Gaussian Mixture Model Based Volume Visualization

Shusen Liu *, Joshua A. Levine †, Peer-Timo Bremer ‡, Valerio Pascucci §

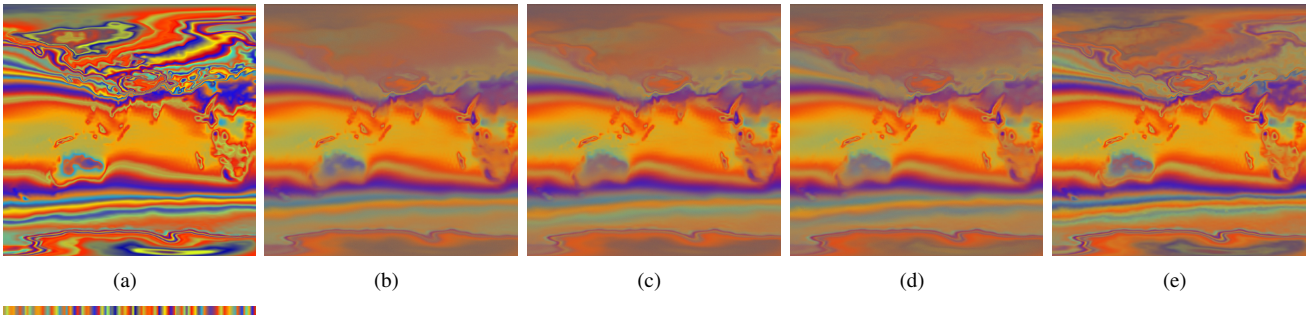

Figure 1: An ensemble of air temperatures created by combining the daily temperatures for the month of January (31 days) over a 29 year period on a three dimensional grid of dimension $256 \times 128 \times 31$. (a) The mean data and transfer function. (b)-(e) Screen space accumulated integration of Gaussian mixture model representations of the data, using one, two, four, and six components, respectively.

ABSTRACT

Representing uncertainty when creating visualizations is becoming more indispensable to understand and analyze scientific data. Uncertainty may come from different sources, such as, ensembles of experiments or unavoidable information loss when performing data reduction. One natural model to represent uncertainty is to assume that each position in space instead of a single value may take on a distribution of values. In this paper we present a new volume rendering method using per voxel Gaussian mixture models (GMMs) as the input data representation. GMMs are an elegant and compact way to drastically reduce the amount of data stored while still enabling realtime data access and rendering on the GPU. Our renderer offers efficient sampling of the data distribution, generating renderings of the data that flicker at each frame to indicate high variance. We can accumulate samples as well to generate still frames of the data, which preserve additional details in the data as compared to either traditional scalar indicators (such as a mean or a single nearest neighbor down sample) or to fitting the data with only a single Gaussian per voxel. We demonstrate the effectiveness of our method using ensembles of climate simulations and MRI scans as well as the down sampling of large scalar fields as examples.

Keywords: Uncertainty Visualization, Volume Rendering, Gaussian Mixture Model, Ensemble Visualization

Index Terms: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

1 INTRODUCTION

With the ever increasing amount of available computing resources scientific simulations are able to represent highly complex phenomena at unprecedented scale and resolution. However, each such simulation only represents one possible outcome of an experiment which typically depends on numerous assumptions about the initial conditions, the physical properties of the species involved, or the accuracy of the underlying model. In particular, in chaotic and highly non-linear systems, such as climate or turbulent combustion simulations, it is becoming increasingly important to instead understand the range of possible outcomes. Currently, the most common approach to representing the uncertainty in the results is to perform an ensemble of simulations each with slightly different assumptions or parameters. However, analyzing and/or visualizing these massive datasets is challenging and the lack of suitable tools increasingly impacts our ability to gain new insights from such ensembles. One of the main problems is the data representation: working with a collection of simulation results one-by-one requires the infeasible processing of hundreds of ensemble members.

One potential solution is to view an ensemble as samples from a distribution of possible outcomes or carry out only one stochastic simulation. In both cases the result is a single set of simulation data in which each point is represented as a (sampled) distribution of values. This perspective is attractive as it can also incorporate, for example, data compression through down sampling or experimental errors. However, new techniques are needed to visualize and analyze such data ideally using paradigms familiar to users and commodity hardware.

In this paper, we consider volume rendering of stochastic fields, created from ensembles or by down sampling, using per-voxel Gaussian mixture models (GMMs). Given a sampled distribution of values at each voxel we fit a GMM with varying number of Gaussian kernels as basis functions. GMMs are attractive as they allow fitting complex data with relatively few parameters. In the simplest case, fitting a single Gaussian at each position encodes the mean and variance of the data and thus requires only two values per voxel. More general, fitting k Gaussians requires $3k$ values (the additional third value is necessary to encode the weight of the Gaussian) compared to a typical ensemble which may contain hundreds of members. In particular, fitting GMMs makes the size of the representation dependent on the complexity of the data rather than the size of the ensemble. In practice, we find that typically fitting around four Gaussian kernels is sufficient and the resulting 12 values per-voxel (encoded as 3 four channel textures) can still be interactively processed on common desktop computers. From the data fitting point of view there are several other alternatives such as polynomial-based model and Fourier-based model that could be considered. However since all of our rendering methods are base on sampling arbitrary distribution, a Gaussian representation is key for efficient random sampling implementation on GPU. In the method section we will go into details show why GMM is used in our algo-

*e-mail: shusenl@sci.utah.edu
†e-mail: jlevine@sci.utah.edu
‡e-mail: bremer5@llnl.gov
§e-mail: pascucci@sci.utah.edu

IEEE Symposium on Large Data Analysis and Visualization 2012
October 14 - 15, Seattle, Washington, USA
978-1-4673-4733-4/12/$31.00 ©2012 IEEE

rithm and an discussion on the fitting precision is also presented.

We extend basic notions of the volume rendering to efficiently handle this statistical representation. In particular, we replace a typical pre-integration table with one indexed by mean and variance of each Gaussian component for a fast color lookup. We have correspondingly restructured integration of the transfer function. Our method allows for sampling the probability density function described by the GMM on a per frame basis during volume rendering. This allows for augmenting the display with direct animations of the probability—where the data is less certain it flickers and appears "fuzzy" to the user, similar to the probabilistic animations of Lundström et al. [8].

We experiment with three types of data: First, we compare large scale data down sampled using both our technique and more traditional single-sample per voxel techniques; Second, we experiment with ensembles of medical image data, in particular highlighting how automatic registration processes are used in a study of over 300 patients; and third, we investigate massive collections of climate simulations, highlighting the distinctions between various climate studies run by scientists through the CMIP5 project hosted by the PCMDI database. In each case, we show how using GMMs to model the data allows for informative visualizations which highlight aggregate statistical behavior while allowing for interactive frame rates on modern GPUs.

2 Related Work

Uncertainty exists in most scientific data, and consequently it plays an increasingly important role in visualization. Especially in the past decade, research has been directed towards augmenting visualizations with techniques that directly communicate uncertainty [6]. Incorporating uncertainty into visualization is particularly important in domain specific applications, such as seismology [2], geographical information systems [9], or climate science [13], where ensembles of data are naturally produced. More generalized approaches to visualizing uncertainty have focused on the human perception element. For example, the use of animation [5, 8] and glyphs [11, 12] have been studied and shown to be effective ways to convey uncertainty when provided appropriate contexts. In this work, we build upon the use of animation to convey uncertainty. A state of art report on uncertainty visualization can be found here [10].

Given its versatility, volume visualization is one of the major research focus for uncertainty visualization. Djurcilov et al. [4] present two of the earliest known methods for volume rendering with uncertain data. In the first they focus on "inline" techniques (where the uncertainty is pushed into the rendering process) for direct volume rendering, where for example, the user is allowed to interact with a 2D transfer function where one axis is uncertainty and the second is scalar value. Their second applies post processing to either dither or texture the volume rendering to highlight uncertainty. In both of these cases, it is assumed that there is a well defined scalar field to encode the uncertainty channel, a luxury we do not explicitly construct here.

As discussed above, the choice of data representation is of key importance, in particular, to our goal of providing a generalized framework for volume visualization. Thompson's hixel representation [17] uses per-voxel histograms to encode a distribution of values for each location in volumetric data. They augment these with contingency information of neighboring voxels. While such sampled distributions are certainly flexible enough to represent arbitrary distributions, they lack the necessary compactness needed to work in the memory constraints of the GPU. Instead, the Gaussian distribution is often a popular choice for representing probability distributions because of its smooth bell shape and ease of deriving analytical and explicit form. Gaussian functions have shown some effectiveness in solving volume rendering integration for high dimensional data [7] and a way to generate better down sampling [18], however are limited in their ability to model complex distributions. Using a weighted sum of Gaussians, or Gaussian mixture models (GMM) are preferred in clustering methods common in pattern recognition [15]. In visualization literature, GMMs have been introduced to model uncertainty in visual analytics by Correa et al. [3]. Using GMMs to model bidirectional reflection distribution functions is another popular use [16], and this work shares a key challenge of its own in understanding how to interpolate GMMs.

3 Methods

Gaussian Mixture Models (GMMs) [1] are one of the most widely used and versatile probabilistic models. Each component of the mixture model is a single Gaussian distribution that is combined through a weighted linear combination into the overall distribution. The unique numerical and statistical property of Gaussians and the linearity of the combination make it an ideal method for converting an arbitrary distribution into a compact yet flexible format. Compactly, at any position x we define a K component GMM $p(x)$ as:

$$p(x) = \sum_{i=1}^{K} \pi_i \mathcal{N}(x|\mu_i, \sigma_i^2)$$

where each component i, μ_i and σ_i^2 are the mean and variance and π_i is the corresponding weight such that $\sum \pi_i = 1$.

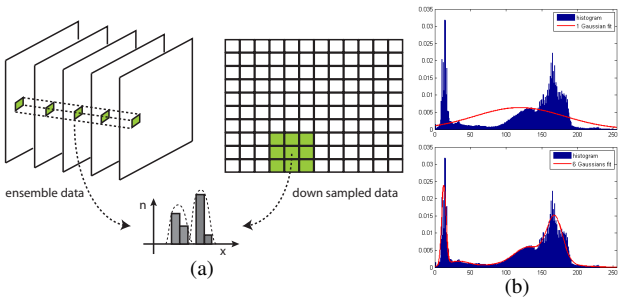

Figure 2: (a) We build GMMs by fitting distributions at the same position across ensembles of images or blocks of data. (b) GMMs (red curves) computed by the EM algorithm. Using only a single component (top) provides a poor fit, but increasing to six components (bottom) results in a more accurate fit of the input distribution.

Computing a GMM from a distribution of values is typically accomplished using the Expectation Maximization (EM) procedure. It uses an iterative method to maximize the likelihood function [1] by estimating the Gaussian parameters. In our work, we have applied the EM algorithm to per-voxel distribution which model the behavior of a single voxel across an ensemble of images or a block of neighboring voxels in an image to be down sampled (see Fig.2).

3.1 Modeling Distributions with GMMs

In addition to its versatility, one of the main advantages of the mixture model is flexibility in controlling the memory footprint of the representation. Compared to a histogram, which needs to store as many values as there are ranges of data, the GMM stores exactly three values per component. Adjusting the number of components (Fig. 2(b)) improves the fit. Since we are targeting the GPU, memory constraints make the GMM a more desirable option than per-voxel histograms. We calculate an error metric in order to guide us choose suitable number of Gaussian. In our design an arbitrary number of Gaussian could be used for better accuracy as long as there is enough video memory, however for most data we tested four Gaussians were sufficient.

One may wonder why we did not choose methods like Fourier decomposition or polynomials approximation to represent the distribution which also have a small storage footprint. The major rea-

son is the ease of generating random numbers. By exploiting the properties of Gaussian distributions, one can easily generate a random variable that belongs to an arbitrary distribution represented by the GMM. Generating such a random sample requires only three steps: (1) generate a uniform random number; (2) select a Gaussian component based on this value; and (3) generate a random number in the normal distribution of the selected Gaussian. Since the CUDA cuRAND library allows for generating both random numbers used in (1) and (3) efficiently, this can be done in a massively parallel fashion. Unfortunately, one can only generate the random numbers between rendering kernels, i.e. between frames. We overcome this limitation by adding a random 3D indexing buffer. With this buffer we have a "random" number at every vertex and though interpolation at any position in the volume. This buffer is updated on a per frame basis exploiting the cuRAND massively parallel random number generator. However, spatially close positions have similar numbers (from interpolation), so we use them to index into a 1D uniform random buffer and 1D Gaussian random buffer for steps (1) and (3), which are updated each frame.

3.2 Rendering with GMMs

Fuzzy Rendering Given efficient GPU-based random sampling on GPUs, a brute force Monte Carlo sampling approach becomes a viable option for volume rendering distributions. Inspired by the animated renderings of Lundström [8], we modified a standard GPU raycaster to include a notion of per-frame sampling. Instead of storing a single scalar volume, we pack each Gaussian component into three channels of a texture. This means we can effectively store a four kernel GMM in three RGBA textures, or up to eight kernels within six textures. For the standard raycasting algorithm a scalar value is required to be passed into the transfer function lookup table, where its color will be accumulated in the ray integral. At each frame, we sample a new scalar value from the distribution and perform a lookup. The result of this random sampling gives us a probabilistic rendering of the distribution, where the color flickers based on variability. A narrow distribution will result in a small range of possible values producing less "flicker," while a wider distribution produces more substantial color variations. These temporarily changing cues provide an intuitive notion of the amount of uncertainty in the distribution.

Monte Carlo Integration in Screen Space A disadvantage of the fuzzy rendering approach is that it requires animation to convey information on the distribution, and thus does not allow for a high quality static image. To address this problem we apply a screen space integration scheme to complement the fuzzy rendering. Since the random sampling is based on a specific distribution, it is possible to accumulate the randomly generated values, resulting in a Monte Carlo integration. However, since the rendering domain is volumetric this accumulation of random samples is expensive and will suffer severely from transfer function pre-classification artifacts. Therefore, instead of accumulating samples in object space we accumulate samples in image space. This enables our random sample based algorithm to produce smoother images that provide a more meaningful description of the data than, for example, simply using the distribution's mean.

4 Results

We demonstrate the effectiveness of our techniques on datasets which are both large ensembles of volumetric images as well as comparing the effects of down sampling higher resolution data to smaller sized volumes. Each experiment runs on a desktop machine with an Intel Core i7-920, 16GB of memory, and a nVidia GTX580 graphics card with 1.5GB of memory. Our implementation uses CUDA.

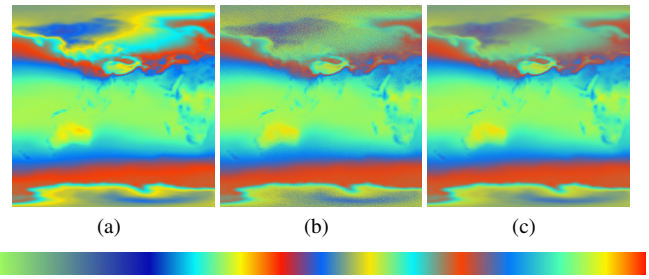

Figure 3: Climate data rendering mode comparison, showing the air temperature field for the 4 component GMM with (a) mean, (b) sampled, and (c) scale space integrated rendering techniques.

4.1 Ensemble Data

Climate Data We experiment with ensembles of data from two sources. Our first source of data is an ensemble from climate science. We use data from the CMIP5 project hosted by the PCMDI database. We construct GMM models of daily averages for air temperature during the month of January over a 29 year period from 1979-2008 for the CNRM-CM5 model. Each dataset of size $256 \times 128 \times 31$ (3.88 MB) is aggregated on the per-voxel values. This produced a dataset with 899 (29×31) images, for a total of 3.40GB of data. Fig. 1 compares taking the mean of the data to constructing GMM models with various numbers of components (one to six). As the number of GMMs increases, we are able to see additional bands of temperature where there was higher variability. Using fewer components, this area is blurred further. Interestingly, comparing Fig. 1(a) to Fig. 1(b) shows that the mean alone fails to capture the regions where there is high uncertainty near the poles of the earth. Fig. 3 compares the mean, sampled, and scale space integrated versions of this data using a four component GMM. Even in a still frame, the speckling in the sampled rendering indicates the uncertainty in the data around the poles, producing an effect similar to Djurcilov et al. [4]. However instead of modulating the data with a post-process filter, we directly sample on the uncertainty.

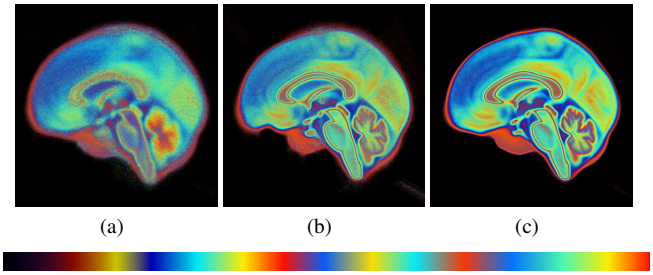

Figure 4: MRI registration ensemble comparison. We can see the (a) unregistered images produce the most amount of fuzziness on account of the lack of alignment. Both the (b) relaxation and (c) greedy optimization of the LDDMM produce less variable results in image space, with the greedy results appearing the least fuzzy.

MRI Atlas As a second ensemble, we use a collection of registered brain images. Initially, the 315 images were produced for a study of mild cognitive impairment, selected from the Alzheimers Disease Neuroimaging Initiative (ADNI) database. To study these patients in aggregate, an atlas of registered images has been produced [14]. Initially, each volume was an MRI scan of size $144 \times 192 \times 160$ (16.88MB), and the 315 images together produce a total dataset of 5.19GB. Registration was performed by optimizing the large deformation diffeomorphic mapping metric (LDDMM) using two different approaches, a relaxation procedure as well as a greedy optimization. One of the goals of our approach was to help illuminate the efficacy and potentially differences between the

different registration techniques. Fig. 4 shows three renderings of the unregistered data as well as registration using both techniques, built using a four component GMM model. We see that the greedy approach produces less variability.

4.2 Down Sampling Large-Scale Data

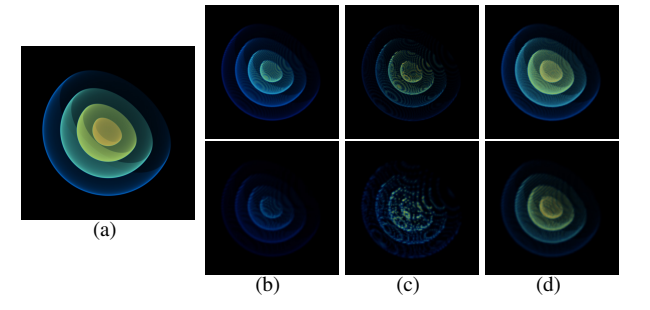

Figure 5: Down sampling the nested spheres. Compare (a) original data with down samplings at 128^3 (top row) and 64^3 (bottom row) of the (b) mean, (c) single sample, and (d) 4 component GMM. At 64^3, neither the mean nor single sample preserve the spherical layers.

Artificial Dataset: Nested Spheres We next compare how GMMs augment down sampled large-scale data. First, we construct an artificial dataset with nested spheres. The goal is to simulate the data with thin layered structures, which are traditionally difficult to down sample correctly. The original resolution of the dataset is 512^3, which has been down sampled to resolutions of 128^3 and 64^3. Here we use three different down sample schemes taking either the mean, a single sample, or four component GMM for the distribution of data stored in each block. Fig. 5 shows a comparison of the effects of down sampling. We can see that in this example, the mean produces data values with a lower range (from averaging the collection of zeros adjacent to each spherical layer) while the single sample produces a sparse surface with a collection of holes. In essence, the mean of the data manages to preserve the structure, but not the values, while the single sample destroys the structure while retaining some of the correct values. By comparison, even at a relatively extreme level of down sampling (8^3 or 512 values per voxel), the GMM rendering preserves both the structure and values of the original data.

Richtmyer-Meshkov Instability Finally, we compare down sampling a volume which was produced as the result of a Richtmyer-Meshkov instability simulation, shown in Fig. 6. In this particular example, examining the surface of turbulent mixing produces a striking result. As the amount of down sampling increases, the mean field produces larger feature lobes of mixing, amalgamating the fine-scale features, while taking a single sample of each block produces a more sporadic representation. Both also produce a low-frequency orange surface between the blue and green regions, a feature not existing in the original data. In the four components GMM, an increasing amount of blurring is shown, preserving the inherent uncertainty encapsulated by taking large regions of turbulent data and compressing them.

5 Conclusions

This work presents a technique for modeling per-voxel distributions of data using GMMs and an efficient volume raycaster that leverages this representation to show uncertainty. All examples run at interactive frame rates between 5Hz to 30Hz. In comparing our work to alternatives for down sampling, one interesting thing to note is that when the data is of high variability, we visualize it using more natural measurements of uncertainty (either flickering in

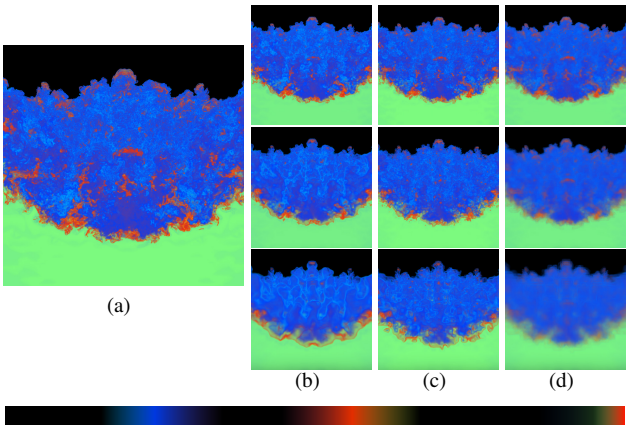

Figure 6: Down sampling the Richtmyer-Meshkov instability. Compared with the (a) original data at 512^3, we look at 3 versions of down sampled (b) mean, (c) single sample, and (d) 4 component GMM representations. From top-to-bottom, each row has resolutions 256^3, 128^3 and 64^3.

the fuzzy renderer or blurring in the screen space integration). At first glance, these images may appear of lower quality (comparing Fig. 6(b) to Fig. 6(d)), however, the blurriness of the data is an inherent byproduct of the uncertainty at those positions. In fact, we consider it a feature that our renderer prefers to not impart a certain "truth" on the data in these regions.

This work is preliminary in the sense that we have only begun to experiment with the basic implementation for rendering and manipulating GMMs. A number of open questions remain. During screen space integration, we aggregate colors, which can produce color values that are no longer representative of the initial data (e.g., red + green = yellow, while the transfer function may not have picked yellow for the color at the values between red and green). Interpolation is also a challenging question. When ray casting, we linearly interpolate a GMM at the position along the ray by interpolating the means, variances, and weights of the aligned mixture models (similar to Tan et al. [16]). However, it is unclear whether modeling the GMM random field this way is always the most appropriate (as opposed to imparting a more expensive statistical model which minimized the Fisher information metric). Future work is needed to find the best compromise between efficiency and accuracy for interpolating GMMs.

Acknowledgements

This work is supported in part by NSF awards OCI-0906379 and OCI-0904631 and DOE awards DOE/NEUP 120341, DOE/MAPD DE-SC000192, DOE/LLNL B597476, DOE/Codesign P01180734, and DOE/SciDAC DE-SC0007446. J. Samuel Preston and Sarang Joshi provided access to data [14]. LLNL-PROC-576992.

References

[1] J. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, Univ. of Berkeley, 1998.

[2] A. Bostrom, L. Anselin, and J. Farris. Visualizing seismic risk and uncertainty. *Ann. of the NY Academy of Sciences*, 1128(1):29–40, 2008.

[3] C. D. Correa, Y.-H. Chan, and K.-L. Ma. A framework for uncertainty-aware visual analytics. In *Proc. IEEE VAST*, pages 51–58, 2009.

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

[5] D. Huber and C. Healey. Visualizing data with motion. In *IEEE Visualization*, pages 527–534, Oct. 2005.

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

[7] J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *IEEE Visualization*, pages 497–504, Oct. 2003.

[8] C. Lundström, P. Ljung, A. Persson, and A. Ynnerman. Uncertainty visualization in medical volume rendering using probabilistic animation. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1648–1655, 2007.

[9] A. M. MacEachren, A. Robinson, S. Hopper, S. Gardner, R. Murray, M. Gahegan, and E. Hetzler. Visualizing geospatial information uncertainty: What we know and what we need to know. *Cartography and Geographic Information Science*, 32(3):139–160, July 2005.

[10] K. Potter. Uncertainty visualization state of the art. Symp. on Stochastic Modeling and Uncertainty Quantification, Rio de Janeiro, Brazil, 2011.

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

[12] J. Sanyal, S. Zhang, G. Bhattacharya, P. Amburn, and R. Moorhead. A user study to compare four uncertainty visualization methods for 1d and 2d datasets. *IEEE Trans. Vis. Comput. Graph.*, 15(6):1209 –1218, 2009.

[13] 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 Trans. Vis. Comput. Graph.*, 16(6):1421 – 1430, 2010.

[14] N. Singh, P. T. Fletcher, J. S. Preston, L. Ha, R. King, J. S. Marron, M. Wiener, and S. Joshi. Multivariate statistical analysis of deformation momenta relating anatomical shape to neuropsychological measures. In *MICCAI*, pages 529–537, 2010.

[15] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, pages 2246–2252, 1999.

[16] P. Tan, S. Lin, L. Quan, B. Guo, and H. Shum. Filtering and rendering of resolution-dependent reflectance models. *IEEE Trans. Vis. Comput. Graph.*, 14(2):412–425, 2008.

[17] D. C. Thompson, J. A. Levine, J. C. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. Pébay. Analysis of large-scale scalar data using hixels. In *Prov. IEEE LDAV Symp.*, pages 23–30, Oct. 2011.

[18] H. Younesy, T. Möller, and H. Carr. Improving the quality of multiresolution volume rendering. In *EuroVis*, pages 251–258, 2006.