Eurographics Symposium on Parallel Graphics and Visualization (2023)
R. Bujack, D. Pugmire, G. Reina (Editors)

FunMC^2: A Filter for Uncertainty Visualization of Marching Cubes on Multi-Core Devices

Zhe Wang[1*], Tushar M. Athawale[1*], Kenneth Moreland[1*], Jieyang Chen[2], Chris R. Johnson[3] and David Pugmire[1]

[1]Oak Ridge National Laboratory, Oak Ridge, TN, USA
[2]University of Alabama at Birmingham, Birmingham, AL, USA
[3]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT, USA
[*]These authors contributed equally to this work.

Abstract

Visualization is an important tool for scientists to extract understanding from complex scientific data. Scientists need to understand the uncertainty inherent in all scientific data in order to interpret the data correctly. Uncertainty visualization has been an active and growing area of research to address this challenge. Algorithms for uncertainty visualization can be expensive, and research efforts have been focused mainly on structured grid types. Further, support for uncertainty visualization in production tools is limited. In this paper, we adapt an algorithm for computing key metrics for visualizing uncertainty in Marching Cubes (MC) to multi-core devices and present the design, implementation, and evaluation for a ***F****ilter for* ***un****certainty visualization of* ***M****arching* ***C****ubes on* ***M****ulti-****C****ore devices (FunMC^2). FunMC^2 accelerates the uncertainty visualization of MC significantly, and it is portable across multi-core CPUs and GPUs. Evaluation results show that FunMC^2 based on OpenMP runs around $11\times$ to $41\times$ faster on multi-core CPUs than the corresponding serial version using one CPU core. FunMC^2 based on a single GPU is around $5\times$ to $9\times$ faster than FunMC^2 running by OpenMP. Moreover, FunMC^2 is flexible enough to process ensemble data with both structured and unstructured mesh types. Furthermore, we demonstrate that FunMC^2 can be seamlessly integrated as a plugin into ParaView, a production visualization tool for post-processing.*

1. Introduction

Exascale computing resources have significantly increased scientists' abilities to model complex physical phenomena and accelerate scientific discoveries. Visualization is important for extreme-scale computing and decision-making because it enables scientists to effectively and efficiently explore interesting features present in large-scale data, understand trends, and perform analysis. Fundamental visualization algorithms, for example, isosurface extraction, volume rendering, topological feature characterization, and streamline tracking, play a critical role in analysis and decision-making across a wide range of scientific domains. To gain insight and make informed decisions, scientists must be able to trust the accuracy of analysis and visualization produced from their data. Most, if not all, scientific data contain varying degrees of uncertainty due to a number of factors. These include parameters and assumptions of simulation models, spatial and temporal discretization, experimental sensor tolerances, data reduction to address storage and I/O constraints, and fixed precision representations.

Unfortunately, a majority of visualization tools and algorithms ignore the uncertainty inherent in the input data and simulation models. Visualization results that do not contain the uncertainty must be carefully analyzed by scientists in order to be trusted. Ignoring uncertainty in data and in algorithms prevents visualization systems from displaying standard deviation or confidence intervals that are indicative of errors in visualizations [Joh04] and may lead to scientists being unaware of the trustworthiness of visualizations. This lack of understanding of the reliability of visualizations can adversely impact analysis and may result in misleading scientific analysis with potentially hazardous consequences. Given the ubiquity of uncertainty throughout computational and visualization pipelines, understanding and effectively communicating uncertainty is critical for educating scientists about risks associated with scientific processes to enable better decision-making.

Marching Cubes (MC) [LC87] is a classic algorithm used for extracting isosurfaces from a scalar data field. Uncertainty visual-

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (`http://energy.gov/downloads/doe-public-access-plan`).

DOI: 10.2312/pgv.20231081

ization of MC has been an active and fruitful area of uncertainty visualization research [PWH11; AE13; ASE16; AJ19; KDJ*21; ASJ21]. Although new uncertainty algorithms of MC have been developed, several challenges remain. One key challenge is that uncertainty-aware visualization of MC algorithms is typically computationally expensive. For example, uncertainty visualization of MC [ASJ21] may require sampling data from a multivariate Gaussian distribution, which is a time-consuming operation to complete on a serial device. A second challenge is understanding the relationship between data uncertainty and the representation of uncertainty in extracted features. In addition, most of the research on uncertainty algorithms has been focused on structured data in uniform grids. Finally, effectively communicating uncertainty for decision-making remains an important challenge.

In light of these challenges, we developed a data **F**ilter that accelerates **un**certainty visualization of **M**arching **C**ubes on **M**ulti-**C**ore devices (FunMC^2). FunMC^2 utilizes VTK-m [MSU*16] to accelerate the computing of uncertainty visualization of Marching Cubes (MC) [LC87; PWH11; ASJ21]. FunMC^2 is implemented using data parallel primitives with multiple backends that can be executed on multi-core CPUs and GPUs, including both NVIDIA GPUs and AMD GPUs. To the best of our knowledge, this is the first work on accelerating uncertainty visualization for the MC isosurface extraction on multi-core CPUs and GPUs, as well as on both structured and unstructured grids. We evaluate FunMC^2 over four data sets on supercomputers at the Oak Ridge Leadership Computing Facility. Evaluation results show that FunMC^2 running on multi-core devices achieves a significant speed-up over its corresponding serial implementation. For example, FunMC^2 based on OpenMP runs between 11$\times$ to 41$\times$ faster on multi-core CPUs than the corresponding serial using one CPU core. Furthermore, FunMC^2 based on a single GPU runs around 5$\times$ to 9$\times$ faster than corresponding OpenMP. Finally, we demonstrate that FunMC^2 can be integrated as a plugin into ParaView [Aya15], an open-source multiple-platform application for interactive scientific visualization.

2. Related Work

Uncertainty is one of the top challenges facing scientific visualization [Joh04], and there has been much research on the subject, which is reviewed in detail by Kamal et al. [KDJ*21], Bonneau et al. [Han], and Potter et al. [PRJ12]. Here, we briefly discuss the most relevant previous work on computing isosurfaces with uncertain data. We also discuss past work on parallel MC algorithms and parallel visualization frameworks leveraged by our work.

2.1. Algorithms for uncertainty visualization of MC

Pöthkow et al. [PWH11] and He el al. [HMH*15] describe a general approach to express the uncertainty of scalar fields for isosurfaces. Their approach is the foundation for representing the uncertainty visualization of MC based on a multivariate Gaussian distribution. The main metric for expressing uncertainty is the probability of a grid cell crossing the specific isovalue, also known as the level crossing probability. Their works mainly focus on the algorithm in a serial implementation. Our work focuses on using parallel primitives to accelerate the computation of uncertainty MC, including level crossing probabilities and other metrics such as entropy-based uncertainty MC visualization [ASJ21].

Thompson et al. [TLB*11] describe a data representation called *hixel*, which stores a histogram generated by values at points in the sampled domain. This representation can express a fuzzy isosurface based on probabilistic MC for the reduced data set downsampled from a large data set. The work of Thompson et al. [TLB*11] mainly focuses on the algorithm description whereas our work describes how to use parallel primitives to accelerate both the downsampling process and the computing process for the isosurface extraction with multiple use case scenarios.

Gillmann et al. [GWHA18; GWHH18] present a methodology to model the uncertainty of geometry utilizing assumptions of uncertainty measurements. Their approach can use uncertainty measurement to find surfaces containing less geometric uncertainty. They consider the uncertainty of underlying graphics elements, such as the positions of points, lines, and triangles. Our work focuses on techniques that visualize the uncertainty of the isosurface based on ensemble data and downsampled data, which is a different use-case scenario of uncertainty surface visualization. The source of uncertainty is from the multiple versions of input data sets instead of the estimation based on uncertainty models.

Athawale et al. [ASJ21] present multiple metrics for describing the uncertainty of probabilistic isosurfaces [LTB*14]. The topology case count and entropy are computed based on the distribution of MC topology cases for the structured data set. However, their work mainly focuses on describing the algorithm in serial implementation. Our work utilizes parallel primitives to reduce the overhead for computing these metrics for uncertainty visualization of MC.

2.2. MC on multi-core devices

Multiple researchers discusses how to utilize GPU and multi-core devices to accelerate the classical MC algorithm [LC87]. The typical optimization strategy is to improve the memory utilization for multi-core devices based on efficient data structures [DZTS08a; SSO*10; CP13]. Another optimization strategy is to change the algorithm from one pass to multiple passes to improve the load balance [MSM10] or decrease redundant vertex access [SMG15]. These works have mainly focused on adapting the MC algorithm to multi-core devices. However, our work focuses on adapting the uncertainty visualization of MC to multi-core devices. Although the classical MC is the foundation of uncertainty visualization of MC, the latter has different input data sets and output values.

Han et al. [HAPJ22] describe a parallel implementation for the uncertainty MC algorithm based on Joblib, and they also explore how to use a deep-learning model to accelerate the prediction of uncertainty in MC. However, their approach is limited to uncertain 2D data and dedicated training data on specific types of devices. The data filter described in our work is more flexible and can be executed on different parallel devices.

2.3. Data parallel frameworks

VTK-m [MSU*16] is a production-ready library for executing data analysis and visualization filters on multi-core devices. It provides flexible parallel primitives based on adaptors for multi-core devices. The goal of this approach is both to minimize algorithm development time and to achieve efficient portable performance on multi-core devices, and studies show that despite the added layers used to achieve these goals, performance is maintained [MMP*21]. Yang et al. [YLW21] present solutions for using the VTK-m's parallel primitives to accelerate computations of histograms and the Gaussian Mixture Model (GMM). However, they did not evaluate the performance of different VTK-m device adaptors and how to integrate associated filters into other interactive visualization tools.

3. Uncertainty visualization of MC

Computing the probability distribution of the field associated with the grid vertex is a fundamental step to computing the uncertainty of MC. For either downsampled data [TLB*11] or ensemble data generated by ensemble simulations [LP08; STZ*20], a field associated with each vertex has uncertainty and contains a distribution of probable values. If we use the random variable D to represent the uncertain field value associated with each vertex, for each given isovalue k, there are $Pr(D < k)$ and $Pr(D > k)$, which are the probability that the field value is under and over the given isovalue, respectively. Based on these two probabilities for each vertex, this work focuses on metrics such as level crossing probability, topology count, and entropy for visualizing the uncertainty of MC in each cell. These metrics are presented by Athawale et al. [ASJ21] in detail.

3.1. Level crossing probability

The level crossing probability [PWH11; HMH*15], $Pr(cross)$, represents the probability that the isosurface crosses a particular cell. To compute the level crossing probability, we can integrate the probability density function of the random variable for a field associated with each vertex. Depending on the assumption about the field variable, there are different strategies to compute the level crossing probability.

For uncertainty models that satisfy the *independent random field assumption*, such as the independent Gaussian distribution, we can compute the *non*crossing probability of each cell by adding the products of the probabilities of each vertex under and over the isovalue, respectively. That is,

$$Pr(\neg cross) = \prod_{v \in C} Pr(D_v < k) + \prod_{v \in C} Pr(D_v > k), \qquad (1)$$

where the v's are the vertices in each cell, C, D_v is the random variable representing the uncertain field value at vertex v, and k is the given isovalue. We can then compute the level crossing probability as 1 minus the *non*crossing probability, $Pr(cross) = 1 - Pr(\neg cross)$.

However, there is no straightforward approach to compute the noncrossing probability of each cell under the *correlated random field assumption*, such as the multivariate Gaussian distribution. Instead of computing the integration of an n-dimensional density function, the commonly adopted strategy is to use Monte Carlo sampling from the distribution function to compute the level crossing probability. If there are M samples among N total samples having a cell that crosses the given isovalue, the level crossing probability of each cell equals $\frac{M}{N}$.

3.2. Topology case count visualization

For the 2D structured data set, each cell contains four vertices, and each field associated with the vertex has two probabilities ($Pr(D < k)$ and $Pr(D > k)$). For each cell, there are $2^4 = 16$ combinations in total. Likewise, there are $2^8 = 256$ cases for the eight vertices in each hexahedron of a 3D structured data set. Based on the process of computing the level crossing probability described in Subsection 3.1, we can get a distribution of the probability for all possible MC topology cases per cell. The case count approach [ASJ21] checks the probabilities of all these cases and counts the number of cases for which their probabilities are greater than a given threshold. High values of a lower threshold can help us understand isosurface positions with relatively high topological uncertainty.

3.3. Entropy-based uncertainty visualization

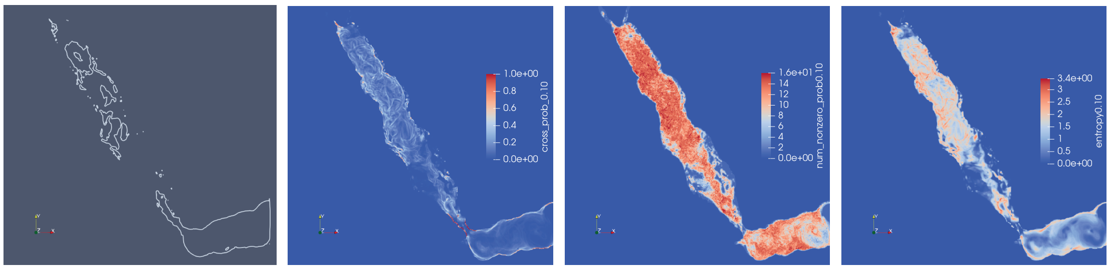
The first step of the entropy-based approach [ASJ21] is the same as for the topology case count approach discussed in Subsection 3.2. After computing the probability of each case, we use the Shannon entropy [Sha48] as an information summary for each cell. For example, for the 2D structured data set, the entropy value for each cell equals $-\sum_{i=1}^{i=2^d} P_i \times log_2(P_i)$, where d is the cell dimension and P_i is the probability of each MC case. A low entropy value represents a more deterministic isocontour, and a high entropy value implies there is more uncertainty in the data field.

Figure 1 shows concrete visualization examples for the uncertainty metrics described in this section. We use a data set generated by the ensemble simulations of variables relevant to oceanology for the Red Sea [STZ*20]. In particular, Figure 1(a) illustrates the isosurface with isovalue 0.1 based on the mean value of the ensemble data without an indication of uncertainty; Figure 1(b) illustrates the level crossing probability; Figure 1(c) illustrates the number of nonzero crossing probabilities (the given threshold is 0); and Figure 1(d) illustrates the entropy value used as the uncertainty visualization result of MC for each cell.

4. Design considerations of FunMC^2

A feature of the MC algorithm is that the computation of the section of the contour contained within a cell is independent of the rest of the computation. Therefore, the MC algorithm lends itself naturally to data parallelism, and many algorithms have taken advantage of this feature [DZTS08b; LSA12; MMM14]. Because our algorithms are based on the probabilities of MC cases, we have similar opportunities for constructing parallel algorithms.

To simplify the parallel implementation of FunMC^2, we leverage the VTK-m software framework [MSU*16], which provides a collection of primitives to construct visualization algorithms in parallel. This section describes design considerations of how FunMC^2 leverages the capability of VTK-m to accelerate the computation of uncertainty visualization of MC discussed in Section 3.



(a) Mean of ensembles (b) Cross probability (c) Number of nonzero cross probability (d) Entropy

Figure 1: *Different visualization results for the Red Sea data set at step* 40 *(the first vertical layer with isovalue* 0.1*).*

4.1. Preparing data sets with uncertainty

A typical way to represent uncertainty data is to use a collection of data about a variable (distribution data) instead of a single scalar value [LKP03]. The distribution data can then be used as the input for generating more flexible representations, such as the probability density function constructed through density estimation. For the data generated by ensemble simulation, there is uncertainty between different ensemble members, and we need to group only the ensemble values associated with each vertex into a vector (distribution field) when loading the data set. For the data sets with structured grids, we execute a downsampling operation and collect all values within a sampling region for the sub-block to construct distribution data. This downsampling process can be accelerated by parallel primitives.

When downsampling the data set with a structured grid mesh, we first specify the size of the downsampling domain. Then we extract the summary information within each downsampling domain. The assumptions of data distribution determine what information we need to extract in each downsampling domain. A uniform distribution requires only the minimal and maximal value for each downsampling domain; an independent Gaussian distribution requires the mean and standard deviation value for each sampling domain; and a multivariate Gaussian distribution requires the mean and all the raw data in each sampling domain for computing the covariance matrix used in the probability distribution function.

The complex part of performing this downsampling in parallel is identifying the members of each block to be reduced and independently combining them. FunMC^2 utilizes two approaches to accelerate the downsampling operation. The default strategy is to first mark each point of the input with an identifier of the unique downsampling block (a simple `WorkletMapField`) and then combine all points with the same downsampling block identifier with a reduce-by-key operation (`WorkletReduceByKey` in VTK-m). In this way, we can specify the different sizes of data blocks for downsampling. For example, we can set a small downsampling domain for interesting data regions and a large block size for regions with no interesting phenomena, such as the background region. If we assume the sizes of the data block are equal for each subdomain, we can use an alternative downsampling approach (the second strategy) where we use VTK-m's `WorkletPointNeighborhood` to compute the indexing of points within each downsampling block using strides and offsets. This approach avoids the overhead of processing the key in parallel, but the subdomain partitioning must be regular. Evaluation results in Section 5.2.1 show more details about the comparison of these two strategies.

4.2. Metrics for visualizing the uncertainty of MC

Whether getting uncertainty data from the downsampling operation described in Section 4.1 or through other means such as ensemble data, FunMC^2 can take these data sets and compute the probabilistic locations of contours. As described in Section 3, contour probability and other metrics are computed by considering the probability of each MC case for each cell. `WorkletVisitCellsWithPoints` of VTK-m allows an algorithm to visit each cell and access the field data on incident points to perform operations of this nature. This worklet provides a mechanism to implement the level crossing probability, topology case count, and entropy metrics.

Algorithm 1 describes detailed operations for computing the uncertainty metrics within each cell. This algorithm is executed by `WorkletVisitCellsWithPoints`, which can iterate through each cell in parallel and access the information associated with each vertex. The inputs of the algorithm include the distribution assumption of the scalar field and associated summary information of the scalar field in each cell. The extraction of the summary information of the scalar field follows the description in Subsection 4.1. Specifically, for the assumption of uniform distribution, the summary information contains the min and the max values of the scalar field associated with each vertex. For the assumption of an independent Gaussian distribution, the summary information contains the mean and the standard deviation values of the scalar field associated with each vertex. For the multivariate Gaussian distribution, the summary information contains the mean value and the raw data of the scalar field. The output of the algorithm are metrics described in Subsection 3, and we define these metrics as *CrossProb*, *NonZeroCases*, and *E(Entropy)* in this algorithm.

From line 1 to line 6, we assume the field variable is represented as a uniform or an independent Gaussian distribution. We can compute the probability density function of the field variable conveniently and derive the probability that the value is greater (less) than a given isovalue. After that, we can compute the probability

Algorithm 1: Pseudocode executed per thread to compute metrics for uncertainty visualization of MC.

```
Input  : Distribution, Summary of the scalar field
Output: CrossProb, NonZeroCases, E (Entropy)
1  if Distribution is uniform or independent Gaussian then
2      for each vertex in the cell do
3          $f \leftarrow$ probability density function ($< min, max >$ or $< mean, stdev >$);
4          $P(d > iso), P(d < iso) \leftarrow$ computing probability($f$);
5          $CasesProb \leftarrow$ updating cases of probability($P$);
6      end
7  else if Distribution is multivariate Gaussian then
8      $\Sigma_c \leftarrow$ computing covariance matrix (summary information for all vertices in the cell);
9      $AA^T \leftarrow$ EigenDecomposition($\Sigma_c$);
10     for 1 ... number of samples do
           // There are m vertices per cell
11         $Y_{m\times 1} \leftarrow$ generating random numbers from $N(0,1)$;
12         $Y_{m\times 1} \leftarrow A_{m\times m} \times Y_{m\times 1} + \mu_{m\times 1}$;
13         $CasesCount \leftarrow$ computing cases list ($Y_{m\times 1}, iso$) ;
14         $CasesProb \leftarrow$ cases of probability ($CasesCount$);
15     end
   // Computing metrics for visualizing uncertainty MC
16 $CrossProb \leftarrow 1 - CasesProb[0] - CasesProb[2^m - 1]$;
17 for $i$ in $0 \ldots 2^m - 1$ do
18     if $CasesProb[i]>0$ then
19         $NonZeroCases \leftarrow NonZeroCases + 1$;
       // Updating entropy value
20     $E \leftarrow E + (-CasesProb[i]) \times log(CasesProb[i])$;
21 end
```

for each MC topology case within the cell (denoted by variable *CasesProb* in Algorithm 1).

From line 7 to line 15 for the more complex multivariate Gaussian distribution, we use the sampling strategy to compute the probabilities for each case. For each sampling operation, we sample a vector Y_m from $N(\mu, \Sigma)$, where μ contains the mean value of each vertex, and Σ is the covariance matrix for vertex values in the corresponding cell. A more detailed explanation of sampling a multivariate Gaussian distribution is provided by Dong and Yao [DY08]. After sampling the multivariate Gaussian distribution, we can compute the probability of each MC topology case.

Line 16 computes the level crossing probability using 1 minus the probability that all vertex values are under or over the given isovalue (described in Section 3.1). In this line, we are assuming that the first and last case indices represent the case of all field values under and over the isovalue, respectively, which is common in MC algorithms. From line 17 to line 21, we iterate the probability table for all cases and compute the number of nonzero probabilities and the entropy value (discussed in Section 3.2 and Section 3.3).

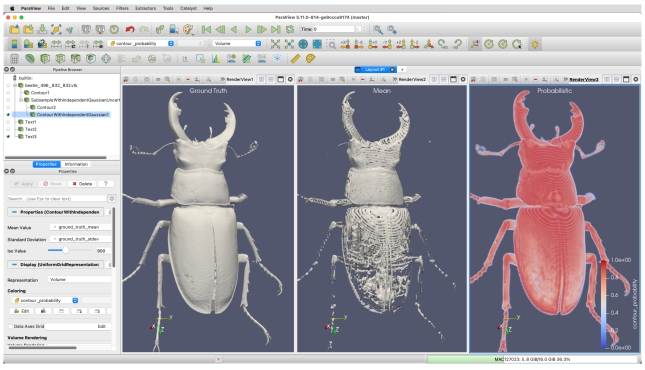


Figure 2: *A plugin makes the algorithms described in this paper easily accessible in the ParaView application.*

4.3. Integrating with the post-processing pipeline

Our ultimate goal is to make uncertainty visualization practical for real use and to put tools in the hands of end users. ParaView [Aya15] is a widely used post-processing tool for visualizing and analyzing scientific data sets. As a proof of concept, we have created a plugin for the ParaView application that makes FunMC^2 available within the tool, as demonstrated in Figure 2. The plugin adds to the ParaView interface new filters that execute uncertain contouring algorithms described in Subsection 4.1 and Subsection 4.2. The ParaView GUI allows direct manipulation of the parameters used by the algorithms (such as the isovalue used for the contour). A video in the supplementary material presents how to use FunMC^2 in ParaView for uncertainty visualization of isosurfaces in detail.

The ParaView filters internally run the previously described VTK-m implementations of the uncertain contour algorithms. Although our plugin's filter implementations are executed in VTK-m, they seamlessly integrate with the other ParaView systems – file readers, other filters, and rendering – regardless of whether these components use or directly support VTK-m. Integrating FunMC^2 into ParaView is achieved through VTK accelerator interface, which allows zero-copy data passing between the two systems.

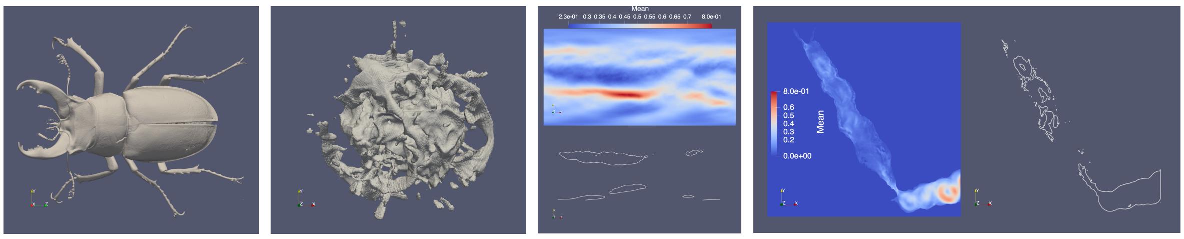
5. Performance Evaluation

The performance evaluation is divided into three subsections. Subsection 5.1 introduces the platform and data sets we targeted in the evaluation. In Subsection 5.2, we evaluate the performance results for downsampling data. Finally, Subsection 5.3 describes performance results for data generated by ensemble data. The corresponding code of all experiments is publicly available[†].

5.1. Platform and data sets

VTK-m uses an abstraction mechanism called a *device adaptor* to provide portability across a number of hardware devices. These include single-core CPU, multi-core CPU, and GPUs. Implementing

[†] https://github.com/wangzhezhe/UCV



(a) Beetle (b) Supernova (c) Wind (d) Red Sea

Figure 3: *Visualization results of the isosurface for different data sets used in the evaluation.*

Table 1: *Performance results of FunMC^2 to sample the structured data sets (time unit is milliseconds) with the Beetle data set. The number in parentheses is the ratio computed using the execution time of the serial adaptor divided by the execution time of the specific parallel device adaptor. In particular, "IndependentG" represents the independent Gaussian distribution, and "MultivariateG" represents the multivariate Gaussian distribution.*

Approach	Distribution	Serial	OpenMP	CUDA	Kokkos_CUDA	Kokkos_HIP
Reduce by key	Uniform	24363	713 (34×)	635 (**38**×)	3188 (7×)	1423 (17×)
	IndependentG	25117	716 (35×)	624 (**40**×)	3183 (7×)	1475 (17×)
	MultivariateG (1000 samples)	24996	725 (**34**×)	826 (30×)	3351 (7×)	1454 (17×)
Fixed block size	Uniform	1075	91 (11×)	171 (6×)	147 (7×)	74 (**14**×)
	IndependentG	1609	92 (17×)	171 (9×)	155 (10×)	78 (**20**×)
	MultivariateG (1000 samples)	1444	93 (15×)	357 (4×)	302 (4×)	94 (**15**×)

FunMC^2 in VTK-m gives it portability across the supported hardware types. We used the Serial, OpenMP, and CUDA device adaptors when conducting experiments on the Summit supercomputer at ORNL [‡]. VTK-m supports AMD GPUs through the Kokkos programming model [TLA*22; DGM*21]. Experiments using the Kokkos_HIP adaptor are executed on Crusher [§], the test system for Frontier [¶] at ORNL. Since Kokkos also supports NVIDIA GPUs, we also ran Kokkos_CUDA experiments on Summit to compare the two VTK-m's CUDA implementation.

We use four data sets in the evaluation. Two of them are structured data sets, and we use the downsampling operation to introduce the uncertainty to the data set in the evaluation. The other two data sets are generated by ensemble simulations, and there is uncertainty between different ensemble members in the data set. The details of the data sets are described as follows:

Beetle data set [‖] is a structured block data set of size $494 \times 832 \times 832$. This data set is generated by a CT scan of a stag beetle. This data set has been frequently used for uncertainty research of MC [TLB*11; ASJ21]. Figure 3(a) illustrates an isosurface of this data set with an isovalue 900.

Supernova data set is generated by a simulation for core-collapse supernovae [SHM*21]. For the purposes of this study, we resampled the original data onto a $400 \times 400 \times 400$ structured mesh. Figure 3(b) illustrates the isosurface of an iron fraction field with an isovalue 0.3.

Wind data set is from the European Center For Medium Range Weather Forecast (EXMWF) Sub-seasonal to Seasonal (S2S) Prediction Project. This data set has been used [HAPJ22] for accelerating the computation of probabilistic MC. This data set consists of a structured grid of size 240×121 with 15 ensemble members. The upper portion of Figure 3(c) shows the mean value wind pressure across all ensemble data, and the lower portion of Figure 3(c) shows the corresponding isosurface with an isovalue 0.3.

Red Sea data set is provided by the 2020 SciVis Contest [**]. It consists of an ensemble of oceanology simulations with varying parameter values. The data set has a spatial resolution of $500 \times 500 \times 50$ with 20 ensemble members. The left side of Figure 3(d) illustrates the mean value velocity magnitude of ensemble members for this data set. The right side of Figure 3(d) shows the corresponding isosurface with an isovalue 0.1.

‡ `https://docs.olcf.ornl.gov/systems/summit_user_guide.html`

§ `https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html`

¶ `https://docs.olcf.ornl.gov/systems/frontier_user_guide.html`

‖ `https://www.cg.tuwien.ac.at/research/publications/2005/dataset-stagbeetle/`

** `https://kaust-vislab.github.io/SciVis2020/index.html`

Table 2: *Performance results of FunMC^2 to compute uncertainty metrics for structured data sets (time unit is milliseconds). In particular, "IndependentG" represents the independent Gaussian distribution, and "MultivariateG" represents the multivariate Gaussian distribution.*

Data sets	Distribution	Serial	OpenMP	CUDA	Kokkos_CUDA	Kokkos_HIP
Beetle	Uniform	11115	266 (41×)	185 (60×)	47 (236×)	28 (**396**×)
	IndependentG	10961	263 (41×)	185 (59×)	46 (238×)	34 (**322**×)
	MultivariateG (1000 samples)	1706650	40951 (41×)	17542 (97×)	7028 (**242**×)	-
	MultivariateG (2000 samples)	3298010	79220 (41×)	35321 (93×)	11148 (**295**×)	-
Supernova	Uniform	2034	52 (39×)	48 (42×)	17 (119×)	6 (**339**×)
	IndependentG	2053	51 (40×)	49 (41×)	17 (120×)	7 (**293**×)
	MultivariateG (1000 samples)	323165	7939 (40×)	5910 (54×)	2003 (**161**×)	-
	MultivariateG (2000 samples)	615802	15557 (39×)	9063 (67×)	2828 (**217**×)	-
Beetle_Small	MultivariateG (1000 samples)	25689	1505 (17×)	1720 (14×)	234 (**109**×)	265 (96×)
Supernova_Small	MultivariateG (1000 samples)	17839	1191 (14×)	1161 (15×)	197 (**90**×)	244 (73×)

5.2. Performance results for downsampled data sets

We first describe the performance results of different downsampling strategies in Subsection 5.2.1. Then, in Subsection 5.2.2, we discuss the performance results for computing uncertainty metrics.

5.2.1. Comparison of downsampling strategies

Table 1 illustrates the performance results for the downsampling operation based on different device adaptors. The first column lists the downsampling approaches described in Subsection 4.1. The second column lists assumptions about the field variable. The block size is four when executing the downsampling process.

"Fixed block size" shows better performance than "Reduce by key" for all adaptors. This result is expected because `WorkletReduceByKey` in VTK-m contains sorting operations based on customized keys, which take extra overhead. The evaluated device adaptors have different performance results for this downsampling, and the speed-up with the highest performance improvement is labeled in bold font. In particular, the CUDA adaptor shows the highest performance improvements among all adaptors when using the "Reduce by key" approach for uniform and independent Gaussian distributions. Kokkos_HIP has the highest performance improvements when using the "Fixed block size" sampling approach. Kokkos_CUDA does not perform as well as the other CUDA device adapter for the "Reduce by key" approach. This is likely due to a known performance issue [RAB*21] with sorting in Kokkos compared with the Thrust library [BH12], and will be addressed in future versions of VTK-m and Kokkos.

With the assumption of a multivariate Gaussian distribution, we need the covariance matrix when computing the metrics for uncertainty MC; therefore, the operation of extracting summary information of the MultivariateG case requires both the mean value and the raw scalar field data for each subsampled domain. Executing the associated kernel function takes longer (around 15 times longer with CUDA adaptor) than the corresponding Uniform case.

5.2.2. Computing metrics for uncertainty visualization of MC

Table 2 illustrates the performance results of computing metrics for the uncertainty of MC (discussed in Section 3) for the Beetle and the Supernova data sets. The meaning of each column in Table 2 is the same as in Table 1. Since the Beetle and Supernova data sets are too large to be properly processed by the Kokkos_HIP adaptor for the case of the multivariate Gaussian distribution, we re-sampled these two data sets into Beetle_Small ($124 \times 208 \times 208$) and Supernova_Small ($152 \times 152 \times 152$) to show the performance of the Kokkos_HIP adaptor for the multivariate Gaussian distribution.

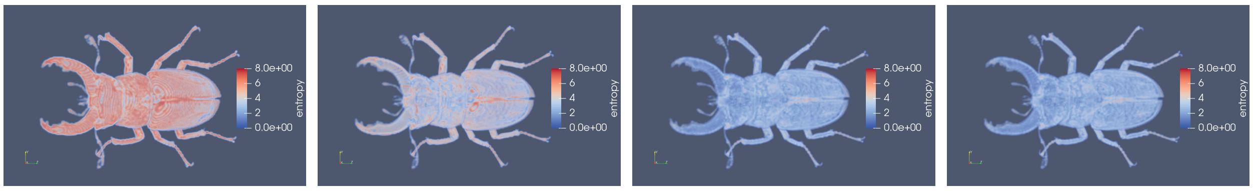
As shown in Table 2, performance results based on parallel device adaptors show significant improvements compared with the serial device adaptor. Specifically, performance results of uniform and independent Gaussian are close to each other since they have the same code structure when computing metrics regarding the uncertainty of MC (shown in algorithm 1). In contrast, the multivariate Gaussian distribution has a heavier workload, and it needs to compute covariance metrics and execute sampling operations, which take a comparatively long execution time for each thread. The heavy workload makes the GPU speed-up particularly important for the multivariate Gaussian metrics.

The Kokkos_HIP adaptor has clear benefits over other device adaptors for the uniform and independent Gaussian distributions for Beetle and Supernova data sets, and the Kokkos_CUDA adaptor shows better performance for the multivariate Gaussian.

Figure 5 illustrates the speed-up ratio of the Supernova data set with different sizes. We resampled the original Supernova data into data sets with different dimension values in this experiment. Results show that there is more speed-up with the increase in the data size. For example, the speed-up of Kokkos_CUDA adaptor increases from around 50× to 230× when the data size increases from 100^3 to 800^3.

Figure 4 and Figure 6 present metrics for the uncertainty visualization of MC discussed in Section 3. Figure 4 illustrates the entropy-based uncertainty visualization of MC with the Beetle data set. The uniform and independent Gaussian distributions tend to overestimate entropy values compared with the multivariate Gaussian distribution because they do not take into account the correlation. There are minor differences for the multivariate Gaussian distribution with 1000 and 2000 random samples.

Figure 6 illustrates two representative visualization results of the



(a) Uniform (b) Independent Gaussian (c) Multivariate Gaussian (1000 random samples) (d) Multivariate Gaussian (2000 random samples)

Figure 4: *The entropy-based uncertainty visualization of MC with different assumptions of distribution for the Beetle data set.*

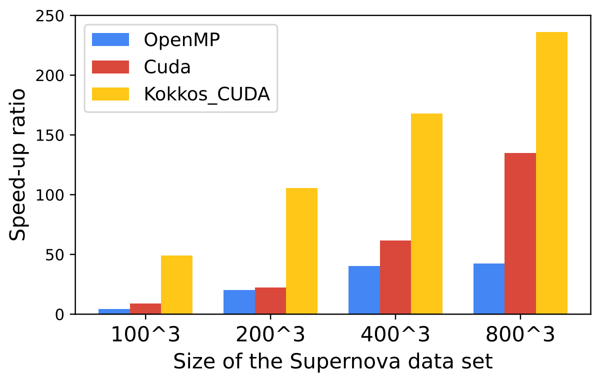
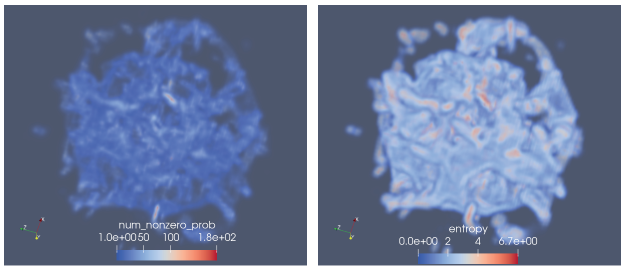


Figure 5: *Speed-up of the Supernova data set with different sizes.*



(a) Number of nonzero probabilities (b) Entropy

Figure 6: *Uncertainty visualization of MC for the Iron field of the Supernova data set.*

uncertainty visualization of MC based on the Iron field for the Supernova data set with an isovalue of 0.3. Figure 6(a) shows the number of MC topology cases with nonzero probability for each cell, and Figure 6(b) shows the entropy of the probability distribution of MC topology cases for visualizing the uncertainty of MC.

5.3. Performance results based on ensemble members

In this subsection, we first present the overall speed-up for the Wind and the Red Sea ensemble data sets in Subsection 5.3.1. Next, in Subsection 5.3.2, we discuss the comparison of execution time with a previous work [HAPJ22] that also uses the Wind data set for accelerating the uncertainty visualization of MC. Lastly, we discuss the flexibility of FunMC^2 for processing ensemble data with different types of meshes in Subsection 5.3.3.

5.3.1. Overall speed-up

Table 3 illustrates the performance results of FunMC^2 for computing metrics for visualizing the uncertainty of MC with the Wind and the Red Sea data sets. We show the results based on the multivariate Gaussian distribution, which considers the correlation between vertices and contains the heaviest workload among all distribution assumptions discussed in this paper. Since both the Wind and the Red Sea data sets are generated by ensemble simulations, we can compute the metrics for visualizing the uncertainty of MC directly based on these ensemble members. As shown in Table 3, all parallel devices show significant performance improvements. With an increasing number of samples, the performance speed-up becomes more apparent. In particular, the Kokkos_CUDA adaptor has the highest speed-up for all measurements, achieving 313× speed-up compared with the serial adaptor (and 17× faster than the OpenMP adaptor) for the Red Sea data when there are 4000 random samples.

5.3.2. Comparison to other implementations

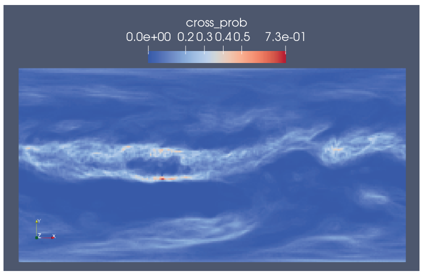


Figure 7: *The visualization of the uncertainty isosurface based on the Wind data set using isovalue* 0.3 *with the multivariate Gaussian distribution.*

The Wind data set used in our experiments is used by Han et al. [HAPJ22], and their work also focuses on improving the performance of the uncertainty visualization of MC. Although their work mainly focuses on using deep learning to improve the speed of computing metrics for visualizing the uncertainty of MC, they also present a parallel version of computing probabilistic MC in

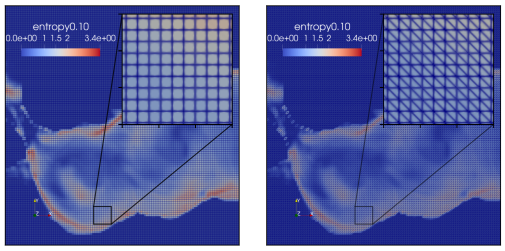
Table 3: *Performance results of FunMC^2 based on ensemble data sets (the unit is ms).*

Data sets	Distribution	Serial	OpenMP	CUDA	Kokkos_CUDA	Kokkos_HIP
Wind	MultivariateG (1000 samples)	5464	193 (28×)	121 (45×)	39 (**140**×)	119 (45×)
	MultivariateG (2000 samples)	9602	339 (28×)	143 (67×)	46 (**208**×)	188 (51×)
	MultivariateG (4000 samples)	17798	627 (28×)	199 (89×)	57 (**312**×)	317 (56×)
Red Sea	MultivariateG (1000 samples)	7264	404 (17×)	145 (50×)	47 (**154**×)	179 (40×)
	MultivariateG (2000 samples)	12717	708 (17×)	203 (62×)	55 (**231**×)	274 (46×)
	MultivariateG (4000 samples)	23547	1309 (17×)	309 (76×)	75 (**313**×)	465 (50×)

the paper. In particular, they use Joblib [††] as the infrastructure to support the parallel running of probabilistic MC. On our test platform, the probabilistic MC implemented in their work takes around 2.8 seconds with 1000 Monte Carlo samples. The concrete performance test depends on the experimental device. Our implementation based on VTK-m shows better performance for both OpenMP (14× faster) and other adaptors, such as CUDA (23× faster) on the same device.

Figure 5.3.2 illustrates the level crossing probability of isosurfaces with an isovalue of 0.3 of the Red Sea data set. We also compare our results with the results using the parallel implementation in [HAPJ22]. In particular, the maximal probability difference is 0.073 on average with a standard deviation of 0.003 if we execute the compared implementation five times. The difference between multiple runs is caused by randomness during the sampling process. The difference between our results and the compared results is 0.067 on average for five runs, and this error is close to the difference caused by random sampling.

5.3.3. Flexibility



(a) Unstructured quads. (b) Unstructured triangles.

Figure 8: *Illustration of entropy with different types of mesh for the Gulf of Aden in the Red Sea.*

One benefit of using primitives within VTK-m is to process different types of meshes using the same function call. As a proof of concept, we convert the original Wind data set into an unstructured mesh of quadrilaterals with explicit cell connections. We also produce a second unstructured mesh by dividing the quadrilaterals into triangles. Figure 8 shows the connectivity of the unstructured meshes. Both of these unstructured meshes along with the original structured mesh are demonstrated as input to FunMC^2.

Figure 1 illustrates different metrics of the uncertainty visualization of MC (discussed in Section 3) based on the original structured mesh of the Red Sea data. Figure 8 illustrates the entropy-based uncertainty visualization MC for the Gulf of Aden in the Red Sea data set with the unstructured meshes. There are minor differences visually between the results for different mesh types, which are dictated by the differences in connectivity and cases of the meshes.

Table 4 illustrates the performance results of FunMC^2 to process the Wind and the Red Sea data with different types of mesh for a multivariate Gaussian distribution with 1000 samples. The performance of processing different types of mesh is influenced by the number of cells in the data and the kernel function VTK-m used to process the data in parallel. The results show that the Kokkos_CUDA can achieve the highest speed-up.

6. Conclusions and Future Work

We have presented the design, implementation, and evaluation for FunMC^2, a filter for uncertainty visualization of Marching Cubes on multi-core devices. The performance results show that FunMC^2 achieves significant performance improvements for computing metrics used for visualizing the uncertainty of MC on multi-core CPU, NVIDIA GPU, and AMD GPU. In addition, FunMC^2 can also support ensemble data sets with different mesh types.

We observe that the parallel processing of the uncertainty visualization algorithms is an important but understudied problem. Properly managing and representing the inherent uncertainty in data processed by scientific visualization is critical to maintain trust in the results. The additional computation required to analyze probabilistic rather than absolute data can add a prohibitive amount of time to visualization systems. Parallel processing is a fruitful and necessary feature to keep uncertainty visualization tractable.

We further observe that uncertainty visualization techniques are seldom integrated into visualization tools, particularly those for scientific visualization. We have shown that scientific visualization tools are capable of adopting uncertainty techniques, and we encourage the visualization community to contribute new algorithms to these shared tools. Although we have demonstrated the use of FunMC^2 in ParaView, there is much work to be done to provide a production quality solution for end users.

The scope of this paper focuses on the parallelization over one

[††] `https://joblib.readthedocs.io/en/latest/`

Table 4: *Performance results of FunMC^2 for processing the Wind data set and the Red Sea data set with different types of mesh. Time unit is milliseconds, and there are* 1000 *samples for the multivariate Gaussian distribution*

Data sets	Mesh type	Serial	OpenMP	CUDA	Kokkos_CUDA	Kokkos_HIP
Wind	Structured	5431	192 (28×)	100 (54×)	26 (**208**×)	114 (47×)
	Unstructured Quads	5473	135 (40×)	48 (114×)	34 (**160**×)	81 (67×)
	Unstructured Triangles	7084	181 (39×)	32 (221×)	21 (**337**×)	63 (112×)
Red Sea	Structured	7264	404 (17×)	145 (50×)	46 (**157**×)	168 (43×)
	Unstructured Quads	7268	379 (19×)	83 (87×)	59 (**123**×)	198 (36×)
	Unstructured Triangles	9419	484 (19×)	42 (224×)	27 (**348**×)	140 (67×)

GPU device. That said, FunMC^2 can be easily extended to support distributed data processing. We have shown that the process of computing metrics of the uncertainty of MC is embarrassingly parallel, and this parallelism could be extended over multiple devices. We leave it to future work to demonstrate this capability by, for example, leveraging the MPI-parallel capabilities of ParaView.

Although our implementation of FunMC^2 demonstrates significant performance improvements for visualizing the uncertainty of MC, we believe there is much room for iterative improvements in future work. We observe that the comparative performance between devices is not completely explained by their respective hardware limits. Further exploration of scheduling and data management could potentially reap more benefits from the hardware and improve, for example, the size of data sets the Kokkos_HIP adapter can operate on. Furthermore, our straightforward parallelization of operations results in some repetition of computation, which may be shared with a different code structure.

The current implementation of FunMC^2 uses parametric distributions to construct probability density functions used for computing uncertainty metrics. We will extend it and support nonparametric distribution in the future.

7. Acknowledgements

Thanks for the help from Mengjiao Han in processing the Wind data set. Thanks to Christine Pickett for proofreading the paper. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This work was supported in part by the U.S. Department of Energy (DOE) RAPIDS SciDAC project under contract number DE-AC05-00OR22725 and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration.

REFERENCES

[AE13] Athawale, Tushar M. and Entezari, Alireza. "Uncertainty Quantification in Linear Interpolation for Isosurface Extraction". *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), 2723–2732. DOI: 10.1109/TVCG.2013.208 2.

[AJ19] Athawale, Tushar M. and Johnson, Chris R. "Probabilistic Asymptotic Decider for Topological Ambiguity Resolution in Level-Set Extraction for Uncertain 2D Data". *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), 1163–1172. DOI: 10.1109/TVCG.2018.2864505 2.

[ASE16] Athawale, Tushar M., Sakhaee, Elham, and Entezari, Alireza. "Isosurface Visualization of Data with Nonparametric Models for Uncertainty". *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), 777–786. DOI: 10.1109/TVCG.2015.2467958 2.

[ASJ21] Athawale, Tushar M, Sane, Sudhanshu, and Johnson, Chris R. "Uncertainty Visualization of the Marching Squares and Marching Cubes Topology Cases". *2021 IEEE Visualization Conference (VIS)*. IEEE. 2021, 106–110 2, 3, 6.

[Aya15] Ayachit, Utkarsh. *The ParaView Guide: A Parallel Visualization Application*. Kitware, Inc., 2015. ISBN: 9781930934306 2, 5.

[BH12] Bell, Nathan and Hoberock, Jared. "Thrust: A productivity-oriented library for CUDA". *GPU computing gems Jade edition*. Elsevier, 2012, 359–371 7.

[CP13] Cirne, Marcos Vinicius Mussel and Pedrini, Hélio. "Marching cubes technique for volumetric visualization accelerated with graphics processing units". *Journal of the Brazilian Computer Society* 19.3 (2013), 223–233 2.

[DGM*21] Dufek, Amanda S, Gayatri, Rahulkumar, Mehta, Neil, et al. "Case study of using Kokkos and SYCL as performance-portable frameworks for Milc-Dslash benchmark on NVIDIA, AMD and Intel GPUs". *2021 International Workshop on Performance, Portability and Productivity in HPC (P3HPC)*. IEEE. 2021, 57–67 6.

[DY08] Dong, Weishan and Yao, Xin. "Unified eigen analysis on multivariate Gaussian based estimation of distribution algorithms". *Information Sciences* 178.15 (2008), 3000–3023 5.

[DZTS08a] Dyken, Christopher, Ziegler, Gernot, Theobalt, Christian, and Seidel, Hans-Peter. "High-speed marching cubes using histopyramids". *Computer Graphics Forum*. Vol. 27. 8. Wiley Online Library. 2008, 2028–2039 2.

[DZTS08b] Dyken, Christopher, Ziegler, Gernot, Theobalt, Christian, and Seidel, Hans-Peter. "High-speed marching cubes using histopyramids". *Computer Graphics Forum*. Vol. 27. 8. Wiley Online Library. 2008, 2028–2039 3.

[GWHA18] Gillmann, Christina, Wischgoll, Thomas, Hamann, Bernd, and Ahrens, James. "Modeling and visualization of uncertainty-aware geometry using multi-variate normal distributions". *2018 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE. 2018, 106–110 2.

[GWHH18] Gillmann, Christina, Wischgoll, Thomas, Hamann, Bernd, and Hagen, Hans. "Accurate and reliable extraction of surfaces from image data using a multi-dimensional uncertainty model". *Graphical Models* 99 (2018), 13–21 2.

[Han] Hansen, Charles D. *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable*. Springer. 2.

[HAPJ22] Han, Mengjiao, Athawale, Tushar M, Pugmire, David, and Johnson, Chris R. “Accelerated Probabilistic Marching Cubes by Deep Learning for Time-Varying Scalar Ensembles”. *2022 IEEE Visualization and Visual Analytics (VIS)*. IEEE. 2022, 155–159 2, 6, 8, 9.

[HMH*15] He, Yanyan, Mirzargar, Mahsa, Hudson, Sophia, et al. “An uncertainty visualization technique using possibility theory: Possibilistic marching cubes”. *International Journal for Uncertainty Quantification* 5.5 (2015) 2, 3.

[Joh04] Johnson, Chris R. “Top Scientific Visualization Research Problems”. *IEEE Computer Graphics and Applications* 24.4 (2004), 13–17 1, 2.

[KDJ*21] Kamal, Aasim, Dhakal, Parashar, Javaid, Ahmad Y, et al. “Recent advances and challenges in uncertainty visualization: a survey”. *Journal of Visualization* 24.5 (2021), 861–890 2.

[LC87] Lorensen, William E. and Cline, Harvey E. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”. SIGGRAPH ’87. New York, NY, USA: Association for Computing Machinery, 1987 1, 2.

[LKP03] Luo, Alison, Kao, David, and Pang, Alex. “Visualizing spatial distribution data sets”. *VisSym*. Vol. 3. 2003, 29–38 4.

[LP08] Leutbecher, Martin and Palmer, Tim N. “Ensemble forecasting”. *Journal of computational physics* 227.7 (2008), 3515–3539 3.

[LSA12] Lo, Li-ta, Sewell, Christopher M, and Ahrens, James P. “PISTON: A Portable Cross-Platform Framework for Data-Parallel Visualization Operators.” *EGPGV@ Eurographics*. 2012, 11–20 3.

[LTB*14] Levine, Joshua A, Thompson, David, Bennett, Janine C, et al. “Analysis of Uncertain Scalar Data with Hixels”. *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization* (2014), 35–44 2.

[MMM14] Miller, Robert, Moreland, Kenneth, and Ma, Kwan-Liu. “Finely-Threaded History-Based Topology Computation.” *EGPGV@ EuroVis*. Citeseer. 2014, 41–48 3.

[MMP*21] Moreland, Kenneth, Maynard, Robert, Pugmire, David, et al. “Minimizing Development Costs for Efficient Many-Core Visualization Using MCD^3”. *Parallel Computing* 108 (Dec. 2021), 102834 3.

[MSM10] Martin, Steven, Shen, Han-Wei, and McCormick, Patrick S. “Load-Balanced Isosurfacing on Multi-GPU Clusters.” *EGPGV@ Eurographics*. 2010, 91–100 2.

[MSU*16] Moreland, Kenneth, Sewell, Christopher, Usher, William, et al. “VTK-m: Accelerating the Visualization Toolkit for Massively Threaded Architectures”. *IEEE computer graphics and applications* 36.3 (2016), 48–58 2, 3.

[PRJ12] Potter, Kristin, Rosen, Paul, and Johnson, Chris R. “From quantification to visualization: A taxonomy of uncertainty visualization approaches”. *IFIP advances in information and communication technology* 377 (2012), 226 2.

[PWH11] Pöthkow, Kai, Weber, Britta, and Hege, Hans-Christian. “Probabilistic marching cubes”. *Computer Graphics Forum*. Vol. 30. 3. Wiley Online Library. 2011, 931–940 2, 3.

[RAB*21] Rajamanickam, Sivasankaran, Acer, Seher, Berger-Vergiat, Luc, et al. “Kokkos kernels: Performance portable sparse/dense linear algebra and graph kernels”. *arXiv preprint arXiv:2103.11991* (2021) 7.

[Sha48] Shannon, Claude E. “A Mathematical Theory of Communication”. *Bell System Technical Journal* 27.3 (1948), 379–423 3.

[SHM*21] Sandoval, Michael A., Hix, W. Raphael, Messer, O. E. Bronson, et al. “Three-dimensional Core-collapse Supernova Simulations with 160 Isotopic Species Evolved to Shock Breakout”. *The Astrophysical Journal* 921.2 (Nov. 2021), 113 6.

[SMG15] Schroeder, William, Maynard, Rob, and Geveci, Berk. “Flying edges: A high-performance scalable isocontouring algorithm”. *2015 IEEE 5th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE. 2015, 33–40 2.

[SSO*10] Schmitz, L, Scheidegger, Luiz F, Osmari, Daniel K, et al. “Efficient and quality contouring algorithms on the GPU”. *Computer Graphics Forum*. Vol. 29. 8. Wiley Online Library. 2010, 2569–2578 2.

[STZ*20] Sanikommu, Sivareddy, Toye, Habib, Zhan, Peng, et al. “Impact of atmospheric and model physics perturbations on a high-resolution ensemble data assimilation system of the Red Sea”. *Journal of Geophysical Research: Oceans* 125.8 (2020), e2019JC015611 3.

[TLA*22] Trott, Christian R., Lebrun-Grandié, Damien, Arndt, Daniel, et al. “Kokkos 3: Programming Model Extensions for the Exascale Era”. *IEEE Transactions on Parallel and Distributed Systems* 33.4 (2022), 805–817 6.

[TLB*11] Thompson, David, Levine, Joshua A, Bennett, Janine C, et al. “Analysis of large-scale scalar data using hixels”. *2011 IEEE Symposium on Large Data Analysis and Visualization*. IEEE. 2011, 23–30 2, 3, 6.

[YLW21] Yang, Hao-Yi, Lin, Zhi-Rong, and Wang, Ko-Chih. “Efficient and portable distribution modeling for large-scale scientific data processing with data-parallel primitives”. *Algorithms* 14.10 (2021), 285 3.