DOI: 10.1111/j.1467-8659.2011.01913.x

Eurographics / IEEE Symposium on Visualization 2011 (EuroVis 2011)
H. Hauser, H. Pfister, and J. J. van Wijk
(Guest Editors)

Volume 30 (2011), Number 3

Visual Coherence for Large-Scale Line-Plot Visualizations

Philipp Muigg[1,2] Markus Hadwiger[3] Helmut Doleisch[2] Eduard Gröller[1]

[1]Vienna University of Technology, Austria
[2]SimVis GmbH, Vienna, Austria [3]King Abdullah University of Science and Technology, Saudi Arabia

Abstract

Displaying a large number of lines within a limited amount of screen space is a task that is common to many different classes of visualization techniques such as time-series visualizations, parallel coordinates, link-node diagrams, and phase-space diagrams. This paper addresses the challenging problems of cluttering and overdraw inherent to such visualizations. We generate a 2x2 tensor field during line rasterization that encodes the distribution of line orientations through each image pixel. Anisotropic diffusion of a noise texture is then used to generate a dense, coherent visualization of line orientation. In order to represent features of different scales, we employ a multi-resolution representation of the tensor field. The resulting technique can easily be applied to a wide variety of line-based visualizations. We demonstrate this for parallel coordinates, a time-series visualization, and a phase-space diagram. Furthermore, we demonstrate how to integrate a focus+context approach by incorporating a second tensor field. Our approach achieves interactive rendering performance for large data sets containing millions of data items, due to its image-based nature and ease of implementation on GPUs. Simulation results from computational fluid dynamics are used to evaluate the performance and usefulness of the proposed method.

1. Introduction

Lines are an integral part of many visualization methods. However, in most cases the number of lines that is rendered is directly proportional to the number of displayed data items. For this reason, such line-based approaches quickly suffer from cluttering and overdraw when large data sets have to be visualized. Standard line rasterization decomposes each line into individual pixels without orientation information. Thus, it can at most yield information about the line density per pixel, e.g., by using additive blending. We propose to generate additional data about line orientation on a per-fragment basis, which can be used to enhance line-based visualizations. In earlier work [MKO*08], a vector field has been used to store averaged line orientations per pixel. However, this is not sufficient for many visualization applications, as a vector field does not contain information about the overall distribution of line orientations. For example, distinguishing image regions where all lines are parallel from regions where many lines are crossing is impossible when only a vector field is used. In contrast, the technique proposed in this work relies on a 2x2 tensor field that is similar to structure tensor fields used in image processing. In both cases, tensors are symmetric positive definite, and the eigenvector corresponding to the largest eigenvalue encodes directional information per pixel. In our approach, this eigenvector corresponds to the most dominant line orientation through a pixel, whereas in a structure tensor it represents the dominant image-gradient direction. Additionally, the relation between eigenvalues λ_1 and λ_2 provides further information that is complementary to the dominant line orientation. We exploit this property to solve the problem of distinguishing regions comprised of parallel lines from regions with more evenly distributed line orientations. Furthermore, we construct the tensors representing the line orientations in such a way that they can be used directly as diffusion tensors. We then apply anisotropic diffusion to a white noise texture, driven by the directional information encoded in the tensor field. The resulting texture is spatially coherent in regions where the orientations of many lines are well aligned. Complementarily, the more an area is comprised of isotropically distributed line orientations, the more the resulting texture approaches isotropically blurred noise. The latter would also result from filtering with a Gaussian kernel. However, in isotropic regions the diffused noise texture does not convey additional information. We therefore adapt the influence of the noise texture on the visualization according to a local measure of anisotropy. Figure 1 compares parallel-coordinates without (top) and with (bottom) the technique proposed in this work. Line density is indicated by brightness/color. Note that especially in regions of uniform line density our approach provides pronounced visual cues that convey general line orientation very well.

Furthermore, many visualization techniques rely on a focus+context rendering approach. Data selected/highlighted by the user are displayed prominently as focus, whereas the remainder of the data set is represented in a less obtrusive style to provide context. In order to properly support this differentiation between focus and context, we propose to use

Published by Blackwell Publishing, 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

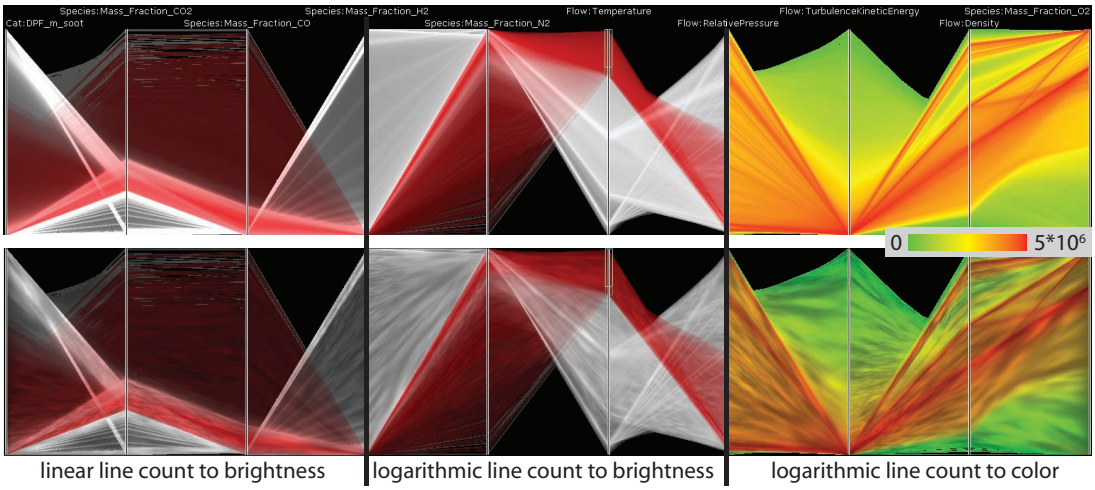

Figure 1: *Comparison between parallel-coordinates plots where line density is linearly/logarithmically mapped to brightness (first/second column, respectively), and color (third column), respectively. Each bottom-row image has been generated by applying our technique to the corresponding top-row plot. In the first and second columns, color is used for distinguishing between focus (red) and context (grey-scale), respectively. Regions where line density does not vary strongly are improved most visibly. The coherent noise texture employed by our method provides important information on line orientations that is otherwise lost.*

two separate tensor fields. One field represents line orientations for the overall data set (i.e., the context), and one field stores line orientations weighted by a selection function in $[0, 1]$ (i.e., the focus). This function is zero for unselected, and one for selected data, respectively. Anisotropic diffusion is either performed for a tensor field that is generated by combining the respective fields for focus and context, or for both tensor fields independently (see Section 4.5).

Since individual features can exist at different spatial scales, choosing just one noise frequency for the whole image plane is often not sufficient. Therefore, we propose to utilize a multi-resolution approach which builds on an image pyramid of the tensor field. Anisotropic diffusion is performed for each level of this pyramid. The input-noise frequency is adapted to the local scale of the tensor field. This results in a second image pyramid containing smeared noise textures which correspond to the different resolution levels of the original tensor field. A final image compositing step is used to blend multiple noise levels. Coarser noise is applied to homogeneous regions, whereas fine-grained noise is used in image areas exhibiting small-scale detail.

To summarize, the contributions of this work are:

- A generic image-space method that can enhance any visualization that draws many overlapping lines in 2D.
- Visual coherence via anisotropic diffusion of noise, driven by a 2x2 tensor field that subsumes line orientations.
- Combined visualization of features at different scales.
- Support for focus+context rendering.
- Interactive performance with straightforward implementation and integration into existing code.

2. Related Work

The technique proposed in this work is quite general and thus can be applied to a large class of visualization problems that render many lines. One of the most prominent examples is the parallel coordinates technique, which represents each data item by one poly-line [Ins85]. For large data sets, the main problem of this technique is cluttering in image space due to line overdraw. Fua et al. [FWR99] propose to perform hierarchical clustering on the data in order to solve this problem. A different approach based on image-based clustering has been proposed by Novotny and Hauser [NH06]. Instead of clustering entire poly-lines, only line segments between coordinate axes are grouped together in order to greatly speed up rendering. Outlier detection is also performed for line segments and not for entire poly-lines. McDonnell and Mueller [MM08] utilize different illustrative techniques in combination with edge-bundling and clustering. Edge-bundling is also used in the context of graph visualization via link-node diagrams [Hol06, HvW09], which could also be enhanced by our technique. Contrary to these approaches, Johansson et al. [JLJC06, JLC07] do not rely on a priori clustering of the data. Instead, a floating point texture is used to accumulate line density data. Transfer functions are then applied in order to reduce cluttering. Figure 1 shows how our technique can be combined with this approach. Sophisticated sampling strategies have also been used to reduce overdrawing [EG06]. Ellis and Dix [ED07] provide a more comprehensive classification of clutter reduction techniques.

Tensor fields are most commonly visualized by mapping tensor properties to glyph shapes such as super quadrics [Kin04]. Depending on the properties of the tensor field, more complex shapes/approaches can be

used [WH06]. In earlier work, focusing on a specific application, Zachow et al. [ZMH*09] briefly mention an image-based approach that also utilizes a 2x2 tensor field to encode line orientations per pixel. However, instead of driving anisotropic diffusion by this tensor field, it is used to perform convolution with locally modified Gaussian filters. Directionally-blurred noise has first been used in the context of flow visualization. Van Wijk uses locally deformed spots [vW91], whereas Cabral and Leedom [CL93] integrate white noise along integral lines of the flow field. Muigg et al. [MKO*08] apply line integral convolution to large-scale time series visualizations, computing a vector field to represent line orientations per pixel. Non-linear anisotropic diffusion is most commonly applied in image processing [Wei98], but has also been utilized for dense flow visualization [DPR00].

The notion of a continuous scale space has been introduced by Witkin [Wit83]. This concept of analyzing data on different scales has been used in many different fields in image processing as well as in visualization. Examples are information-visualization approaches relying on hierarchical clustering [FWR99], or flow-visualization methods based on a multi-resolution representation of the underlying flow field [TS06]. Rasterizing discrete lines for data items which are samples of a continuous data domain is not always sufficient. Thus, Bachthaler and Weiskopf [BW08] propose a continuous data representation for scatterplots, which has also been extended to parallel coordinates [HW09]. Note that even though no actual lines are drawn by this technique, data-density information is still transferred between coordinate axes along certain trajectories. For example, for two perfectly correlated attributes, these trajectories would all be horizontal lines between the two corresponding axes. Therefore, the tensor field driving our method could also be generated for the continuous case.

3. Dense Encoding of Line Orientations

Standard line rasterization can at most provide line-density information per pixel. Let us consider computing a line density function $C(x,y) \in \mathbb{N}$ during rasterization. This gives the number of lines crossing the pixel at position (x,y), but discards all information on line orientation. In principle, the gradient ∇C can be used to reconstruct some directional information in image space. Consider an image with a single rasterized line. The gradient at the edge of the line is always perpendicular to the line, and can thus be used to obtain information about line orientation without actually requiring the original line. As soon as line density increases, however, overdraw resulting from crossing or overlapping lines rapidly reduces the accuracy of image-space gradients. In regions of constant $C(x,y)$, no orientation can be reconstructed at all. In order to solve these problems, we generate additional data during rasterization. A 2D vector field $V(x,y) \in \mathbb{R}^2$ can be used to store the averaged line orientation for the pixel at position (x,y). On GPUs, such a field can be generated easily by calculating multiple output values per fragment during rasterization. These can be accumulated per pixel and renormalized in a subsequent rendering pass. However, a vector field lacks information about the coherence of line orientations through the pixel. No information about the actual distribution of line orientations is retained. Averaging different orientations may result in the same average. In this case, it is impossible to distinguish a pixel through which parallel lines are passing from one that is crossed by lines of varying orientations. Treating both cases equally can lead to misleading visualizations. Instead of averaging vectors, we propose to employ symmetric positive-definite 2x2 tensors. This is somewhat similar to the use of structure tensors in image processing. The structure tensor for a 2D image I and a 2D windowing function w is defined as

$$S_w(\mathbf{p}) = \sum_{\mathbf{u}} \left(w[\mathbf{p}-\mathbf{u}] \begin{bmatrix} I_x(\mathbf{u})^2 & I_x(\mathbf{u})I_y(\mathbf{u}) \\ I_x(\mathbf{u})I_y(\mathbf{u}) & I_y(\mathbf{u})^2 \end{bmatrix} \right), \quad (1)$$

where $\mathbf{u}$ covers a neighborhood of $\mathbf{p}$. I_x and I_y denote the partial derivatives of the image in x and y, respectively. The eigenvector of $S_w(\mathbf{p})$ corresponding to the largest eigenvalue λ_1 represents the predominant gradient orientation in a local neighborhood of $\mathbf{p}$. This neighborhood is defined by the windowing function w. The degree of anisotropy of the gradient in a region (i.e., how unevenly gradient orientations are distributed) can be expressed by the relative anisotropy:

$$f_r = \frac{\lambda_1 - \lambda_2}{\lambda_1 + \lambda_2}. \quad (2)$$

If the gradient within the windowed region is constant, then the smaller eigenvalue $\lambda_2 = 0$, and hence $f_r = 1$. If the gradient orientations are distributed uniformly (isotropically), $\lambda_1 = \lambda_2$, and thus $f_r = 0$.

We exploit these properties to encode the orientations of lines passing through a pixel $\mathbf{p}$, instead of the gradient orientations in the neighborhood of the pixel. Instead of using a windowing function to define a region for which directional information is accumulated, we sum over all normalized orientations $\mathbf{v} = (v_x, v_y)$ of lines passing through a pixel $\mathbf{p}$, with

$$O(\mathbf{p}) := \sum \mathbf{v}^{\mathrm{T}}\mathbf{v} = \begin{bmatrix} \sum v_x^2 & \sum v_x v_y \\ \sum v_x v_y & \sum v_y^2 \end{bmatrix} \quad (3)$$

defining the line-orientation tensor-field O. Analogously to structure tensors, the eigenvector corresponding to λ_1 represents the predominant orientation of all lines passing through $\mathbf{p}$. Likewise, the anisotropy measure f_r can be used to obtain information about the overall distribution of orientations. The tensor field O can be computed efficiently in a GPU fragment shader during line rasterization, computing $O(\mathbf{p})$ for each fragment covering a pixel $\mathbf{p}$. This approach requires almost no change of the underlying visualization method determining what and where lines should be drawn.

4. Visually Coherent Noise for Line Plots

Our method builds on the construction of the dense line-orientation tensor-field O described above, in order to produce coherent line visualizations in image space. Figure 2 provides an overview of the overall rendering pipeline.

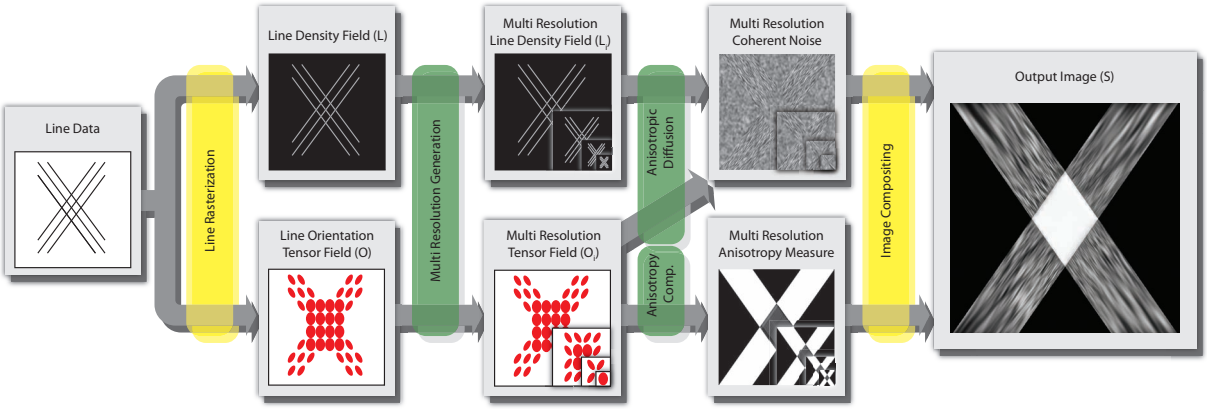

Figure 2: *The rendering pipeline that we use to synthesize the final coherent noise texture. Except for the line rasterization stage itself, all operations are image-based. This makes the technique scalable to very large data sizes. The steps indicated in yellow can be used to merge input from multiple subsets of the data set in order to accommodate focus+context visualizations.*

During line rasterization, we compute the line orientation tensor $O(\mathbf{p})$ for each pixel $\mathbf{p}$, as well as a scalar line density $L(\mathbf{p})$, by summing up the contribution of each line passing through the pixel. Additionally, in order to allow for focus+context techniques, multiple tensor fields representing different subsets of the data can be generated. The two stages highlighted in yellow in Figure 2 can be used to integrate the data in the focus with the data in the context for a cohesive focus+context visualization (Section 4.5). The coherent noise texture which is used to indicate predominant line orientations in a dense way is synthesized via nonlinear anisotropic diffusion. In order to do this, $O(\mathbf{p})/L(\mathbf{p})$ is directly used as the diffusion tensor (Section 4.2). In order to represent features at different spatial scales this diffusion process is performed at multiple image resolutions (Section 4.3). The output image is generated in a final compositing step that uses the coherent noise textures and an anisotropy measure of $O(\mathbf{p})/L(\mathbf{p})$ as input (Section 4.4).

4.1. Line Rasterization

The primary goal of the line-rasterization stage is the computation of $O(\mathbf{p})$ and $L(\mathbf{p})$ for each pixel $\mathbf{p}$. The contribution to all pixels covered by a single line between points $\mathbf{p_0}$ and $\mathbf{p_1}$ is equal to $\mathbf{v}^T\mathbf{v}$ with $\mathbf{v} = (\mathbf{p_1} - \mathbf{p_0})/|\mathbf{p_1} - \mathbf{p_0}|$. Additive blending into a floating point render target can be used to perform the component-wise summation of each line's contribution to O and L. In total, only four floating point values per pixel are required to store both fields: three floats for the symmetric 2x2 tensor $O(\mathbf{p})$, and one float for the number of lines $L(\mathbf{p})$ that contribute to the pixel. A single four-component floating point render target is sufficient to store all information required for coherent line visualization. Current GPUs support simultaneously writing to multiple render targets. This makes it straightforward to adapt existing visualization methods to generate both fields without requiring additional rendering passes. Furthermore, this approach can easily be combined with techniques that cluster lines and draw only one representative line per cluster. In this case, the contribution of each representative line must simply be weighted according to the number of lines in the cluster. For n lines in a cluster, $n\mathbf{v}^T\mathbf{v}$ has to be added to $O(\mathbf{p})$ at every pixel, and $L(\mathbf{p})$ must be increased by n instead of by 1.

4.2. Line-Orientation Driven Diffusion

After the fields O and L have been generated as described above, they can be used to visualize the encoded orientations in a coherent manner. It is important that this is done in a non-intrusive way that can be added easily to existing visualization techniques, while retaining their overall properties. For this reason, instead of using tensor visualization techniques such as glyphs, we employ dense noise textures that result from performing anisotropic diffusion of white noise. The resulting noise textures are straightforward to integrate into existing visualizations in image space by using them to modulate brightness per pixel.

Diffusion is the process of equilibrating concentrations within a spatial domain without generating or destroying mass. The flux j that corresponds to the direction along which mass is transported by the diffusion process is defined by Fick's Law:

$$j = -D \cdot \nabla u \tag{4}$$

Here, $u = u(\mathbf{x}, t)$ is a scalar function that defines the temporal evolution of concentrations over the spatial domain. In our case, $u \in [0,1]$ is the pixel intensity. D denotes the symmetric positive definite diffusion tensor, which is used to steer the strength and direction of the flux j. When the eigenvalues of D are equal (i.e., $\lambda_1 = \lambda_2$), the diffusion that results from the flux is isotropic, because then j is parallel to ∇u. Anisotropic diffusion results when $\lambda_1 \neq \lambda_2$. In the extreme case of $\lambda_2 = 0$, flux parallel to the corresponding eigenvector is not permitted at all. Thus, the diffusion tensor locally

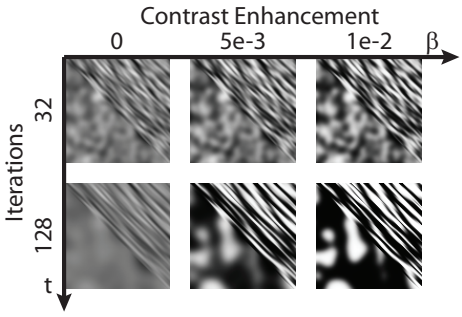

Figure 3: *Comparison of the influence of the contrast enhancement parameter β, and the number of iterations for diffusion, respectively. A small number of iterations requires higher values of β in order to generate comparable image contrast. The reason for this is that the influence of the contrast enhancement function f increases over time.*

steers the directional properties of the diffusion process. The diffusion can be formulated as a partial differential equation over time [Wei98]:

$$\partial_t u = \mathrm{div}(D \cdot \nabla u) \tag{5}$$

This means that the change of concentration over time is equal to the negative divergence of the flux. As proposed in the context of flow visualization [DPR00], this formulation can be used to create a dense visualization of directional information by diffusing a white noise texture and carefully selecting the diffusion tensor D. For a vector field V, D can be constructed such that the eigenvector corresponding to the largest eigenvalue is always parallel to V. The remaining degrees of freedom can be used to encode $|V|$. In our method, the normalized tensor field O/L has already been created with diffusion in mind, and can thus be used directly as the diffusion tensor, i.e., we use $D = O/L$. The normalization by L decouples diffusion strength from line density. It restricts the values of λ_1 and λ_2 to the unit interval $[0,1]$, regardless of the number of lines contributing to a given pixel. Without this normalization, the diffusion in regions with a higher line density would be stronger than in sparse image regions. Even if this effect would be desirable, a diffusion tensor with large eigenvalues generates large fluxes, which would require very small time steps for solving Equation 5.

Another important consideration is that diffusion generally reduces image contrast over time. Diewald et al. [DPR00] proposed to introduce a contrast enhancement function f. This function pushes pixels in the range $[0.0, 0.5)$ towards 0.0, and pixels in the range $[0.5, 1.0]$ towards 1.0. This can be incorporated into the diffusion equation as

$$\partial_t u = f(u) + \mathrm{div}(D \cdot \nabla u) \tag{6}$$

The contrast function that we use in our method is

$$f(u) := \beta \cdot (-(2u-1)^3 + 2u - 1), \tag{7}$$

where $\beta \in [0,1]$ represents a scalar that is defined by the user and controls the strength of contrast enhancement. Figure 3 shows how different settings for β influence the diffusion.

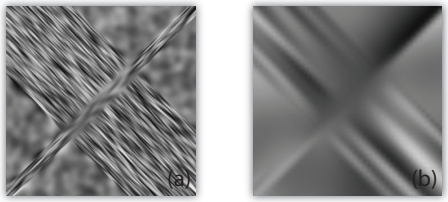

Figure 4: *Two different resolution levels of a coherent noise texture synthesized with our technique. The input were two sets of crossing lines. The set of lines that are packed tightly together is represented well by the high-frequency noise (left image), whereas the set of lines spread out more broadly is also represented by low-frequency noise (right image).*

The coherent noise texture $u(\mathbf{x}, t)$ is computed for time t by performing explicit integration of Equation 6, evaluating

$$u_i = u_{i-1} + \Delta_t \left(f(u_{i-1}) + \mathrm{div}(D \cdot \nabla u_{i-1}) \right) \tag{8}$$

for a desired number of iterations. The initial image u_0 contains black-and-white noise. The two major parameters that influence the result of the diffusion are the number of iterations, and the contrast enhancement parameter β. More iterations result in longer coherent structures, which is illustrated in Figure 3. All results in this paper have been created using between 128 to 256 iterations of Equation 8. We use a second-order discretization of $\mathrm{div}(D \cdot \nabla u_{i-1})$, [Wei98] (page 95), with $\Delta_t = 10^{-3}$ and pixel sizes of 1x1.

4.3. Visualizing Features of Different Scales

Multiple directionally-blurred noise textures to visualize features of different spatial scale have been used in the context of 2D vector field visualization [TS06]. Depending on the actual underlying line-based visualization approach, the tensor field O can also exhibit features of different spatial scales. Figure 4 shows an illustrative example where two groups of lines are crossing. The group of lines from the top-left to the bottom-right is distributed broadly and represents a large-scale feature. Lines from the bottom-left to the top-right are packed tightly, and thus can be considered to be a small-scale feature. The region where both line groups are crossing can also be regarded as a small-scale feature.

In Figure 4 (a), high-frequency noise has been used for the diffusion process. The resulting coherent noise texture shows both, the small-scale, and the large-scale features. However, the small length of the coherent structures and the high noise frequency result in a cluttered image. This specifically affects the region of the large-scale feature, because in that area a large number of short, narrow "blobs" is used to represent orientations. The length of these structures is limited by the number of iterations in the diffusion process.

This number is in turn limited by the requirement that interactive frame rates should be retained during user interaction. However, the length of the coherent noise structures can be greatly increased by reducing the resolution of the domain discretization on which the diffusion process is performed. The number of "blobs" can be decreased by reducing the spatial frequency of the input noise. To summarize, in order to create fewer, longer coherent noise structures which better represent large-scale features, lower-frequency input noise and lower-resolution representations of O/L can be used. Figure 4 (b) also shows the small- and large-scale feature. This time, a low-resolution version of O/L, as well as low-frequency noise are used as input to the diffusion process, which is likewise performed on a domain discretized with lower resolution. This results in an image that is less cluttered and contains much longer coherent structures along the large-scale feature. However, the small-scale features are less pronounced. Section 4.4 describes the blending technique that we employ to combine multiple noise resolutions in order to capture both small-scale and large-scale features.

We exploit the automatic mip-map generation on GPUs for the creation of multi-resolution image pyramids. The averaging filter that is utilized by most GPUs per default is sufficient to generate all mip-map levels of the four-component floating point texture that contains the fields O and L. We use O_i and L_i to denote the i-th mip-map level, where $i = 1$ represents the original field. The resolution is reduced by one half along each axis between subsequent levels. Here, the field O is used instead of the field O/L since the latter does not account for line density. Using O/L to generate the mip-map pyramid would overemphasize directional information from regions with low line densities. After mip-map generation, the diffusion step of the rendering pipeline is applied to each renormalized tensor field O_i/L_i separately. This creates coherent noise images N_i in the noise pyramid N.

4.4. Final Image Compositing

In the final image compositing step, the results of previous pipeline stages are combined into a single, grey-scale output image S. This image is then used to enhance the underlying line-based visualization technique by, for example, modulating image brightness per pixel. All of the presented examples (Section 5) use simple brightness modulation in order to combine a computed visualization result with the coherent noise image S.

The primary task of this pipeline step is the blending of the different coherent noise textures N_i described above. These textures are weighted so that noise frequency corresponds to local feature scale. Regions where small-scale features are present use higher-frequency noise, whereas homogeneous regions use lower-frequency noise. In the following, the salient features of the tensor fields O_i/L_i are considered to be regions of high anisotropy and similarly oriented eigenvectors. We have chosen anisotropy because the diffusion process only generates coherent information on line-orientation in regions of strong anisotropy. The scale of a feature at pixel position $\mathbf{p}$ can be determined by considering how the anisotropy of $O_i(\mathbf{p})/L_i(\mathbf{p})$ changes with increasing i. Large-scale features retain higher anisotropy values for larger i than small-scale features. The averaging process that is applied during mip-map generation reduces the anisotropy of image regions corresponding to smaller features faster. Therefore, the following weighted sum over a user-defined number $n > 1$ of noise resolutions can be used to accumulate the output image S:

$$S = (1 - A_1) + A_1 \left(\sum_{i=1}^{n} w_i N_i \right) / \left(\sum_{i=1}^{n} w_i \right) \tag{9}$$

$$w_i = \begin{cases} \sum_{j=1}^{d} \max(-A''_j, 0), & \text{if } i = 1 \\ \sum_{j=n+d-1}^{m} \max(-A''_j, 0), & \text{if } i = n \\ \max(-A''_{i+d-1}, 0), & \text{else} \end{cases} \tag{10}$$

A_i denotes fractional anisotropy for resolution level i and is derived from the tensor-field image-pyramid by applying Equation 2. It is not desirable to show the noise texture in regions of low anisotropy, because these regions do not contain well-defined coherent structures. Therefore A_1 is used to blend between the noise representation and a uniform white background. While n denotes the lowest resolution mip-map level from which noise is sampled, m is the smallest overall mip-map level. A''_i represents a numerical approximation to the second-order derivative of A_i with respect to i. It is computed by using central differences:

$$A'_i = A_{i+1} - A_{i-1}, \quad A''_i = A'_{i+1} - A'_{i-1} \tag{11}$$

The noise weight w_i for mip-map level i is determined by this second-order derivative since it indicates the resolution at which most of the anisotropy is lost due to the averaging process. Large negative values in a region of A''_i indicate that isotropy will start to decrease quickly in the following lower-resolution levels. A noise level j, which can represent features at level i, has to be selected. The level j is not equal to level i, because it is not possible to represent one pixel of directional information with just one pixel of a coherent noise texture. The integer parameter d is used to account for this, where $2^d \times 2^d$ noise pixels are used to represent one pixel of directional data. Typically, $d = 5$ in our examples. Since N_1 is the highest frequency noise, it is used to represent all levels between 1 and d. This is accounted for by the sum in row $i = 1$ in Equation 10. The case $i = n$ handles the weight computation for the noise of lowest resolution.

4.5. Focus + Context Visualization

The basis of all focus+context visualization techniques is the definition of one or multiple subsets of the data which

are of interest to the user. These subsets are called the focus, which have to be visualized in a prominent way. A context representation is used in order to provide the user with information on how the focus relates to the remainder of the data set. Many line-based visualization techniques rely on focus+context approaches, e.g., via different brushing/selection techniques. Our method facilitates easy integration of focus+context rendering. Primarily, a way to represent subsets of the data has to be established. Since the proposed technique is image-based, this can be achieved by supporting multiple input fields O^i and L^i, where O^1 and L^1 denote fields representing the entire data set. In order to create a proper focus+context representation, these fields have to be combined to form a single output image. This is somewhat similar to multi-volume rendering [CS99]. Combination can be performed at different stages in the rendering pipeline. A fast way to integrate l focus and context representations is to blend all input fields directly after the line rasterization stage:

$$O = \left(\sum_{i=1}^{l} b_i O^i\right) / \left(\sum_{i=1}^{l} b_i\right), \quad L = \left(\sum_{i=1}^{l} b_i L^i\right) / \left(\sum_{i=1}^{l} b_i\right) \tag{12}$$

The weights b_i can be used to emphasize data in the focus over data in the context. After this blending step, the rendering pipeline is executed as described before.

Another alternative is to integrate focus and context in the image compositing step. The entire rendering pipeline can be executed for all image pairs O^i, L^i separately. This results in coherent noise image pyramids N^i, and subsequently in multiple blended images S^i. These images can then either be combined by using the weights b_i, just as in the previous case, or the underlying visualization can use them individually to enhance the focus and context layers separately.

Both of these blending approaches have very distinctive advantages and disadvantages. Blending all input fields after line rasterization requires only a single diffusion step, which is the most time-consuming operation in our rendering pipeline. Furthermore, no additional texture memory is required to represent the intermediate information that is generated throughout the pipeline. The main drawback of this approach is that prior blending of the orientation tensor fields results in a field that does not exclusively represent the line orientations of the focus. This is illustrated in Figure 5 (a). Here, the line orientations of the focus (indicated in red) are only barely recognizable in the regions where context lines are crossed. Figure 5 (b) clearly indicates the orientations of lines in the focus throughout the whole visualization. Additionally, the generation of just one output image S restricts the way the underlying visualization method can integrate our technique. This is not the case when multiple output images S^i are generated by executing all pipeline steps multiple times for each data subset. Here, the main drawbacks are the increased memory requirements, and the necessity to compute multiple separate diffusions.

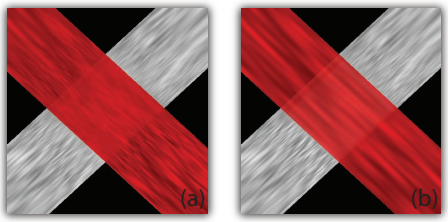

Figure 5: *Difference of blending the focus and the context, at different stages in our rendering pipeline. (a) the two tensor fields have been created separately for focus and context during line rasterization, but are then immediately combined to form a single composite field. (b) the entire rendering pipeline is executed for both tensor fields separately, until the final image compositing step merges their coherent noise textures into the result image.*

5. Evaluation and Discussion

Our visualization technique generates grey-scale images of coherent noise which illustrate general line orientation. This grey-scale output can be integrated into a multitude of visualization techniques which rely on lines as rendering primitives. We demonstrate this in the following by extending three exemplary visualization methods accordingly. In all three cases, the blended coherent noise texture S is combined with the unmodified visualization V by means of opacity modulation:

$$\gamma SV + (1-\gamma)V \tag{13}$$

The parameter γ is used to adjust how prominently the coherent noise should influence the original image. This approach retains the color information of the original visualization, which is used to distinguish focus from context. If V is a grey-scale image, image brightness instead of color has to be retained. In this case a color transfer-function f_{tf} can be applied to S before the blending step. To prevent $f_{\text{tf}}(S)$ from influencing image brightness, only colors of equal brightness have to be selected for f_{tf}.

The diffusion process has been applied to the focus and the context portions of the data separately. For diffusion, 128 (Figures 1 and 7), or 256 iterations (Figure 6) have been computed, using a contrast enhancement parameter of $\beta = 1.28e-3$. All application examples show data from the same time-dependent and high-dimensional computational fluid dynamics (CFD) data set. This data set contains simulation results from a model of the regeneration process within a diesel particulate filter [DMG*04]. The combustion and dissipation of soot within the filter has been simulated. The data set is comprised of 20 time steps and 260K cells per time step. Overall, 5.2 million data items, each consisting of 32 scalar values, are stored.

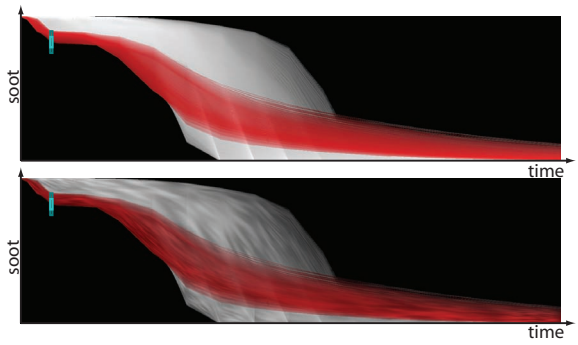

Figure 6: *Comparison between an original time series visualization (top), and our enhanced version (bottom). The structure that is added by our technique helps the user to discern general line orientation, especially in regions of low image contrast. Here, the scalar attribute represents the amount of soot within a CFD cell. The user-defined selection highlights portions of soot that initially burns at a fast rate.*

5.1. Parallel Coordinates

The parallel coordinates implementation that we have extended with our technique is based on an approach presented by Novotny and Hauser [NH06], and thus scales well with data size. Figure 1 shows two result images which have been created by visualizing 10 different dimensions of the diesel particulate filter data set while using three different line density transfer functions. A linear/logarithmic mapping is used in the left/center images respectively whereas a color transfer function is applied in the right-most visualizations. All 20 time steps are shown, which implies that 5.2M poly-lines have to be drawn to create these results. Each poly-line is made up of nine line segments. The right-most visualizations indicate that some pixels are crossed by nearly all of the lines. The top-row images show the original visualization results. Color is used to differentiate focus from context in the first two visualizations. Red indicates selected data items, whereas the context is visualized using a grey-scale transfer-function. Our technique has been applied to the top-row images in order to create the bottom-row images. Especially evenly populated regions benefit from the addition of the coherent noise image, as it adds visual cues that indicate line orientations. Furthermore, regions of parallel lines can be clearly distinguished from regions where lines are crossing, since anisotropy is used to attenuate the influence of the coherent noise texture. Here, the parameters in Equation 9 have been set to $n = 3$ and $d = 5$, respectively.

5.2. Time-Series Visualization

The time-series visualization presented in this section is based on a technique proposed by Muigg et al. [MKO*08]. For each data item, i.e., cell in the case of the CFD data set, one curve is drawn which represents the development of a scalar attribute over time. Therefore, 260K poly-lines, each comprised of 19 line segments, are displayed. The bottom image in Figure 6 has been created by modulating the top image with the coherent noise texture. Structure is added in regions where the line-density to brightness mapping is creating uniform regions which lack contrast. Similarly to the parallel-coordinates implementation individual lines are clustered which results in a highly scalable visualization approach. Selected data is indicated in red, whereas the context is again visualized using a grey-scale representation. The visualizations in Figure 6 show plots of the amount of soot per cell. Since soot is burning up over time, all curves are decreasing monotonously. The user-defined selection represents portions of the data set where soot is initially burning at a fast rate. Although the dissipation of soot starts early in the selected regions, it takes longer to vanish completely compared to the remainder of the data set. Here, the parameters in Equation 9 have been set to $n = 3$ and $d = 5$, respectively.

5.3. Phase-Space Diagram

A phase-space diagram is a generalization of the time series visualization presented in the previous section. Instead of plotting the evolution of one scalar value over time, curves representing the evolution of each data item with respect to two arbitrarily selectable data attributes are rendered. Again 260K poly-lines, each containing 19 line segments, are displayed in Figure 7. The x-axis is mapped to the temperature within a cell, and the y-axis represents the amount of contained oxygen O_2. The focus represents the same subset of the data as in Figure 6. In Figure 7 (right), our technique has been used to enhance the phase-space plot shown in the left image. The coherent structures added in the right figure nicely depict regions where data items evolve over time in a similar fashion. The combustion process of the soot in the selected regions can be tracked easily. The combustion starts after a short heating period at the region indicated by the blue ellipse. After this, temperature and oxygen concentration drop off significantly. Here, the parameters in Equation 9 have been set to $n = 3$ and $d = 4$, respectively.

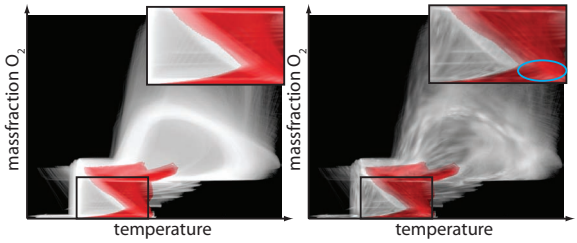

Figure 7: *Enhancing phase-space diagrams with our method (right image). The focus data is the same as in Figure 6. For comparison, the original visualization is shown in the left image.*

5.4. Impact on Rendering Performance

We evaluate the performance for the parallel-coordinates visualization rendered to a 1024×768 view port on an NVIDIA Geforce GTX 480 graphics card. Basic line rasterization has to be performed for most visualization approaches which employ lines as rendering primitives. The line rasterization stage introduced in Section 4.1 can be performed as part of the basic line rasterization by exploiting multiple render targets. Therefore, no separate rendering pass has to be used to generate the fields O and L. Writing to additional render targets increases the fill-rate requirements of the original algorithm. However, we found that this does not strongly impact performance since the line rasterization process seems to be limited by geometry throughput. We also found the difference between rendering to a single or to three floating-point render targets to be also barely measurable at 1 ms. If the original visualization method uses only a single 8-bit fixed-point render target, the overhead incurred by the additional two floating-point targets increases to 50 ms.

However, the most important property of the proposed technique with respect to rendering performance is that the remainder of the rendering pipeline is purely image-based. For these stages, the data size does not influence rendering speed at all. Instead, the output image resolution is the limiting factor. In image space, computing the anisotropic diffusion requires the most time, since a high number of iterations has to be computed, where each iteration performs a large number of texture fetches. For 128 iterations for both focus and context data, i.e., 256 iterations in total, our system needed 104 ms. Since mip-map generation is a hardware feature of current GPUs, the impact on the overall rendering time for the generation of the multi-resolution image pyramid is negligible (below 1 ms). The evaluation of an anisotropy measure also barely influences performance. The final image-compositing step requires only 1 ms to blend multiple noise levels for the final coherent noise texture.

5.5. Comparison to Previous Work

The techniques proposed in this paper extend previous work published by Zachow et al. [ZMH*09] in several important ways. Our approach tightly integrates an anisotropy measure into the visualization pipeline. This allows to better distinguish between regions of parallel lines and areas of more uniformly-distributed line orientations. An example is given in Figures 8 (a) and (b), which depict zoom-ins from the parallel coordinates plot shown in Figure 1. Our improved technique (b) can depict a group of lines which cross most of the remaining lines at a right angle. The method proposed by Zachow et al. (a) hides large parts of this important feature (see blue rectangle). Figure 8 (d) depicts how multiple noise resolutions are used to represent large-scale features (yellow rectangle) and small-scale features (green rectangle). Also, Zachow et al. use only high-resolution noise (Figure 8(c)). An additional difference is that focus+context visualization

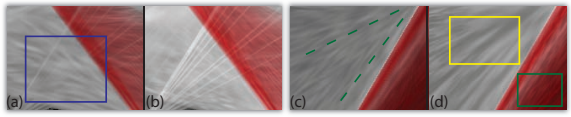

Figure 8: *Comparison of our method (b, d) and the technique proposed by Zachow et al. [ZMH*09] (a, c).*

is deeply integrated into our coherent noise-generation approach since multiple separate tensor fields are used. This is not the case in the previous technique, which handles only a single tensor field representing the whole data set. In addition to these conceptual differences, the noise diffusion itself has been improved. Instead of using convolution with locally-deformed Gaussian kernels to emulate the diffusion process without computing actual diffusion, this work solves a partial differential equation iteratively via a stencil developed by Weickert [Wei98]. This alleviates the problem that certain orientations (indicated as green dashed lines in Figure 8 (c)) are overemphasized when using simple convolution to perform the blurring.

6. Conclusions and Future Work

This work introduces a novel, generic image-based technique that can be used to enhance many existing visualization methods that depict a large number of lines in 2D. A 2D tensor field, subsuming line orientations per pixel, is used to drive anisotropic non-linear diffusion of white noise. The resulting coherent noise texture improves the perception of overall line orientations. Multiple noise frequencies are combined, with blending weights determined by a local feature-scale measure. Using this approach, line orientations in large-scale features can be illustrated with low-frequency noise, whereas high-frequency noise is used for small-scale features. We presented two different ways of how focus+context rendering can be integrated. The general applicability and usefulness of our method has been demonstrated on three different line-based visualization methods.

Future work will deal with extending continuous visualization techniques, such as continuous parallel coordinates [HW09], or continuous phase space diagrams. The latter would be a time-dependent extension to continuous scatter plots [BW08, LT10]. Furthermore, additional blending strategies for the combination of different noise levels could be explored.

7. Acknowledgements

The authors thank Thomas Schultz and Andrea Kratz for valuable input on tensor-related topics, and Wolfgang Freiler for help with figures. The diesel particulate filter data set is courtesy of AVL List GmbH, Graz, Austria. Parts of this work were funded by the Austrian Research Funding Agency (FFG) in the scope of the project “AutARG” (No. 819352)

References

[BW08] BACHTHALER S., WEISKOPF D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1428–1435. 3, 9

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In *Proceedings SIGGRAPH '93* (1993), pp. 263–272. 3

[CS99] CAI W., SAKAS G.: Data intermixing and multi-volume rendering. *Computer Graphics Forum (Eurographics '99) 18*, 3 (1999), 359–368. 7

[DMG*04] DOLEISCH H., MAYER M., GASSER M., WANKER R., HAUSER H.: Case study: Visual analysis of complex, time-dependent simulation results of a diesel exhaust system. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)* (2004), pp. 91–96. 7

[DPR00] DIEWALD U., PREUSSER T., RUMPF M.: Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces. *IEEE Transactions on Visualization and Computer Graphics 6*, 2 (2000), 139–149. 3, 5

[ED07] ELLIS G., DIX A. J.: A taxonomy of clutter reduction for information visualisation. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1216–1223. 2

[EG06] E.BERTINI, G.SANTUCCI: Give chance a chance: modeling density to enhance scatter plot quality through random data sampling. *Information Visualization 5*, 2 (2006), 95–110. 2

[FWR99] FUA Y.-H., WARD M. O., RUNDENSTEINER E. A.: Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings IEEE Visualization '99* (1999), pp. 43–50. 2, 3

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 741–748. 2

[HvW09] HOLTEN D., VAN WIJK J. J.: Force-directed edge bundling for graph visualization. *Computer Graphics Forum 28*, 3 (2009), 983–990. 2

[HW09] HEINRICH J., WEISKOPF D.: Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1531–1538. 3, 9

[Ins85] INSELBERG A.: The plane with parallel coordinates. *Visual Computer 1*, 4 (1985), 69–91. 2

[JLC07] JOHANSSON J., LJUNG P., COOPER M.: Depth cues and density in temporal parallel coordinates. In *EuroVis07: Joint Eurographics - IEEE VGTC Symposium on Visualization* (2007), pp. 35–42. 2

[JLJC06] JOHANSSON J., LJUNG P., JERN M., COOPER M.: Revealing structure in visualizations of dense 2d and 3d parallel coordinates. *Information Visualization 5*, 2 (2006), 125–136. 2

[Kin04] KINDLMANN G.: Superquadric tensor glyphs. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)* (2004), pp. 147–154. 2

[LT10] LEHMANN D., THEISEL H.: Discontinuities in continuous scatter plots. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1291–1300. 9

[MKO*08] MUIGG P., KEHRER J., OELTZE S., PIRINGER H., DOLEISCH H., PREIM B., HAUSER H.: A four-level focus+context approach to interactive visual – analysis of temporal features in large scientific data. *CGF 27*, 3 (2008), 775–782. 1, 3, 8

[MM08] MCDONNELL K. T., MUELLER K.: Illustrative parallel coordinates. *Computer Graphics Forum 27*, 3 (2008), 1031–1038. 2

[NH06] NOVOTNY M., HAUSER H.: Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 893–900. 2, 8

[TS06] TELEA A., STRZODKA R.: Multiscale image based flow visualization. In *Proc. of SPIE-IS&T Electronic Imaging, Visualization and Data Analysis (VDA) 2006* (2006), vol. 6060, pp. 1–11. 3, 5

[vW91] VAN WIJK J.: Spot noise: Texture synthesis for data visualization. In *Proceedings SIGGRAPH '91* (1991), pp. 309–318. 3

[Wei98] WEICKERT J.: *Anisotropic Diffusion in Image Processing*. Teubner-Verlag, 1998. 3, 5, 9

[WH06] WEICKERT J., HAGEN H.: *Visualization and Processing of Tensor Fields*. Springer, 2006. 3

[Wit83] WITKIN A. P.: Scale-space filtering. In *Proc. International Joint Conference on Artificial Intelligence* (1983), pp. 1019–1022. 3

[ZMH*09] ZACHOW S., MUIGG P., HILDEBRANDT T., DOLEISCH H., HEGE H.-C.: Visual exploration of nasal airflow. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1407–1414. 3, 9