Eurographics Conference on Visualization (EuroVis) 2012
S. Bruckner, S. Miksch, and H. Pfister
(Guest Editors)

Volume 31 (2012), Number 3

Procedural Texture Synthesis for Zoom-Independent Visualization of Multivariate Data

R. Khlebnikov[1], B. Kainz[1], M. Steinberger[1], M. Streit[2] and D. Schmalstieg[1]

[1]Institute for Computer Graphics and Vision, Graz University of Technology, Austria
[2]Institute of Computer Graphics, Johannes Kepler University Linz, Austria

Abstract

Simultaneous visualization of multiple continuous data attributes in a single visualization is a task that is important for many application areas. Unsurprisingly, many methods have been proposed to solve this task. However, the behavior of such methods during the exploration stage, when the user tries to understand the data with panning and zooming, has not been given much attention.

In this paper, we propose a method that uses procedural texture synthesis to create zoom-independent visualizations of three scalar data attributes. The method is based on random-phase Gabor noise, whose frequency is adapted for the visualization of the first data attribute. We ensure that the resulting texture frequency lies in the range that is perceived well by the human visual system at any zoom level. To enhance the perception of this attribute, we also apply a specially constructed transfer function that is based on statistical properties of the noise. Additionally, the transfer function is constructed in a way that it does not introduce any aliasing to the texture. We map the second attribute to the texture orientation. The third attribute is color coded and combined with the texture by modifying the value component of the HSV color model. The necessary contrast needed for texture and color perception was determined in a user study. In addition, we conducted a second user study that shows significant advantages of our method over current methods with similar goals. We believe that our method is an important step towards creating methods that not only succeed in visualizing multiple data attributes, but also adapt to the behavior of the user during the data exploration stage.

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

1. Introduction

The visualization of multiple continuous variables is often necessary for analyzing data in many disciplines [BH07]. While this can be done by showing several juxtaposed visual representations of univariate data (small multiples), studies show that simultaneous visualizations are advantageous over adjacent representations in solving complex tasks [VCL11]. Furthermore, in adjacent visualizations, it is difficult to notice slight differences in similar datasets, which may be important for data analysis [Tay02].

Showing multiple data attributes in a single visualization presents several challenges. First, the number of communication channels that a user can perceive in a visualization is very limited. To a certain extent all of them have been employed in various multivariate visualization methods, for example, color [Mil07], or the combination of texture and color [HvWM06, UIM*03]. Second, when users try to understand the data at hand by using a specific visualization method, they usually need to interact with the representation. For example, a user may zoom out to gain the overview over the data or zoom in to see more details. However, little attention is given to the behavior of multivariate visualization methods during this data exploration stage.

Methods that employ texture for the visualization of data attributes are especially sensitive during interaction. On the one hand, if the texture is created so that all the small details would be represented in the visualization, it may become aliased and too fine for the depicted values to be perceived when the user zooms out to gain a global overview of the data. On the other hand, if the texture is created for global overview, zooming in will reveal a lack of detail that may hide important features of the data. Therefore, creating

a texture that adapts to the current zoom level would be advantageous.

In this paper, we propose a method that uses texture synthesis based on procedural noise to create zoom-independent visualizations. The procedural nature of our texture synthesis model allows us to adapt the texture to any zoom level. As a consequence, the resulting textures are alias-free and data values can be interpreted at any scale. The base for our texture synthesis model is random-phase Gabor noise [LLD11], which has predictable statistical properties and allows a high degree of control. To encode the data values, we map them to two texture properties. First, we map data values to the texture frequency. Second, we use a transfer function for the noise values to correlate the texture density with the data values. Our method is also able to produce anisotropic textures, which can be used to visualize one more scalar value with the orientation of the texture. To produce the final image, the generated texture is composed with underlying color information by modifying the value component within the HSV color model as described in Section 3. The limitations of our method are discussed in Section 5. In Section 7, we have conducted a preliminary user-study to find suitable contrast levels for our method and we show strong evidence that our method is effective by providing the results of a second comparative user study, which shows that our method surpasses commonly used multivariate visualization methods.

2. Related work

We subdivide the work that is related to our method in three distinct categories. The first category contains methods, which do not use noise, but aim for visualizations of continuous multivariate data. The second category consists of algorithms using noise-based methods for visualization. In the third category, we discuss a separate area of research – stylized rendering and animation. The goals of stylized animation almost coincide with our goal of zoom-independent texture synthesis.

2.1. Visualization of continuous multivariate data

Tang *et al.* propose to use natural textures for the visualization of weather data [TQWZ06]. While this method is comparable to our method, the authors do not discuss the behavior of their method during user interaction. Given that the texture is generated once per dataset and not adapted to the zoom level, this method may lead to strong aliasing artifacts during data exploration. A similar problem can be found using the *color weaving* method proposed by Urness *et al.* [UIM*03]. They visualize several scalar fields in a texture obtained with line integral convolution (LIC). To display several variables, the authors propose to select highly saturated and perceptually iso-luminant colors first. Subsequently, they build a 2D colormap for each variable and use the saturation component to show the actual value of the variable. The value component is used to maintain the original contrast features of the LIC texture. The authors also propose to use multiple frequencies for LIC texture generation to highlight the regions of interest. To alleviate the problem of under-representing, the authors propose to super-sample the data before applying their algorithm. This may again lead to aliasing artifacts if the super-sampled dataset resolution exceeds the display resolution at an overview level.

The attribute blocks method by Miller subdivides the screen into a regular grid of blocks [Mil07]. Each block corresponds to an array of visual representations (*lenses*). In each lens, a single attribute is displayed with color maps similar to [UIM*03] (saturation of a distinct color). The author proposes two possibilities for the behavior of attribute blocks when the user scales the map. The first possibility is to link the attribute blocks to the object space, *i. e.* , the attribute blocks are scaled along with the map. This may lead to undesired results on large and small scales, especially if the size of a single attribute lens becomes smaller than a pixel. The second possibility is to link the attribute block size to screen space. This leads to an effect resembling a 'screen door of lenses', as the map moves and the grid of attribute blocks stays static. In contrast, our method adds or removes the details smoothly in-place, which results in a more consistent visualization *during* the scaling process.

Healey *et al.* employ perceptually-based brush strokes for non-photorealistic visualizations [HTER04]. The authors discuss the behavior of their method during scaling. In contrast to our work, they discuss the display of additional data rather than the adaptation of the approach for particular zoom level.

Kosara *et al.* refer to blur based methods as *semantic depth of field* (SDOF) [KMH01]. Their main idea is to blur information which is not relevant for the current cognitive task. However, the authors showed in later work [KMH*02] that SDOF cannot be used as a full visualization dimension because users are not able to distinguish between different levels of blur.

2.2. Noise-based methods

Noise-like textures were used for information display in the work by van Wijk based on 'spot noise' [vW91], which has been extended for multivariate data visualization by Botchen *et al.* [BWE05]. However, these methods require a complex definition of a constructing element (*i. e.* , spot shape), which should be adapted for every task at hand. In the work by Holten *et al.* [HvWM06], the authors propose a method to generate textures with desired properties for filling several disjoint areas (*e. g.* , member states of the USA). The drawback of direct spectral methods of texture synthesis is that they lack local spatial control and this almost prohibits their use for the visualization of continuous data. While these methods are very powerful, they are not zoom-independent

and their performance may be compromised during the data exploration stage.

In recent work, Coninx *et al.* propose to use animated Perlin noise in conjunction with classic color maps to convey the information about uncertainty of one scalar field displayed on 3D surfaces [CBDT11]. The authors use the intensity of Perlin noise multiplied by the amount of uncertainty as lookup offset in a colormap. As a result, the data appears noisy in areas where data is uncertain. Thorough analysis of low-level perception of noise contrast allows to adjust the noise to ensure that the uncertain areas are visible in the resulting visualization. While our method shares the idea of using noise for visualization purposes, our method aims at visualizing arbitrary data and is not limited to uncertainty of scalar fields.

2.3. Stylized rendering

Preserving texture appearance and scale is a common task in stylized rendering and animation. In this context, it is referred to as *flatness requirement* [BBT11]. Two other requirements need to be met to assure high quality animation with stylized rendering. *Motion coherence* refers to the correlation between the motion of a 3D scene and the motion of stylization primitives in screen space, and *temporal continuity*, corresponds to the absence of popping and flickering artifacts. These three goals are inherently contradictory and therefore a tradeoff between them must be made. However, the motion coherence requirement is not relevant for our application. Therefore, we only need to preserve flatness and temporal continuity. We use one of the methods that satisfies these two requirements well, namely random-phase Gabor noise as proposed by Lagae *et al.* [LLD11], as a basis for our visualization method.

3. Method

We adapt the texture created with random-phase Gabor noise for the purpose of visualizing continuous multivariate data. To employ texture for visualization, one needs to understand how users perceive texture characteristics. The main texture characteristics perceived by the human visual system are spatial frequency and orientation. The human visual system is fine-tuned for the perception of a specific range of spatial frequencies, which under normal viewing conditions, corresponds to a range of 4 to 40 pixels [WK95]. We ensure that texture frequency lies within this range at arbitrary zoom levels. Furthermore, the flexibility of random-phase Gabor noise allows us to avoid flickering and popping artifacts during zooming.

Using random-phase Gabor noise we can convey three distinct scalar data attributes. To visualize the first attribute, we use local texture frequency (Section 3.2). The second attribute is shown with local texture orientation (Section 3.3). To strengthen the perception of the data attribute shown with texture frequency, we map the noise values to texture intensity using a noise transfer function (Section 3.4). We then use the resulting texture to modulate the 'value' component of the color-coded third attribute using the HSV color model (Section 3.5).

3.1. Random-phase Gabor noise

Before we show how the data attributes are displayed with the texture, we give a short overview of the random-phase Gabor noise that serves as a basis for its construction. For a more detailed description, please refer to the original work by Lagae *et al.* [LLDD09, LLD11, LD11].

Random-phase Gabor noise is based on the convolution of random positions using the phase-augmented Gabor kernel:

$$g(\mathbf{x}; a, \omega, \phi) = e^{-\pi a^2 |\mathbf{x}|^2} \cos(2\pi \mathbf{x} \cdot \omega + \phi), \qquad (1)$$

where a is the bandwidth, ω and ϕ are the frequency and phase of the harmonic. The value of random-phase Gabor noise with bandwidth a and frequency ω at an arbitrary position $\mathbf{x}$ is then computed as:

$$\mathcal{N}(\mathbf{x}) = \sum_i g(\mathbf{x} - \mathbf{x}_i; a, \omega, \phi_i), \qquad (2)$$

where $\mathbf{x}_i$ are positions distributed according to an n-dimensional Poisson impulse process with impulse density λ. n is the dimensionality of the desired noise. ϕ_i is the random phase associated with an impulse at position $\mathbf{x}_i$. Phases ϕ_i are distributed according to uniform distribution in the interval $[0, 2\pi)$

Obviously, the Gabor kernels (Eq. 1) have infinite support, so computation of the exact value at a single position would require significant time. To overcome this issue, Gabor kernels are first truncated at some threshold level t. Then, a virtual uniform grid with cell size equal to the radius of the truncated kernel G is introduced. Consequently, to compute a noise value at a certain position, it is only necessary to account for impulses in the cell this position belongs to, and its immediate neighbors.

We obtain the spatially varying random-phase Gabor noise according to Lagae *et al.* [LLD11]. This technique allows us to adapt the frequency of the noise according to the continuously changing value that has to be displayed.

3.2. Frequency control

Without loss of generality, let us assume that the value v that has to be conveyed in the visualization lies in the interval $v \in [0, 1]$. To avoid sliding artifacts during panning and zooming, we compute the noise values using the coordinates in the original data space. However, to ensure that the screen space frequency of the resulting texture lies in the range perceived well by the human visual system, we adjust the noise parameters according to the current zoom level. To achieve that, we choose the grid size G in data space so that

its screen space size lies within this range and the exact size corresponds to the value being visualized:

$$G = Z^{-1} \cdot s \cdot 2^{(1-v)\cdot \log_2(S/s)}, \tag{3}$$

where s and S are the desired minimum and maximum screen-space sizes of the noise, and Z is the current zoom level defined by the size in pixels of one data grid cell on the screen. The exponential dependence on the visualized value is due to the fact that repeatedly doubling the texture frequency results in equal perceptual steps, while adding a constant value does not [HvWM06]. The bandwidth a is then computed as:

$$a = \frac{\sqrt{-\ln(t)/\pi}}{G}, \tag{4}$$

and the radial frequency f is computed as [War04, p. 164]:

$$f = 1/G. \tag{5}$$

3.3. Orientation control

The isotropic random-phase Gabor noise is obtained if the parameters $\{\omega_i\}$ associated with impulse positions $\mathbf{x}_i$ are computed in spherical coordinates with the radial component set to the desired frequency f (Eq. 5) and the angular frequency (or frequencies in case of 3D noise) is randomized [LD11]. If the randomization is not performed, the noise will be anisotropic. By choosing the angular component of $\{\omega_i\}$ according to one of the displayed data attributes, we are able to link the local orientation of the resulting texture to the value of that attribute.

3.4. Noise transfer function

To strengthen the perception of the attribute visualized with texture frequency, we redundantly encode this attribute with average texture intensity. The texture intensity at a position $\mathbf{x}$ in space is computed according to the noise value $\mathcal{N}(\mathbf{x})$ at this point and a *noise transfer function*. The noise transfer function is similar to the regular transfer functions used in direct volume rendering but it maps the data values to a single scalar instead of a RGBA quadruplet.

To correlate the noise with the actual values, we impose the following constraint on the noise transfer function $\alpha_n(t)$:

$$\forall v : \int_{-\infty}^{\infty} p(t) \cdot \alpha_n(t) \mathrm{d}t = v, \tag{6}$$

where $p(t)$ is the probability density function (PDF) of the intensity distribution of the noise. Note that imposing such a constraint for random-phase Gabor noise is possible because its PDF is predictable. Please refer to Appendix A for the exact formulation of the random-phase Gabor noise PDF and the proof of its independence from the size of the virtual grid G.

To construct a transfer function that satisfies condition in Eq. 6, we define a family of compactly supported *decay* functions $\mathcal{D} : \mathbb{R} \to \mathbb{R}$. In order to compute the intensity value at position $\mathbf{x}$, we first choose the concrete decay function $D \in \mathcal{D}$ that satisfies the condition in Eq. 6 for a particular value v of the scalar field at position $\mathbf{x}$. We then evaluate the procedural noise at position $\mathbf{x}$ to get the noise value $\mathcal{N}(\mathbf{x})$. The texture intensity at position $\mathbf{x}$ is finally set to $D(\mathcal{N}(\mathbf{x}))$.

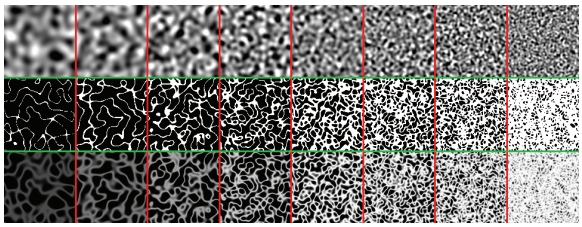

Figure 1: *Comparison of effect of noise transfer functions. The value that is visualized changes from zero to one. Top: underlying noise without modifications. Middle: after application of step decay function. Bottom: After application of spline decay function. To see the details, please zoom in or refer to Appendix C.*

Additionally, we avoid aliasing that may occur when using a transfer function (see the middle row of Fig. 1 for an example of aliasing occurring when applying a simple step transfer function). In order to achieve this, we avoid high frequencies in the resulting function. Therefore, it is not enough to know the power spectrum of the noise, but also how the transfer function modifies it. According to Bergner *et al.* [BMWM06], the power spectrum is modified by the transfer function in the following way:

$$f = f_n \cdot \max_t \left| D'(t) \right|, \tag{7}$$

where f is the maximum frequency of the resulting function, f_n is the maximum frequency of the underlying noise function and $D'(t)$ is the derivative of the transfer function. To avoid introducing high frequencies to the resulting function, we use a family of C^2-continuous cubic spline decay functions of the following form:

$$\mathcal{D}^C_{\gamma,q}(t) = \begin{cases} \frac{2q}{\gamma^3} \cdot |t|^3 - \frac{3q}{\gamma^2} \cdot |t|^2 + q & \text{if } |t| \le \gamma \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $D^C_{\gamma,q}$ has the maximum value q and is nonzero for $t \in [-\gamma;\gamma]$. It is straightforward to show that $\max_t \left| D'(t) \right| = 3q/2\gamma$. If we know the screen-space frequency of the noise, we can choose this parameter in a way that it does not introduce aliasing. In our implementation, we ensure that $\max_t \left| D'(t) \right| \le 1$. It should be noted that we cannot maintain the value of 1 for $D^C(0)$, so we reduce the intensity of this peak in order to guarantee that the requirement in Eq. 6 is met and the frequency requirement is not violated. See Figure 2 for several example functions from the family $\mathcal{D}^C_{\gamma,q}$.

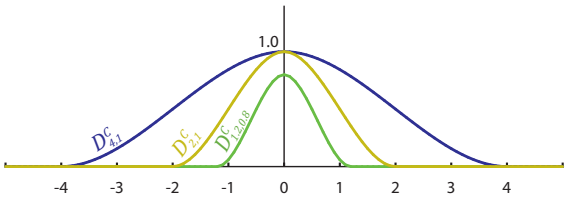

Figure 2: *Several representative cubic spline decay functions with varying parameters. The parameters γ and q are computed so that the condition in equation 6 is satisfied for a value v and that* $\max_t \left|D'_{\gamma,q}(t)\right| < 1, \forall D_{\gamma,q} \in \mathcal{D}^C$.

There is no closed form solution for the parameter γ. Therefore, we precompute a lookup table by evaluating the integral in Eq. 6, invert it to get a lookup table for $\gamma(v)$, and use linear interpolation to obtain intermediate values.

3.5. Color blending

Blending the obtained noise texture with color deserves special attention. According to Holten *et al.* [HvWM06], additive, multiplicative or alpha blending do not work well. They use a combination of additive and multiplicative blending depending on the value encoded in the texture. However, we observed that this method does not work well with textures obtained with our method as too little of the original color stays visible. Therefore, in order to combine our texture with color, we convert the RGB color value to the HSV color space and modify the value component (v-component) according to the computed texture intensity at this point in space. This allows us on the one hand to maintain the hue and saturation of the original color and on the other hand have enough contrast for the texture to be perceptible. Furthermore, we can keep a significant amount of the original v-component. There are two reasons for doing so. First, keeping a part of the original v-component helps us to maintain the resolution of the underlying color image, as zero values of the noise transfer function do not result in a black color. Second, we do not need the full value component range to make the texture perceptible. Overall, the value component of the color is computed as:

$$\hat{C}_v = \beta \cdot C_v + (1-\beta) \cdot \alpha_n, \tag{9}$$

where β is a constant, whose value controls the amount of the original v-component in the resulting image, C_v is the original and $\hat{C}_v$ is the modified v-component. In our implementation, we set $\beta = 0.6$ (see Section 7).

4. Applications

A common task in many application fields is to correlate multiple dimensions of a dataset with each other in order to draw conclusions from it. We have applied the proposed method to two 2D application scenarios. In both examples, we encode additional information on top of an existing base visualization. Additionally, we show that our method scales to three dimensions by providing a 3D use case.

4.1. Encoding additional information in video data

The first example is concerned with the evaluation of data acquired during perceptual experiments performed with an eye-tracker, as described in detail in [VMFS11]. In a user study, the authors evaluated in which way and how much the visual bottom-up saliency [Itt05] of a scene can be altered to steer the attention of a subject to a certain piece of information without noticeable changes for the subject. In order to evaluate the success of their method, the authors needed to compare three videos in a side-by-side fashion, as shown in Fig. 3 (top row). The first video is the original source video. The second video visualizes the user's gaze while watching the first video during the experiment. The time a subject was looking at a certain spot in the shown scene is color-coded. The third video shows the calculated visual saliency for the original source video. In this evaluation scenario the visual variable *color* is already used by the gaze heatmap and therefore cannot be used for visualizing the modulated saliency on top of the same video. We applied our method to to integrate the visual saliency as well as the user's gaze concurrently on top of the video. We set the v-component of the user's gaze color map to 1 so that the texture does not affect the values read from it. The rest of the frame is only context information and therefore v-component changes are not required. Figure 3 (middle and bottom rows) shows the combined result for a single sample video frame. By inspecting the combined image, the expert who evaluates the experiment can identify that users tend to look more on saliency-modulated spots – which supports the hypothesis made in [VMFS11].

4.2. Encoding additional information on top of geospatial data

A second example is the visualization of weather data stemming from various sources. We have selected freely available global data from NOAA (http://www.noaa.gov/) at a resolution of approximately 11 km/pixel (4096 x 2048 pixels). We have chosen to combine color-coded sea temperature with the amount of precipitation over the sea and the precipitation's optical flow, which encodes the direction of movement of precipitation areas. Precipitation is encoded with texture frequency and texture intensity. The optical flow of the precipitation density is encoded as texture orientation. As we obtained data for several days, we are able to display an animated sequence of the changes in time for these values. Figure 4 illustrates different zoom levels and possible findings from a map encoding precipitation and sea temperature from the 14^{th} of September 2011.

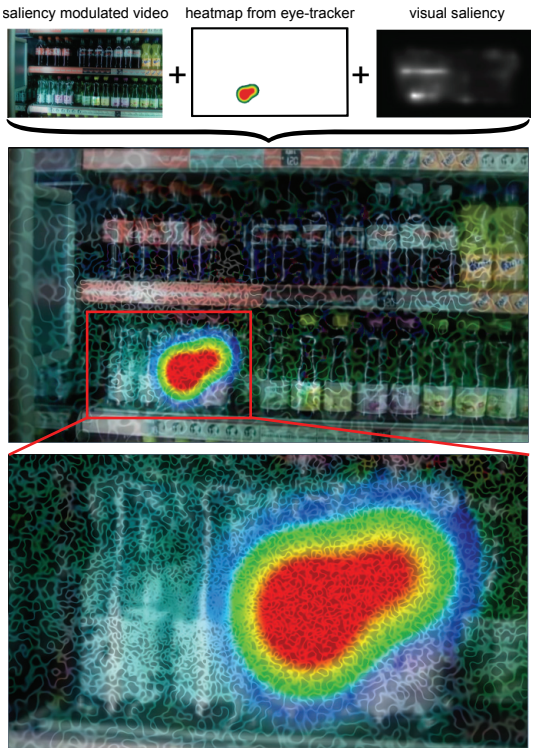

Figure 3: *This figure shows our approach during the analysis of an eye-tracker experiment with visual saliency modulated videos [VMFS11]. The visual saliency is visualized as texture frequency and texture intensity. Highly salient regions are encoded with high texture frequency and intensity. Concurrently, we show the time the user looked at a specific region with a color-coded heatmap. To see the details, please zoom in or refer to Appendix C.*

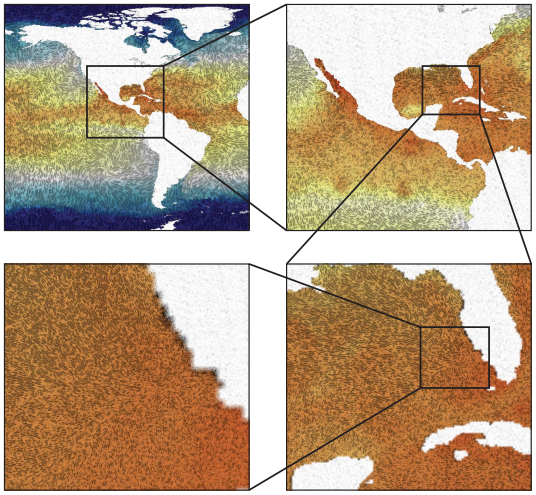

Figure 4: *The shown world map color codes the average sea temperature from 14^{th} of September 2011. On top, the procedural noise encodes the amount of precipitation (noise frequency and texture intensity) with the optical flow of the precipitation between 14^{th} and 15^{th} of September 2011 (texture orientation). Encoding the optical-flow value between two points in time as second visualization channel, allows the user to estimate the evolution of the precipitation value as well. Note the increasing amount of details with increasing zoom level. To see the details, please zoom in or refer to Appendix C.*

4.3. Three-dimensional use case

As our procedural texture method is not limited to 2D we also applied it to convey additional values in 3D volume rendering. This use case demonstrates that the simplicity of our method allows it to be directly scaled to 3D. In this example, we display positional uncertainty of isosurfaces. We encode the probability of a ray crossing the uncertain isosurface [PH11] with the 3D texture density (higher probability – denser texture). Furthermore, we estimate the spatial deviation of the surface using the stochastic distance function [PRW11] and map it to the texture frequency (higher deviation – lower frequency). Please refer to Appendix B for the mathematical formulation of these quantities.

The resulting visualization, obtained with direct volume rendering, is shown in Fig. 5 and in the accompanying video. The level-crossing probabilities can be perceived in the view-perpendicular directions (*i. e.* , along the silhouette of the isosurface), while the spatial deviation can be perceived in the view-parallel directions. While the method by Pfaffelmoser *et al.* [PRW11] also allows the visualization of spatial variability of an uncertain isosurface in view-perpendicular directions, we believe that our method will perform better if the user needs to understand the spatial relationship between two or more uncertain isosurfaces because it is possible to choose a distinct color for each of the corresponding 3D textures. Nevertheless, a thorough examination of 3D noise-based texture perception and a comparison to the existing methods [DKLP02, PRW11, PWH11] is required to judge the effectiveness of our method in 3D applications. We are excited to explore these topics in future work.

5. Limitations

While our method provides a way to display both, an overview and details at different zoom levels, it is prone to some data hiding similar to other methods that employ texture-based visualization. This problem is not given much attention in the related work. Urness *et al.* propose to upsample the data in order to not hide any information in the original data [UIM*03]. Shenas and Interrante propose to choose a texture, such that its frequency exceeds the data frequency [SI05]. Taylor also points out the same problem and

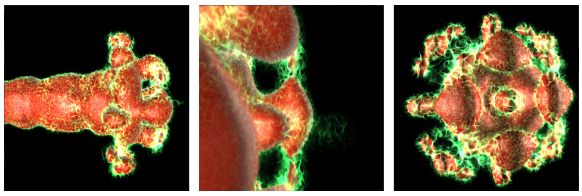

Figure 5: *Several views of a 3D isocontour uncertainty visualization of the Fuel dataset using our visualization method. Texture density shows the level-crossing probability and texture frequency shows the estimate of spatial deviation of uncertain isosurface, with higher frequencies corresponding to lower deviations. To see the details, please zoom in or refer to Appendix C.*

relies on user interaction with the visualization to see the details [Tay02]. Our method is also limited in this respect, *i. e.* , our method cannot show the data variations whose frequency exceeds that of the texture. However, as the texture is distinguishable at any scale, we can at least show the general trends within the dataset and rely on the user to zoom in the areas of high interest. For our method, mapping the most important and/or most varying data attribute to the underlying color simplifies the task of finding regions of interest, because the perceivable resolution is higher for the color-mapped attribute than for the others.

6. Implementation and performance

We implemented our method for execution on GPUs using NVIDIA CUDA [NVI11]. As the evaluation of procedural noise for every position is independent from others, the parallelization of the evaluation procedure is straightforward. The visualized datasets are stored as textures on the GPU, which enables hardware accelerated bilinear interpolation and texture caching.

The implementation is based on the Visualization Toolkit (VTK) [SML06]. We use VTK as the framework for basic interaction, data loading and rendering of color-coded data. For every frame, we compute a viewport-size overlay with our procedurally generated texture. The texture itself is generated using CUDA. We compute a transformation matrix from screen space to data-texture space and pass this matrix to the texture-synthesis procedure so it is able to sample the values of the data textures. The acquired values, along with the screen-space requirements for noise, are then used to control the noise parameters.

The performance of our method depends mostly on the performance of the procedural noise evaluation. As all the noise parameters are computed on-the-fly, no additional heap or global memory in CUDA is required. On our test system (Intel Core i7-870, 8GB RAM, NVIDIA GeForce GTX 480), the synthesis of a 2D texture with a resolution of 1680x988 pixels takes on average 19ms in case anisotropic noise is used, and 24ms if isotropic noise is used. In total, the frame rate of the whole system does not fall below 30 frames per second.

7. Optimal color blending

In order to evaluate the influence of the parameter β (Eq. 9) on the ability to read data values, we conducted an experiment with 79 participants. In a pilot study we identified a β range of 0.5 to 0.9 to be suitable. For the main experiment we tested the β values 0.5, 0.6, 0.7, 0.8, and 0.9, each creating one group. Participants were randomly assigned to these groups.

As test data, we used 20 triplets of data values drawn from uniform distributions with a minimum value of 0 and a maximum of 1. These triplets were encoded using *texture frequency*, a *color gradient* from red to blue, and *texture orientation*. Using the different β values we created 20 images of 64×64 pixels for each group (see Fig. 6). The participants were shown these 20 images in a randomized order. With the help of a legend, they tried to read the encoded variables and reported their results numerically. The entire task took them approximately 10 minutes.

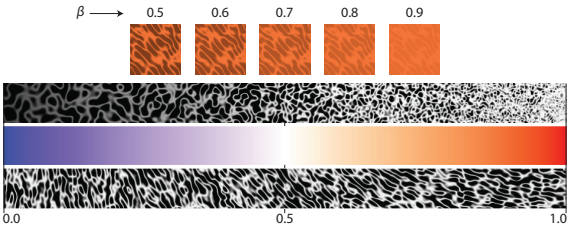

Figure 6: *1^{st} row: an example data triple (0.521, 0.633, 0.312) for the five tested β values. $2^{nd}, 3^{rd}, 4^{th}$ rows: the legend for our method that was used in both our user studies. To see the details, please zoom in or refer to Appendix C.*

We compared the obtained results to the original data values computing the mean squared error (MSE), as depicted in Fig. 7. After outlier removal, we computed a one-way ANOVA. We found no statistically significant difference between the groups for frequency ($F_{4,73} = 1.713, p = .156$), or color gradient ($F_{4,71} = 0.696, p = .597$). There was a main effect for orientation ($F_{4,73} = 4.114, p = .005$). A Tukey post-hoc test revealed that the MSE for $\beta = 0.6$ and $\beta = 0.7$ was statistically significantly lower than for $\beta = 0.9$.

The results indicate that a β value of 0.6 or 0.7 works well, as the MSEs are low for all three attributes. For our further experiments we thus chose $\beta = 0.6$.

8. Comparative user study

We conducted a controlled experiment to evaluate the effectiveness of our visualization method in comparison with

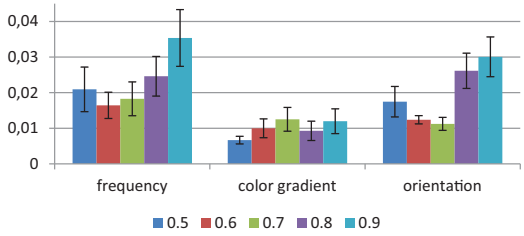

Figure 7: *Average mean squared error for different* β *values between* 0.5 *and* 0.9. *Choosing* β *between* 0.6 *and* 0.7 *seems to create the best results, yielding an average mean squared error below* 2%.

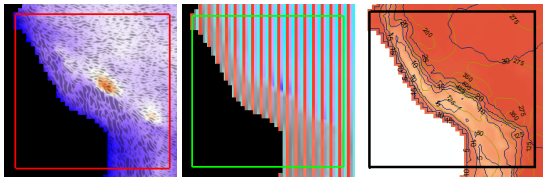

Figure 8: *Examples of the visualizations compared in our user study (from left to right): our method (noise-based texture), attribute blocks [Mil07], and labeled contours (produced with ArcGIS 10). To see the details, please zoom in or refer to Appendix C.*

other multivariate visualization methods. We recruited 18 participants (aged 22 to 35, 15 males, 3 females) from a local university with self-reported normal or corrected-to-normal vision. The participants were from the fields of computer science and economics. Fourteen participants indicated that they had experience with visual data analysis.

We compared three techniques for multivariate data analysis in our user study (Fig. 8):

- **Noise-based procedural texture (N)** - our algorithm
- **Attribute blocks (A)** algorithm [Mil07]
- **Labeled contours (C)** with one variable mapped to color and others mapped to labeled contour lines with different contrast colors.

8.1. Task and Procedure

We designed the tasks relevant for an exploration of geographic data according to Hagh-Shenas *et al.* [HSKT09]. The visualization methods displayed the Climate Research Unit (CRU) high resolution climate data, obtained via Intergovernmental Panel on Climate Change (IPCC). We used temperature, precipitation, and amount of water vapor for January, averaged over 30 years (1961 - 1990). In these datasets, the Earth's surface is sampled with 0.5° intervals, which results in images with a resolution of 720×360 pixels. The exact mapping of variables for each method is shown below. The diverging color scale was used for all methods (attribute blocks required only half of it).

	Our method	Attr. blocks	Contours
Temperature	Frequency	Grey-red	Blue-red
Precipitation	Blue-red	Grey-blue	Contour 1
Water vapor	Orientation	Grey-cyan	Contour 2

The users were provided with a computer with two 22" monitors, one showing a maximized window with the data and the other displayed a corresponding legend. The viewing distance was approximately 60 cm. The study was conducted as a within-subjects experiment with three experimental conditions (technique) and three tasks per condition:

- **'global correlate'** - correlate all pairs of variables on the full map: "If 'a' is small, is 'b' then also small? (yes / no / not clear)"
- **'region comparison'** - determine the location of highest/lowest value for each variable
- **'detail comparison'** - correlate all pairs of variables for a certain region of the map: "Is the value of 'a' generally greater or smaller as the value of 'b' in the pointed area? (yes / no / not clear)".

We have selected three rectangular regions (North-East of South America, a region from India to Vietnam, and Papua New Guinea) for the execution of the 'detail comparison' task. The users were encouraged to interact with the visualization by using the mouse to pan and zoom.

The participants started each condition with a warm up phase reading out values on random locations on the map. The inter-task correctness was averaged to yield one correctness value per task and condition per participant. After each condition, the users completed a questionnaire assessing subjective satisfaction. Upon completion of the experiment, they were asked to assess their overall preference. To reduce the influence of learning effects, the sequence of the conditions was counter-balanced and the sequence of the locations in the 'detail comparison' task was randomized.

Hypothesis Our hypothesis was that the noise-based procedural texture performs better than the other techniques in both quantitative and qualitative assessment.

8.2. Results

Correctness measures were evaluated using repeated measures ANOVA ($\alpha = .05$) with Bonferroni adjusted post-hoc comparisons. Questionnaire answers, which were given on a seven-point Likert scale, were analyzed using Friedman non-parametric tests for main effects and post-hoc comparisons using Wilcoxon Signed Rank tests with Bonferroni adjustments. The questionnaire items concerning preference were analyzed using repeated measures non-parametric Chi Square tests with Bonferroni adjusted post-hoc comparisons. The results are illustrated in Fig. 9.

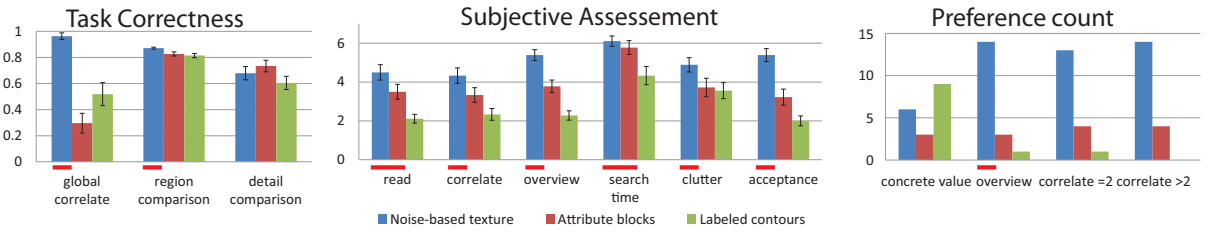

Figure 9: *The results of our user study. From left to right: results of correctness of completing the quantitative tasks by the users, a qualitative evaluation of the methods, and overall preference of the users for choosing a certain visualization method. A higher value is better for all the cases. The red underline shows the methods that were superior in the cases where the results are statistically significant.*

Correctness We found a significant main effect for correctness on 'global correlate' ($F_2 = 29.759, p < .001$) and 'region comparison' ($F_2 = 7.108, p = .003$). Post-hoc comparison revealed that correctness was higher for N than A and C in both cases. We did not find a significant main effect on 'detail comparison' ($F_{1.366} = 2.771, p = .077$).

Subjective Assessment The questionnaire items assessing readability ($\chi^2(2) = 16.687, p < .001$), ability to correlate different data values ($\chi^2(2) = 12.091, p = .002$), ability to find a certain point in the data ($\chi^2(2) = 13.059, p = .001$), and overall acceptance of the technique ($\chi^2(2) = 20.086, p < .001$) were rated significantly higher for N and A than for C. The ability to gain an overview of the data ($\chi^2(2) = 26.226, p < .001$) revealed an order of the techniques with N > A > C. The questionnaire item assessing visual clutter ($\chi^2(2) = 5.848, p > .05$) was not significant.

While the preference counts for reading a concrete data value ($\chi^2(2) = 3.000, p > .05$) was not significant, preferences for getting an overview ($\chi^2(2) = 16.333, p < .001$) were significantly higher for N than for A and C. The preference counts for correlating two data values ($\chi^2(2) = 16.333, p < .01$), and correlating more than two data values ($\chi^2(2) = 13.000, p = .002$) were significantly higher for N than for C, but the differences between N and A and between A and C were not significant.

8.3. Discussion

Our hypothesis was supported by the results of the user study. We found that for global correlation of data and for region comparisons the noise-based procedural texture method performs significantly better than the other two methods. In the subjective assessment, our method achieved the highest score in all questioned asked, with four answered being statistically significant. Furthermore the personal preference count shows a clear support for our method. Only for reading concrete values from the map labeled contours achieved a higher rating, which was not statistically significant. Reading exact data values is not the goal for any of the tested methods. Other techniques, such as interactive data probing, should be used to complement them. Please refer to Fig. 9 for more details.

9. Conclusions and future work

We have presented a method for synthesizing a texture for the display of multivariate information using a procedural noise function. The main advantage of our method is that it gives a possibility to adjust the texture in a zoom-independent manner to avoid aliasing at low scales and keep the data readable at large scales. With our user study we have shown that our method outperforms other methods that are commonly used for similar tasks.

Animating the noise might increase the potential of our method, creating time-varying visualizations of static data. As motion is an additional communication channel, it may be possible to extend our method for visualizing more data attributes, *i. e.* , faster/slower animation for higher/lower values. However, the usage of motion is a very controversial topic in the visualization community, as it may attract a user's attention to unimportant portions of the data and lead to eye fatigue. Nevertheless, we anticipate that with careful evaluation, animation may give additional advantage to our method and we leave it for future work. We will also explore the use of glyph like textures [TQWZ06] for increasing the number of displayed attributes. Using controlled orientation, our method can also be applied to display vector field data in conjunction with scalar data. We will conduct additional studies to determine how our method performs in comparison to current methods for vector field visualization.

Acknowledgements

This research was funded by the Austrian Science Fund (FWF): P23329 and the Austrian Research Promotion Agency (FFG) under the BRIDGE program: project #822702 (NARKISSOS).

References

[BBT11] BÉNARD P., BOUSSEAU A., THOLLOT J.: State-of-the-art report on temporal coherence for stylized animations. *Computer Graphics Forum 30*, 8 (December 2011), 2367–2386. 3

[BH07] BÜRGER R., HAUSER H.: Visualization of multi-variate scientific data. In *EuroGraphics 2007 State of the Art Reports (STARs)* (2007), pp. 117–134. 1

[BMWM06] 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. 4

[BWE05] BOTCHEN R. P., WEISKOPF D., ERTL T.: Texture-based visualization of uncertainty in flow fields. In *In Proceedings of the IEEE Symposium on Visualization* (2005), pp. 647–654. 2

[CBDT11] CONINX A., BONNEAU G.-P., DROULEZ J., THIBAULT G.: Visualization of uncertain scalar data fields using color scales and perceptually adapted noise. In *Applied Perception in Graphics and Visualization* (2011). 3

[DKLP02] DJURCILOV S., KIM K., LERMUSIAUX P., PANG A.: Visualizing scalar volumetric data with uncertainty. *Computers and Graphics 26* (2002), 239–248. 6

[HSKT09] HAGH-SHENAS H., KIM S., TATEOSIAN L.: Multivariate visualization of continuous datasets, a user study. In *Proceedings of the IEEE Symposium on Information Visualization* (2009). 8

[HTER04] HEALEY C. G., TATEOSIAN L., ENNS J. T., REMPLE M.: Perceptually based brush strokes for nonphotorealistic visualization. *ACM Trans. Graph. 23* (2004), 64–96. 2

[HvWM06] HOLTEN D., VAN WIJK J. J., MARTENS J.-B.: A perceptually based spectral model for isotropic textures. *ACM Trans. Appl. Percept. 3* (2006), 376–398. 1, 2, 4, 5

[Itt05] ITTI L.: Models of Bottom-Up Attention and Saliency. In *Neurobiology of Attention*, Itti L., Rees G., Tsotsos J. K., (Eds.). Elsevier, 2005, pp. 576–582. 5

[KMH01] KOSARA R., MIKSCH S., HAUSER H.: Semantic depth of field. In *Proceedings of the IEEE Symposium on Information Visualization* (2001), IEEE Computer Society, pp. 97–104. 2

[KMH*02] KOSARA R., MIKSCH S., HAUSER H., SCHRAMMEL J., GILLER V., TSCHELIGI M.: Useful properties of semantic depth of field for better F+C visualization. In *Proceedings of the Symposium on Data Visualisation* (2002), Eurographics Association, pp. 205–210. 2

[LD11] LAGAE A., DRETTAKIS G.: Filtering solid Gabor noise. In *ACM SIGGRAPH 2011 papers* (2011), SIGGRAPH '11, ACM, pp. 51:1–51:6. 3, 4

[LLD11] LAGAE A., LEFEBVRE S., DUTRÉ P.: Improving Gabor noise. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (2011), 1096 –1107. 2, 3

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2009) 28*, 3 (2009), 54:1–54:10. 3

[Mil07] MILLER J. R.: Attribute blocks: Visualizing multiple continuously defined attributes. *IEEE Computer Graphics and Applications 27* (2007), 57–69. 1, 2, 8

[NVI11] NVIDIA: *NVIDIA CUDA Programming Guide 4.0*. NVIDIA Corporation, 2011. 7

[PH11] PÖTHKOW K., HEGE H.-C.: Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE Transactions on Visualization and Computer Graphics 17*, 10 (2011), 1393–1406. 6

[PRW11] PFAFFELMOSER T., REITINGER M., WESTERMANN R.: Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. *Computer Graphics Forum 30*, 3 (2011), 951–960. 6

[PWH11] PÖTHKOW K., WEBER B., HEGE H.-C.: Probabilistic marching cubes. *Comput. Graph. Forum 30*, 3 (2011), 931–940. 6

[SI05] SHENAS H. H., INTERRANTE V.: Compositing color with texture for multi-variate visualization. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (New York, NY, USA, 2005), GRAPHITE '05, ACM, pp. 443–446. 6

[SML06] SCHROEDER W., MARTIN K., LORENSEN B.: *Visualization Toolkit: An Object-Oriented Approach to 3D Graphics, 4th Edition*, 4th ed. Kitware, 2006. 7

[Tay02] TAYLOR R.: Visualizing multiple fields on the same surface. *IEEE Computer Graphics and Applications 22* (2002), 6–10. 1, 7

[TQWZ06] TANG Y., QU H., WU Y., ZHOU H.: Natural textures for weather data visualization. In *Proceedings of the conference on Information Visualization* (2006), IEEE Computer Society, pp. 741–750. 2, 9

[UIM*03] URNESS T., INTERRANTE V., MARUSIC I., LONGMIRE E., GANAPATHISUBRAMANI B.: Effectively visualizing multi-valued flow data using color and texture. In *In Proceedings of the IEEE Symposium on Visualization* (2003), VIS '03, IEEE Computer Society, pp. 115–121. 1, 2, 6

[VCL11] VIARD T., CAUMON G., LEVY B.: Adjacent versus coincident representations of geospatial uncertainty: Which promote better decisions? *Computers and Geosciences 37*, 4 (2011), 511–520. 1

[VMFS11] VEAS E. E., MENDEZ E., FEINER S. K., SCHMALSTIEG D.: Directing attention and influencing memory with visual saliency modulation. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (2011), CHI '11, ACM, pp. 1471–1480. 5, 6

[vW91] VAN WIJK J. J.: Spot noise texture synthesis for data visualization. In *Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991), SIGGRAPH '91, ACM, pp. 309–318. 2

[War04] WARE C.: *Information Visualization: Perception for Design*. Morgan Kaufmann Publishers Inc., 2004. 4

[WK95] WARE C., KNIGHT W.: Using visual texture for information display. *ACM Trans. Graph. 14* (1995), 3–20. 3