Screen-space Silhouettes for Visualizing Ensembles of 3D Isosurfaces

Ismail Demir* Johannes Kehrer† Rüdiger Westermann‡

Computer Graphics and Visualization Group, Technische Universität München, Germany

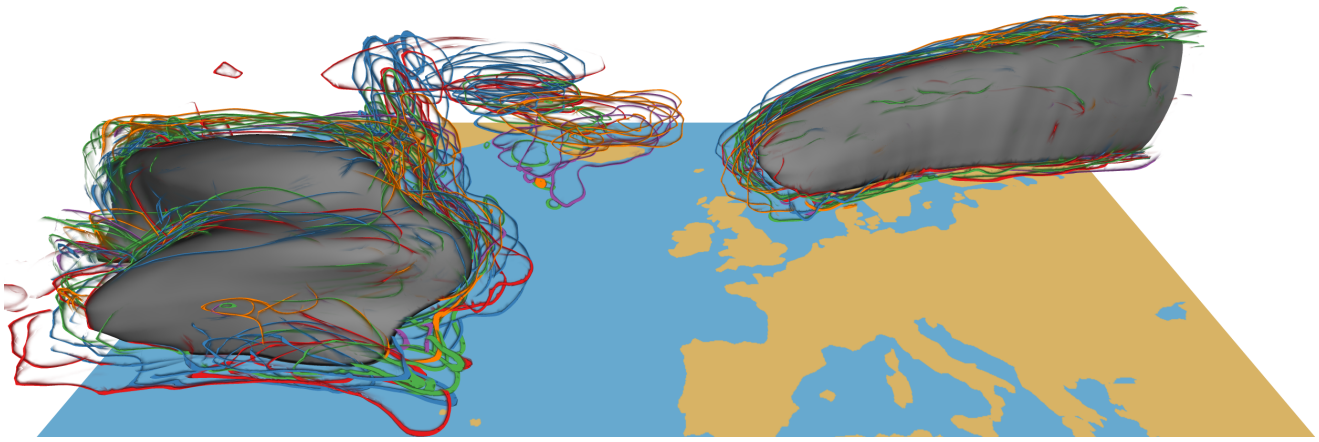

Figure 1: Visualization of a weather forecast ensemble from the ECMWF Prediction System [29]. This ensemble comprises wind velocity data and consists of 50 members. For one isovalue, all members are visualized as silhouettes of isosurfaces. Additionally, a mean ensemble member is rendered as a gray isosurface to enhance the visual perception of the spatial context. Color is used to cluster members by their similarity.

Abstract

Visualizing sets of isosurfaces from 3D scalar ensemble fields is a difficult task due to inherent occlusion effects, yet it is often required to analyze the uncertainty represented by such an ensemble. In this paper, we present a novel visualization technique for ensembles of isosurfaces based on screen-space silhouettes. By using silhouettes, the displayed information is reduced to avoid occlusions, yet the major shape of the surfaces can be maintained. Our approach preserves spatial coherence and does not make any assumption about the underlying surface distribution. By providing additional mechanisms, i.e., picking, clustering, cutting and animation, we enable the user to explore an ensemble of surfaces interactively.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1 Introduction

Scalar field ensembles play an important role in many areas of science and engineering. Ensembles are typically generated by $N \in \mathbb{N}$ repeated simulation runs, where different input models or parameters are used in each run. Each simulation generates a possible occurrence of the simulated phenomenon, the so-called ensemble members. For 3D scalar fields, this can be described formally as a mapping $\{1,...,N\} \times \mathbb{R}^3 \rightarrow \mathbb{R}$. Visualizing 3D scalar field ensembles is a challenging task, since a large amount of data at each spatial location has to be conveyed to the user, such that important information is neither lost nor occluded.

*e-mail: ismail.demir@mytum.de
†e-mail: johannes.kehrer@tum.de
‡e-mail: westermann@tum.de

Different solutions exist to approach this problem. For instance, volume rendering is an effective technique to visualize the information stored in 3D scalar fields, especially with the help of transfer functions. By interactively adjusting this function, the shape of relevant features can be revealed. In this context, isosurfaces are of particular interest, i.e., locations exhibiting a constant value [18]. However, volume rendering suffers from occlusion effects, even if only a single scalar field is rendered. Consequently, this effect is amplified when multiple ensemble members are superimposed, which makes an experts analysis quickly unfeasible. As an alternative, different members can be visualized by drawing them next to each other [15]. However, in this case, the spatial context between similar features in different members is lost. Furthermore, this method is only feasible for a relatively small number of members due to the screen space limitation. Another approach is to provide the user with statistical data derived from the original ensemble instead of the individual members [35, 37]. For instance, mean and standard deviation can be computed at each spatial location and then presented to the user. In such a scenario, however, differences between members are smoothed out and important information, such as outliers, is lost.

To remedy the problem of occlusion, techniques from information visualization such as parallel coordinates or histograms can be combined with volume rendering by utilizing brushing and linking [12, 16, 26, 45]. The idea behind these approaches is to show multiple linked views displaying different aspects of the original data. In this scenario, the user selects regions of interest in one view and is then provided with visual feedback in all linked views. Another approach is to cut the original volume into slices and to render each slice separately. Then, ensemble members can be visualized via drawing spaghetti plots, by overlaying isocontours of each member [38]. However, in this representation, the spatial relationships between different slices are lost.

In this work, we propose a novel rendering approach to interactively visualize 3D isosurface ensembles. Our method preserves

spatial coherence, while occlusion effects do not disturb the visual perception to a considerable degree. By providing interactive mechanisms, we enable the user to further explore the data on a more detailed level. In particular, the main contributions of our approach are:

- A novel efficient visualization technique for 3D isosurface ensembles based on rendering semi-transparent silhouettes. Our method is spatially coherent both within each member and between different members.
- An efficient implementation to convert 3D scalar field ensembles into polygonal meshes for different isolevels. In this process, all information necessary for rendering is precomputed and stored at the vertex level. Thus, the workload during rendering is minimized, meaning that even a large number of ensemble members can be displayed at interactive frame rates.
- By integrating movable cutting planes directly into the 3D view, we enable the user to gain more detailed insights at specific spatial locations without losing spatial coherence.
- Different hierarchical clustering algorithms based on isolevels enable the user to group and analyze members according to their similarity. Since we compute all clusters in a preprocess, the user can select and explore them interactively.
- Picking and brushing mechanisms as well as animations that enable the user to view selected members of interest as complete isosurfaces, thus enhancing the spatial recognition.

2 Related Work

Ensemble visualization belongs to the area of uncertainty visualization, which has been an important research topic in visualization for almost two decades [25, 34]. Many useful approaches have been published and a number of surveys exist [4, 17, 20, 26]. For example, diagram-based techniques have been proposed to augment the spatial context by visual cues, such as glyphs, charts or box plots [10, 24, 39, 42]. Clustering algorithms are used to break down larger data sets into groups of similar characteristics [2, 5, 7, 13, 23]. In our work, we use clustering to classify silhouettes wrt. their similarity. Recently, research has also focused on analyzing spatio-temporal ensemble data [19, 28, 30, 40].

Spaghetti plots are a powerful technique to visualize 2D scalar field ensembles by simultaneously rendering an isocontour per member [21, 22, 36, 38, 41]. We extend this method to 3D data sets and provide interactive techniques to aid the user in the visual analysis, such as animation and picking. Several methods have been proposed to visualize isosurfaces extracted from 3D ensemble data, e.g., by means of animation [6], volume rendering [11, 44], or confidence envelopes [35, 37, 48]. Other methods aim at rendering multiple isosurfaces into a single 3D view [1, 8]. However, only a few ensemble members can be visualized simultaneously due to cluttering and occlusion. Matković et al. [32] visualize ensemble data as families of data surfaces in combination with cutting planes showing intersections with the surfaces. We pursue a similar approach by using cutting planes to guide the user towards specific regions of interest.

A number of methods exist for reducing 3D shapes to feature lines such as silhouettes, suggestive contours, ridge and valley lines, or apparent ridges [9, 33, 46, 47]. We compute silhouettes per member based on curvature due to its simplicity and computational efficiency [27], although other methods could be used as well.

3 Preprocess

Let us consider a scalar field ensemble given at the vertices of a 3D Cartesian grid, $f : \{1,...,N\} \times \mathbb{R}^3 \to \mathbb{R}$. That is, at each grid point a scalar value in $\mathbb{R}$ is associated with each member. Moreover, a set of isovalues is specified, $I \subset \mathbb{R}$. Now, we generate polygonal meshes and clustering distributions for each isovalue:

Mesh-Generation. Given a scalar field, i.e., one ensemble member, on a Cartesian grid and an isovalue, we first make use of the marching cubes algorithm to extract the corresponding isosurface in the form of a polygonal mesh. The marching cubes algorithm, originally proposed by Lorensen et al. [31], however, produces a uniformly resolved mesh. Consequently, polygonal primitives are generated at a very fine resolution, thus wasting memory. To overcome this drawback, we apply a mesh decimation algorithm that takes the underlying geometry into account. Our implementation is based on the quadric mesh simplification method as proposed by Garland et al. [14]. As a result, we end up with an adaptively resolved triangle mesh. Next, for every vertex x, we compute the following attributes and use them in the rendering process:

- We compute the *vertex normal* by taking the normalized gradient, i.e., $n = -g/\|g\|$, where $g = \nabla f_i(x)$ denotes the gradient and i refers to the member index. Since the ensemble is given on a Cartesian grid, we make use of finite differences to compute the respective derivatives numerically.
- The *principal curvatures* κ_1, κ_2 as well as the first *principal curvature direction* (PCD) p are calculated according to the method described by Kindlmann et al. [27]. Let $P = I - n \cdot n^\mathsf{T}$, $H = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x)\right)$, the Hessian matrix, and $G = -PHP/\|g\|$. Next, we compute the trace T and the Frobenius norm F of G. Now, we obtain $\kappa_{1,2} = \frac{T \pm \sqrt{2F^2 - T^2}}{2}$. Finally, p is obtained as the eigenvector of G corresponding to the eigenvalue κ_1.
- We compute the *outlyingness* as the mean squared distance to all members, i.e., $o = \sum_{j=0}^{N} \left(f_j(x) - f_i(x)\right)^2 / N$.

To save memory, these values are stored in a packed format. Position, principal curvatures and the outlyingness are stored in 16-bit floats per component. Since the normal and PCD are of unit length, we can store these vectors as a 32-bit unsigned integer. Here, the x and $y-$component are stored in the first 16 bits and in the next 15 bits, respectively. The sign of the $z-$component is stored in the last bit. These values are later reconstructed in a shader program. In total, 20 bytes are consumed per vertex. As a result, we end up with N meshes for each isovalue, where $N \in \mathbb{N}$ denotes the number of ensemble members, meaning that $N \cdot |I|$ meshes are generated in total. See the results section for an analysis of the memory requirement.

Clustering. For clustering, we compute a similarity matrix of size $N \times N$, where at each entry (i, j) the difference between member i and j is stored. We compute the difference as $d(i,j) = \sum_x \left(\sqrt{|f_i(x) - \mu|} - \sqrt{|f_j(x) - \mu|}\right)^2 / m$, where μ and m denote the current isovalue and the number of grid points, respectively. By taking the square root, changes in values closer to the isovalue are considered of greater importance.

Clusters are then generated based on the similarity matrix by using an agglomerative hierarchical clustering method [23]. We begin with N clusters, where each member is assigned to one unique cluster. Then, we proceed by merging a pair of clusters of minimal distance. This process is repeated until one cluster remains. Now, for each step we store the respective cluster distribution. Here, distance is computed in the following way. Suppose, each cluster contains exactly one member. Then, we obtain the distance as the corresponding entry in the similarity matrix. Otherwise, we compute the distance by using a linkage criterion. For instance, the average linkage criterion, defined as $d(A,B) = \sum_{a \in A, b \in B} d(a,b) / (|A||B|)$, yields suitable results in our implementation [3]. Again, this process is carried out for each isovalue.

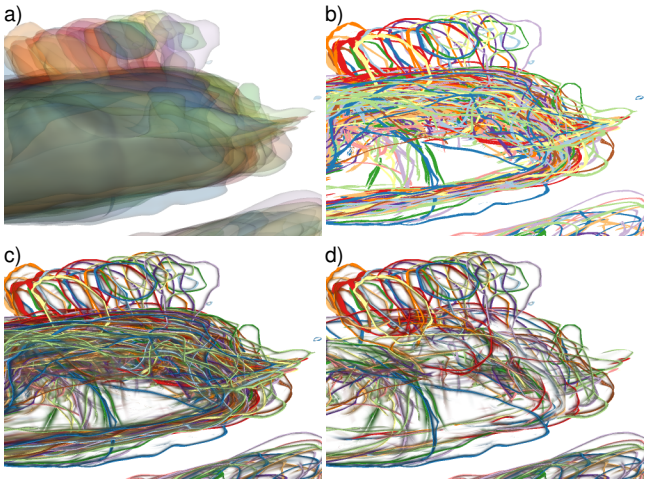

Figure 2: Visualization using different rendering techniques. a) Semi-transparent surfaces. b) 3D spaghetti plots of silhouettes. c) Shaded silhouettes. d) Shaded silhouettes with density-based removal.

4 Rendering Shaded Silhouettes

In this section, we explain the proposed rendering technique for isosurface ensembles based on the precomputed per-vertex attributes and clusters. For a user-selected isovalue and each selected member, the corresponding mesh is rendered. Note that rendering solid isosurfaces suffers from occlusion effects, in particular if multiple members are rendered simultaneously. Although drawing isosurfaces with transparency reduces occlusion effects, perceptually multiple layers of transparency cannot easily be distinguished, and quickly becomes infeasible when too many surfaces are overlaid. This can be observed in Fig. 2a. Therefore, we propose a different approach based on rendering only silhouettes instead of whole surfaces. Silhouette rendering is not a novel technique, yet to the best of our knowledge, it has not been used for ensemble visualization so far. Fig. 1 shows a weather forecast ensemble containing 50 members that is rendered using our method. In this view, the gray isosurface represents the mean of all members, which was generated by computing the average value at each grid point. Clearly, the members can be distinguished and the spatial context is preserved. By rotating the camera, a further improved recognition of the spatial structures is achieved.

Once the attributes are reconstructed in the vertex shader, rendering silhouettes is done in the fragment stage. By using the normal n, the curvature along view direction κ and the view vector v, we compute the silhouette coefficient as $\sigma = \left|v \cdot n^{\mathsf{T}}\right| / \sqrt{\tau\kappa(2-\tau\kappa)}$. Here, τ denotes the silhouette thickness, such that a silhouette is present iff $\sigma \leq 1$. Let us refer to [27] for further details on the derivation. To efficiently compute κ in the fragment shader, we make use of the precomputed principal curvatures $\kappa_{1,2}$ and the PCD p as introduced in the previous section. First, we obtain the angle φ between the PCD and the view direction, projected onto the plane aligned with the normal at the fragment position, as $\cos\varphi = \langle v - \langle n, v\rangle n, p\rangle$. Now, we calculate the curvature in view direction as $\kappa = \kappa_1 \cdot \cos^2\varphi + \kappa_2 \cdot \sin^2\varphi$. This equation holds, because PCDs are always aligned orthogonal to each other. Note that it is not necessary to actually compute φ, since $\cos^2\varphi + \sin^2\varphi = 1$. By drawing fragments with $\sigma \leq 1$ as opaque using a solid color, and discarding all other fragments, we obtain 3D spaghetti plots as can be seen in Fig. 2b. However, this approach suffers from the drawback that distinguishing different members of the same color is infeasible without rotating the camera. To overcome this limitation, we make use of shading and transparency effects. Given an RGB-color value c and silhouette coefficient σ, we return the RGBA-color value $(c,1)$ if $\sigma < \frac{1}{2}$, $\left(c \cdot \frac{1}{2}, 1 - \left(\sigma - \frac{1}{2}\right)/\frac{1}{2}\right)$ if $\frac{1}{2} \leq \sigma \leq 1$, and discard the fragment otherwise. The result is depicted in Fig. 2c.

To reduce the amount of visual clutter, consider the observation that there are some regions where the density of silhouettes is greater. Hence, we can eliminate a number of silhouettes in such regions without losing important information, since all these silhouettes share a similar shape. In doing so, unnecessary data is removed from the visualization, thus making it easier to grasp for the user. To identify silhouettes that are suitable for removal, we resort to the outlyingness o as explained in the previous section. This value indicates how much difference there is between the value of the current member and all other members at a given position wrt. the selected isovalue. Thus, lower values imply a greater density of isosurfaces and therefore a greater density of silhouettes. This gives rise to the following algorithm, implemented in the fragment shader. Let i_M, s_G denote the member index and group size. Here, the group size determines the number of nuances for silhouette removal. Then, we obtain the group index by $i_G = i_M \pmod{s_G}$. Let $o^*_{Min}, o^*_{Max} \in \mathbb{R}$ denote the global minimum and maximum outlyingness threshold. Now, we compute the thresholds for group index i_G as $o_{Min}(i_G) = o^*_{Min} + i_G \cdot \left(o^*_{Max} - o^*_{Min}\right)/s_G, o_{Max}(i_G) = o^*_{Min} + (i_G+1) \cdot \left(o^*_{Max} - o^*_{Min}\right)/s_G$. Finally, we compute the outlyingness-based opacity as $\alpha(i_G, o) = \text{clamp}\left(\frac{o - o_{Min}(i_G)}{o_{Max}(i_G) - o_{Min}(i_G)}, 0, 1\right)$. If the result equals 0, the fragment is discarded. Otherwise, the alpha-component of the fragment shader output is multiplied by $\alpha(i_G, o)$. To identify regions where most members agree, the user can start out with a lower degree of silhouette removal and then gradually increase this threshold to focus on finer structures. This can be done by dynamically adjusting o^*_{Min} and o^*_{Max}. The result is shown in Fig. 2d. As an alternative to altering the opacity, we could also resort to color-coding the silhouette density or adjusting the thickness, although this is not shown here.

5 Visualizing Details on Demand

According to the information seeking mantra "Overview first, zoom and filter, then details-on-demand" proposed by Shneiderman [43], we have implemented several mechanisms to visualize user-selected details.

Cutting Planes. We have integrated movable cutting planes into our implementation to enable the user to gain more detailed insights at specific locations of interest. This process is illustrated in Fig. 3b,d. Rendering cutting planes is accomplished in the fragment stage. First, we compute the distance d from the fragment's position to the cutting plane along the view direction v. To prevent occlusion, fragments lying in front of the plane, i.e., $d < 0$, are discarded. To render intersection curves of thickness ε, we consider a fragment as lying on the plane iff the following conditions are met: a) The distance along the view direction, projected onto the surface normal, is within the plane thickness, i.e., $\langle v, n\rangle d \leq \varepsilon$; b) the distance along the view direction is within the plane thickness, along the view direction, projected onto the plane normal n_P, i.e., $d \leq \langle v, n_P\rangle \varepsilon$. In this case, the fragment is rendered by using the term $\langle v, n\rangle d$ to compute shading and opacity as explained for the silhouette coefficient.

Clustering. We utilize hierarchical clustering algorithms to group members wrt. their similarity. Due to the preprocess, the clusters can be selected interactively. E.g., in Fig. 1 members are clustered into five groups. By showing or hiding clusters separately, the user can focus on cluster-specific features. In this way, outliers can quickly be identified and trends can be analyzed in more detail.

Picking and Brushing. We have developed picking and brushing functionality by checking the mouse coordinates against the screen space coordinates in the fragment stage. Thereby, we provide means to view selected members as isosurfaces to enhance to

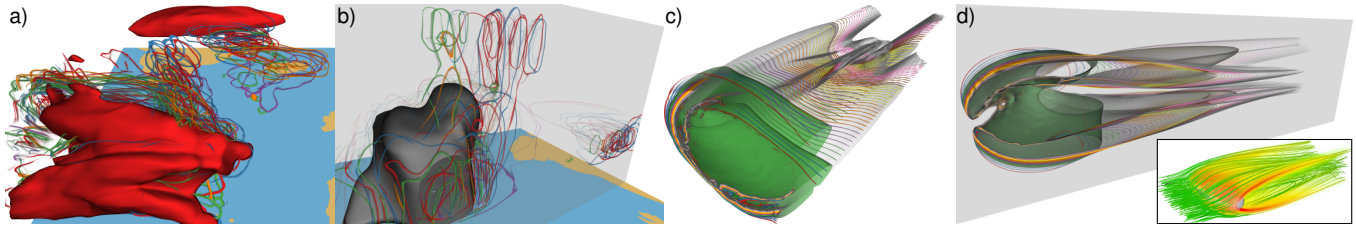

Figure 3: Results from two ensemble data sets. a) An outlier was selected in the ECMWF data set (bottom left) to get a better understanding of its shape. b) A cutting plane allows us to analyze a feature in more detail. c) Overview of a Navier-Stokes flow simulation comprising 56 runs. d) A cutting plane reveals the inner structures. The underlying flow is depicted at the bottom right.

visual recognition. This also serves as a point of reference to facilitate the spatial perception of the silhouettes. In Fig. 1, the mean surface is visualized in gray. Note that this method is only effective for displaying one or a few members as otherwise occlusion effects prevail.

Animation. To quickly discover differences and similarities between a selected subset of members of interest, we provide the user with the ability to animate over members. He first selects a subset of members by picking and brushing. Then, during the animation, one member after another is visualized by a solid isosurface. To make the transitions easier to understand, members are sorted by cluster index and similarity.

6 Results

We now discuss some results to demonstrate the potential of our visualization technique. To begin with, let us consider an ensemble comprising wind velocities of a weather forecast simulation by the ECMWF Ensemble Prediction System [29]. This ensemble consists of 50 members of resolution $128 \times 256 \times 16$. In Fig. 1, for an isovalue of $30\frac{m}{s}$ the resulting isosurfaces are visualized as silhouettes. An additional mean member is included, as explained before. To store the data for visualizing all 51 members (including the mean), roughly 16 MiB are consumed per isovalue. On a standard desktop PC, the process of precomputation takes ≈ 3 minutes for generating all meshes and another ≈ 2 minutes for computing the similarity matrix. The time for computing the clusterization based on the similarity matrix is negligible. Note that the computational cost is proportional to the grid resolution and the number of members. Rendering the image in Fig. 1 took $3.6ms$ measured on an NVIDIA GeForce GTX 580 graphics card at a viewport resolution of 1900×1200. This means that our implementation enables the user to interactively analyze an ensemble of isosurfaces.

By observing the visualization in Fig. 1, we can spot two regions, where most, if not all, simulation runs agree on a wind velocity of $30\frac{m}{s}$. One such region is located above the Baltic Sea and the other region above the North Atlantic Ocean south of Greenland. Note that we have also determined that the wind velocities within these regions are greater than $30\frac{m}{s}$ by studying other isovalues, although this is not shown here. Moreover, there is another region located in the middle where only some members exhibit the aforementioned wind velocity. By clustering the members into five groups and using color to indicate the cluster membership, we can discover the main trends arising at that region. Here, we can also identify an outlier and investigate the shape of that member's isosurface by picking the respective silhouette. This is shown in Fig. 3a. Due to the preprocess, the user can change the number of clusters interactively.

Let us now focus again on the second region. Here, we find a structure of curls on the top, where only some members agree as this area is clearly separated from the mean isosurface. If we place a cutting plane as shown in Fig. 3b, we can analyze this feature in more detail and discover the inner structures. By moving the plane, the spatial perception is enhanced, and ellipsoid zones of higher wind velocity are revealed, indicating isolated air turbulences predicted by some simulation runs.

As a second example, let us consider an ensemble featuring an incompressible fluid flow evolving around an ellipsoid obstacle. 56 Navier-Stokes simulation runs were performed with different viscosities on a Cartesian grid of resolution $145 \times 49 \times 49$. After a given simulation time, the vorticity magnitude was saved as the scalar field ensemble. The result is presented in Fig. 3c using our visualization technique. Clearly, it is easily possible to visually distinguish members as only minimal occlusion effects occur. By inserting a cutting plane, we can also gain insight into the inner regions and study the behavior of the flow simulation in more detail as shown in Fig. 3d. In this data set, the use of animation proved to be helpful, since the shape changes gradually along the sequence of all members. We demonstrate this in the accompanying video clip.

7 Conclusion

In this work, we have presented a novel visualization technique for 3D scalar field ensembles. Our approach is based on the idea of rendering silhouettes instead of solid isosurfaces. Hence, occlusion artifacts are minimized without losing spatial coherence. We have also implemented several ways to provide a more detailed analysis of the ensemble. That is, by enabling the user to place cutting planes in the 3D view and by providing means of clustering and animation, the ensemble can be investigated according to different criteria. To study individual members in the context of the whole ensemble, the user can resort to picking and brushing functionality. Moreover, by precomputing all computationally involved parts in a preprocess, we have accomplished rendering at interactive rates. We have demonstrated the effectiveness of our approach by visualizing and analyzing two ensembles of very different characteristics. In both cases, our method was able to reveal relevant features and supported the user to gain insight into the spatial structures.

In the future, we plan to extend our implementation to time-varying ensembles. In this regard, the concept of animation might prove as an effective tool. We also aim at integrating mechanism to automatically detect relevant features and place cutting planes or other visual clues accordingly. By utilizing the GPU, the preprocess could be significantly sped up, although we do not consider this as a critical issue, as it has no effect on the rendering performance. We further plan to investigate our results in a user study in collaboration with domain experts to evaluate the practical benefits of our method.

Acknowledgments

Access to ECMWF prediction data has been kindly provided in the context of the ECMWF special project "Support Tool for HALO Missions". We are grateful to the special project members Marc Rautenhaus and Andreas Dörnbrack for providing the ECMWF ENS dataset of 17 October 2012. This work was supported by the European Union under the ERC Advanced Grant 291372 SaferVis - Uncertainty Visualization for Reliable Data Discovery.

REFERENCES

[1] O. S. Alabi et al. Comparative visualization of ensembles using ensemble surface slicing. In *Proc. SPIE Vis. and Data Analysis*, 2012.

[2] M. Beham, W. Herzner, M. E. Gröller, and J. Kehrer. Cupid: Cluster-based exploration of geometry generators with parallel coordinates and radial trees. *IEEE TVCG*, 20(12):1693–1702, 2014.

[3] E. J. Bijnen. *Cluster analysis – Survey and evaluation of techniques*. Springer Netherlands, Dordrecht, 1973.

[4] G.-P. Bonneau, H.-C. Hege, C. R. Johnson, M. M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. Overview and state-of-the-art of uncertainty visualization. In *Scientific Visualization*, pages 3–27. Springer, 2014.

[5] U. D. Bordoloi, D. L. Kao, and H.-W. Shen. Visualization techniques for spatial probability density function data. *Data Science Journal*, 3:153–162, 2004.

[6] R. Brown. Animated visual vibrations as an uncertainty visualisation technique. In *Proc. GRAPHITE*, pages 84–89, 2004.

[7] S. Bruckner and T. Möller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE TVCG*, 16(6):1468–1476, 2010.

[8] S. Busking, C. Botha, L. Ferrarini, J. Milles, and F. Post. Image-based rendering of intersecting surfaces for dynamic comparative visualization. *The Visual Computer*, 27:347–363, 2011.

[9] F. Cole, K. Sanik, D. Decarlo, A. Finkelstein, T. Funkhouser, S. Rusinkiewicz, and M. Singh. How well do line drawings depict shape. *ACM Trans. on Graph. (Proc. of SIGGRAPH)*, 2009.

[10] I. Demir, C. Dick, and R. Westermann. Multi-charts for comparative 3D ensemble visualization. *IEEE TVCG*, 20(12):2694–2703, 2014.

[11] S. Djurcilov, K. Kim, P. Lermusiaux, and A. Pang. Visualizing scalar volumetric data with uncertainty. *Computers & Graphics*, 26(2):239–248, 2002.

[12] H. Doleisch and H. Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. *J. WSCG*, 10(1–3):147–154, 2002.

[13] F. Ferstl, K. Bürger, and R. Westermann. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE TVCG*, 22(1):767–776, Jan 2016.

[14] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. SIGGRAPH '97*, pages 209–216, 1997.

[15] M. Gleicher, D. Albers, R. Walker, I. Jusufi, C. D. Hansen, and J. C. Roberts. Visual comparison for information visualization. *Information Visualization*, 10(4):289–309, 2011.

[16] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A system for visually linking 3-D and statistical visualizations, applied to cardiac simulation and measurement data. In *Proc. IEEE Visualization*, pages 489–492, 2000.

[17] H. Griethe and H. Schumann. The visualization of uncertain data: Methods and problems. In *Proc. SimVis*, pages 143–156, 2006.

[18] C. Hansen. *The visualization handbook*. Elsevier Butterworth-Heinemann, Burlington, MA, 2005.

[19] L. Hao, C. Healey, and S. Bass. Effective visualization of temporal ensembles. *IEEE TVCG*, 22(1):787–796, Jan 2016.

[20] C. Heinzl, S. Bruckner, M. E. Gröller, A. Pang, H.-C. Hege, K. Potter, R. Westermann, T. Pfaffelmoser, and T. Möller. Uncertainty and parameter space analysis in visualization. IEEE VisWeek Tutorial, 2012.

[21] T. Höllt, A. Magdy, G. Chen, G. Gopalakrishnan, I. Hoteit, C. Hansen, and M. Hadwiger. Visual analysis of uncertainties in ocean forecasts for planning and operation of off-shore structures. In *Proc. IEEE Pacific Visualization Symposium*, pages 185–192, 2013.

[22] T. Höllt, A. Magdy, P. Zhan, G. Chen, G. Gopalakrishnan, I. Hoteit, C. D. Hansen, and M. Hadwiger. Ovis: A framework for visual analysis of ocean forecast ensembles. *IEEE TVCG*, 20(8):1114–1126, 2014.

[23] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666, 2010.

[24] M. Jarema, I. Demir, J. Kehrer, and R. Westermann. Comparative visual analysis of vector field ensembles. In *Proc. IEEE VAST*, pages 81–88, 2015.

[25] C. R. Johnson and A. R. Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Comput. Graph. Appl.*, 23(5):6–10, 2003.

[26] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE TVCG*, 19(3):495–513, 2013.

[27] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proc. IEEE Visualization*, pages 513–520, 2003.

[28] P. Kothur, M. Sips, H. Dobslaw, and D. Dransch. Visual analytics for comparison of ocean model output with reference data: Detecting and analyzing geophysical processes using clustering ensembles. *IEEE TVCG*, 20(12):1893–1902, 2014.

[29] M. Leutbecher and T. Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227(7):3515–3539, 2008.

[30] L. Liu, M. Mirzangar, R. M. Kirby, R. Whitaker, and D. H. House. Visualizing time-specific hurricane predictions, with uncertainty, from storm path ensembles. *Comput. Graph. Forum*, 34(3):371–380, 2015.

[31] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, Aug. 1987.

[32] K. Matković, D. Gračanin, B. Klarin, and H. Hauser. Interactive visual analysis of complex scientific data as families of data surfaces. *IEEE TVCG*, 15(6):1351–1358, 2009.

[33] M. Mirzargar, R. T. Whitaker, and R. M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE TVCG*, 20(12):2654–2663, 2014.

[34] A. T. Pang, C. M. Wittenbrink, and S. K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[35] T. Pfaffelmoser, M. Reitinger, and R. Westermann. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. *Comput. Graph. Forum*, 30(3):951–960, 2011.

[36] T. Pfaffelmoser and R. Westermann. Visualizing contour distributions in 2D ensemble data. In *EuroVis-Short Papers*, pages 55–59, 2013.

[37] K. Pothkow and H.-C. Hege. Positional uncertainty of isocontours: Condition analysis and probabilistic measures. *IEEE TVCG*, 17(10):1393–1406, 2011.

[38] K. Potter et al. Ensemble-Vis: A framework for the statistical visualization of ensemble data. In *Proc. IEEE Int'l. Conf. Data Mining Workshops*, pages 233–240, 2009.

[39] K. Potter, J. Kniss, R. Riesenfeld, and C. Johnson. Visualizing summary statistics and uncertainty. *Comput. Graph. Forum*, 29(3):823–832, 2010.

[40] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, V. Pascucci, and C. Johnson. A flexible approach for the statistical visualization of ensemble data. In *Proc. IEEE ICDM Workshop Knowledge Discovery from Climate Data*, 2009.

[41] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. J. Moorhead. Noodles: A tool for visualization of numerical weather model ensemble uncertainty. *IEEE TVCG*, 16(6):1421–1430, 2010.

[42] T. Schultz, L. Schlaffke, B. Schölkopf, and T. Schmidt-Wilcke. Hifive: A hilbert space embedding of fiber variability estimates for uncertainty modeling and visualization. *Comput. Graph. Forum*, 32(3):121–130, 2013.

[43] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symposium on Visual Languages*, pages 336–348, 1996.

[44] D. C. Thompson, J. A. Levine, J. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. P. Pébay. Analysis of large-scale scalar data using hixels. In *Proc. IEEE Symposium on Large Data Analysis and Visualization*, pages 23–30, 2011.

[45] M. Q. Wang Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for using multiple views in information visualization. In *Proc. Working Conference on Advanced Visual Interfaces*, pages 110–119, 2000.

[46] R. T. Whitaker, M. Mirzargar, and R. M. Kirby. Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE TVCG*, 19(12):2713–2722, 2013.

[47] K. Wu and S. Zhang. A contour tree based visualization for exploring data with uncertainty. *Int'l. J. Uncertainty Quantification*, 3(3):203–223, 2013.

[48] B. Zehner, N. Watanabe, and O. Kolditz. Visualization of gridded scalar data with uncertainty in geosciences. *Computers & Geosciences*, 36(10):1268–1275, 2010.