = \sum_{i=1}^{N} w_i s_i(\mathbf{x}), \qquad (1)$$

where every individual kernel $s_i(\mathbf{x})$ is obtained from a compactly supported shape function $s(\mathbf{x})$ by

$$s_i(\mathbf{x}) = s\left(\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right),$$

where $h_i$ defines a scaling radius of $s_i(\mathbf{x})$. In our work we focus on isotropic kernels such that $s(\mathbf{x}) = s^r(\|\mathbf{x}\|)$. Our goal is to find proper weights $w_i$ to achieve $s_{\text{overall}}(\mathbf{x}) \approx \text{const}$. Since this condition simply means that all parts of domain $X$ are equally important, the exact value of the constant does not matter. Then, if one integrates the relation (1) over the volume, one deduces

$$C_1 \cdot \text{Volume}(X) = \sum_{i=1}^{N} w_i \int_{\mathbb{R}^n} s_i(\mathbf{x})\, d\mathbf{x} = C_2 \sum_{i=1}^{N} w_i h_i^n .$$

with constants C1 and C2. This means that $w_i h_i^n$ stand for volume fractions associated with samples. Therefore, one takes $w_i = 1$ for all $i$ to get $s_{\text{overall}}(\mathbf{x}) \approx \text{const}$.

In the following, we study the case $n = 3$ and $m = 2$ as the most practically important one, since it is common practice to investigate pairs of attributes defined over a volume by means of scatterplots.

Let $X_i$ be the support of the function $s_i(\mathbf{x})$. The construction of a continuous scatterplot is based on additive mapping of all $X_i$ weighted by $w_i s_i(\mathbf{x})$ by means of $\tau(\mathbf{x})$ to the attribute domain $Q$. Recall that discrete scatterplots map the data sample by sample resulting in the set $\{\mathbf{q}_i\}$. A continuous scatterplot is composed by a collection of images of $X_i$. It is shown in [HBW11] that spherical kernels $s_i(\mathbf{x})$ result in elliptical footprints. Each footprint is equipped with a density distribution $\sigma_i(\mathbf{q})$, which represents a counterpart of the spatial density $s_i(\mathbf{x})$, see Figure 2. Hence, a continuous scatterplot $\sigma_{\text{overall}}(\mathbf{q})$ is the direct sum of densities $\sigma_i(\mathbf{q})$.

The key relation between the two density functions is given by

$$\int_{X_0} s_{\text{overall}}(\mathbf{x})\mathrm{d}^n\mathbf{x} = \int_{Q_0} \sigma_{\text{overall}}(\mathbf{q})\mathrm{d}^m\mathbf{q}, \qquad (2)$$

for any $X_0 \subset X$ and $Q_0 = \tau(X_0)$. This condition uniquely defines $\sigma_{\text{overall}}(\mathbf{q})$. Equation (2) with $X_0 \equiv X$ and $Q_0 \equiv Q$ is called *total mass conservation*.

## 3.2 Direct Approach for Generation of Continuous Scatterplots

Following the method proposed in [HBW11], a spherical neighborhood of sample $\mathbf{x}_i$ of radius $kh_i$ is mapped as an elliptical splat centered at $\mathbf{q}_i = (q_1(\mathbf{x}_i), q_2(\mathbf{x}_i))$ using a linear approximation of the mapping $\tau$ in a neighborhood of sample $\mathbf{x}_i$. The density $\sigma_i(\mathbf{q})$ is non-zero within this footprint. Besides its center, a footprint is uniquely defined by its extents $S_{i,x}$ and $S_{i,y}$ (semi-axes of the screen-space ellipse) and its rotation angle $\theta_i$. These parameters are defined as follows (cf. [Wes90, HBW11]):

$$S_{i,x} = kh_i\sqrt{\lambda_1}, \quad S_{i,y} = kh_i\sqrt{\lambda_2}, \qquad (3)$$

$$\cos\theta_i = \frac{b}{\sqrt{b^2 + (\lambda_1 - a)^2}}, \qquad (4)$$

where $k$ is a global smoothing parameter discussed below, $h_i$ is a local support size which may vary from sample to sample and reflect the samples' density in the neighborhood of the $i$-th instance,

$$\lambda_1 = \frac{a+c+e}{2}, \quad \lambda_2 = \frac{a+c-e}{2}, \quad e = \sqrt{(a-c)^2 + 4b^2}$$

$$a = \nabla q_1(\mathbf{x}_i)\cdot\nabla q_1(\mathbf{x}_i), \qquad b = \nabla q_1(\mathbf{x}_i)\cdot\nabla q_2(\mathbf{x}_i),$$

$$c = \nabla q_2(\mathbf{x}_i)\cdot\nabla q_2(\mathbf{x}_i),$$

where $\cdot$ stands for the scalar product. Note that the computation of $\cos\theta_i$ can be numerically instable for small $b$ when applying Equation (4) naively. If $|b| < \epsilon$ for some threshold $\epsilon > 0$, the splat is not rotated, i.e., $\theta_i = 0$. Thus, one can explicitly set $\cos\theta_i = 1$.

The user is allowed to change the value of parameter $k$ which simultaneously scales the extents $S_{i,x}$ and $S_{i,y}$ of all footprints. Thus, the global smoothness of the continuous scatterplot can be controlled: For small values of $k$ the result is almost a discrete plot, while large values of $k$ make the resulting image blurred. This scaling is equivalent to the respective change of support sizes of spatial kernels $s_i(\mathbf{x})$. Here, their supports shrink to samples $\mathbf{x}_i$ as $k$ vanishes and unboundedly grow when $k \to \infty$. Note that for larger values of $k$, the number of locally overlapping spatial kernels increases.

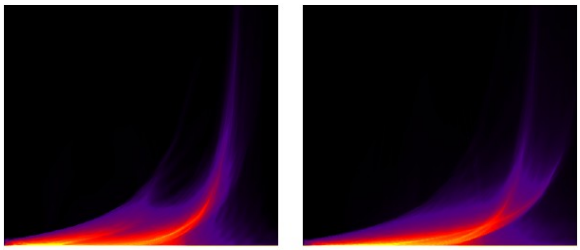
All elliptical footprints are added as textures to the final image. In the following we refer to this procedure as a *direct (splatting) approach* in contrast to the progressive method proposed in this paper. After a normalization step which makes the range of the accumulated density $\sigma_{\text{overall}}(\mathbf{q})$ equal to $[0,1]$, a transfer function can be applied. The main bottleneck of the method is related to the rendering of large splats (relative to the screen resolution), since their contributions to many pixels have to be computed. This situation occurs in particular when producing high-resolution images, when zooming into a smaller region of the plot, or when choosing high values of the global smoothing parameter $k$.

## 3.3 Progressive Approach

To overcome the issue of insufficient efficiency of the method for interactive visual analyses, a progressive splatting approach was proposed in [HBW11]. When splitting the input data into small portions and operating on them separately, the number of operations per chunk is reduced. It makes it possible to generate images for each chunk faster.

Gradual accumulation of the generated images results in a progressively changing continuous scatterplot. However, every intermediate result intrinsically depends on which part of the data is already processed and which not. In other words, it depends on the order in which the chunks are accumulated. To illustrate this effect we generated a "Tornado" dataset, courtesy of [CM93], sampled at $40^3$ regularly distributed nodes. Velocity magnitude and the $z$-component of the velocity field are taken

(a) 25% of samples (b) 40% of samples



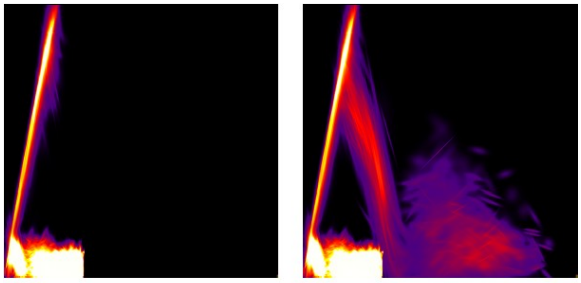
(c) 50% of samples (d) 100% of samples

Figure 3: Dependence of intermediate images in progressive splatting on the fraction of processed data ("Tornado" dataset with $40^3$ gridded nodes).

as dimensions of the scatteplot. Intermediate results after operating 25%, 40% and 50% of samples as well as the final result are shown in Figure 3. All four images are qualitatively very different, so that there is no monotonic behavior of the results along iterations. Hence, this unpredictability of the intermediate results will make the user wait until all computations are done, which destroys the essence of progressive rendering (see Requirements 1 and 2 formulated in the Introduction). Of course, the intermediate images depend on how the chunks of data have been generated and a different choice may have produced qualitatively better approximations early on, but the issue of not knowing whether the intermediate results reflect the final result well is apparent.

We also note that the normalization of the density, which is necessary before a transfer function can be applied, is very sensitive to the skipping of data portions. In fact, skipping a few samples contributing to the pixel with the highest density obviously changes the normalization factor, which affects the appearance of the whole picture, although the transfer function remains the same.

The situation becomes even worse when dealing with unstructured spatial datasets, which are discussed in the next section. The reason is that there is no standard way of ordering data in memory and splitting data into chunks can be absolutely arbitrary. Note that even if the data are regularly sampled, after a zooming operation, the data to be displayed generally do not belong to any spatially regular structure. These considerations motivated us to



(a) 33,3% of samples (b) 50% of samples

Figure 4: Dependence of intermediate images in progressive splatting on the fraction of processed data ("two White Dwarfs" dataset with 24k unstructured samples).

design an approach, where all samples are handled uniformly and simultaneously and where no particular spatial arrangement of the samples is assumed.

Our solution to the issue above relies on the spectral representation of continuous scatterplots. Instead of splitting the data, the progressive part of our approach is based on a consecutive computation of the Fourier frequencies advancing from the lowest most contributing modes to the highest less important ones. In the following section we show that large splats have a nice (rapidly decaying) representation in the frequency domain that makes it possible to significantly speed up the construction of continuous scatterplots.

We note that spatially unorganized data can be stored in a file in arbitrary order. Thus, when applying the progressive splatting idea from [HBW11], the distribution of samples among chunks of data is very accidental. This fact leads to the effect shown in Figure 4. Shown is a dataset representing an astrophysical simulation of a binary system consisting of two White Dwarfs. The dimensions of scatterplot are the internal energy values and temperature at samples' positions. Images for 33.3% and 50% of the samples provide a poor approximation to the final result shown in Figure 5 (e).

## 3.4 Spatially Unstructured Data

We generalize the approach of Heinrich et al. [HBW11] by allowing the local scales to vary from sample to sample. It makes it possible to apply the direct approach to irregularly sampled data. In order to define an individual scale $h_i$, we estimate the local density of samples: In the regions, where samples are dense, $h_i$ is small, while sparsely distributed $\mathbf{x}_i$ yield large $h_i$. Only the relative variation of scales is important, since the absolute magnitude can be changed by adapting parameter $k$.

For regular data, the samples' density is constant everywhere, hence, all $h_i$ are equal.

The ability to handle irregular data is very valuable for practical applications. For instance, various scientific measurement techniques and particle-based numerical methods may produce large multivariate unstructured spatial datasets which are to be visualized and analyzed. Of course, it is always possible to re-sample unstructured data regularly, but this comes at the cost of introducing interpolation errors and losing spatial adaptivity which may affect the result. Therefore, it is advantageous to operate on the original data.

## 4. SPECTRAL REPRESENTATION OF CONTINUOUS SCATTERPLOTS

For our purposes, it is important that density $\sigma_i(\mathbf{q})$ in the footprint domain can be computed from some template kernel function $\sigma(\mathbf{q})$ via an affine transformation composed by scaling, rotation, and translation, i.e.,

$$\sigma_i(\mathbf{q}) = \alpha_i \ \sigma(L_i(\mathbf{q}-\mathbf{q}_i)),$$

$$L_i = \begin{pmatrix} S_{i,x}^{-1} & 0 \\ 0 & S_{i,y}^{-1} \end{pmatrix} \cdot \begin{pmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{pmatrix}.$$

In other words, to compute $\sigma_i(\mathbf{q})$ at some location, one has to perform translation by $\mathbf{q}_i$, rotation by angle $\theta_i$, scaling by $(S_{i,x}, S_{i,y})$, and then evaluate the shape-function $\sigma$ at the obtained location. This is just an analytical expression of the splatting, where a pre-computed texture containing values of $\sigma(\mathbf{q})$ is placed at the right position in the frame with right scaling and right orientation resulting in the footprint $\sigma_i(\mathbf{q})$. The coefficient $\alpha_i = w_i h_i^3 \cdot S_{i,x}^{-1} \cdot S_{i,y}^{-1}$ serves to conserve the mass associated with the $i$-th sample:

$$\int_Q \sigma_i(\mathbf{q}) \mathrm{d}^2\mathbf{q} = \alpha_i \int_Q \sigma(L_i(\mathbf{q}-\mathbf{q}_i)) \mathrm{d}^2\mathbf{q}$$

$$= \frac{\alpha_i}{|L_i|} \int_Q \sigma(\mathbf{p}) \mathrm{d}^2\mathbf{p} = w_i h_i^3 \int_X s(\mathbf{y}) \mathrm{d}^3\mathbf{y} = w_i \int_X s_i(\mathbf{x}) \mathrm{d}^3\mathbf{x}.$$

Here we assumed that the $L_1$-norms of template functions $s$ and $\sigma$ are equal and used $|L_i| = S_{i,x}^{-1} \cdot S_{i,y}^{-1}$. Summation over $i$ provides the total conservation of mass (2).

The final density distribution is given as a composition of individual contributions of all samples, i.e.,

$$I(q_1, q_2) = \sum_{i=1}^{N} \sigma_i(q_1, q_2).$$

Here and further we use $I \equiv \sigma_{\text{overall}}$ for short. Now we examine the density function in the frequency domain. By the linearity property, the Fourier transform of $I(q_1, q_2)$ reads

$$\mathcal{F}[I](u,v) = \sum_{i=1}^{N} \mathcal{F}[\sigma_i](u,v).$$

It is known that scaling, rotation, and translation operators become scaling, rotation and modulation after the Fourier transform, correspondingly. In particular, $\mathcal{F}[f(A\mathbf{x})](\mathbf{q}) = |A^{-1}| \cdot \mathcal{F}[f(\mathbf{x})](A^{-1}\mathbf{q})$. Thus,

$$\mathcal{F}[\sigma_i](u,v) = \mathrm{e}^{i(uq_1(\mathbf{x}_i) + vq_2(\mathbf{x}_i))} \alpha_i |L_i^{-T}| \cdot \mathcal{F}[\sigma](L_i^{-T}(u,v)),$$

where $-T$ denotes inversion and transposition, with

$$L_i^{-T} = \begin{pmatrix} S_{i,x} & 0 \\ 0 & S_{i,y} \end{pmatrix} \cdot \begin{pmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{pmatrix}.$$

Finally,

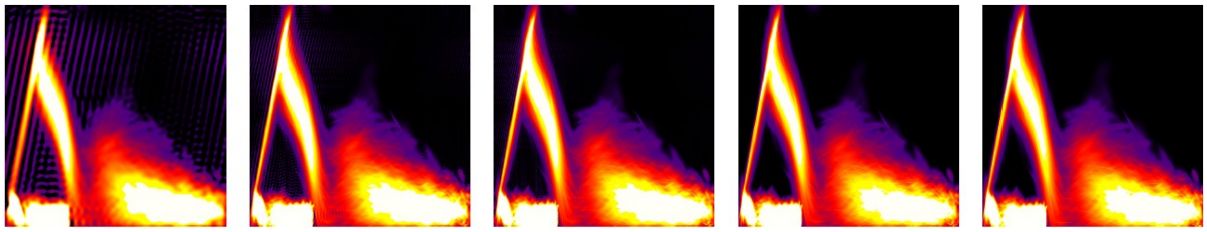
$$\mathcal{F}[I](u,v) = \sum_{i=1}^{N} w_i h_i^3 \ \mathrm{e}^{i(uq_1(\mathbf{x}_i) + vq_2(\mathbf{x}_i))} \mathcal{F}[\sigma](L_i^{-T}(u,v)). \quad (5)$$

When the spectral representation $\mathcal{F}[I]$ is computed, one can apply the inverse Fourier transform to obtain the continuous scatterplot. Practically, we use the discrete Fourier transform rather than the Fourier transform itself. This approximation introduces errors which vanish if the resolution of textures (number of frequencies taken into account) grows. We also need to comment on the periodicity of the discrete Fourier transform: Footprints located close to the boundary of the computed region may have their parts appearing at the opposite part of the boundary instead of being cut. To overcome this effect, the region should be extended to reserve additional space for such footprints.

## 5. PROGRESSIVE ALGORITHM

Our complete computation of spectral representation $\mathcal{F}[I]$ is much slower than the standard splatting-based technique for continuous scatterplot generation. Obviously, every footprint has usually non-zero contributions to all frequencies $(u,v)$, whereas the splat may have very small size. Thus, if the resolution of the screen is $R_u \times R_v$, complexity of $\mathcal{F}[I]$ computation is $N \times R_u \times R_v$. Recall that this representation has to be transformed by the inverse FFT and not the full resolution can be shown, since some border region of the texture is reserved to eliminate the periodicity effect of DFT. However, there is a strong side of $\mathcal{F}[I]$.

It is well-known that the decay of Fourier coefficients depends on the smoothness of a function:



(a) spectral, 1 iter. (b) spectral, 6 iter. (c) spectral, 12 iter. (d) spectral, all iter. (e) direct

Figure 5: Continuous scatterplots of merging White Dwarfs dataset. When all footprints are progressively computed using spectral method, small splats cause noise, which is visible at first iterations and disappears later. After all iterations are completed, the result is identical to one obtained by the direct approach, i.e., when all elliptical footprints are blended as textures.

The smoother the kernel $\sigma(\mathbf{q})$, the faster the decay of $|\mathcal{F}[\sigma](u,v)|$. In fact, for many footprints, high frequencies do not significantly affect the result and therefore may be neglected. Based on this observation, we propose the following progressive algorithm for continuous scatterplot construction: Evaluate $\mathcal{F}[I](u,v)$ for some low frequencies, perform the inverse FFT and render the result as the first approximation to the final picture. In subsequent iterations higher frequencies are added to the Fourier representation of the scatterplot and the rendered picture is updated after performing the inverse FFT. Since high frequencies of $\mathcal{F}[I]$ computed at later iterations are composed by vanishing contributions from individual splats, later iterations have generally less impact on the final result for smooth kernels. Thus, Requirements 1-3 in the Introduction are fulfilled. The number of frequencies computed in each iteration depends on the hardware performance and the data size. It can be fixed or vary from one step to another.

## 6. HYBRID ALGORITHM

Large footprints have fast decaying Fourier spectra and therefore can be very efficiently and with high precision represented by a few lowest frequencies. Small footprints can be fast rendered directly to the scatterplot. Here "small" means that at least one of the extents $S_{i,x}$ and $S_{i,y}$ is less than a prescribed threshold. To profit from the best of both worlds, we divide all footprints into two groups according to their size and operate accordingly. The grouping is based only on values $S_{i,x}$ and $S_{i,y}$, i.e., no additional computations are needed. Moreover, the critical size of a splat below which it is labeled as small, depends only on the screen resolution. Since parameter $k$ scales all splats, its value affects the grouping.

The idea of the hybrid approach is illustrated in Figure 1. We used the "Bucky Ball" dataset (courtesy AVS, USA) with $32^3$ gridded samples. Small and large splats rendered by means of the direct method and their spectral representation, respectively, see Figure 1 (a) and (b). Both plots are combined to produce the final result (c) which is very close to the scatterplot computed by direct splatting (d).

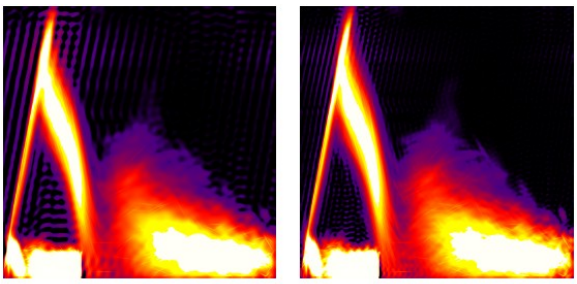
## 7. RESULTS

All numerical tests were performed on a PC with graphic card NVIDIA GTX 680 and implemented in CUDA. We used the same color scheme to render all continuous representations of projections, see Figure 1 (e). The texture size used for computation is $1024^2$ pixels. We reserved 10% for the border, such that the size of the rendered texture is $921^2$ pixels. Computation of the whole spectral representation requires $1024^2 / 4096 = 256$ iterations, where parameter 4096 is chosen to achieve a high occupancy of the GPU. If other is not specified, $s_{\text{overall}}(\mathbf{x}) = \text{const}$ is used. Other information about parameters and datasets is shown in Table 1.

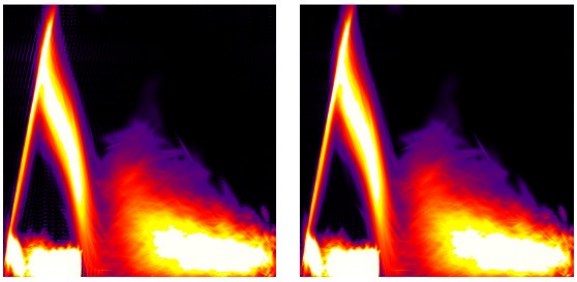
| dataset | k | threshold | Total number of points | Number of large splats |
|---|---|---|---|---|
| Bucky Ball | 2.0 | $0.02 * R_u$ | 32768 | 13986 |
| White Dwarfs | 1.6 | $0.008 * R_u$ | 24149 | 21676 |
| Tornado | 0.6 | $0.01 * R_u$ | 262144 | 82856 |

Table 1: Parameters and thresholds for all experiments. Shown are values of the global smoothing parameter $k$, the threshold that defines which splats are labeled as small, the size of dataset, and the number of splats computed by the spectral method for the given threshold.

First, we demonstrate the convergence of the pure spectral approach. The dataset represents an astrophysical simulation of two merging White

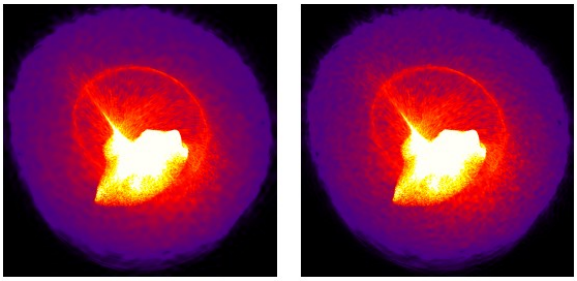


(a) hybrid, 1 iter. (b) hybrid, 3 iter.

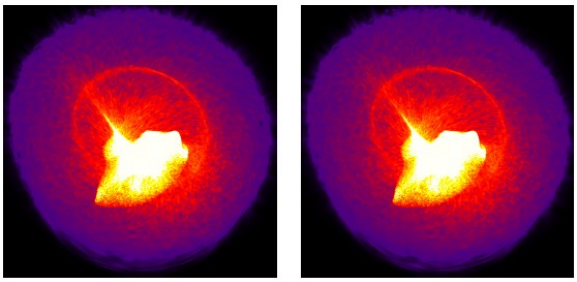


(c) hybrid, 6 iter. (d) hybrid, 12 iter.

Figure 6: Hybrid approach applied to the merging White Dwarfs dataset.



(a) hybrid, 1 iter. (b) hybrid, 3 iter.



(c) hybrid, 6 iter. (d) direct

Figure 7: Hybrid approach applied to the "Tornado" dataset.

Dwarfs. The simulation was executed by means of the Smoothed Particle Hydrodynamics method [Luc77, GM77]. The data includes unstructured nodes' positions $\mathbf{x}_i$, lengths $h_i$, several scalar attributes and their gradients. Dimensions of the scatterplot are internal energy and temperature fields in horizontal and vertical directions, correspondingly. Results for $s_{\text{overall}}(\mathbf{x})$ to be equal to the density field are shown in Figure 5. Small splats have poor spectral representation and, thus, serve as a source of noise. This noise is eliminated during progressive computation of Fourier magnitudes. After all iterations are completed, the result is identical to one obtained by the direct method. When the hybrid approach is applied to the same dataset, the initial level of noise is much lower, see Figure 6.

Next, we used the "Tornado" dataset, courtesy of [CM93]. It is given as an analytical function describing a velocity profile in a volume. We sampled the velocity field at $64^3$ random uniformly distributed locations. Results obtained by the hybrid approach are presented in Figure 7. Velocity components in $x$ and $y$ directions serve as the two axes of the scatterplots. We intentionally chose a relatively small $k$ to demonstrate the effect of detailization during progressive rendering. Small global smoothness makes the sizes of footprints small, thus, even some individual splats remain distinguishable in the continuous scatterplot. The first iteration of the hybrid approach results in a slightly smeared image with more and more details appearing at later steps.

Computational times are listed in Table 2. Both the spectral and the hybrid approaches deliver preliminary results much earlier than the direct splatting method. Though, the first iteration of the hybrid method takes a longer time than one iteration of the spectral method, its later iterations are faster and the quality of the result is higher. The accompanying video shows a comparison of the quality of the results obtained with spectral and hybrid approach. Note that the first iterations of the video are played in slow-motion to have enough time to observe the intermediate images.

To measure the error of the proposed methods at $j$ -th iteration, we computed the root-mean-square error

$$L_j = \left( \frac{1}{R_u \cdot R_v} \sum_{p_x=1}^{R_u} \sum_{p_y=1}^{R_v} \left( \bar{\sigma}(p_x, p_y) - \bar{\sigma}_{\text{direct}}(p_x, p_y) \right)^2 \right)^{1/2},$$

where $p_x$ and $p_y$ stands for pixel indices, $\bar{\sigma}_{\text{direct}}$ is the density computed by the direct approach, $\bar{\sigma}$ can be computed either by the spectral or by the hybrid method, and the bars denote that the densities are normalized (individually). Results are shown in Figure 8. It is evident that the error is gradually improved along the progressive computations. The error of the hybrid method after first iteration is significantly less than the analogous error of the spectral one. Moreover, only a few first iterations of the hybrid method suffice to closely approach a stable state.

The hybrid approach has better error behavior though the first step takes more computation time.

| dataset | direct | Spectral, per iter. | Hybrid, 1st iter. | Hybrid, from 2nd iter. |
|---|---|---|---|---|
| Bucky Ball | 3942 ms | 119 ms | 435 ms | 48 ms |
| White Dwarfs | 895 ms | 95 ms | 269 ms | 76 ms |
| Tornado | 1254 ms | 591 ms | 282 ms | 203 ms |

Table 2: Computation times for construction of continuous scatterplots using direct approach, pure spectral representation and the hybrid algorithm (all methods are implemented on GPU).

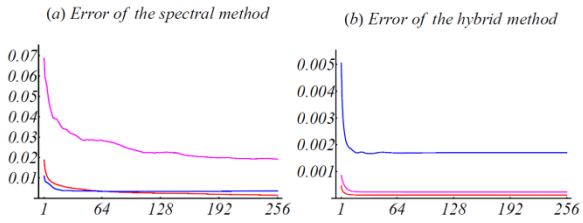



Figure 8: Error behavior along the iterative construction of continuous scatterplots. The plots present the root-mean-square errors of the spectral algorithm and the hybrid method. Results for the "Bucky Ball" (red), two White Dwarfs (blue) and the "Tornado" (magenta) datasets are shown. The error of the hybrid approach applied to the "Bucky Ball" dataset (red line in (b)) is multiplied by a factor of $10^3$ to be visible.

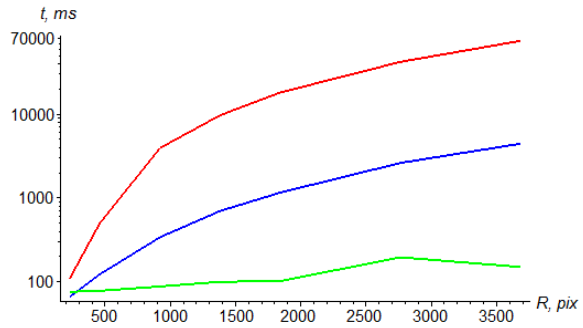



Figure 9: Plots of calculation times of appearence of the first meaningful image against output resolution (pixels) computed for "Bucky Ball" dataset. Results for the direct splatting (red), the hybrid method (blue) and the spectral approach (green) are shown.

As follows from the data in Table 2, almost 7 iterations of the spectral method can be finished before the first iteration of the hybrid method is completed. However, the initial error of the hybrid approach is lower than the error of the spectral algorithm after 7 iterations and decreases faster in the consequent steps. This holds for all datasets. By the time, when the first result of the hybrid approach is delivered only 11% ("Bucky Ball"), 30% ("two White Dwarfs") and 22% ("Tornado") of samples can be processed by the direct method, which is insufficient for performing a reliable analysis.

The benefit of our approach becomes extremely significant when producing high-resolution outputs. Figure 9 demonstrates the dependence of calculation times for the first meaningful picture on the output resolution. In particular, the speed-up of 500 times is achieved for $R_u = 3686$ pixels. Note that the values of time (vertical axis) are scaled logarithmically. Non-monotonic behavior of the green line is related to higher efficiency of FFT when R is a power of 2.

## 8. CONCLUSION

We have seen that when producing high-resolution continuous scatterplots, when dealing with highly adaptive sampling in physical space, when zooming into a region of interest on a scatterplot, or when increasing the global smoothness of the result, sizes of some footprints become large up to the range of the screen size. In such situation, a direct accumulation of the splats in the final image gets extremely slow and significantly affects the overall performance, which hinders interactivity. Our method allows overcoming this drawback by the effective use of the spectral representation of large footprints.

The proposed progressive rendering approach fulfills all three requirements formulated in the Introduction section. First, small splats are directly aggregated into the density texture at the very first step of the algorithm. These splats usually have highest density values and thus affect the overall picture most, since they mostly determine the range of the transfer function being applied. Second, due to decay of the Fourier coefficients of a smooth function, each next iteration in the Fourier domain will have less impact than the previous step. Thus, changes in the displayed picture lessen over the iteration steps and stabilize to the final state quite rapidly, which was shown in a number of tests. Finally, only missing modes of spectral representation are added in each subsequent step, so that no re-computation of any part of earlier obtained results is needed.

We have been able to significantly speed up the appearance of the first result of the hybrid method when compared to the full-resolution direct splatting approach. In our tests, the speed-up varied from 3.3 times to two orders of magnitude depending on the resolution of the output. This allows for computation times suitable for interactive analysis, e.g., in scatterplot matrices.

## 9. ACKNOWLEDGMENTS

The authors wish to thank Marius Dan and Stephan Rosswog for sharing datasets. This work was supported in part by a DFG grant LI 1530/6-2.

## 10. REFERENCES


[BFGS86] Bergman L., Fuchs H., Grant E., Spach S.: Image rendering by adaptive refinement. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 29–37.

[BMW*06] Bergner S., Möller T., Weiskopf D., Muraki D.: A spectral analysis of function composition and its implications for sampling in direct volume visualization. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1353–1360.

[BW08] Bachthaler S., Weiskopf D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization) 14*, 6 (2008), 1428–1435.

[BW09] Bachthaler S., Weiskopf D.: Efficient and adaptive rendering of 2-d continuous scatterplots. *Comput. Graph. Forum (Proc. Eurovis 09) 28*, 3 (2009), 743 – 750.

[CBB06] Carr H., Duffy B., Denby B.: On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1259–1266.

[CBPS06] Callahan S. P., Bavoil L., Pascucci V., Silva C. T.: Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1307–1314.

[CM93] Crawfis R., Max N.: Texture splats for 3D vector and scalar field visualization. In *Proceedings Visualization '93* (1993), IEEE CS Press, 261–266.

[EDF08] Elmqvist N., Dragicevic P., Fekete J.-D.: Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation. *IEEE Transactions on Visualization and Computer Graphics 14* (2008), 1141–1148.

[FKLT10] Feng D., Kwock L., Lee Y., Taylor R.: Matching visual saliency to confidence in plots of uncertain data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 980–989.

[FP04] Farrugia J.-P., Péroche B.: A progressive rendering algorithm using an adaptive perceptually based image metric. In *Eurographics conference proceedings* (2004).

[GM77] Gingold R. A., Monaghan J. J.: Smoothed particle hydrodynamics — theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc. 181* (1977), 375–389.

[HBW11] Heinrich J., Bachthaler S., Weiskopf D.: Progressive splatting of continuous scatterplots and parallel coordinates. *Comput. Graph. Forum 30*, 3 (2011), 653–662.

[Hop96] Hoppe H.: Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), SIGGRAPH '96, ACM, 99–108.

[HW09] Heinrich J., Weiskopf D.: Continuous parallel co-ordinates. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1531–1538.

[LPD*04] Linsen L., Pascucci V., Duchaineau M. A., Hamann B., Joy K.: Wavelet-based multiresolution with nth-root-of-2 subdivision. *Journal on Computing special edition* (2004).

[LT10] Lehmann D. J., Theisel H.: Discontinuities in continuous scatter plots. *IEEE Transactions on Visualization and Computer Graphics 16* (2010), 1291–1300.

[Luc77] Lucy L. B.: A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal 82* (1977), 1013–1024.

[MFL13] Molchanov V., Fofonov A., Linsen L: Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling. *Comput. Graph. Forum (Proc. Eurovis 13)* (2013), to appear.

[PB00] Pascucci V., Bajaj C. L.: Time critical isosurface refinement and smoothing. In *Proceedings of the 2000 IEEE symposium on Volume visualization* (2000), VVS '00, ACM. 33–42.

[PEPM12] Poco J., Eler D. M., Paulovich F. V., Minghim R.: Employing 2d projections for fast visual exploration of large fiber tracking data. *Comp. Graph. Forum 31*, (2012), 1075–1084.

[PS89] Painter J., Sloan K.: Antialiased ray tracing by adaptive progressive refinement. *SIGGRAPH Comput.Graph. 23*, 3 (1989), 281–288.

[RC96] Reddy B. S., Chatterji B. N.: An FFT-based technique for translation, rotation, and scale-invariant image registration. *Trans. Img. Proc. 5*, 8 (1996), 1266–1271.

[SDS96] Stollnitz E. J., Derose T. D., Salesin D. H.: *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.

[SSD*08] Scheidegger C., Schreiner J., Duffy B., Carrh., Silva C.: Revisiting histograms and isosurface statistics. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (2008), 1659 –1666.

[Wes90] Westover L.: Footprint evaluation for volume rendering. In *Computer Graphics* (1990), 367–376.