Exploration and Analysis of Ensemble Datasets with Statistical and Deep Learning Models

Dissertation

Presented in Partial Fulfillment of the Requirements for the Degree Doctor of Philosophy in the Graduate School of The Ohio State University

By

Wenbin He, B.E.

Graduate Program in Computer Science and Engineering

The Ohio State University

2019

Dissertation Committee:

Han-Wei Shen, Advisor

Rephael Wenger

Huamin Wang

Abstract

Ensemble simulations are becoming prevalent in various scientific and engineering disciplines, such as computational fluid dynamics, aerodynamics, climate, and weather research. Scientists routinely conduct a set of simulations with different configurations (e.g., initial/boundary conditions, parameter settings, or phenomenological models) and produce an ensemble of simulation outputs, namely an ensemble dataset. Ensemble datasets are extremely useful in studying the uncertainty of the simulation models and the sensitivities of the initial conditions and parameters. However, compared with deterministic scientific simulation data, visualizing and analyzing ensemble datasets are challenging because the ensemble datasets introduce extra dimensions into the field data (i.e., each spatial location is associated with multiple possible values instead of a deterministic value) and extra facets (e.g., simulation parameters).

Over the last decade, various approaches have been proposed to visualize and analyze ensemble datasets from different perspectives. For example, the variability of isocontours is modeled and visualized by a collection of techniques [115, 118, 120, 121, 148]. Coordinated multiple views are frequently used to visualize the simulation parameters and outputs simultaneously and linked together to study the influence of different simulation parameters [20, 22, 34, 106, 108, 117, 145, 155]. However, to handle different types of ensemble datasets (e.g., unstructured grid data, time-varying data, and extreme-scale data) and address

various visualization tasks (e.g., uncertainty modeling and parameter space exploration), more work needs to be done in terms of ensemble data visualization and analysis.

In this dissertation, we focus on visual exploration and analysis of ensemble datasets using statistical and deep learning models. Specifically, we explore and analyze ensemble datasets from three perspectives. First, we focus on modeling and visualizing the variability of ensemble members for 1) features (e.g., isosurfaces) derived from the field datasets and 2) raw simulation outputs (i.e., field datasets). For the derived features especially surface-like features (e.g., isosurfaces and streamsurfaces), we propose a statistical approach to study the variability of features. We model the positional uncertainty of the surfaces by extending kernel density estimate (KDE) from discrete data points to the infinite set of points on the input surfaces. For the field datasets, we focus on modeling and visualizing the variability of scalar values at each spatial location. To this end, we treat the scalar value at each spatial location as an independent random variable and model the distribution of the random variable using KDE. To visualize and explore the distributions, we decompose and summarize the distributions over a few representative subranges by cumulative probabilities. Then we build a visual interface to explore and analyze the field of distributions based on the cumulative probabilities over subranges. Second, we study the influence of different initial conditions, parameterizations, and/or phenomenological models of simulations on the simulation outputs. To model the mapping between simulation inputs and outputs, which are often highly complex and nonlinear functions, we embrace the emerging deep learning techniques. With the assistance of trained deep learning models, users can predict the visualization of simulation outputs given arbitrary simulation inputs and study the sensitivity of different simulation parameters through interactive explorations. Third, we extend our study from visualizing and analyzing members within an ensemble to comparing

multiple ensembles, for which we focus on studying how, when, and where the ensembles agree/disagree with each other. To this end, we train a discriminative network to differentiate members of one ensemble from members of the other. After training, we can use the discriminative network to approximate the overall difference between the two ensembles and identify members and spatial locations the two ensembles agree/disagree with each other. This dissertation ends with potential future research directions, as well as a summary of our contributions.

Dedicated to Xiaoxiang He, Yuhua Chen, and Xi Yi

Acknowledgments

My doctoral study at The Ohio State University has been a truly life-changing experience for me and it would not have been possible to do without the support and guidance that I received from many people.

Firstly, I would like to express my sincere gratitude to my advisor Dr. Han-Wei Shen for the continuous support of my Ph.D. study and related research. His patience, motivation, and immense knowledge have been invaluable to me. I could not have imagined having a better advisor and mentor for my Ph.D. study.

Besides my advisor, I would like to thank the rest of my dissertation committee, Dr. Rephael Wenger and Dr. Huamin Wang, for their insightful comments and suggestions to help improve my dissertation.

The internship experience at Argonne National Laboratory contributed significantly to my Ph.D. study. I would like to sincerely thank my mentor Dr. Hanqi Guo and Dr. Tom Peterka, who helped me a lot in developing research ideas and writing clear and concise research papers. I would also like to thank my collaborators at Argonne National Laboratory, Scott M. Collis, Jonathan J. Helmus, Sheng Di, Franck Cappello, Mukund Raj, and Youssef S. G. Nashed.

I must also thank my fellow lab-mates and friends for the stimulating discussions and for the sleepless nights we were working together before deadlines. They are: Junpeng Wang, Xiaotong Liu, Ko-Chih Wang, Teng-Yok Lee, Kewei Lu, Soumya Dutta, Subhashis

Hazarika, Xiaonan Ji, Chun-Ming Chen, Ayan Biswas, Cheng Li, Xin Tong, Jiayi Xu, Tzu-Hsuan Wei, Rui Li, Abon Chaudhuri, Ming-Yi Su, Haoyu Li, Jingyi Shen, Yamei Tu, Neng Shi, Rachel Wang, Hyunjean Choi, Skylar Wurster, and Piyush Chawla.

Last but not least, I would like to thank the unconditional love, care, and encouragement from my parents and my lovely wife. It is impossible for me to complete my Ph.D. study without their support.

Vita

February 4, 1990	Born - Dongtai, Jiangsu, China
2008 - 2012	B.E. Beijing Institute of Technology, Beijing, China
May - July, 2015	Research Aide, Argonne National Laboratory, Lemont, IL, USA
May - August, 2016	Research Aide, Argonne National Laboratory, Lemont, IL, USA
January - May, 2017	Graduate Teaching Associate, The Ohio State University, Columbus, OH, USA
May - July, 2019	Summer Intern, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA
2013 - 2019	Graduate Research Associate, The Ohio State University, Columbus, OH, USA

Publications

Research Publications

Wenbin He, Junpeng Wang, Hanqi Guo, Ko-Chih Wang, Han-Wei Shen, Mukund Raj, Youssef S. G. Nashed, and Tom Peterka "InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations". *IEEE Transactions on Visualization and*

Computer Graphics (SciVis 2019), 2019 (Early Access). **[IEEE SciVis 2019 Best Paper Award]**.

Wenbin He, Hanqi Guo, Han-Wei Shen, and Tom Peterka "eFESTA: Ensemble Feature Exploration with Surface Density Estimates". *IEEE Transactions on Visualization and Computer Graphics*, 2019 (Early Access).

Hanqi Guo, Wenbin He, Sangmin Seo, Han-Wei Shen, Emil Mihai Constantinescu, Chunhui Liu, and Tom Peterka "Extreme-Scale Stochastic Particle Tracing for Uncertain Unsteady Flow Visualization and Analysis". *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2710–2724, 2019.

Wenbin He, Hanqi Guo, Tom Peterka, Sheng Di, Franck Cappello, and Han-Wei Shen "Parallel Partial Reduction for Large-Scale Data Analysis and Visualization". In *Proceedings of 2018 IEEE Symposium on Large Data Analysis and Visualization*, pp. 45–55, 2018. **[IEEE LDAV 2018 Best Paper Honorable Mention Award]**

Wenbin He, Xiaotong Liu, Han-Wei Shen, Scott M. Collis, and Jonathan J. Helmus "Range Likelihood Tree: A Compact and Effective Representation for Visual Exploration of Uncertain Data Sets". In *Proceedings of 2017 IEEE Pacific Visualization Symposium*, pp. 151–160, 2017.

Hanqi Guo, Wenbin He, Tom Peterka, Han-Wei Shen, Scott M. Collis, and Jonathan J. Helmus "Finite-Time Lyapunov Exponents and Lagrangian Coherent Structures in Uncertain Unsteady Flows". *IEEE Transactions on Visualization and Computer Graphics (PacificVis 2016)*, vol. 22, no. 6, pp. 1672–1682, 2016.

Wenbin He, Chun-Ming Chen, Xiaotong Liu, and Han-Wei Shen "A Bayesian Approach for Probabilistic Streamline Computation in Uncertain Flows". In *Proceedings of 2016 IEEE Pacific Visualization Symposium, Visualization Notes*, pp. 214–218, 2016.

Ayan Biswas, David Thompson, Wenbin He, Qi Deng, Chun-Ming Chen, Han-Wei Shen, Raghu Machiraju, and Anand Rangarajan "An Uncertainty-Driven Approach to Vortex Analysis Using Oracle Consensus and Spatial Proximity". In *Proceedings of 2015 IEEE Pacific Visualization Symposium*, pp. 223–230, 2015.

Fields of Study

Major Field: Computer Science and Engineering

Studies in:

Computer Graphics	Prof. Han-Wei Shen
Artificial Intelligence	Prof. Mikhail Belkin
Database	Prof. Srinivasan Parthasarathy

Table of Contents

Page

Abstract ii

Dedication v

Acknowledgments vi

Vita viii

List of Tables xv

List of Figures xvii

1. Introduction 1

    1.1 Background and Motivation 1
    1.2 Research Problems 2
    1.3 Proposed Solutions 5
        1.3.1 Exploration of Ensemble Surface Features with Density Estimation 5
        1.3.2 Range Likelihood Tree: A Compact and Effective Representation for Visual Exploration of Uncertain Datasets 6
        1.3.3 Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations 7
        1.3.4 Collective Ensemble Comparison and Visualization using Deep Neural Networks 7
    1.4 Outlines 8

2. Related Work 9

    2.1 Ensemble Visualization 9
        2.1.1 Ensemble Feature Extraction and Visualization 9
        2.1.2 Parameter Space Exploration of Ensemble Simulations 10

2.1.3 Collective Comparison between Ensembles 11
2.2 Uncertainty Visualization 13
2.2.1 Uncertain scalar field visualization 13
2.2.2 Uncertain vector field visualization 14
2.3 In Situ Visualization 15
2.4 Density Estimation 15
2.5 Deep Learning for Visualization 17

3. eFESTA: Ensemble Feature Exploration with Surface Density Estimates 18

3.1 Overview 20
3.2 Surface Density Estimate 22
3.2.1 SDE Definitions 22
3.2.2 SDE Calculation 23
3.3 Bandwidth Selection with Least Squares Cross Validation (LSCV) 27
3.4 Visual Exploration of SDE 29
3.4.1 Hierarchical Representation of Density Fields 31
3.4.2 Visual Exploration with Hierarchical Clustering Tree 31
3.5 Implementation Details and Performance Evaluation 33
3.6 Results 34
3.6.1 Synthetic Ensemble Surfaces 35
3.6.2 Isosurfaces in Ensemble Scalar Fields 41
3.6.3 Lagrangian Coherent Structures in Uncertain Flow Fields 48
3.6.4 Streamsurfaces in Ensemble Flow Fields 50
3.7 Conclusion, Limitations, and Future Work 52

4. Range Likelihood Tree: A Compact and Effective Representation for Visual Exploration of Uncertain Datasets 54

4.1 Overview 57
4.2 Methods 58
4.2.1 Transforming Distribution into Range Likelihoods 58
4.2.2 A Multi-level Data Model for Range Likelihood Fields 61
4.2.3 Range Likelihood Based Probabilistic Classification 63
4.3 Range Likelihood Tree Guided Exploration Framework 65
4.3.1 User Interface 65
4.3.2 Exploration Guidelines 69
4.4 Results 71
4.4.1 Massachusetts Bay Sea Trial Ensemble Dataset 72
4.4.2 Temporally Down-Sampled Hurricane Isabel Dataset 74
4.4.3 Ensemble HRRR Simulation Dataset 77
4.5 Discussion and Future Work 78

4.6 Conclusion . . . 80

5. InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations . . . 82

5.1 Overview . . . 85
5.2 In Situ Training Data Collection . . . 86
5.3 InSituNet Architecture and Training . . . 88
5.3.1 Network Architecture . . . 89
5.3.2 Loss Function . . . 94
5.3.3 Techniques to Stabilize Training . . . 97
5.3.4 Training Process . . . 99
5.4 Parameter Space Exploration with InSituNet . . . 100
5.4.1 Inference of Visualization Results . . . 101
5.4.2 Sensitivity Analysis on Simulation Parameters . . . 101
5.5 Results . . . 103
5.5.1 Ensemble Simulations . . . 103
5.5.2 Implementation and Performance . . . 104
5.5.3 Model Evaluation for Different Hyperparameters . . . 106
5.5.4 Comparison with Alternative Methods . . . 112
5.5.5 Parameter Space Exploration . . . 114
5.6 Limitations, Discussion, and Future Work . . . 118
5.7 Conclusion . . . 119

6. CECAV-DNN: Collective Ensemble Comparison and Visualization using Deep Neural Networks . . . 120

6.1 Overview . . . 123
6.2 Collective Ensemble Comparison . . . 125
6.2.1 Discriminative Networks . . . 126
6.2.2 Three-Level Comparative Analysis of Two Ensembles . . . 130
6.2.3 Extending to Multiple Pairs of Ensembles . . . 133
6.3 Visualization System for Collective Comparison . . . 134
6.3.1 User Interface . . . 135
6.3.2 Exploration Guidelines . . . 138
6.4 Case Studies . . . 139
6.4.1 Climate Ensembles with Different Simulation Models . . . 139
6.4.2 CFD Ensembles with Different Spatial Resolutions . . . 143
6.5 Implementation and Performance . . . 147
6.6 Discussion and Future Work . . . 148
6.7 Conclusion . . . 150

7. Conclusion and Future Work 152

7.1 Conclusion 152
7.2 Future Work 154

Appendices 156

A. Bivariate Normal Integral Calculation 156

Bibliography 158

List of Tables

Table **Page**

3.1 Data specifications and performance (in seconds). t_d: timings for density estimation; t_c: timings for hierarchical clustering; t_r: response time for interactive queries (i.e., selecting a node in the hierarchical clustering tree). 34

4.1 Datasets and timings: t_c is the runtime (in seconds) for the clustering algorithm. 72

5.1 Datasets: k controls the size of InSituNet to cope with datasets in different complexities; diversity [153] measures how diverse the generated images are. 103

5.2 Timings: t_{sim}, t_{vis}, and t_{tr} are timings for running ensemble simulations, visualizing data in situ, and training InSituNet, respectively; t_{fp} and t_{bp} are timings for a forward and backward propagation of the trained InSituNet, respectively. 105

5.3 Quantitative evaluation of InSituNet trained with different loss functions. The model trained with the combination of $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$ generates images with the best EMD and FID and only a slightly lower PSNR and SSIM compared with the model trained with $\mathscr{L}_{mse}$ or $\mathscr{L}_{feat}$. 108

5.4 Evaluating the weight λ of $\mathscr{L}_{adv\_R}$: $\lambda = 0.01$ provides the results that balance the PSNR, SSIM, EMD, and FID. 110

5.5 Size and training time of different network architectures controlled by k for the Nyx dataset. 111

5.6 Evaluation of the number of ensemble runs used for training. 112

5.7 Quantitative comparison of images generated with interpolation, GAN-VR, and InSituNet. 112

6.1 Architecture details and training time of the discriminative networks used in the two case studies. 147

6.2 Defined μ and υ functions for different types of divergences between distributions. $\mathscr{F}$ is a class of real-value bounded functions, and $\|\mathscr{D}_\phi\|_1$ stands for 1-Lipschitz functions. 149

List of Figures

Figure **Page**

3.1 Overview of the exploration workflow. We begin with an ensemble of surface features (a), whose corresponding density fields (b) are generated based on density estimation. The resulting density fields are organized into a hierarchical representation (c) based on their pairwise distances. The hierarchical representation is then used to guide the visual exploration of the surfaces (d) and the density fields (e). 21

3.2 Density fields generated for a triangular patch: (a) and (b) show KDEs calculated based on 10^4 and 10^5 sample points, respectively; (c) shows SDE calculated analytically based on the triangular patch. 24

3.3 Mapping a 3D triangle (a) into 2D (b) by parameterizing the triangle in the 2D space defined by the plane of the triangle. The origin $\mathbf{o}$ of the 3D space is projected onto the plane of the triangle at $\mathbf{w}$ that is treated as the origin of the 2D space. 2D equivalent of each point $\mathbf{p}'$ within the triangle is denoted as $\mathbf{q}$. 25

3.4 Computing the bivariate normal integral over a triangle $\overline{T}$, denoted as $\alpha_{\overline{T}}$. Integrals α_{AB}, α_{BC}, and α_{CA} are first computed based on the method proposed by Owen [109]. Then $\alpha_{\overline{T}}$ can be computed by combining α_{AB}, α_{BC}, and α_{CA}, for which there are two cases. (a) The origin is inside the triangle and (b) The origin is outside the triangle. 27

3.5 (a) Density field of an ensemble of isocontours in 2D. (b)-(d) Three possible shapes of the isocontours. 30

3.6 Hierarchical clustering tree view, where ring layout represents surfaces ordered by their IDs from S_0 to S_{N-1}. When a node is selected, the corresponding ring segments are enlarged and highlighted. 32

3.7 Visualization of the three synthetic ensembles of spheres: (a) distributions of radius; (b) ground truth density fields; (c)-(e) density estimation results of IC, IKDE, and our method, respectively. 35

3.8 Quantitative analysis of the proposed method for the three synthetic ensembles of spheres with different ensemble sizes from 128 to 2,048 for every power of 2: (a) bandwidth σ suggested by the proposed bandwidth selection method, (b) PSNR (db) between the density estimation results of the proposed method and the ground truth. 37

3.9 Example of counting the number of circles intersecting grid cells. Because of the discretization, errors are introduced into the resulting density field. For example, points P and Q should have the same density, but different densities are assigned to P and Q in the result. 38

3.10 Quantitative comparison for IC, IKDE, and our method: (a), (c), and (e) PSNR (db) between the density estimation results and the ground truth; (b), (d), and (f) timings for density estimation. 40

3.11 Visualization of the isosurfaces extracted from the synthetic ensemble scalar fields using different methods: (a) spaghetti plots, (b) CPP [80], (c) fuzzy isosurfacing [148], (d) SDVR [131], (e) LCP [119], and (f) SDE. Notice that different techniques quantify the positional uncertainty of isosurfaces with different metrics. Hence, the results have different value ranges and have to be visualized using specific transfer functions. 41

3.12 Example of correlation between random variables at different spatial locations: (a) locations P and Q that random variables are defined on, (b) histogram of scalar values at P, (c) histogram of scalar values at Q, and (d) scatter plot of scalar values at P and Q. 44

3.13 Visualization of the density fields and the underlying isosurfaces for three representative clusters selected from the hierarchical clustering tree for the synthetic ensemble scalar fields. The major trends of the isosurfaces are extracted and highlighted. 45

3.14 Visualization of the isosurfaces extracted from the CHL variable of the MBST-98 ensemble data with an isovalue 1.4 using (a) spaghetti plots and (b)-(d) SDE with different transfer functions. The vertical axis represents water depth in meters, which increases from bottom to top. 46

3.15 Visualization of the density fields for four representative clusters based on the isosurfaces generated from the CHL variable of the MBST-98 ensemble data with an isovalue 1.4. Distinct features are highlighted at Boston Harbor and Stellwagen Bank for the four clusters. 48

3.16 Visualization of the uncertain LCSs extracted from the HRRR ensemble dataset: (a) spaghetti plots of the surfaces, (b) SDE, and (c) density field created by SPH. 49

3.17 Visualization of the streamsurfaces generated from the synthetic ensemble flow around a confined square cylinder, where the flow is moving from left to right: (a) spaghetti plots of the streamsurfaces (b) volume rendering of the SDE, and (c) two clusters of density fields that highlight different flow features. 51

4.1 Visual analysis of probability distribution fields using conventional techniques. (a) Visualizing a distance field constructed from a 2D probability distribution field; regions associated with two distinct distributions can not be readily distinguished from their distances to the target. (b) Visualizing a low-dimensional embedding of a 3D probability distribution field using principal component analysis (PCA); no clear separation of clusters can be found. 56

4.2 Overview of the analytical workflow. 57

4.3 Three probability distributions from different spatial locations in an uncertain scalar field. They are summarized by two subranges (in two distinct colors) with the probabilities that each distribution falls within the subranges as $[0.7, 0.3]$, $[0.5, 0.5]$, and $[0.3, 0.7]$, respectively. 59

4.4 Transforming distribution into range likelihoods. (a) Visualization of scalar fields sampled from a 2D distribution field. (b) Visualization of range likelihood fields (RLFs) transformed from the distribution field. 60

4.5 An illustration of three range likelihood fields (RLFs), highlighting grid points with non-zero likelihoods. Based on spatial locality of grid points, RLF-a is similar to RLF-b, while RLF-c is different from RLF-a and RLF-b. 62

4.6 The composite radial visualization of a range likelihood tree with distribution density map. 66

4.7 Visualizing distributions of the material ensemble dataset using superimposed curves (top) and curve density estimation (bottom). 66

4.8 Multi-field transfer function widget using parallel coordinates axes (top) and flexible coordinates axes (bottom). 68

4.9 An illustration of our range likelihood tree guided exploration framework using the Massachusetts Bay Sea Trial Ensemble dataset. Starting from the initial range likelihood tree view (1), users can hover on different nodes in the tree to see their corresponding subranges (2), and examine the likelihoods of associated grid points in the range likelihood field view (3); after selecting a few subranges of interest, a multi-field transfer function widget (4) is created, which can be manipulated by the user to explore and classify grid points based on their likelihoods in different subranges in the multi-field classification view (5). 70

4.10 Exploring ranges at different levels in the RLT of the Massachusetts Bay Sea Trial Ensemble dataset. 72

4.11 Experiments on the Temporally Down-Sampled Hurricane Isabel dataset. See Section 4.4.2 for details about the exploration process. 75

4.12 Experiments on the Ensemble HRRR Simulation dataset. See Section 4.4.3 for details about the exploration process. 77

4.13 Comparison of different clustering approaches using the 2D material ensemble data set. Each image represents a RLF with respect to one cluster. (a): Hierarchical clustering based on the Euclidean distances between range likelihood fields. (b): Hierarchical clustering applied on likelihood distributions (scaled range likelihood fields) using the Euclidean distance. (c): K-means clustering using the Euclidean distances between likelihood distributions. . 79

5.1 Workflow of our approach. Ensemble simulations are conducted with different simulation parameters on supercomputers, and visualization images are generated in situ for different visual mapping and view parameters. The generated images and the parameters are collected into an image database. A deep image synthesis model (i.e., InSituNet) is then trained offline based on the collected data, which is later used for parameter space exploration through an interactive visual interface. 85

5.2 Our in situ training data collection pipeline. Simulation data, generated with different simulation parameters, are visualized in situ with different visual mapping and view parameters. The in situ visualization generates a large number of images, which are collected along with the corresponding parameters for the training of InSituNet offline. 86

5.3 Overview of InSituNet, which is a convolutional regression model that predicts visualization images from input parameters. During training, the regression model is trained based on the losses computed with the assist of a pretrained feature comparator and a discriminator. 88

5.4 Architecture of R_ω, which encodes input parameters into a latent vector with fully connected layers and maps the latent vector into an output image with residual blocks (a). The size of R_ω is defined by k, which controls the number of convolutional kernels in the intermediate layers. 90

5.5 Architecture of D_υ. Input parameters and the predicted/ground truth image are transformed into latent vectors with fully connected layers and residual blocks (a), respectively. The latent vectors are then incorporated by using the projection-based method [99] (b) to predict how likely the image is a ground truth image conditioning on the given parameters. Similar to R_ω, the size of D_υ is controlled by the constant k. 92

5.6 Architecture of F (i.e., VGG-19 network), where each layer is labeled with its name. Feature maps are extracted through convolutional layers (e.g., `relu1_2`) for feature-level comparisons. 93

5.7 Visual interface developed for parameter space exploration. (a) The three groups of parameters: simulation, visual mapping, and view parameters. (b) The predicted visualization images and the sensitivity analysis results. . . . 100

5.8 Qualitative comparison of InSituNet trained with different loss functions. Combining $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$ gives the results of high quality. 107

5.9 Images generated by InSituNet trained with $\mathscr{L}_{feat}$ that uses different layers after the first pooling layer of VGG-19: (a) `relu2_1` and (b) `relu3_1`. Checkerboard artifacts are introduced. 109

5.10 Quantitative evaluation of different network architectures controlled by k with (a) PSNR and (b) EMD. 110

5.11 Comparison of the images generated using interpolation, GAN-VR, and InSituNet with the ground truth images. 113

5.12 Parameter space exploration with the Nyx simulation. For the selected parameter values, the sensitivity of different parameters is estimated and visualized as line charts on the left, whereas the predicted image is visualized on the right. 114

5.13 Comparison of the visual appearance of the predicted images using different h values to see the effect of this simulation parameter. 115

5.14 Predicted images of the MPAS-Ocean dataset for different isosurfaces and viewpoints, which reasonably reflect the change of view projections and shading effects. 116

5.15 Forward prediction (top row) and backward subregion sensitivity analysis (bottom row) for different *BwsA*. Regions that influenced by *BwsA* (i.e., regions a and b) are highlighted by the sensitivity map. 117

6.1 Workflow of CECAV-DNN, which takes a sequence of ensemble pairs as input, and trains a discriminative network for each ensemble pair. After training, three levels of comparative analysis are provided with an interactive visualization system to compare the ensemble pairs. 124

6.2 Architecture of discriminative network $\mathscr{D}$, which begins with a sequence of convolutional layers, followed by a few fully connected layers. 126

6.3 Comparing two ensembles using the two distributions of likelihood scores produced by the trained discriminative network $\mathscr{D}_\phi$. 131

6.4 Saliency map of a member computed using backpropagation. 132

6.5 User interface of CECAV-DNN demonstrated by the climate ensembles generated with two different simulation models: (a) the parallel violins plot view presents the overall dissimilarities (a1) as well as the likelihood score distributions (a2) of the ensemble pairs in comparing; (b) after selecting an ensemble pair of interest and brushing on the corresponding violin plot, members whose likelihood score are within the brushed ranges are visualized in the member view; (c) by feeding the selected members into the trained discriminative network, saliency maps are generated using backprogation and visualized in the saliency map view. 134

6.6 (a) Jointing two PDFs as side-by-side views for comparison. (b) A band is created after brushing on a violin plot, which reveals the distribution of the currently selected members in other ensembles. 135

6.7 Comparison between two models in day 13: (a) likelihood score distributions and selected ranges; (b) aggregated saliency map of the selected members; (c), (d), (e), and (f) mean of the selected members within the ranges (a1), (a2), (a3), and (a4), respectively. 141

6.8 Aggregated saliency map of all members for each day. Positive gradients cover larger spatial regions than negative gradients. 143

6.9 The PVP for the CFD ensembles: (a) overall dissimilarities across three resolutions; (b) likelihood score distributions of E_l and E_h. 144

6.10 Visualization of the selected members in time 18: (a) the likelihood distributions of E_l and E_h as well as the selected ranges; (b)−(e) mean fields of the selected members within the ranges shown in (a1)−(a4), respectively; (g)−(j) slice on the top of the domain for the four mean fields; (f) and (k) mean saliency map of selected members and the slice of the saliency map. . 145

A.1 (a) Region over which $\Gamma(h,a)$ gives the integral of the standard bivariate normal distribution. (b) and (c) Bivariate normal integral over areas defined by an edge of a given triangle. 156

Chapter 1: Introduction

1.1 Background and Motivation

Simulation models in various scientific and engineering disciplines are often associated with different model configurations (e.g., initial/boundary conditions, parameter settings, and phenomenological models). By conducting a set of simulations with different configurations and compare the outputs of individual runs, scientists are able to study the variability of the simulation models and the sensitivity of the configurations. The output of the simulation runs is a collection of spatio-temporal results, namely an ensemble dataset, and the result of a single run is often referred to as an ensemble member or simply a member. To explore and analyze ensemble datasets effectively, visualization techniques are playing an important role. However, compared with conventional scientific simulation data, visualizing and analyzing ensemble datasets are challenging for two reasons. First, the ensemble datasets introduce extra dimensions into the field data. For example, given an ensemble dataset, each spatial location is associated with multiple possible values instead of a deterministic value. Moreover, scientists often need to generate multiple ensembles for different phenomenological models or spatial resolutions, which introduce another dimension (i.e., the ensemble dimension [154]) into the data. Second, the simulation outputs are associated with the input configurations, and scientists often need to explore

the simulation parameter space to study the influence and sensitivity of different input configurations.

Over the last decade, various ensemble visualization and analysis approaches have been proposed. For example, a group of approaches focused on studying features derived from ensemble datasets, such as isocontours [115,118,120,121,148] and streamlines [47]. Temporal trend of time-varying ensemble datasets has been studied in [49,106]. The influence and sensitivity of simulation parameters to the simulation outputs have been visualized and analyzed with visual linking of coordinated multiple views [20,22,29,34,36,94,106,108,117,145,155]. Despite the advances in recent ensemble visualization approaches, however, ensemble data visualization and analysis remain challenging as scientists often require techniques to address various analysis tasks (e.g., uncertainty modeling and parameter space exploration) and to handle different types of ensemble datasets (e.g., unstructured grid data, time-varying data, and extreme-scale data).

In this dissertation, we propose several statistical and deep learning approaches to study ensemble datasets from different perspectives. With statistical modeling techniques such as kernel density estimation (KDE), we analyze and visualize the uncertainty of various types of ensemble datasets as well as derived features. With the assist of recent advances in deep learning, we are able to study the mapping between simulation inputs (e.g., parameters) and outputs (i.e., ensemble datasets). In addition, we can differentiate members of different ensembles with deep neural networks.

1.2 Research Problems

In this dissertation, we focus on addressing four specific research problems for ensemble data exploration and analysis, which include:

1. How to model and visualize the variability of surface features (e.g., isosurfaces, ridge surfaces, and streamsurfaces) derived from ensemble datasets when the field datasets are no longer available? Although various techniques have been proposed to visualize the variability of surface features, isosurfaces in particular [115, 118, 120, 121, 148], most of the existing approaches rely on the field datasets (e.g., scalar fields) instead of studying the derived surfaces directly. As the field datasets are not always available or useful for modeling the variability of the surfaces, a technique applied on the surfaces directly needs to be developed. For example, given an ensemble of vector fields, the uncertainty of the streamsurfaces cannot be derived from the field datasets directly because the uncertainty can be propagated through the vector fields. Instead, streamsurfaces have to be extracted from individual members first, and then the variability of the extracted surfaces have to be studied. Moreover, scientists need to explore the surfaces effectively such that the major trends and outlines of the surfaces can be identified.
2. How to model and visualize the uncertainty of values at each spatial location? Given an ensemble of scalar fields, as each spatial location is associated with a set of possible values instead of a discriminate one, conventional scalar field visualization techniques such as volume rendering cannot be applied directly. A common approach is to treat the value at each spatial location as a random variable and study the probability distributions of the random variables. In order to visualize and analyze the field of probability distributions, various techniques have been proposed. For example, the derived statistics (e.g., means and standard deviations) could be visualized with conventional techniques such as volume rendering. Distributions of selected spatial locations could be visualized directly with line charts. However, how to effectively

explore and analyze the complex distributions and preserves the spatial information of the distributions remain challenging.

3. How to explore the parameter space of ensemble datasets in extreme scale? Parameter space exploration plays an important role in analyzing the influence and sensitivity of different simulation parameters, and various techniques with visual linking of coordinated multiple views [20, 22, 29, 34, 36, 94, 106, 108, 117, 145, 155] have been proposed to address this task. However, as the computational power of modern supercomputers continues to grow, the spatial and/or temporal resolutions of ensemble simulations are getting finer. Due to the IO and storage bottleneck, scientists are not able to store the simulation outputs onto disk and perform post-hoc exploration and analysis with the previous techniques. As a result, the simulation outputs are often visualized in situ and only the visualization results are stored for post-hoc exploration and analysis. However, as the simulation data are no longer available, the flexibility of the post-hoc parameter space exploration is limited.

4. How to analyze and visualize the differences between different ensembles? In modeling complex scientific phenomena, scientists often produce multiple ensembles using different phenomenological models and/or spatial resolutions. To understand the differences between phenomenological models and/or spatial resolutions, collective comparison between different ensembles plays an important role. However, the collective comparison is non-trivial because the ensembles are collections of simulation outputs that reside in a very high dimensional space, which represent high dimensional distributions. Modeling and comparing high dimensional distributions are challenging, especially scientists often interest in when and where the distributions are different other than just the overall difference. Although ensemble visualization

has been extensively studied recently, the majority of previous works focus more on comparing members within an ensemble other than different ensembles. Several previous works that tackle the collective comparison between ensembles focus more on individual spatial locations [68] or simulation parameters [20, 155] other than the field datasets. Hence, the collective comparison between ensembles remains challenging for understanding the differences between the underlying models.

1.3 Proposed Solutions

In this dissertation, we propose four approaches in exploring and analyzing ensemble datasets from different perspectives and addressing the research problems discussed in Section 1.2.

1.3.1 Exploration of Ensemble Surface Features with Density Estimation

To model and visualize the variability of surface features (e.g., isosurfaces, ridge surfaces, and streamsurfaces) for 3D ensemble simulation data, we propose *surface density estimate* (SDE). The inputs of SDE computation are surface features represented as polygon meshes, and no field datasets are required (e.g., scalar fields or vector fields). The SDE extends KDE from discrete points to the infinite set of points on the input surfaces. We propose a method to calculate the surface densities of triangular patches analytically and accumulate the surface densities of the triangular patches to approximate the SDE of given surfaces represented as triangular meshes. We also propose an algorithm to guide the selection of a proper kernel bandwidth for SDE computation. An ensemble feature exploration method based on SDE is then proposed to extract and visualize the major trends of ensemble surface features. For an ensemble of surface features, each surface is first transformed into a density

field based on its contribution to the SDE, and the resulting density fields are organized into a hierarchical representation based on the pairwise distances between them. The hierarchical representation is then used to guide visual exploration of the density fields as well as the underlying surface features. We demonstrate the application of the proposed method using isosurface in ensemble scalar fields, Lagrangian coherent structures in uncertain unsteady flows, and streamsurfaces in ensemble fluid flows.

1.3.2 Range Likelihood Tree: A Compact and Effective Representation for Visual Exploration of Uncertain Datasets

To model and analyze the uncertainty of ensemble scalar fields at every spatial location, probability distributions are often used. Exploring ensemble scalar fields modeled as a probability distribution field is a challenging task because the underlying features are often complex, and the data associated with each grid point are high dimensional. In this work, we present a compact and effective representation, called range likelihood tree, to summarize and explore probability distribution fields. The key idea is to decompose and summarize each complex probability distribution over a few representative subranges by cumulative probabilities, and allow users to consider the roles that different subranges play in understanding the probability distributions. In our method, the value domain is first partitioned into subranges, then the distribution at each grid point is transformed according to the cumulative probabilities of the point's distribution in those subranges. Organizing the subranges into a hierarchical structure based on how these cumulative probabilities are spatially distributed in the grid points, the new range likelihood tree representation allows effective classification and identification of features through user query and exploration. We present an exploration framework with multiple interactive views to explore probability distribution fields and provide guidelines for visual exploration using our framework. We

demonstrate the effectiveness and usefulness of our approach in exploratory analysis using several representative ensemble datasets.

1.3.3 Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations

To support parameter space exploration for ensemble simulations that are visualized in situ, we propose InSituNet, a deep learning based surrogate model. In situ visualization, generating visualizations at simulation time, is becoming prevalent in handling large-scale simulations because of the I/O and storage constraints. However, in situ visualization approaches limit the flexibility of post-hoc exploration because the raw simulation data are no longer available. Although multiple image-based approaches have been proposed to mitigate this limitation, those approaches lack the ability to explore the simulation parameters. Our approach allows flexible exploration of parameter space for large-scale ensemble simulations by taking advantage of the recent advances in deep learning. Specifically, we design InSituNet as a convolutional regression model to learn the mapping from the simulation and visualization parameters to the visualization results. With the trained model, users can generate new images for different simulation parameters under various visualization settings, which enables in-depth analysis of the underlying ensemble simulations. We demonstrate the effectiveness of InSituNet in combustion, cosmology, and ocean simulations through quantitative and qualitative evaluations.

1.3.4 Collective Ensemble Comparison and Visualization using Deep Neural Networks

We perform collective ensemble comparisons to help scientists understand the differences between phenomenological models and/or spatial resolutions by comparing their ensemble

simulation outputs. Because the simulation outputs reside in a very high dimensional space, it is challenging to compare the collections of simulation outputs that model high dimensional distributions. To address the issue, we choose to train a deep discriminative neural network to differentiate members of one ensemble from members of the other, such that we are able to identify members that the ensembles agree/disagree with each other based on the output of the discriminative neural network. Moreover, by analyzing which spatial location are more sensitive to the discriminative neural network, we are able to identify the locations that the ensembles differ the most. We also design and develop a visualization system to help scientists understand the collective comparison results based on the discriminative network. We demonstrate the effectiveness of our approach with two real-world applications, including the ensemble comparison of the community atmosphere model (CAM) and the rapid radiative transfer model for general circulation models (RRTMG) [163] for climate research, and the comparison of computational fluid dynamics (CFD) ensembles with different spatial resolutions.

1.4 Outlines

In the following, the dissertation is organized as follows. We first discuss the related works in chapter 2. Then, we provide the details of the four proposed approaches in Chapter 3 to 6. In Chapter 7, we conclude the dissertation and discuss future research opportunities.

Chapter 2: Related Work

In this chapter, we review and discuss previous works that are closely related to this dissertation, which are categorized as ensemble visualization, uncertainty visualization, in situ visualization, density estimation, and deep learning for visualization.

2.1 Ensemble Visualization

The previous ensemble visualization techniques cover various data types (e.g., curves, surfaces, and volumes) and address different analysis tasks (e.g., comparison, clustering, and parameter space exploration). Comprehensive surveys on this topic can be found in [107, 154]. In this section, we review ensemble visualization techniques that are closely related to this dissertation, which includes ensemble visualization techniques on feature extraction, parameter space exploration, and collective comparison.

2.1.1 Ensemble Feature Extraction and Visualization

Spaghetti plots are a well-known visual analysis technique that overlays an ensemble of features such as isocontours to compare among ensemble members [123]. Phadke et al. [116] proposed two ensemble visualization methods, pairwise sequential animation and screen door tinting, for visualizing ensembles that contain numerous members. Sanyal et al. [134] proposed a tool named Noodles that uses glyphs and confidence ribbons to

visualize uncertainties for operational meteorologists. Whitaker et al. [159] and Mirzargar et al. [97] proposed contour and curve box-plots, respectively, to visualize ensembles of curves based on the concept of statistical band depth. Raj et al. [125] extended contour box-plots to 3D for brain atlas construction and analysis. Zehner et al. [164] proposed a method to render multiple isosurfaces with a specific color scheme. Ferstl et al. [47] presented a technique to explore the major trends in an ensemble of streamlines based on principal component analysis and hierarchical clustering. More recently, Ferstl et al. [48] presented an approach that uses signed distance functions to generate contour variability plots for the visual analysis of an ensemble of isocontours. They further extended the approach for visualizing the spatial and temporal evolution of isocontours in ensembles of 2D time-varying scalar fields [49]. Demir et al. [38] proposed a technique to visualize ensembles of isosurfaces based on screen-space silhouettes.

However, the previous works mainly focus on line features (e.g. contour lines); and for surface features, the previous works can not show point-wise quantitative information. Unlike previous works, we analyze the variability of ensemble surfaces using density estimation, thus the probability density of surfaces passing through any given location can be quantified.

2.1.2 Parameter Space Exploration of Ensemble Simulations

The parameter space exploration of ensemble simulations is used to study the impact of simulation parameters on the simulation outputs. Existing approaches focus mainly on visualizing the simulation parameters and relating the parameters with the simulation outputs.

The simulation parameters are typically treated as multidimensional vectors and can be visualized by traditional visualization techniques, such as parallel coordinate plots [106,155], radial plots [29, 34, 36], scatter plots [94, 108, 145], line charts [20], matrices [117], and glyphs [22].

To relate the simulation parameters with the simulation outputs, we categorize previous approaches into four groups: brushing and linking, clustering, sensitivity analysis, and direct manipulations. The first group uses brushing and linking of coordinated multiple views to visualize the correspondences between the simulation inputs and outputs [22, 34, 94]. The second group uses clustering algorithms to analyze simulations with a large number of parameter settings [29, 106, 117, 145, 155] and reveal the dominant patterns in those settings. The third group uses sensitivity analysis to identify more influential parameters [20]. The fourth group uses direct manipulations [36, 108], which allow users to directly interact with the visualizations of simulation inputs and outputs simultaneously, to study the influence of simulation parameters.

Our work is distinguished from these approaches in two aspects. First, different from works that studied a limited number of simulation inputs and outputs collected from ensemble runs, we train a surrogate model to extend our study to arbitrary parameter settings within the parameter space, even if the simulations were not executed with those settings. Second, our approach is incorporated with in situ visualization, which is widely used in large-scale ensemble simulations.

2.1.3 Collective Comparison between Ensembles

Comparisons between ensembles play an important role in analyzing various simulation models, investigating different spatial resolutions, and exploring the temporal evolution

of ensembles. A few techniques have been proposed to tackle the problem of collective ensemble comparison, and much work remains to be done. Existing techniques include comparison between ensembles of scalar values, simulation parameters, and isocontours.

Höllt et al. [68] compared two ensembles of scalar values at different spatial locations across time by visualizing the distributions of the two ensembles of scalar values side-by-side at each timestep. Köthur et al. [79] extended the use of the windowed cross-correlation matrix to support a correlation-based comparison between two ensembles of time series. However, the methods of Höllt et al. and Köthur et al. are limited to ensembles of scalar values and not feasible for the comparison between ensembles of scalar fields.

Wang et al. [155] and Biswas et al. [20] proposed methods to investigate climate ensembles of different spatial resolutions for the analysis of the simulation parameters and the prediction accuracy. Wang et al. introduced the nested parallel coordinates plot to visualize intra-resolution and inter-resolution parameter correlations. Biswas et al. analyzed and visualized the sensitivity and accuracy of the ensembles with respect to the simulation parameters across different spatial resolutions. However, the methods of Wang et al. and Biswas et al. mainly focused on investigating the influence of different spatial resolutions on the sensitivity of the simulation parameters and the accuracy of the prediction results compared with the observed ground truth. Comparisons between ensembles of simulation fields to investigate where the ensembles agree or disagree with each other are missing.

Pfaffelmoser et al. [114] proposed a technique to indicate the difference between ensembles of isocontours using spaghetti plots colored by gradients along the isocontours and backgrounds colored by probabilities of scalar values greater than the isovalue. Ferstl et al. [49] analyzed ensembles of 2D isocontours at different timesteps using time-hierarchical

clustering and visualized the temporal evolution of the clusters using stacked contour variability plots. However, the methods of Pfaffelmoser et al. and Ferstl et al. mainly focused on visual comparison between two collections of 2D isocontours. Collective comparison and visualization of the difference between distributions of simulation outputs remain challenging.

2.2 Uncertainty Visualization

In this section, we summarize the closely related studies on uncertainty quantification and visualization for scalar and vector fields. Comprehensive reviews of uncertainty visualization can be found in [28, 110, 122].

2.2.1 Uncertain scalar field visualization

To visualize the uncertainty of scalar fields, various techniques have been proposed. One group of techniques focuses on modeling the uncertainty of scalar values and encoding the uncertainty into additional visual channels. For example, glyph-based methods that encode uncertainties by glyphs were explored by Hlawatsch et al. [66] and Wittenbrink et al. [160]. Pfaffelmoser et al. [114] proposed methods to visualize the variability of gradients in 2D uncertain scalar fields. Fout and Ma [50] designed a system that provides verifiable volume rendering via uncertainty analysis. Djurcilov et. al. [40] incorporated uncertainty into volume rendering. Luo et at. [91] introduced distributions as a new data type and discussed several techniques for visualizing spatial distribution data. Sakhaee and Entezari [131] presented a spline-based framework to quantify and propagate the uncertainty through the volume rendering pipeline. Another group of techniques visualizes the uncertainty of scalar fields by modeling the uncertainty of isocontours. Thompson et al. [148] introduced a new data representation for uncertain scalar data, called hixels, which enables fuzzy

isosurface visualization to indicate the possible isosurface locations in the data. Pöthkow and Hege [118] proposed a technique that approximates the level crossing probabilities (LCPs) [120] with parametric models for uncertainty analysis of isocontours. Based on this method, Pöthkow et al. proposed the probabilistic marching cubes algorithm [121]. Recently, Pöthkow and Hege proposed a nonparametric statistical analysis framework for uncertain isosurfaces visualization [119]. Athawale and Entezari [11] presented a closed-form computation of LCPs for studying the interaction between linear interpolation and data uncertainty quantified by the uniform distribution. More recently, Athawale et al. presented an isosurface extraction algorithm [12] for uncertain scalar fields where the data uncertainty is modeled by nonparametric statistics. Hazarika et al. [61] proposed a coupla-based technique for LCP approximation. Pfaffelmoser et al. proposed [115] a method to determine more reliable LCPs by considering correlations in the data. Kumpf et al. proposed contour probability plots [80] to depict lobes, which indicate the probability that a contour line is locally contained in the lobe.

2.2.2 Uncertain vector field visualization

Various techniques have been proposed for visualizing uncertain vector datasets. A class of techniques focuses on encoding uncertainties in vector datasets as additional visual channels, such as glyphs [66, 90, 160] and textures [23]. Another class of techniques analyzes and visualize features extracted from uncertain vector datasets by extending methods used in deterministic flow fields such as streamlines and stream ribbons [110]. Petz et al. [113] presented a statistical analysis framework to extract probabilistic local features from uncertain vector fields considering their spatial correlation structure. Bhatia et al. [16] presented streamwaves to visualize spatial errors of edge maps used in particle tracing. The

concept of LCS has also been extended to visualize and analyze uncertain time-varying flow fields by Haller [58] and Guo et al. [56].

2.3 In Situ Visualization

Based on the output, in situ visualization can be categorized into image-based [3, 4], distribution-based [44], compression-based [39, 82], and feature-based [26] approaches. The proposed technique in this dissertation is an image-based approach, which visualizes simulation data in situ and stores images for post-hoc analysis. Tikhonova et al. [149–151] generated images of multiple layers in situ to enable the adjustment of transfer functions in post-hoc analysis. Frey et al. [51] proposed volumetric depth images, a compact representation of volumetric data that can be rendered efficiently with arbitrary viewpoints. Fernandes et al. [46] later extended volumetric depth images to handle time-varying volumetric data. Biedert and Garth [17] combined topology analysis and image-based data representation to preserve flexibility for post-hoc exploration and analysis. Ahrens et al. [3, 4] proposed Cinema, a framework that stores visualization images in situ and performs post-hoc analysis via exploration and composition of those images.

Compared with these approaches, our work supports not only the exploration of various visual mapping and view parameters but also the creation of visualizations under new simulation parameters without actually running the simulation.

2.4 Density Estimation

One of the proposed techniques in this dissertation is closely related to density estimation. In this section, we review studies on density estimation for various applications in the context

of data analysis and visualization, including density estimation of particles, graphs, parallel coordinate plots (PCPs), trajectories, and surfaces.

Density estimation of particles has been extensively studied for visualization and analysis. Peterka et al. [112] reviewed and evaluated density estimation methods that transform particles to a probability density field, including cloud in cell (CIC) [18], smoothed particle hydrodynamics (SPH) [100], tessellation (TESS) [136, 137], and adaptive cloud in cell (ACIC) [19]. CIC [18] is a first-order method that linearly interpolates the particle's mass to points of estimation within a fixed-size hypercube. The size and shape of the region over which the particle's mass is distributed were further adjusted for each particle in the SPH [100] and TESS [136, 137]. Concepts of CIC and SPH were then combined in ACIC [19], in which the size of the hypercube adapts to cover a given number of particles as in SPH.

Density estimation has also been extended to analyze and visualize more complex data types, including graphs, PCPs, trajectories, and surfaces. Zinsmaier et al. [170] proposed a technique that aggregates node density with KDE [142] and generates meta edges for visualizing large graphs interactively with adjustable levels of detail. Hurter et al. [71] proposed a method to compute bundled layouts of general graphs based on density estimates of graph edges. Heinrich and Weiskopf [63] proposed continuous parallel coordinates, which visualizes PCPs using density estimation. Zhou et al. [167] proposed a method to visualize PCPs as density fields using splatting. Muigg [101] employed anisotropic diffusion of noise textures to visualize line orientations for PCPs. Scheepens et al. presented a method to explore density maps of trajectories with multiple attributes [139] and extended it by compositing multiple density fields for more comprehensive analysis [138]. Lampe et al. presented KDE-based visual analysis methods for scatterplots of dynamic data [84] and

spatiotemporal trajectories [85]. They further presented a technique for rendering smooth curves independent of frequencies, zoom level, and models based on KDE [83]. Guo et at. [56] proposed a method to transform uncertain LCSs into a density field based on SPH and a zero-order kernel function that is uniformly distributed in the volume of a sphere. Compared with U-LCS, our work generalizes kernel density estimation with higher-order kernel functions (e.g., the Gaussian kernel) for transforming ensemble surfaces to density fields.

2.5 Deep Learning for Visualization

The visualization community has started to incorporate deep learning in visualization research. For example, Hong et al. [69] used long short-term memory [67] to estimate access pattern for parallel particle tracing. Han et al. [60] used autoencoders [130] to cluster streamlines and streamsurfaces. Xie et al. [161] used neural network embeddings to detect anomalous executions in high performance computing applications. Berger et al. [15] proposed a deep learning approach to assist transfer function design using generative adversarial networks (GANs) [54], which is closely related to this dissertation. Specifically, we focus on parameter space exploration of ensemble simulations instead of transfer function design for volume rendering.

Our work is related to deep learning based image synthesis, which has been used in various applications, including super-resolution [41, 73, 86], denoising [162, 166], inpainting [111, 162], texture synthesis [53, 168], text-to-image synthesis [126], style transfer [52,73,169], and rendering [15,43]. We investigate and combine different state-of-the-art deep learning techniques on image synthesis (e.g., perpetual losses [73, 86] and GANs [54]) to improve the quality of our image synthesis results.

Chapter 3: eFESTA: Ensemble Feature Exploration with Surface Density Estimates

Modeling and visualizing the variability of features extracted from ensemble simulation data are playing an important role in ensemble visualization. In this chapter, we focus on studying surface features such as isosurfaces, ridge surfaces, and streamsurfaces, which are widely used in many scientific applications. Uncertainty quantification of surface features based on polygon meshes instead of field datasets (e.g., scalar fields or vector fields) is important because field datasets that can be used to quantify the positional uncertainty of surface features are often not available. For example, for specific applications such as ridge surfaces and streamsurfaces, the positional uncertainty of the surfaces can not be quantified based on the vector fields without known the geometries of the surfaces. Only using the field data, the application is limited to modeling the positional uncertainty of isosurfaces, for which various techniques [11, 12, 80, 115, 118, 120, 121, 131, 148] have been proposed. Moreover, the majority of the previous techniques [11, 12, 80, 118, 131, 148] model the uncertainty of the scalar values at each grid point as an independent random variable without considering correlations between the random variables at different spatial locations. Only a few techniques [115, 120, 121] consider the correlations of the random variables to generate more reliable results. In this work, isosurfaces are extracted from each ensemble member

independently, such that spatial correlations of scalar values within each ensemble member are preserved.

In this work, we propose *surface density estimate* (SDE), which generalizes the kernel density estimate (KDE) from discrete sample points to the infinite set of points on input surfaces. We approximate SDE of the input surfaces by accumulating the surface densities of triangular patches, which can be calculated based on bivariate normal integrals with efficient GPU computation.

We also propose an algorithm to guide the selection of a proper kernel bandwidth for SDE computation. Like KDE, the most important parameter for computing a SDE is the bandwidth, which determines the degree of smoothing induced. An improper bandwidth may cause under- or oversmoothing problems. Our bandwidth selection method is based on the variability of input surfaces. We extend the least squares cross validation (LSCV) method to approximate the mean integrated squared error (MISE) between the SDE and the target density field. The bandwidth is then selected by minimizing the approximated MISE.

To further extract and visualize the major trends of ensemble surface features, we develop an efficient interactive visualization framework called *e*FESTA (ensemble Feature Exploration based on Surface densiTy EstimAtes). In this framework, surface of each ensemble member is first transformed into a density field based on its contribution to the SDE. The density fields are then organized into a hierarchical representation based on their pairwise distances. The hierarchical representation is then used to guide the visual exploration of the major trends of the underlying surfaces.

We compare the accuracy and performance of our density estimation method with alternative approaches using synthetic uncertain surfaces that have known spatial distributions.

Our method is able to generate more accurate density estimation results when the computation cost is similar to the alternative approaches. We also demonstrate the effectiveness and usefulness of the proposed method for different applications including visualizing and analyzing variabilities of isosurfaces in ensemble simulations, Lagrangian coherent structures (LCSs) [59] in uncertain unsteady flows, and streamsurfaces in ensemble fluid flows. For the application of isosurfaces in ensemble simulations, the proposed method generates comparable results to other state-of-the-art approaches that rely on scalar fields. Our method can also handle applications that are depend on extracted surface features such as LCSs and streamsurfaces. Moreover, by applying similarity guided visual exploration, the major trends of ensemble surfaces can be extracted and visualized effectively through user interaction. In summary, the contributions of this study are threefold:

- A density estimation method to visualize and analyze ensemble surfaces
- An algorithm to guide the selection of kernel bandwidth for SDE computation
- A visual exploration framework for the derived density fields

3.1 Overview

In this work, we start with an ensemble of surface features represented as polygon meshes. Instead of visualizing the surfaces directly, such as with overlaid rendering, we visualize and analyze the surfaces based on their spatial distribution. To model the spatial distribution of the surfaces, we present *surface density estimate* (SDE) (details in Section 3.2), which we define as the kernel densities of the infinite set of points on the input surfaces. By computing the SDE, surface features are transformed into a density field, which is then visualized using existing volume exploration techniques such as volume rendering. A bandwidth selection

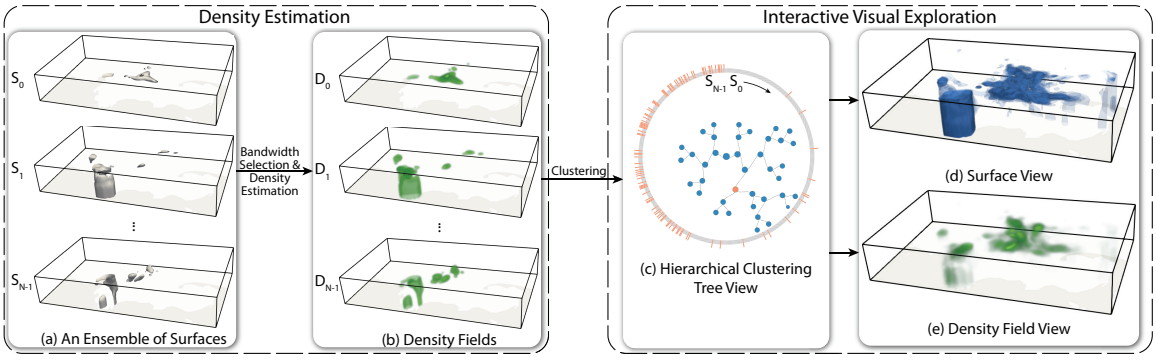


Figure 3.1: Overview of the exploration workflow. We begin with an ensemble of surface features (a), whose corresponding density fields (b) are generated based on density estimation. The resulting density fields are organized into a hierarchical representation (c) based on their pairwise distances. The hierarchical representation is then used to guide the visual exploration of the surfaces (d) and the density fields (e).

method (details in Section 3.3) is also proposed to guide the selection of a proper kernel bandwidth.

To further extract and visualize the major trends of ensemble surface features based on the SDE, we present an efficient interactive visualization framework called *e*FESTA (details in Section 3.4). Our system has two major modules: the density estimation module and the exploration module, as shown in Figure 3.1. Given an ensemble of surfaces S_0, S_1, ..., S_{N-1} (Figure 3.1(a)), we first select a bandwidth for SDE computation based on the variability of the surfaces. Then for each surface S_i, a density field D_i (Figure 3.1(b)) is generated based on its contribution to the SDE, where the density value at each grid point represents the probability density of the point being on the current surface. The resulting density fields from the density estimation module are then used in the exploration module for comparative analysis and interactive visualization of the underlying surfaces. For efficient visual exploration, we organize the density fields into a hierarchical representation (Figure 3.1(c)) that groups similar density fields into clusters. This representation is then

used to guide the visual exploration of the surfaces (Figure 3.1(d)) and the corresponding density fields (Figure 3.1(e)).

3.2 Surface Density Estimate

We present SDE, which generalizes KDE from discrete sample points to surfaces. In this section, we first give a formal definition of SDE and then describe how to approximate the SDE from a given surface.

3.2.1 SDE Definitions

We start with a review of multidimensional KDE for discrete data points and then extend KDE for SDE. Given a set of discrete data points $\mathbf{x}_0$, $\mathbf{x}_1$, ..., $\mathbf{x}_{n-1}$ in $\mathbb{R}^d$, where d is the dimensionality of space, the KDE $\hat{f}_{\mathbf{H}}(\mathbf{x})$ is defined as a convolution of the data points with respect to a smoothing kernel $K_{\mathbf{H}}$ at position $\mathbf{x}$,

$$\hat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=0}^{n-1} K_{\mathbf{H}}(\mathbf{x}-\mathbf{x}_i), \tag{3.1}$$

where $K_{\mathbf{H}}$ is defined as

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}} K(\mathbf{H}^{-\frac{1}{2}}\mathbf{x}), \tag{3.2}$$

$\mathbf{H}$ is a symmetric positive definite bandwidth matrix that controls the degree and orientation of smoothing, and K is the kernel function, which is a symmetric multivariate probability density function. Among various of kernel functions, the multivariate Gaussian kernel,

$$K(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d}} e^{-\frac{1}{2}\mathbf{x}^T\mathbf{x}}, \tag{3.3}$$

is the most widely used kernel function for KDE, we thus use the multivariate Gaussian kernel to demonstrate the concept of SDE in this study.

Given a surface S, we define the SDE as the kernel densities of the infinite set of points on the surface. In other words, the SDE is the convolution of every point $\mathbf{p} \in S$ with respect to a smoothing kernel $K_{\mathbf{H}}$ at any position $\mathbf{x}$ in $\mathbb{R}^3$,

$$\hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{Area(S)} \iint_S K_{\mathbf{H}}(\mathbf{x}-\mathbf{p})\mathrm{d}S, \tag{3.4}$$

where $Area(S)$ is the area of the surface S.

3.2.2 SDE Calculation

To calculate the SDE for a given surface S, we first discretize the surface as a triangular mesh, which comprises a set of triangular patches $T_0, T_1, \ldots, T_{n-1}$. Then, for each triangular patch T, we compute its surface density $\tau_{\mathbf{H}}(\mathbf{x}, T)$ with respect to the surface S. The $\tau_{\mathbf{H}}(\mathbf{x}, T)$ is defined as the convolution of every point $\mathbf{p} \in T$ with respect to a smoothing kernel $K_{\mathbf{H}}$ at target position $\mathbf{x}$ and normalized by the area of the surface S,

$$\begin{aligned} \tau_{\mathbf{H}}(\mathbf{x}, T) &= \frac{1}{Area(S)} \iint_T K_{\mathbf{H}}(\mathbf{x}-\mathbf{p})\mathrm{d}T \\ &= \frac{1}{Area(S)} |\mathbf{H}|^{-\frac{1}{2}} \iint_T K(\mathbf{H}^{-\frac{1}{2}}(\mathbf{x}-\mathbf{p}))\mathrm{d}T. \end{aligned} \tag{3.5}$$

We then approximate the SDE of the surface S by accumulating the surface densities contributed by its triangular patches,

$$\hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x}) = \sum_{i=0}^{n-1} \tau_{\mathbf{H}}(\mathbf{x}, T_i). \tag{3.6}$$

The surface density of a triangular patch can be approximated with the KDE over points sampled from the triangular patch. Figures 3.2(a) and (b) show two example density fields constructed for a single triangular patch as KDEs over 10^4 and 10^5 sample points, respectively. More accurate results can be generated by increasing the number of sample points, but the computation cost increases. In this work, we present a method to compute

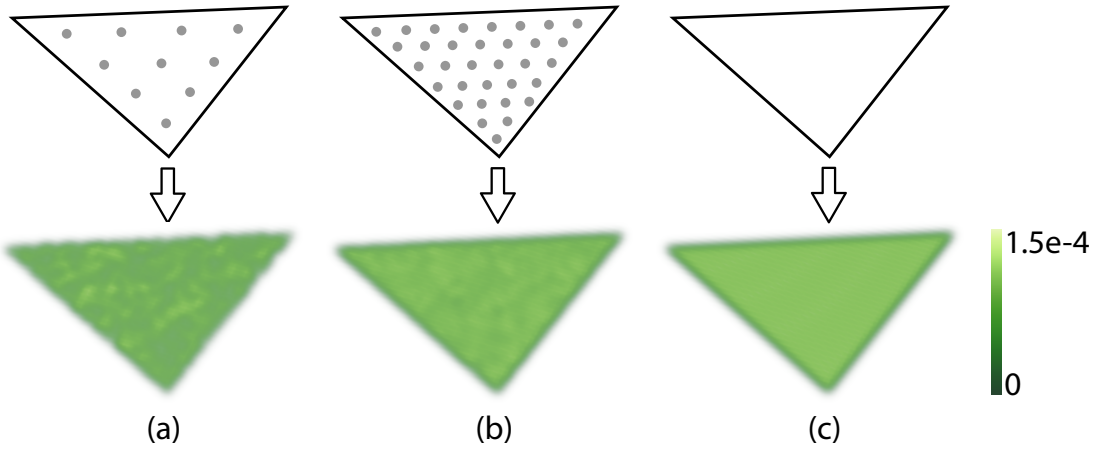


Figure 3.2: Density fields generated for a triangular patch: (a) and (b) show KDEs calculated based on 10^4 and 10^5 sample points, respectively; (c) shows SDE calculated analytically based on the triangular patch.

SDE analytically instead. The density field constructed for the same triangular patch using our method is shown in Figure 3.2(c). We can see that our method can generate more accurate results compared with KDE over discrete sample points. Also, by computing SDE analytically instead of performing KDE on a large number of sample points, the computation cost reduces. Our method transforms the calculation of SDEs into the calculation of integrals of the standard bivariate normal distribution over 2D triangular areas by parameterizing each triangular patch in 2D space as explained below.

3.2.2.1 Transforming Surface Density into Bivariate Normal Integral

The surface density for a triangle T is defined as a surface integral of the kernel $K_{\mathbf{H}}$ over T, as shown in Equation 3.5. By parameterizing the triangle in 2D space, this surface integral can be transformed into an ordinary integral over a 2D triangular area. To achieve this, we first transform T based on the target position $\mathbf{x}$ and the bandwidth matrix $\mathbf{H}$ by

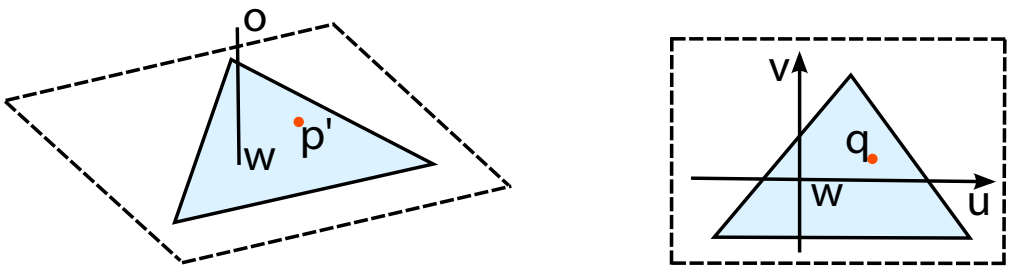
applying the transformation

$$\mathbf{p}' = \mathbf{H}^{-\frac{1}{2}}(\mathbf{x} - \mathbf{p}) \tag{3.7}$$

for each vertex $\mathbf{p} \in T$. After the transformation, the triangle T is transformed into a new triangle T', and the surface density of T becomes an integral of the kernel function K over the triangle T',

$$\tau_{\mathbf{H}}(\mathbf{x}, T) = c_T \iint_{T'} K(\mathbf{p}')\mathrm{d}T', \tag{3.8}$$

where $c_T = |\mathbf{H}|^{-\frac{1}{2}} \frac{\mathrm{d}T}{\mathrm{d}T'} = |\mathbf{H}|^{-\frac{1}{2}} \frac{Area(T)}{Area(T')}$. After the transformation, the surface density of T depends only on the kernel function K and the transformed triangle T'.



(a) (b)

Figure 3.3: Mapping a 3D triangle (a) into 2D (b) by parameterizing the triangle in the 2D space defined by the plane of the triangle. The origin **o** of the 3D space is projected onto the plane of the triangle at **w** that is treated as the origin of the 2D space. 2D equivalent of each point $\mathbf{p}'$ within the triangle is denoted as **q**.

We now parametrize the triangle T' in the 2D space and transform the surface integral over the triangle T' into the integral over a 2D triangular area. We first project the 3D origin **o** onto the plane of the triangle T' and use the projected point **w** as the origin of the 2D space, as shown in Figure 3.3(a). 2D equivalent of each point $\mathbf{p}' \in T'$ is then defined as $\mathbf{q} = (u, v)^T$ in $\mathbb{R}^2$ with w as the origin, as shown in Figure 3.3(b). Because the standard

multivariate normal kernel, which is rotation invariant, is used as the kernel function K, the integral of K over the triangle T' is transformed to the integral over a 2D region $\overline{T}$:

$$\tau_{\mathbf{H}}(\mathbf{x}, T) = c_T \iint_{\overline{T}} K((u, v, \lambda)^T) \mathrm{d}u\mathrm{d}v, \tag{3.9}$$

where $\lambda = \|\mathbf{w} - \mathbf{o}\|$. The standard multivariate normal kernel K is further decomposed into the product of lower dimensional standard normal distributions, and the integral of K over the 2D region $\overline{T}$ can be defined as a product of the density of a 1D standard normal distribution at position λ and an integral of the 2D standard normal distribution over the 2D region $\overline{T}$,

$$\tau_{\mathbf{H}}(\mathbf{x}, T) = c_T N(\lambda) \iint_{\overline{T}} N_2((u, v)^T) \mathrm{d}u\mathrm{d}v, \tag{3.10}$$

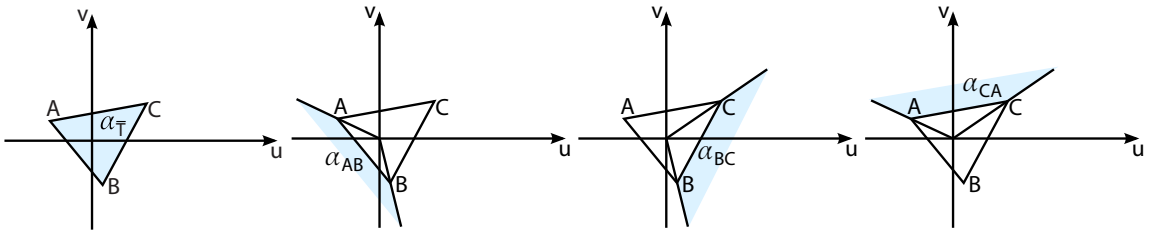
where $N(\lambda)$ and $N_2((u, v)^T)$ are the standard uni- and bivariate normal distributions, respectively.

3.2.2.2 Computing Bivariate Normal Integral

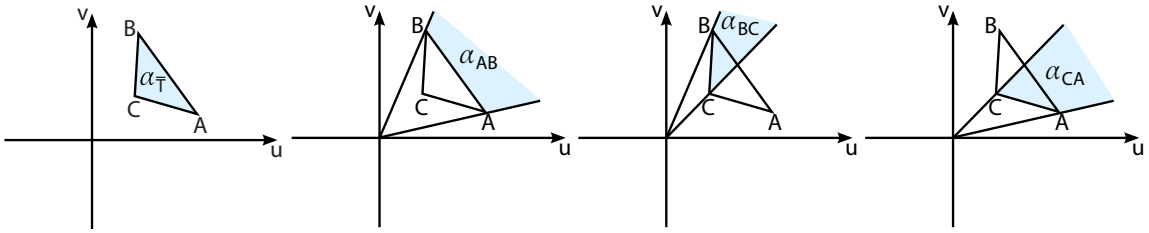
To calculate the bivariate normal integral over $\overline{T}$, denoted as $\alpha_{\overline{T}}$, we use the method proposed by Owen [109] and extended by Donnelly [42]. Let O be the origin and AB be an edge of a given triangle, Owen's method gives the bivariate normal integral over an area defined by line OA, OB, and beyond line AB, denoted as α_{AB}. (Details in Appendix A.)

To calculate the integral of the standard bivariate normal distribution over a triangle $\overline{T}$ defined by vertices A, B, and C, we first calculate integrals α_{AB}, α_{BC}, and α_{CA}, and then combine them to get the integral $\alpha_{\overline{T}}$. There are two cases for combining α_{AB}, α_{BC}, and α_{CA}, as illustrated in Figure 3.4. In Figure 3.4(a), the origin is inside the triangle, and the integral $\alpha_{\overline{T}}$ is computed with

$$\alpha_{\overline{T}} = 1 - \alpha_{AB} - \alpha_{BC} - \alpha_{CA}. \tag{3.11}$$



(a)



(b)

Figure 3.4: Computing the bivariate normal integral over a triangle $\overline{T}$, denoted as $\alpha_{\overline{T}}$. Integrals α_{AB}, α_{BC}, and α_{CA} are first computed based on the method proposed by Owen [109]. Then $\alpha_{\overline{T}}$ can be computed by combining α_{AB}, α_{BC}, and α_{CA}, for which there are two cases. (a) The origin is inside the triangle and (b) The origin is outside the triangle.

If the origin is outside the triangle, as shown in Figure 3.4(b), the integral $\alpha_{\overline{T}}$ is computed with

$$\alpha_{\overline{T}} = \alpha_{BC} + \alpha_{CA} - \alpha_{AB}. \tag{3.12}$$

3.3 Bandwidth Selection with Least Squares Cross Validation (LSCV)

We propose a method to guide the selection of the bandwidth based on the variability of ensemble surfaces. The crucial parameter for computing SDEs is the bandwidth matrix **H**, because it controls the degree and orientation of smoothing induced [152]. There

are three classes of the bandwidth matrix $\mathbf{H}$: positive scalars times the identity matrix $\mathbf{H} = \sigma^2\mathbf{I}$, which have the same amount of smoothing in all directions; diagonal matrices $\mathbf{H} = diag(\sigma_0^2, \sigma_1^2, \ldots, \sigma_{d-1}^2)$, with individual smoothing in each of the dimensions; and symmetric positive definite matrices, which allow arbitrary amount and orientation of smoothing. In this work, we use the first class of the bandwidth matrix $\mathbf{H} = \sigma^2\mathbf{I}$, which assumes a single smoothing for all directions that is controlled by a positive constant value σ^2. Note that σ may result in different scales in the physical space, if the grid coordinate system scales differently on each dimension.

To select a proper σ, we use the most common optimality criterion, which is the mean integrated squared error (MISE) between the SDE $\hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x})$ with respect to a bandwidth matrix $\mathbf{H}$ and the target probability density function $\mathscr{S}(\mathbf{x})$:

$$\mathrm{MISE}(\mathbf{H}) = \mathrm{E}\left[\int_{\mathbb{R}^3} (\hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x}) - \mathscr{S}(\mathbf{x}))^2 \,\mathrm{d}\mathbf{x}\right]. \tag{3.13}$$

Because the target probability density function $\mathscr{S}(\mathbf{x})$ is unknown, we use the LSCV method [25, 129] to approximate the MISE. The existing LSCV method is defined for a set of discretized data points $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$, which approximates the MISE as

$$\mathrm{LSCV}(\mathbf{H}) = \int_{\mathbb{R}^3} \hat{f}_{\mathbf{H}}(\mathbf{x})^2 \,\mathrm{d}\mathbf{x} - \frac{2}{n}\sum_{i=0}^{n-1} \hat{f}_{-i}(\mathbf{x}_i), \tag{3.14}$$

where $\hat{f}_{\mathbf{H}}(\mathbf{x})$ is the KDE at position $\mathbf{x}$, and $\hat{f}_{-i}(\mathbf{x}_i)$ is the leave-one-out estimator that estimates the density at position $\mathbf{x}_i$ for the input data points except $\mathbf{x}_i$:

$$\hat{f}_{-i}(\mathbf{x}_i) = \frac{1}{n-1}\sum_{j=0, j\neq i}^{n-1} K_{\mathbf{H}}(\mathbf{x}_i - \mathbf{x}_j). \tag{3.15}$$

We extend the LSCV for ensemble surfaces by redefining the leave-one-out estimator. For an ensemble of surfaces $S_0, S_1, \ldots, S_{N-1}$, we subsequently take out one surface S_i, then estimate the surface density for the remaining surfaces. Like LSCV, we take out one surface

S_i each time, then compute the leave-one-out estimator on points sampled from the surface S_i and accumulate the results. The LSCV for ensemble surfaces is then defined as

$$\mathrm{LSCV}(\mathbf{H}) = \int_{\mathbb{R}^3} \hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x})^2 \, \mathrm{d}\mathbf{x} - \frac{2}{A(S)} \sum_{i=0}^{N-1} \iint_{S_i} \hat{\mathscr{S}}_{-i}(\mathbf{p}) \mathrm{d}S_i. \tag{3.16}$$

To pick a proper bandwidth matrix based on the smoothing factor σ^2, we minimize the function $\mathrm{LSCV}(\mathbf{H})$ for σ values within the interval $(0, +\infty)$. To this end, we use the golden-section search method [75], which successively narrows the interval of σ in which the minimum of $\mathrm{LSCV}(\mathbf{H})$ occurs until the interval is small enough. In this work, we start from an interval $(0, a]$, which satisfies a condition that within $(0, a]$ we can find at least one σ that gives a $\mathrm{LSCV}(\mathbf{H})$ smaller than the $\mathrm{LSCV}(\mathbf{H})$ given by a.

When the ensemble surfaces have similar shape and spatial location, the leave-one-out estimator decreases more slowly than $\int \hat{\mathscr{S}}_{\mathbf{H}}(\mathbf{x})^2 \, \mathrm{d}x$ increases when the smoothing factor σ^2 of the bandwidth matrix decreases, because the other surfaces are close to the removed surface. On the other hand, if the ensemble surfaces are far away from each other, the leave-one-out estimator decreases rapidly when the smoothing factor σ^2 of the bandwidth matrix decreases. Because there is no analytic solution for the evaluation of $\mathrm{LSCV}(\mathbf{H})$, we evaluate it numerically based on Riemann sums in this work.

3.4 Visual Exploration of SDE

We design a visual exploration framework called *e*FESTA to visualize and analyze ensemble surfaces based on SDE. Based on the method presented in preceding sections, an ensemble of surfaces S_0, S_1, …, S_{N-1} can be transformed into a SDE, which represents the spatial distribution of the surfaces. The SDE is represented as a scalar field where the scalar value at a given grid point represents the probability density of the surfaces passing

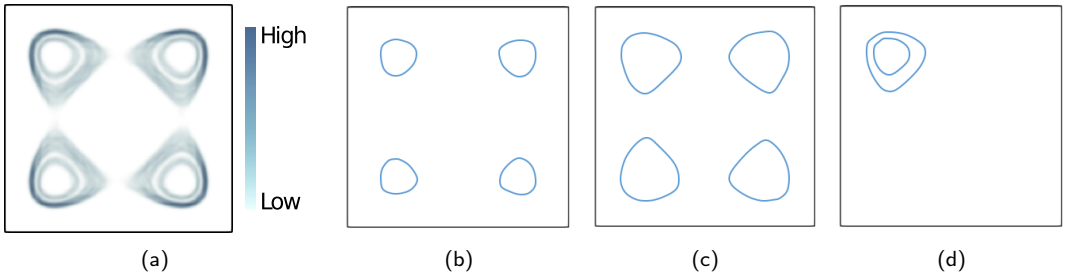


Figure 3.5: (a) Density field of an ensemble of isocontours in 2D. (b)-(d) Three possible shapes of the isocontours.

through it. By visualizing the SDE using existing volume exploration techniques such as volume rendering, the user is able to get an overview of how the surfaces are distributed in the spatial domain. However, the shape and the major trends of the surfaces require further analysis and exploration. For example, in Figure 3.5(a) the density estimation result for an ensemble of isocontours in 2D is visualized. Based on the visualization, it is difficult to determine the shape of the isocontours. Various possible shapes of the isocontours can be inferred from the visualization of the SDE, such as the three possible shapes shown in Figures 3.5(b)-(d). It is also difficult to get the percentage of a particular class of shape. To answer these questions, our framework transforms the surfaces S_0, S_1, ..., S_{N-1} to an ensemble of density fields D_0, D_1, ..., D_{N-1} based on their contribution to the SDE. In other words, the SDE equals to the sum of D_0, D_1, ..., D_{N-1}. We then organize the density fields into a hierarchical representation based on the similarity between them. The resulting hierarchical representation is used to guide visual exploration of the density fields as well as their corresponding surfaces. Below we describe the method to organize density fields hierarchically and the design considerations and choices of our interface.

3.4.1 Hierarchical Representation of Density Fields

For efficient visual exploration, we organize density fields into a hierarchical representation based on the similarity between them. We first compute a distance matrix containing the distances between every pair of the density fields and organize the density fields into a binary tree based on hierarchical clustering. Below are the key steps of constructing the hierarchical representation.

We treat each density field as a point in high-dimensional space $\mathbb{R}^m$, where m equals to the number of grid points in the density field. Then the distance between the density fields is computed as the l^2-norm distance in $\mathbb{R}^m$, and stored in a distance matrix. Based on the distance matrix, agglomerative hierarchical clustering is then performed to build a hierarchy of clusters (an unbalanced binary tree) of the density fields in a bottom-up manner. Starting with each density field in a separate cluster, the hierarchy is built by merging pairs of similar clusters until all density fields are contained in a single cluster. Distance between clusters is determined based on Ward's method [157], which calculates the distance between a pair of clusters as the change of the total sum of squares after merging. In this way, the sum of the distances between data points and the new cluster center after merging is minimized. After clustering, similar density fields are grouped into the same subtree, from which the major trends of the surfaces can be extracted and visualized by selecting subtrees.

3.4.2 Visual Exploration with Hierarchical Clustering Tree

We design a system for interactive visual exploration of density fields and their corresponding surfaces guided by the hierarchical clustering tree generated before. Our system consists of two linked views: a hierarchical clustering tree view and a spatial view.

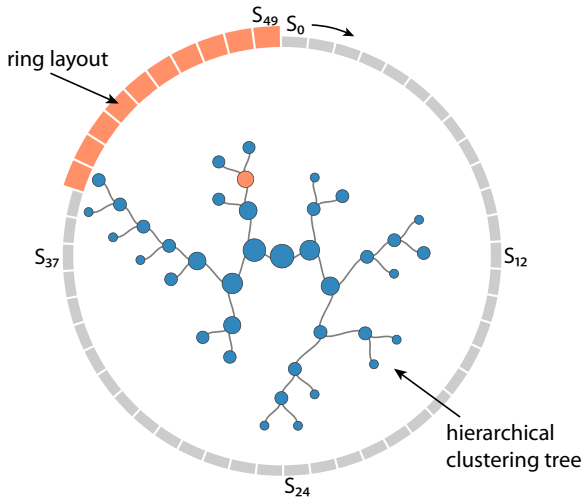


Figure 3.6: Hierarchical clustering tree view, where ring layout represents surfaces ordered by their IDs from S_0 to S_{N-1}. When a node is selected, the corresponding ring segments are enlarged and highlighted.

The hierarchical clustering tree view is provided to help users select and explore ensemble surfaces at different hierarchies. In this work, a radial tree layout is chosen to visualize the hierarchical clustering tree. Compared with a linear tree layout, the radial tree layout utilizes space more effectively. The root of the hierarchical clustering tree is placed at the center of the layout, and the distance of a node from the root encodes the level of the node. A ring layout is then drawn around the hierarchical clustering tree to indicate the surface features extracted from each ensemble member. The surface features are ordered based on their IDs in the ring layout. When the user selects a node of interest, the corresponding ring segments are simultaneously enlarged and highlighted. The user can also adjust the total number of leaf nodes (i.e., the maximum number of clusters in the hierarchical clustering) in the tree layout to reduce visual clutter. In this work, the total number of leaf nodes is set to 20. Figure 3.6 shows the visualization of an example hierarchical clustering tree view.

The spatial view has two subviews: a density field view and a surface view. The density field view is used to visualize a density field of interest, and the surface view shows the corresponding surfaces. After the user selects one node in the hierarchical clustering tree view, a density field is constructed on the fly by aggregating the density fields in the leaf nodes of the selected subtree; the density field view is updated interactively with direct volume rendering. We color the density fields using a single-hue colormap from dark green to light green, and the user can adjust the opacity transfer function to highlight regions with different density values. The surfaces of corresponding ensemble members are shown in the surface view as spaghetti plots. The surfaces are rendered with transparency such that the interior of the surfaces can be visualized. The spatial view is linked with the hierarchical clustering tree view. When the user hovers over a node of interest, the spatial view is simultaneously updated to visualize the density field and the surfaces of the selected ensemble members.

3.5 Implementation Details and Performance Evaluation

We parallelize the calculation of SDE over grid points using a GPU to handle millions of triangular patches over a density field discretized by millions of grid points. In our implementation, each GPU thread estimates the surface density at a grid point by iterating over all the triangles of the given surface. To further improve the performance of the SDE calculation, we filter out triangles whose contribution to the density at a target grid point is negligible. For each triangle, we first construct its axis-aligned bounding box and enlarge the bounding box for 4σ, where σ is the bandwidth. The reason for using 4σ is because the probability of a Gaussian distribution over the region within 4σ is greater than 99.99%, and we hence neglect the contribution of a Gaussian distribution to the density at a location

Table 3.1: Data specifications and performance (in seconds). t_d: timings for density estimation; t_c: timings for hierarchical clustering; t_r: response time for interactive queries (i.e., selecting a node in the hierarchical clustering tree).

Dataset	Application	Number of Members	Total Number of Triangles	Resolution of Density Field	Bandwidth σ	Performance (s)		
						t_d	t_c	t_r
Tangle	Isosurface	50	330,772	$249 \times 249 \times 249$	0.185	1.3×10^2	1.78	$3.6 \times 10^{-3} \sim 4.8 \times 10^{-2}$
MBST-98	Isosurface	600	307,044	$179 \times 105 \times 31$	0.38	0.9×10^1	2.26	$8.4 \times 10^{-4} \sim 6.1 \times 10^{-2}$
HRRR	LCS	100	4,773,666	$224 \times 132 \times 36$	0.67	1.9×10^2	1.37	$1.3 \times 10^{-3} \sim 2.7 \times 10^{-2}$
Square Cylinder	Streamsurface	40	43,636	$383 \times 127 \times 95$	0.3	0.8×10^1	0.82	$2.2 \times 10^{-3} \sim 1.3 \times 10^{-2}$

whose distance to the mean is greater than 4σ. For a given grid point, we test whether it is within the bounding box or not. If the grid point is within, we evaluate the surface density of the triangle over the grid point; if not, then we consider the triangle has zero contribution to the density at the grid point so no density estimation for that triangle is performed. The resulting density fields are stored as textures in the GPU memory for further visualization and exploration.

The prototype system is implemented with C++ and CUDA. All the experiments were performed on a workstation with two Intel Xeon E5620 CPUs (2.40 GHz), 12 GB memory, and an Nvidia Tesla K40c GPU. Table 3.1 reports the timings for the proposed ensemble exploration framework. The density estimation takes 1 to 3 minutes, and the hierarchical clustering takes a couple of seconds. The response time after the user selecting a node in the hierarchical clustering tree is less than 0.1 second.

3.6 Results

In this section, we first compare SDE with two alternative approaches using a synthetic ensemble of surfaces that have known spatial distributions in Section 3.6.1. The comparison shows that the proposed method generates more accurate density estimation results than the other two approaches when the computation costs are similar. Then we show the usefulness

and generality of SDE by applying it on ensemble surfaces for three applications, including ensemble isosurfaces in Section 3.6.2, LCSs in uncertain unsteady flows in Section 3.6.3, and streamsurfaces in ensemble fluid flows in Section 3.6.4. For all three applications, the proposed method is able to characterize how the surfaces are distributed in the spatial domain by only using polygon meshes. Moreover, for the application of ensemble isosurfaces, the proposed method is able to generate comparable results to previous methods that require scalar fields to be available.

3.6.1 Synthetic Ensemble Surfaces

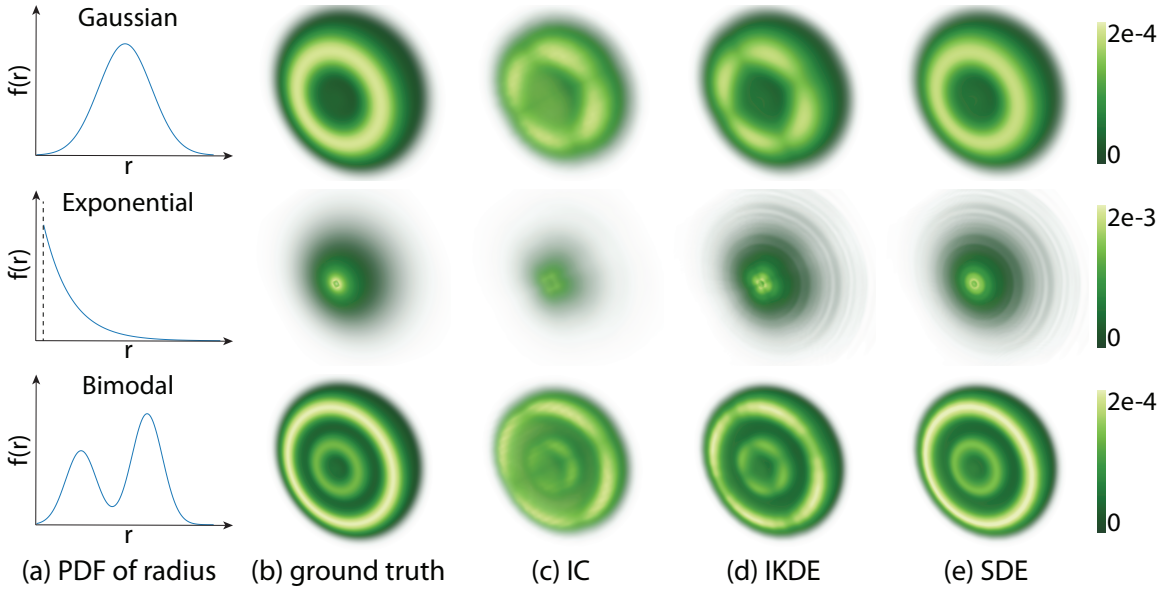


Figure 3.7: Visualization of the three synthetic ensembles of spheres: (a) distributions of radius; (b) ground truth density fields; (c)-(e) density estimation results of IC, IKDE, and our method, respectively.

In this experiment, we focused on evaluating the accuracy of the proposed density estimation method and comparing it with alternative approaches. To this end, we first synthesized

ensemble surfaces with respect to a predefined spatial distribution. Density estimation approaches were then performed on the generated surfaces. Finally, the resulting density fields were compared with the spatial distribution that was used to guide the generation of the surfaces.

The ensemble surfaces used for evaluation were based on a sphere centered at the origin with an uncertain radius. The uncertainty of the radius was regarded as a random variable R that was defined as a PDF $f(r)$. Three different $f(r)$ were tested in the experiment, including a Gaussian distribution, a shifted exponential distribution, and a bimodal distribution (i.e., a Gaussian mixture modeled with two Gaussian components) as shown in Figure 3.7(a). Based on $f(r)$, the probability density of a spatial location $\mathbf{x} \in \mathbb{R}^3$ on the sphere can be computed analytically as follows. We first compute the distance d from $\mathbf{x}$ to the center of the sphere and then evaluate $f(r)$ at d. In the end, we normalize the result such that the density values integrate to 1 for all $\mathbf{x} \in \mathbb{R}^3$. In the experiment, the density values were evaluated on the grid points of a regular grid with a spatial resolution of 256^3. The resulting density field was treated as the ground truth, as shown in Figure 3.7(b) using volume rendering. A clipping plane is placed at the center of the sphere to make the interior visible. An ensemble of spheres was then generated based on each $f(r)$. A set of radiuses was first sampled randomly based on $f(r)$, and then a sphere mesh was constructed for each radius by subdividing the sphere along θ and ϕ in spherical coordinates.

The SDE and two alternative density estimation approaches were applied on the ensemble of spheres, which are detailed as follows.

SDE The proposed bandwidth selection method was first applied on the ensemble of sphere meshes. Then the SDE was calculated based on the selected bandwidth.

Intersection count (IC) First, a regular grid was defined over the spheres. Then, the count of sphere meshes intersecting each grid cell was computed based on triangle-box intersection [45]. Finally, the count of each cell was normalized by the total count to estimate the density field.

Intersection-based KDE (IKDE) Similar to IC, a regular grid was defined over the sphere meshes first. Then, for each sphere mesh, we extracted active cells (i.e., grid cells that intersect the sphere meshes). Finally, we performed point-based KDE over the center of the active cells to estimate the density field. For comparison, the bandwidth used for KDE was the same as SDE.

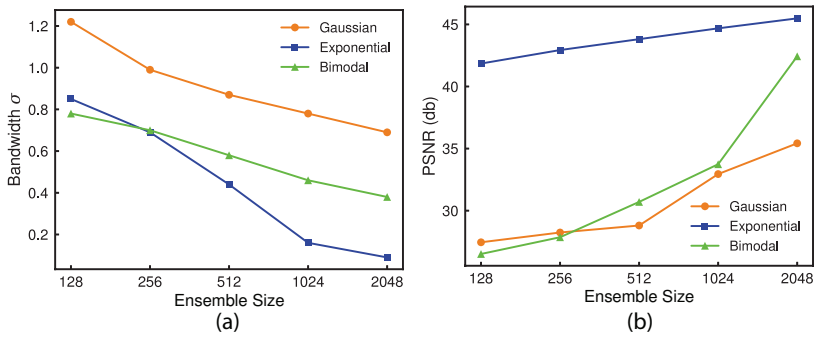


Figure 3.8: Quantitative analysis of the proposed method for the three synthetic ensembles of spheres with different ensemble sizes from 128 to 2,048 for every power of 2: (a) bandwidth σ suggested by the proposed bandwidth selection method, (b) PSNR (db) between the density estimation results of the proposed method and the ground truth.

The resulting density fields were then compared with the ground truth to evaluate the accuracy and performance of the three approaches.

We first evaluated the accuracy of the proposed method by applying it to ensembles of spheres with different ensemble sizes (i.e., number of ensemble members) from 128 to 2,048 for every power of 2. The SDEs were computed on a regular grid with a spatial resolution

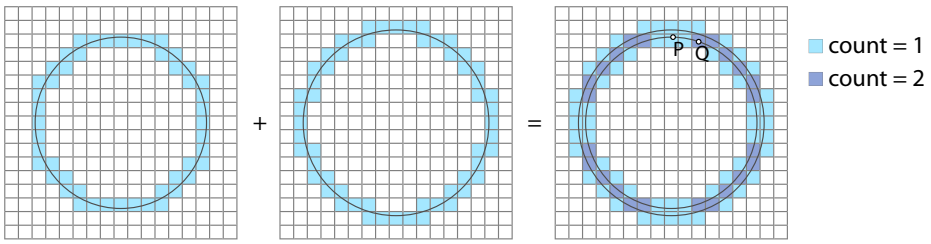


Figure 3.9: Example of counting the number of circles intersecting grid cells. Because of the discretization, errors are introduced into the resulting density field. For example, points P and Q should have the same density, but different densities are assigned to P and Q in the result.

of 256^3. The bandwidth selection results with respect to different ensemble sizes are shown in Figure 3.8(a), we can see that more ensemble members lead to a smaller bandwidth. Figure 3.8(b) shows the peak signal-to-noise ratio (PSNR) between the SDE and the ground truth. We can see that the PSNR increases when the ensemble size increases, because more samples can better fit the underlying distribution. Also, our method can generate density fields that are close to the ground truth. For example, the PSNRs are greater than 30db for all three $f(r)$ when the ensemble size is greater than 1,024.

We then compared the accuracy and performance of the proposed method with IC and IKDE. Because the grid resolution plays an important role in the accuracy of IC and IKDE, we compared the three approaches on different grid resolutions from 8^3 to 256^3 for every power of 2. In order to compare with the ground truth, the resulting density fields were then upsampled to 256^3 using trilinear interpolation. The ensemble size is fixed to 2,048 in this comparison. For all three approaches, the computation was parallelized over grid points using GPU.

The visualization of the resulting density fields generated by IC, IKDE, and our method are shown in Figures 3.7(c)-(e), respectively. The grid resolution of each approach in

Figure 3.7 is the resolution that corresponds to the best density estimation result, which are 32^3, 256^3, and 256^3 for IC, IKDE, and our method, respectively. We can see that the density fields generated by our method are visually similar to the ground truth, and undesirable artifacts are introduced into the density fields generated by IC and IKDE. This is because after discretizing surface patches with respect to a given grid, the information of the surface patches (e.g., location, orientation, and shape) within each grid cell is lost, which will introduce discretization error into the density estimation results. Figure 3.9 shows a 2D example of such discretization error. In this example, a density field is estimated for two circles using IC. We can see that different density values are assigned to spatial locations that should have the same density value (e.g., points P and Q). This discrepancy is caused by the discretization of space. Although increasing the grid resolution can reduce the discretization error, it will reduce the accuracy and performance of IC and IKDE, which are discussed in detail as follows.

The PSNRs between the density fields and the ground truth are shown in Figures 3.10(a), (c), and (e). Also, the performance of the three approaches are shown in Figures 3.10(b), (d), and (f). We can see that IC takes the least amount of time, but generates the worst results for all three $f(r)$. Also, the accuracy of IC increases in the beginning and starts decreasing when the grid resolution is greater than 32^3. This is because on one hand, the discretization error decreases as the grid resolution increases, and on the other hand, the size of grid cells is too small to capture the spatial distribution of the surfaces when the grid resolution is too high. More specifically, if the size of grid cells is too small, most grid cells only intersect individual surface patches once. As a result, the density field becomes sparse and noisy. IKDE does not suffer from this problem when the grid resolution is too high, because even most grid cells only intersect individual surface patch, the intersected cells

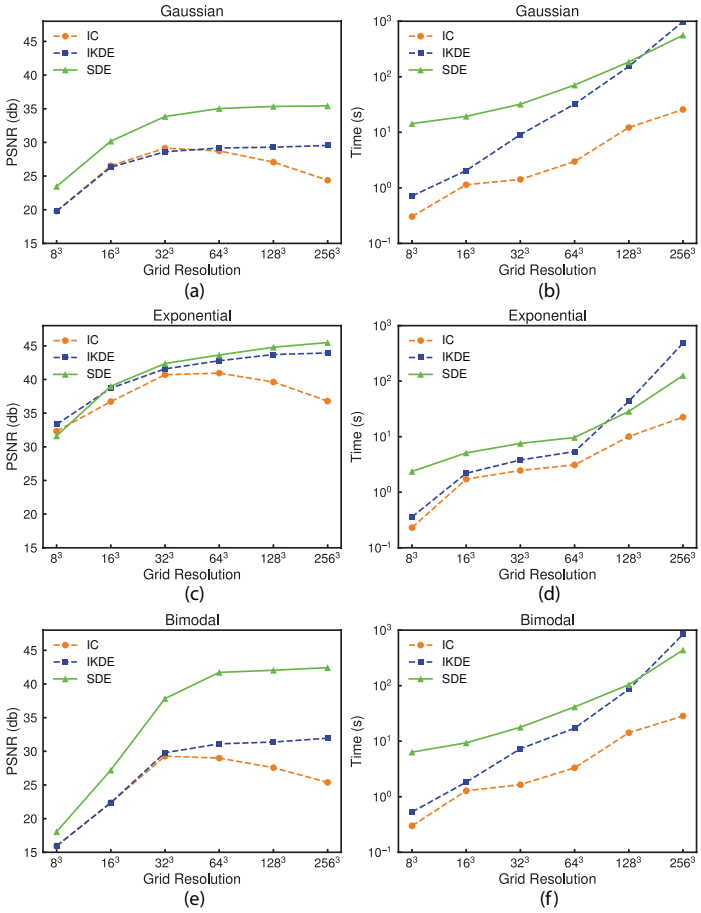


Figure 3.10: Quantitative comparison for IC, IKDE, and our method: (a), (c), and (e) PSNR (db) between the density estimation results and the ground truth; (b), (d), and (f) timings for density estimation.

can contribute density to neighboring cells by using KDE. Hence, increasing grid resolution can improve the accuracy of IKDE, as shown in Figures 3.10(a), (c), and (e). However, when the grid resolution increases, the number of active cells increases as well. As a result, IKDE needs to perform KDE on more points. In contrast, our method performs density estimation on the same number of triangular patches when the grid resolution changes. Hence, the computation cost of IKDE increases faster than our method when the grid

resolution increases, as shown in Figures 3.10(b), (d), and (f). When the grid resolution equals to 256^3, our method takes less time than IKDE and produces more accurate results.

3.6.2 Isosurfaces in Ensemble Scalar Fields

We applied the proposed method on two ensemble scalar fields: a synthetic ensemble dataset based on the tangle function [78] and a real-world ensemble simulation data in environmental science. For the synthetic dataset, we compared the proposed method with the state-of-the-art approaches on uncertain isosurface visualization.

3.6.2.1 Synthetic Ensemble Scalar Fields

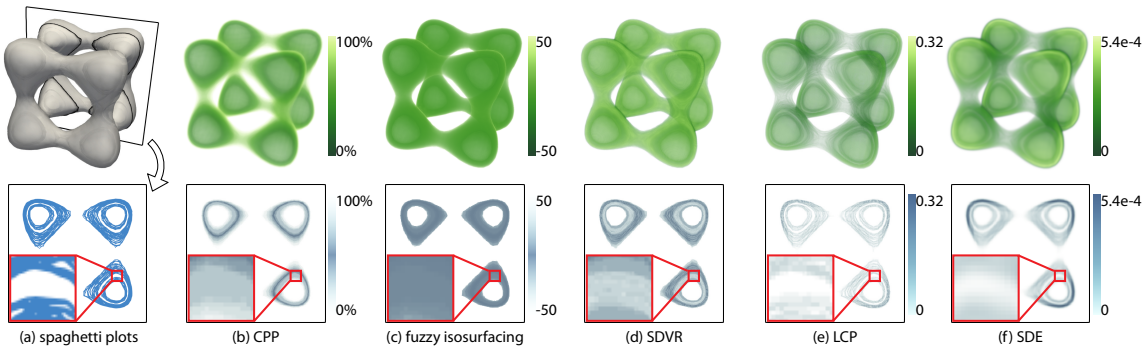


Figure 3.11: Visualization of the isosurfaces extracted from the synthetic ensemble scalar fields using different methods: (a) spaghetti plots, (b) CPP [80], (c) fuzzy isosurfacing [148], (d) SDVR [131], (e) LCP [119], and (f) SDE. Notice that different techniques quantify the positional uncertainty of isosurfaces with different metrics. Hence, the results have different value ranges and have to be visualized using specific transfer functions.

Our first experiment was performed on a synthetic ensemble dataset based on the tangle function. The ensemble was obtained by injecting multiple realizations of noise in the scalar field. The noise samples were generated by sampling 10 Gaussian kernels with shifted means and a fixed standard deviation 0.18, with 5 samples per kernel, to create an ensemble

scalar dataset with 50 members. Seven kernels were placed closer to the underlying value, and three kernels were placed farther away. An ensemble of isosurfaces was then extracted for the isovalue of -0.6. Figure 3.11(a) shows the spaghetti plots of the isosurfaces with one slice extracted and highlighted. The slice is defined by a plane at the location $(0,0,21.75)$ with a normal $(0,0,1)$.

When scalar fields are available, various techniques can be used to model the positional uncertainty of isosurfaces. In this experiment, we compared our method with four techniques that are working on uncertain scalar fields, including: contour probability plots (CPPs) [80], fuzzy isosurfacing [148], statistical direct volume rendering (SDVR) [131], and level crossing probabilities (LCPs) [119] detailed as follows.

CPP For each grid point, the CPP is defined as the percentage of scalar values that are greater than the isovalue, which is from 0% to 100%. Given an ensemble of scalar fields, the number of ensemble members that have scalar values greater than the isovalue is first computed at each grid point. Then, the counts are normalized by the total number of ensemble members to get CPPs. Notice that locations that are considered having a higher chance to be on the isosurface are the locations with CPPs around 50%. Hence, a triangle function centered at 50% is often used as the transfer function to render a CPP field.

Fuzzy Isosurfacing Similar to CPP, fuzzy isosurfacing first computes the number of scalar values above and below the isovalue at each grid point, denoted as a and b, respectively. The likelihood of the presence of the isosurface is then computed for each grid point as $g = \frac{a}{b} - \frac{b}{a}$. If $a = b$, g is set to zero; if $a = 0$ or $b = 0$, g is set to $-N$ and N, respectively, where N is the number of ensemble members. Because we have 50 ensemble members in this experiment, the value of g is between -50 and 50. Fuzzy isosurfacing is typically visualized using volume rendering with a transfer function defined by triangle function

centered at 0, because locations that are considered having a higher chance to be on the isosurface are the locations with $a = b$.

SDVR In SDVR, the distribution of scalar values at each grid point is modeled as box-splines, which enables interpolation of distributions at arbitrary location along the viewing rays. The transfer function is then integrated against the interpolated distribution at each sample point along the viewing rays to obtain the expected color and opacity. The expected colors and opacities are then blended to get the volume rendering results. To visualize uncertain isosurface of a give isovalue using SDVR, a triangle function centered at the isovalue is used as the transfer function.

LCP Unlike the aforementioned methods, the LCP is defined on grid cells instead of grid points. The LCP of a grid cell is defined as the probability of the isosurface intersecting the cell, which is from 0 to 1. To get the LCP of a grid cell defined by several grid points, the joint distribution is first constructed for the random variables defined on the grid points. The LCP is then approximated by sampling from the joint distribution and computing the percentage of samples that the isosurface intersecting the cell.

SDE In contrast with previous methods, SDE is computed based on the geometries of the isosurfaces. First, the proposed bandwidth selection method is applied on the isosurfaces, which suggests to use a bandwidth equals 0.185. Then, SDE is computed based on the selected bandwidth. Because SDE represents the PDF of the spatial distribution of the isosurfaces, SDE integrates to 1 in the spatial domain.

The volume rendering results of the CPP, Fuzzy Isosurfacing, SDVR, LCP, and SDE are shown in Figure 3.11(b)-(f), respectively. Compared with the isosurfaces shown in Figure 3.11(a), we can see that the results of CPP, Fuzzy Isosurfacing, and SDVR assign nonzero probabilities to locations that no isosurfaces are passing through (e.g., empty regions

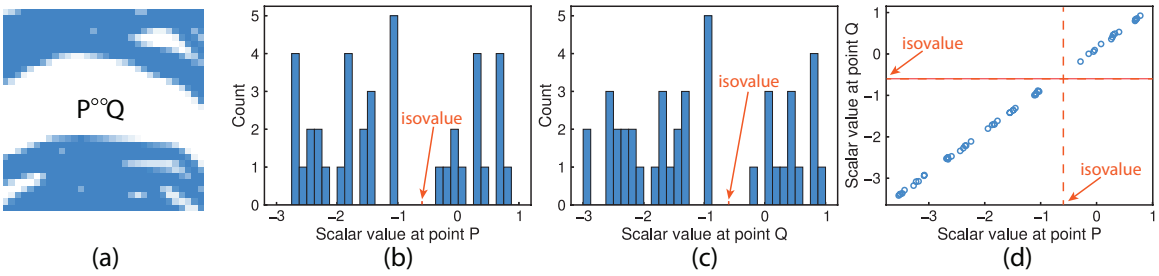


Figure 3.12: Example of correlation between random variables at different spatial locations: (a) locations P and Q that random variables are defined on, (b) histogram of scalar values at P, (c) histogram of scalar values at Q, and (d) scatter plot of scalar values at P and Q.

enlarged in Figure 3.11(a)). This is because the three techniques model the uncertainty of the scalar values at each grid point as an independent random variable without considering correlations between the random variables at different spatial locations. For example, for the adjacent points P and Q selected from the empty regions in Figure 3.12(a), the distributions of the scalar values at points P and Q are shown in Figure 3.12(b) and (c), respectively. If we model the uncertainty of the scalar values at the two points as independent random variables. The probability that the isosurface passing through any points between P and Q is nonzero, because it is possible that the scalar value at P is greater than the isovalue and the scalar value at Q is less than the isovalue. However, the random variables at P and Q are correlated. Figure 3.12(d) shows the scatter plots of the scalar values at P and Q. We can see that when the scalar value at P is greater than the isovalue, the scalar value at Q is also greater than the isovalue, and vice versa. Hence, the isosurface does passing through any points between P and Q. By considering the correlations of the random variables within each grid cell, the result of LCP model the positional uncertainty of isosurface more correctly. Because the SDE is computed based on isosurfaces that are extracted from each ensemble member independently, spatial correlations of scalar values within each ensemble member

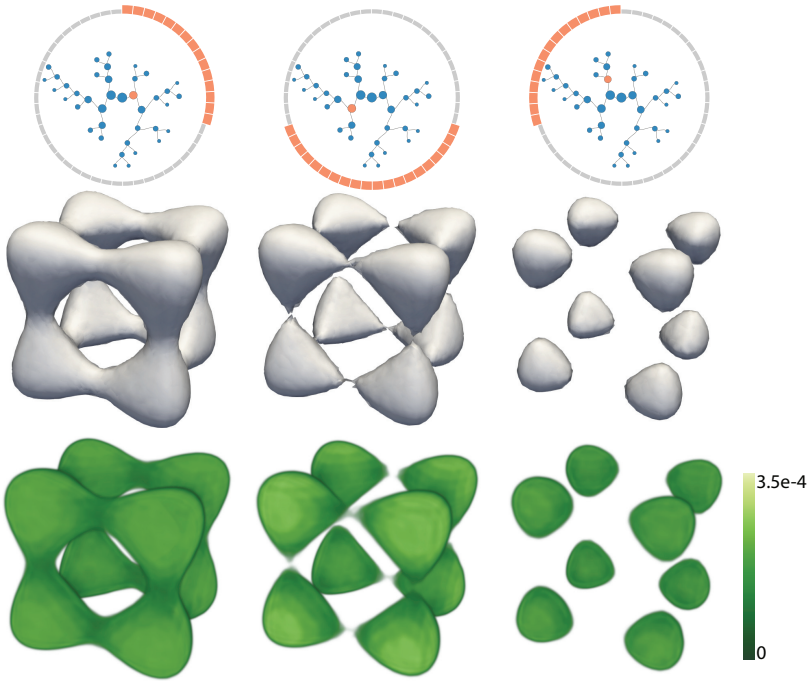


Figure 3.13: Visualization of the density fields and the underlying isosurfaces for three representative clusters selected from the hierarchical clustering tree for the synthetic ensemble scalar fields. The major trends of the isosurfaces are extracted and highlighted.

are preserved. Hence, the result of SDE is comparable to the result of LCP. However, the previous techniques require uncertain scalar fields as input to model the positional uncertainty of isosurfaces. For applications such as LCSs and streamsurfaces, no field data can be directly used to model the positional uncertainty of the surfaces. Unlike the previous techniques, our method is able to model the positional uncertainty of derived surfaces features, which can be applied on LCSs and streamsurfaces as discussed in the following sections 3.6.3 and 3.6.4. Moreover, unlike the previous techniques which only produce one final result for uncertain isosurfaces, our approach organizes ensemble isosurfaces into a hierarchical representation based on their similarities and enables interactive visual exploration of the major trends. Figure 3.13 shows the results of the proposed visual

exploration method. By hierarchical clustering, the isosurfaces are clustered based on the similarities between their contribution to the SDE, and a hierarchical clustering tree is visualized to guide the exploration of the isosurfaces. Three representative clusters of isosurfaces are shown in Figure 3.13, with each column representing one cluster. The contribution of each cluster to the SDE is represented as a density field and visualized using volume rendering. Compared with previous methods, the proposed method is able to extract and highlight the major trends of the isosurfaces.

3.6.2.2 Real-world Ensemble Simulation Data

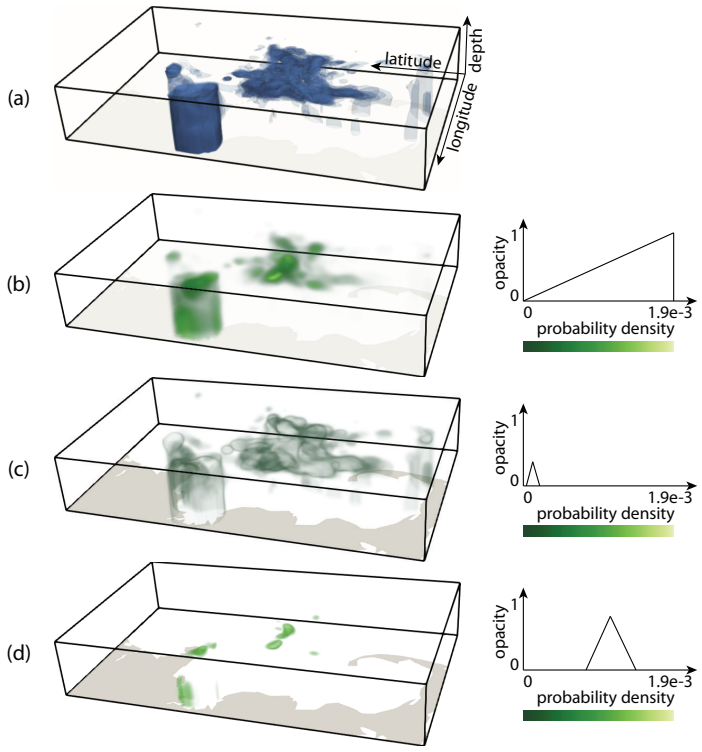


Figure 3.14: Visualization of the isosurfaces extracted from the CHL variable of the MBST-98 ensemble data with an isovalue 1.4 using (a) spaghetti plots and (b)-(d) SDE with different transfer functions. The vertical axis represents water depth in meters, which increases from bottom to top.

Our second experiment was performed on the Massachusetts Bay Sea Trial (MBST-98) ensemble dataset, which is an interdisciplinary (i.e., physical—-biogeochemical) forecast simulation based on the Littoral Ocean Observing and Predicting System [87, 128]. The simulation generated an ensemble of 600 members with more than 30 variables from August 17 to October 5, 1998 in Massachusetts Bay. In this experiment, we select the CHL variable, which represents the chlorophyll-a concentration, at the time step representing September 2, 1998. We extracted the isosurface at isovalue 1.4 mg/m^3 for each ensemble member, because phytoplankton are contaminated in regions with CHL equals 1.4 mg/m^3. The SDE is then computed for the extracted isosurfaces. The isosurfaces are visualized using spaghetti plots as shown in 3.14(a). By applying the proposed bandwidth selection method, the bandwidth σ was set to 0.38. The SDE is visualized using volume rendering with three different transfer functions as shown in Figure 3.14(b)-(d). A linear opacity transfer function is used in Figure 3.14(b). By comparing with the spaghetti plots, we can see that the SDE can reveal the distribution of the surfaces. Moreover, by adjusting the transfer function, the low and medium probability density regions can be highlighted in Figure 3.14(c)-(d). We can see that the low probability density regions are corresponding to the outline surfaces.

Figure 3.15 visualizes four selected clusters of isosurfaces based on their contribution to the SDE, where each row represents one cluster. The contribution of each cluster to the SDE is represented as a density field and visualized using volume rendering. Overall, the isosurfaces with chlorophyll-a concentration 1.4 mg/m^3 have a higher likelihood to be located at Boston Harbor and Stellwagen Bank compared with other regions. By comparing the density fields across clusters, distinct features are highlighted. For example, the density field in the second cluster reveals that the isosurfaces have lower likelihood at Boston Harbor compared with other clusters; and density fields in the first, third, and fourth clusters

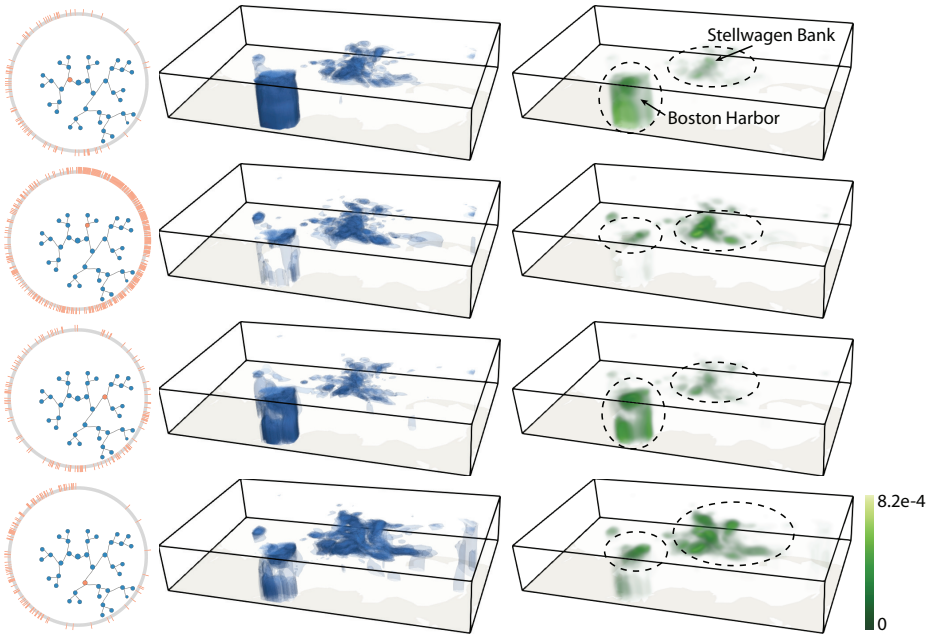


Figure 3.15: Visualization of the density fields for four representative clusters based on the isosurfaces generated from the CHL variable of the MBST-98 ensemble data with an isovalue 1.4. Distinct features are highlighted at Boston Harbor and Stellwagen Bank for the four clusters.

highlight different regions at Boston Harbor. Many other features can also be identified by exploring the density fields guided by the hierarchical clustering tree.

3.6.3 Lagrangian Coherent Structures in Uncertain Flow Fields

We also used the proposed technique for visualizing and analyzing uncertain LCSs [59] for uncertain unsteady flow fields. The LCSs are surfaces that represent the boundaries between attracting or repelling particles in unsteady flow fields. Because the positional uncertainty of LCSs can not be derived from the flow fields directly, Guo et al. [56] presented a technique to visualize and analyze uncertain LCSs by extracting an ensemble of LCSs and then constructing a density field through a density estimation method based on SPH

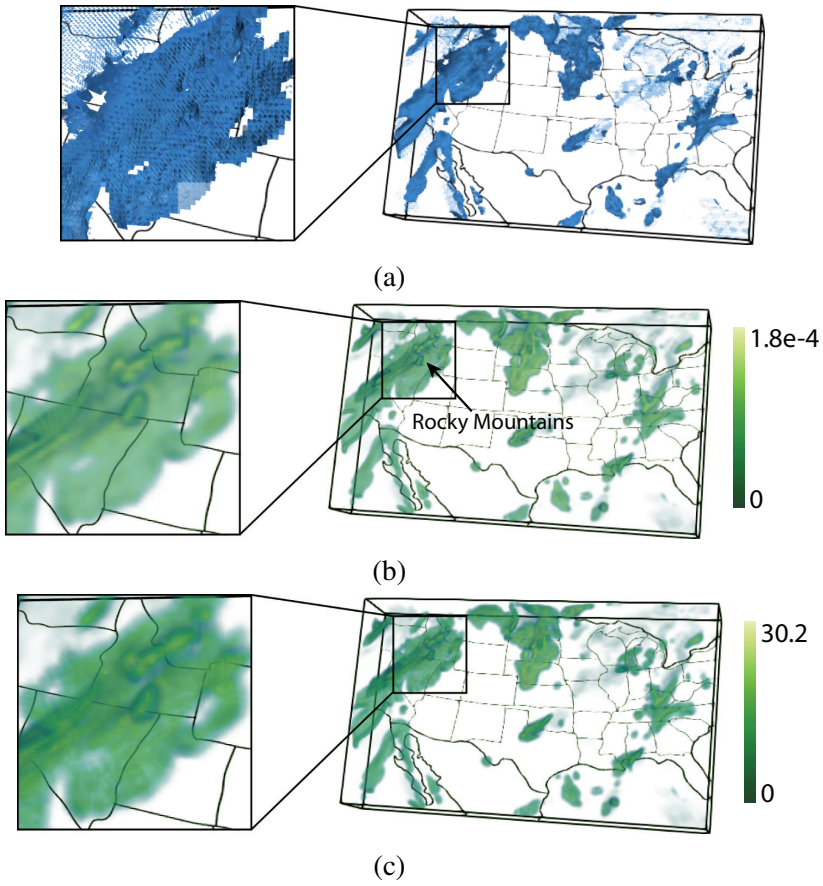


Figure 3.16: Visualization of the uncertain LCSs extracted from the HRRR ensemble dataset: (a) spaghetti plots of the surfaces, (b) SDE, and (c) density field created by SPH.

and zero-order kernel functions. They calculated the density of an arbitrary location **x** by calculating the area of the surfaces within a sphere centered on **x**, and then normalized the results by the volume of the sphere.

In this experiment, we calculated SDE for the uncertain LCSs extracted from High-Resolution Rapid Refresh (HRRR) data [6] and compared with SPH. The HRRR is a National Oceanic and Atmospheric Administration real-time atmospheric model. The HRRR is updated hourly and produces a forecast up to 16 hours. It is available to the public through the Unidata's THREDDS Data Server 2. We used an ensemble of the

HRRR simulation outputs to extract uncertain LCSs, and we used the means and standard deviations of the wind field on each grid point across the ensemble members to model the uncertainty. Then we extracted uncertain LCSs through Monte Carlo particle tracing, finite-time Lyapunov exponents computation, and ridge detection as presented in [56]. The starting time of particle tracing was 00:00:00 UTC, August 27, 2015, and the advection time was 5 hours. Figures 3.16(b) and (c) compare the visualization results of SDE and SPH with the spaghetti plots shown in Figure 3.16(a) as reference. The radius of the sphere was set to 2 for SPH. To compare with SPH, the bandwidth σ was set to 0.67 for SDE calculation, such that the points on the surface with distances to the target location greater than 2 have negligible contribution to the density estimation result. In contrast with SPH, we use higher-order kernel functions to calculate surface densities, which gives more smooth results. Based on the visualization of SDE, regions near the Rocky Mountains that with upward and downward motions have higher likelihoods containing LCSs than other regions.

3.6.4 Streamsurfaces in Ensemble Flow Fields

We use SDE to explore streamsurfaces generated from a synthetic ensemble fluid flow dataset based on a 3D flow around a confined square cylinder [31]. The ensemble is obtained by injecting multiple realizations of noise in the vector field along the y direction. The noise samples are generated by sampling 8 Gaussian kernels with shifted means and standard deviation 0.01, with 5 samples per kernel, to create an ensemble vector dataset with 40 members. An ensemble of streamsurfaces are then generated for a seeding curve close to the square cylinder. By applying the proposed bandwidth selection method, the bandwidth σ was set to 0.3. Figure 3.17 shows the visualization of the density estimation results generated based on the streamsurfaces. The streamsurfaces are visualized using

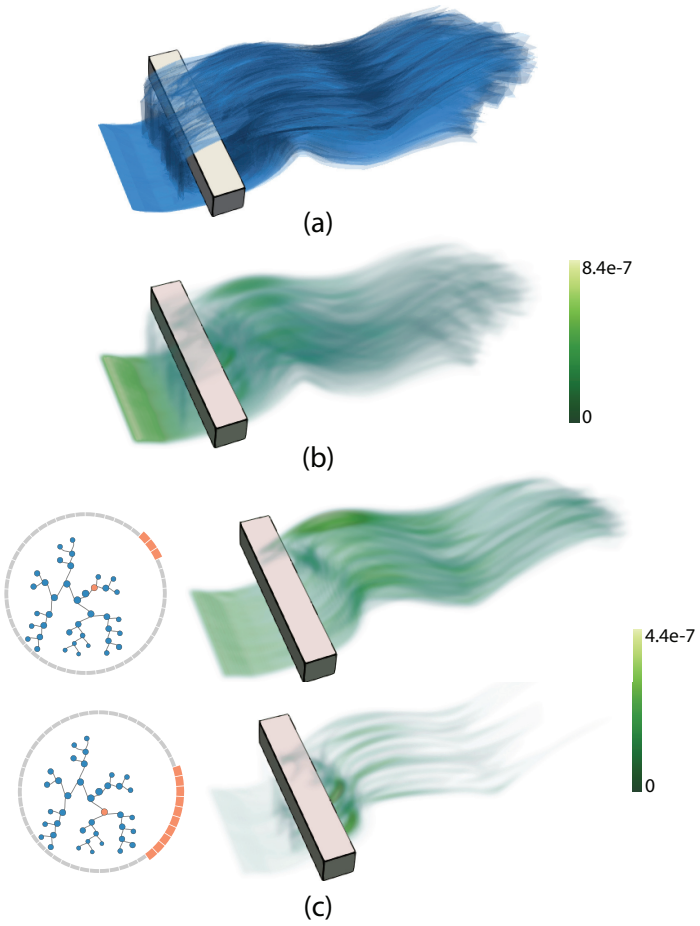


Figure 3.17: Visualization of the streamsurfaces generated from the synthetic ensemble flow around a confined square cylinder, where the flow is moving from left to right: (a) spaghetti plots of the streamsurfaces (b) volume rendering of the SDE, and (c) two clusters of density fields that highlight different flow features.

spaghetti plots as shown in Figure 3.17(a), and the SDE is visualized in Figure 3.17(b). Compared with the streamsurfaces, we find that the SDE clearly reveals the variability of the streamsurfaces. Before the particles reach the square cylinder, the streamsurfaces are close to each other, shown in the SDE as high surface densities. After the particles reach the square cylinder, the flows are separated by the square cylinder, also highlighted in the SDE. The visualization of the SDE also highlights regions near the square cylinder on the

right with relatively higher surface densities than regions far away from the square cylinder on the right, which can not be shown by overlaid rendering of the streamsurfaces due to occlusion. Figure 3.17(c) visualizes two clusters of streamsurfaces as density fields selected in the hierarchical clustering tree, which highlight distinct flow features.

3.7 Conclusion, Limitations, and Future Work

In this work, we presented SDE to model positional uncertainty of surface features in 3D ensemble simulation data. SDE is computed based on surfaces represented as polygon meshes with no need of field datasets. Hence, our method can be applied on applications that no field datasets are available to model the positional uncertainty of the surfaces. We also presented a bandwidth selection method for SDE computation. Major trends and outliers of ensemble surfaces can also be extracted based on SDE. To this end, we transformed ensemble surfaces into density fields based on their contribution to SDE and organized the resulting density fields into a hierarchical representation based on their pairwise distances. We compared the proposed method with alternative density estimation methods in terms of accuracy and performance on synthetic ensemble surfaces that have known spatial distributions. We then demonstrated the effectiveness and usefulness of the proposed method for variate applications.

There are a few limitations of our approach that we would like to address in the future. First, we would like to reduce the computation time of SDE for interactive bandwidth selection and exploration. Second, the bandwidth selection method is limited to single smoothing for all directions currently. We would like to extend it to handle more complex bandwidth matrices. Third, the distance between density fields for clustering is limited to point-wise distance metrics for density values over the entire domain. In the future, we

would like to study different distance metrics as well as considering sub-domains instead of the entire domain for clustering.

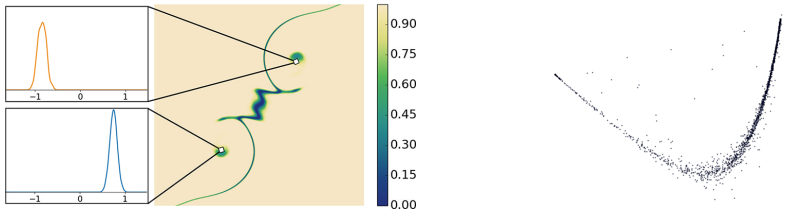
Chapter 4: Range Likelihood Tree: A Compact and Effective Representation for Visual Exploration of Uncertain Datasets

Understanding uncertainty is one of the major scientific challenges in the era of big data. It has a great influence on many applications such as weather forecast and analysis of fluid flows [14, 34, 37, 56]. As computing power continues to grow, state of the art simulations are able to model uncertainty with increasing complexities. For instance, to help estimate climate changes, scientists perform ensemble simulations with multiple parameterizations and stochastic initial conditions to study different outcomes and analyze the inherent uncertainty in the simulations. Uncertainty quantification and visualization play a fundamental role in validating such simulation outcomes and understanding the underlying scientific phenomena [14, 34, 37]. To visualize uncertainty, one common choice is to model uncertainty at each spatial location as a probability distribution of possible simulation outcomes and form a *probability distribution field* in the entire domain [88, 91, 119, 148]. However, with the increasing complexity of uncertainty (e.g., from a single gaussian to a multi-modal distribution), visual exploration of scalar uncertain data sets is a challenging task because the underlying features are often complex, and the data associated with each grid point are high dimensional. Consequently, conventional scalar field visualization techniques such as isocontour extraction and direct volume rendering are not readily applicable.

The visualization community has made a continuing effort to tackle the challenge of visualization and visual analysis of probability distribution fields [8, 9, 14, 34, 91, 119, 121]. Existing research is mostly focusing on the use of *statistical summaries* and *dissimilarity measures*. Common statistical summaries such as mean and standard deviation are among the most straightforward approaches, but they fail to reveal uncertain behaviors modeled by bimodal or multi-modal distributions [14, 34]. For classification purpose, one can specify a target distribution as a feature of interest, and classify the distribution field based on how dissimilar a distribution is to the target [8, 9, 91] (Figure 4.1(a)). However, ambiguities may occur when dissimilar distributions have similar distances with respect to the target [8, 9]. In addition, due to the ever increasing complexity of scientific phenomena, it is often difficiult to define distributions of interest in a precise manner. While automatic cluster analysis can work in the absence of precise target definition, meaningful clusters of probability distributions may not be easily obtained from a large number of probability distributions of complex characteristics. The choice of an appropriate dissimilarity measure is also non-trivial and often domain-specific [34]. Furthermore, low-dimensional embedding that respects the dissimilarities among probability distributions, using common projections techniques such as principle component analysis (PCA) and multi-dimensional scaling (MDS), can generate results that are difficult to understand (Figure 4.1(b)).

In this work, we present a compact and effective representation called *Range Likelihood Tree (RLT)*, to summarize and explore probability distribution fields. The key idea is to consider the different roles that *subranges* (subspaces of the value domain) may play in understanding probability distributions, and decompose and summarize each complex probability distribution over a few representative subranges by cumulative probabilities. In our method, the value domain of the entire field is first partitioned into small subranges, and

the distribution at each grid point is transformed based on these subranges. For each subrange, a *range likelihood field (RLF)* is formed where the scalar value at each grid point is computed as a cumulative probability of the point's distribution within the subrange. Based on how these cumulative probabilities are distributed spatially in the grid points, RLFs are organized into a hierarchical representation. The use of the new RLT representation allows effective classification and identification of features through user query and exploration. We present an exploration framework with multiple interactive views to explore probability distribution fields through RLFs, and provide guidelines for visual exploration using our framework. We show that RLTs can assist users to progressively gain knowledge by exploring RLFs, examining the corresponding probability distribution field, and classifying the field through interactive transfer function manipulation. We demonstrate the effectiveness and usefulness of our approach in exploratory analysis using several representative uncertain data sets, and verify the visualization results with domain scientists in environment science.



(a) (b)

Figure 4.1: Visual analysis of probability distribution fields using conventional techniques. (a) Visualizing a distance field constructed from a 2D probability distribution field; regions associated with two distinct distributions can not be readily distinguished from their distances to the target. (b) Visualizing a low-dimensional embedding of a 3D probability distribution field using principal component analysis (PCA); no clear separation of clusters can be found.

4.1 Overview

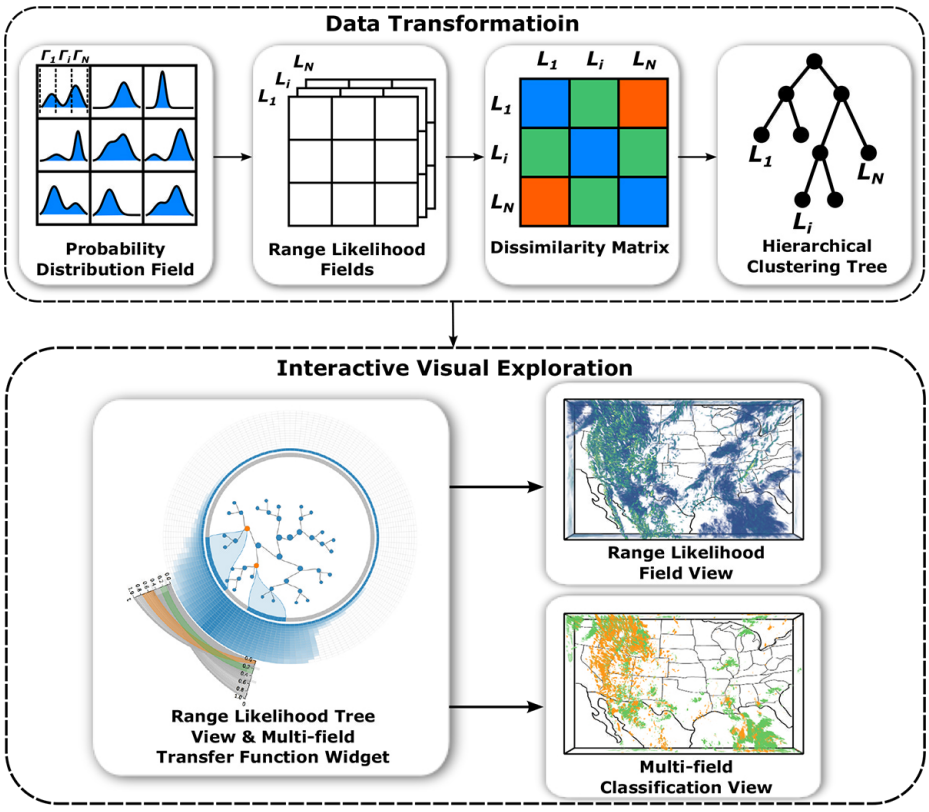


Figure 4.2: Overview of the analytical workflow.

The main objective of this work is to guide visual exploration of uncertain data sets modeled by probability distributions. Essentially, we transform the problem of distribution-based uncertain data exploration to the problem of exploring a set of subranges that are more intuitive and explanatory. Our system has two major modules: the data transformation module and the interactive visualization module, as shown in Figure 4.2. Given a probability distribution field, the value domain is first partitioned into subranges, and the distribution at each grid point is transformed according to the cumulative probabilities of the point's

distribution in those subranges. For each subrange, a *range likelihood field (RLF)* is formed where the scalar value at each grid point is computed as a cumulative probability of the point's distribution within that subrange. Based on these cumulative probabilities in each RLF, we compute a distance matrix of RLFs, and organize them into a hierarchical representation, called *Range Likelihood Tree (RLT)*, which group similar RLFs into clusters for efficient visual exploration. The resulting RLT from the data transformation module is then used in the visualization module for visual exploration and classification of the underlying probability distribution field.

4.2 Methods

In this section, we first introduce the concept of range likelihood which transforms a probability distribution into multiple cumulative probabilities over several subranges. Next, we describe how to organize the subranges using a multi-level data model called *Range Likelihood Tree (RLT)*. Finally, we discuss how to classify a probability distribution field using RLT.

4.2.1 Transforming Distribution into Range Likelihoods

A probability distribution field PF assigns every point p in Euclidean space with a probability density function:

$$PF(p) \sim f_X(p,x) \tag{4.1}$$

where X is a random variable over the value domain D.

Since each grid point is associated with a probability distribution, exploring all distributions in a probability distribution field with different shapes and modalities is difficult. To obtain an effective overview of a complex and heterogeneous probability distribution field,

Figure 4.3: Three probability distributions from different spatial locations in an uncertain scalar field. They are summarized by two subranges (in two distinct colors) with the probabilities that each distribution falls within the subranges as $[0.7, 0.3]$, $[0.5, 0.5]$, and $[0.3, 0.7]$, respectively.

a probability distribution can be summarized using simpler cumulative probabilities over several subranges. For example, probability distributions in Figure 4.3 can be represented by two subranges $[0, 0.5)$ and $[0.5, 1]$ with cumulative probabilities as $[0.7, 0.3]$, $[0.5, 0.5]$, and $[0.3, 0.7]$, respectively. Formally, given N subranges $\{\Gamma_1, \Gamma_2, ..., \Gamma_N\}$ that partition the value domain D of a probability distribution field, the distribution at each grid point is transformed into these subranges. For each subrange Γ_i, a *range likelihood field (RLF)* L_X is formed where the scalar value at each grid point p is computed as a cumulative probability of the point's distribution $f_X(p,x)$ within the respective subrange:

$$L_X(p;\Gamma_i) = \int_{\Gamma_i} f_X(p;x)dx, i = \{1, ..., N\}. \tag{4.2}$$

We denote the transformation from a probability distribution into a range likelihood field as *range likelihood transformation*.

Essentially, a RLF is a univariate scalar field, which can be visualized through isocontour extraction or direct volume rendering. Through range likelihood transformation of the distribution at every grid point p, a probability distribution field can be represented by

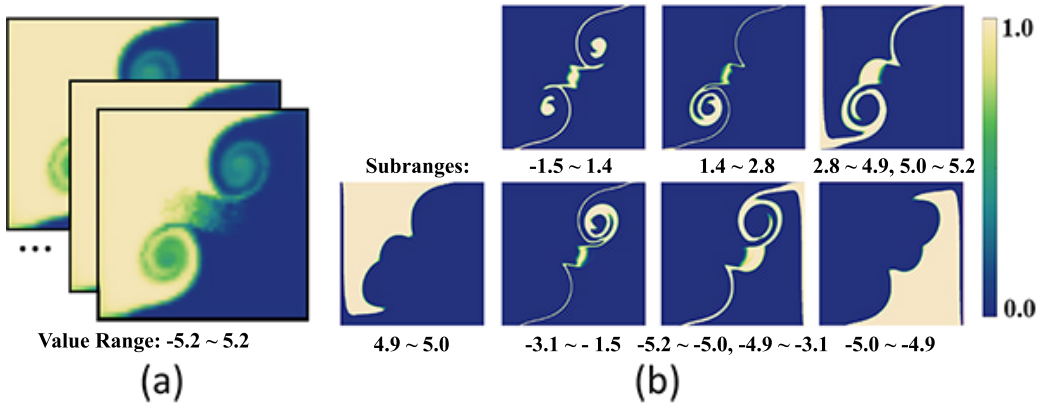


Figure 4.4: Transforming distribution into range likelihoods. (a) Visualization of scalar fields sampled from a 2D distribution field. (b) Visualization of range likelihood fields (RLFs) transformed from the distribution field.

multiple RLFs (forming a *multi-field*), while each RLF corresponds to the cumulative probabilities of a subrange that may highlight a salient feature from the distribution field. As shown in Figure 4.4, a probability distribution field (visualized as several sampled scalar fields in Figure 4.4(a)) is transformed into several representative RLFs of different ranges (Figure 4.4(b)). Such range likelihood transformation of a distribution field has several key benefits:

1. Each RLF indicates a set of grid points of non-zero likelihoods in its corresponding subrange, which may represent a particular feature. The corresponding set of grid points in the distribution field can be quickly retrieved given a RLF.

2. Associating visualization results with subranges allows users to consider the roles that different subranges play in understanding the probability distributions and to explore and relate a feature to the domain knowledge.

3. Given multiple RLFs, every grid point is associated with multiple likelihoods in different subranges, which can be flexibly combined to classify the distribution field for feature identification.

4.2.2 A Multi-level Data Model for Range Likelihood Fields

Since a value domain D can be partitioned into subranges in a number of ways, an appropriate partitioning is critical to the effectiveness of range likelihood transformation. A simple and popular approach is to uniformly partition the value domain into N subranges of equal sizes (e.g., histograms with equal sizes of bins). However, a more effective partitioning should take the input data into consideration, as features may not be uniformly distributed in the value domain. As a result, when the domain is not partitioned fine enough (N is too small), distinct features may not be separated; on the other hand, when the domain is overly partitioned (N is too large), features may be broken into small fragments, and storage overhead may become too large as each subrange corresponds to a range likelihood field.

To provide an effective partition of the value domain, we take a divide-and-conquer approach: (1) the value domain is first partitioned into N subranges, and the distribution at each grid point is transformed into its subranges; the initial value of N is set as 256 to balance the accuracy and storage overhead; (2) a distance matrix is computed between every pair of the N RLFs based on a similarity measure; (3) the N RLFs are organized into a binary tree using hierarchical clustering; and (4) a compact tree representation is produced after merging similar RLF clusters. We call this data representation *Range Likelihood Tree (RLT)*. Below we describe the key steps to construct such a representation.

Measuring similarity of RLFs Since each grid point has a likelihood in a given RLF, we consider two RLFs *similar* if their associated grid points of non-zero likelihoods are

spatially similar (illustrated in Figure 4.5). Hence we consider RLFs as distributions with *spatial location* being the random variable for comparison:

$$\hat{L_X}(p;\Gamma_i) = \frac{L_X(p;\Gamma_i)}{\int_R L_X(p;\Gamma_i)dp}, \tag{4.3}$$

which scales the likelihood at a spatial location p for subrange Γ_i by the total sum of the likelihoods over the entire spatial domain R. To compute the distance between two distributions, we employ a statistical measure *Jensen–Shannon divergence (JSD)* [32].

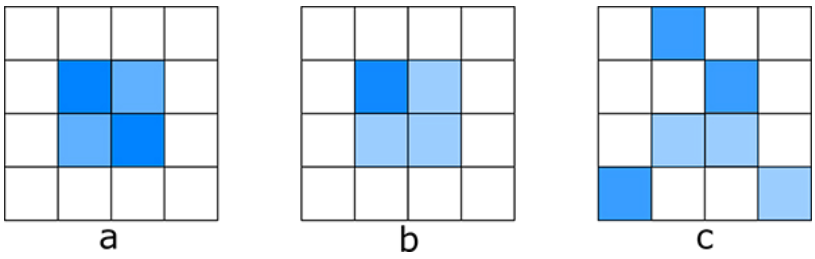


Figure 4.5: An illustration of three range likelihood fields (RLFs), highlighting grid points with non-zero likelihoods. Based on spatial locality of grid points, RLF-a is similar to RLF-b, while RLF-c is different from RLF-a and RLF-b.

Hierarchical Clustering of RLFs Based on the similarity measure, we compute a distance matrix for every pair of RLFs. Agglomerative hierarchical clustering is then applied to the distance matrix to create an unbalanced, binary clustering tree of RLFs in a bottom-up manner. Starting with each RLF in a separate cluster, pairs of resulting clusters are merged successively until all RLFs are contained in one large cluster. Distance between clusters is determined based on average distance of all pairs of RLFs between clusters. Using agglomerative hierarchical clustering, the number of clusters is not required beforehand and the clustering tree can be split easily into the desired number of clusters. In this way, RLFs

of similar features can be clustered into one subtree, while RLFs of distinct features will be organized into different branches.

Pruning of Hierarchical Clustering Tree As a result of hierarchical clustering, a hierarchical clustering tree of RLFs is constructed, which is called *Range Likelihood Tree (RLT)*. Each node of the tree contains a cluster of RLFs and RLF clusters at a higher level are more distinct than those at a deeper level. To keep a compact representation of the underlying probability distribution field with representative RLFs, the hierarchical-clustering tree is further pruned by merging similar RLF clusters. Starting from the root, for each node, we evaluate the dissimilarity between its child nodes. If the dissimilarity is below a predefined threshold D_t, nodes in the subtree are reduced into one node and the RLFs in the leaf nodes $\{L_1, L_2, ..., L_n\}$ of this subtree are merged into one RLF L based on:

$$L(p) = \sum_{i=1}^{n} L_i(p) \tag{4.4}$$

If the dissimilarity is above the threshold D_t, then we keep this node and apply the same operation on its child nodes. The pruning of a RLT also reduces the storage cost because fewer RLFs needs to be kept to represent the underlying probability distribution field.

4.2.3 Range Likelihood Based Probabilistic Classification

By selecting multiple RLFs from the RLT, every grid position p is now associated with a *feature vector* of likelihoods $\vec{l}$, where each element in this vector represents the cumulative probability of the distribution at position p over a subrange. The problem of probability distribution field classification is then transformed to the problem of multi-field classification. To enable visual classification of multi-field data, we start with a set of user selected feature vectors and cluster them into several clusters. Each cluster is represented as a mean vector and a covariance matrix. Then these clusters are used to generate a multi-field transfer

function, which is later used to assign the color and opacity to each feature vector hence the corresponding voxel of the data.

To cluster user-selected feature vectors which can be treated as points in multi-dimensional space, we use a Gaussian mixture model (GMM) to find clusters as well as their mean vectors and covariance matrices. Since the number of clusters could change for different user-selected feature vectors, instead of using the expectation maximization (EM) algorithm which requires to set number of clusters before clustering, we employ the Minimum Volume Ellipsoid Estimator (MVE) method [74] to automatically cluster feature vectors into an optimal number of clusters. It partitions the multi-field space into candidate clusters, and iteratively performs the Kolmogorov-Smirnov test to find the best fitting clusters, while each cluster is fit into a Gaussian function. Finally, the user-selected feature vectors are fit by a GMM and the membership of a feature vector for one Gaussian cluster is evaluated by:

$$G(\vec{l},\vec{\mu},\Sigma) = e^{-\frac{1}{2}(\vec{l}-\vec{\mu})'\Sigma^{-1}(\vec{l}-\vec{\mu})} \tag{4.5}$$

where $\vec{l}$ is a feature vector, $\vec{\mu}$ is the mean vector that the Gaussian is centered over, and Σ is the covariance matrix of the Gaussian.

The fitted GMM is then used to guide the visual classification of the multi-field data by treating each Gaussian component of the GMM as an Ellipsoid Gaussian transfer function [57, 77, 156], which maps a given feature vector $\vec{l}$ with an opacity α:

$$\alpha(\vec{l}) = \alpha_{max} G(\vec{l},\vec{\mu},\Sigma) \tag{4.6}$$

where $\vec{\mu}$ and Σ is a mean vector and a covariance matrix of a Gaussian component of the fitted GMM. α_{max} is a constant to scale the Gaussian transfer function. The final color C and the opacity α are evaluated by combining multiple Gaussian transfer functions together:

$$C = \frac{\sum \alpha_i C_i}{\sum \alpha_i} \quad and \quad \alpha = \sum \alpha_i \tag{4.7}$$

where C_i and α_i are the color and opacity generated by the i-th Gaussian transfer function.

4.3 Range Likelihood Tree Guided Exploration Framework

Modeling probability distribution fields as range likelihood trees, we transform the problem of distribution-based uncertain data exploration to the problem of exploring the cumulative probabilities in a set of value ranges that are meaningful to users. We develop a RLT guided exploration framework with interactive visualization techniques for exploring probability distribution fields. In this section, we describe the design considerations and choices of our interface, and provide guidelines for visual exploration using our framework.

4.3.1 User Interface

The user interface consists of three components: the range likelihood tree (RLT) view, the multi-field transfer function widget, and the spatial view. We provide design details for each component as follows.

4.3.1.1 Range Likelihood Tree View

The RLT view is to facilitate users in understanding the underlying probability distribution field through visual exploration of ranges at different hierarchies. To achieve an efficient and tidy arrangement of RLT while illustrating the distributions regarding selected ranges, we employ a composite radial visualization (Figure 4.6), which consists of two-layered radial visualizations: a RLT visualization in the center, surrounded by a density visualization of distributions over the entire value domain.

RLT Layout. To provide a compact visualization of RLT, a radial tree layout is chosen in this work. Compared with a linear tree layout, the radial tree layout is more compact and

with shorter distance between each node and its corresponding value range. The Reingold-Tilford drawing algorithm [30] is employed to create the radial tree layout. In our design, the root of a RLT is placed at the center of the visualization, and the distance of a node from the root encodes the level of the node (as shown in Figure 4.6).

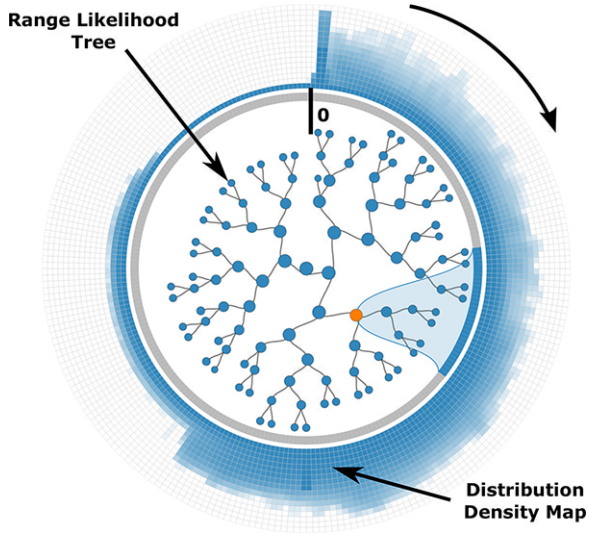


Figure 4.6: The composite radial visualization of a range likelihood tree with distribution density map.

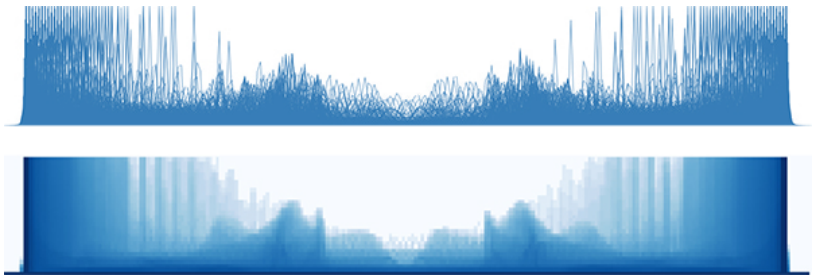


Figure 4.7: Visualizing distributions of the material ensemble dataset using superimposed curves (top) and curve density estimation (bottom).

Distribution Density Map. Since a probability distribution field often contains a large number of distributions, a simple plotting of all distribution as curves will likely lead to problems with overplotting and visual clutter. To provide an effective overview of distributions in the underlying probability distribution field, we employ a curve density estimation technique [83] to create a density-based visualization of distributions, as shown in Figure 4.7. The density map in the RLF view is drawn using a ring metaphor that surrounds a RLT to form a composite radial visualization, where values start at the 12 o'clock position (marked with a line bar in Figure 4.6) and increase in a clockwise order.

Visual Linking. When the user selects a node of interest, it is desired that the underlying range is simultaneously highlighted for examining the distributions within the selected range. Explicit visual linking in which curves are rendered between related elements has been shown expressive and effective [147]. This also motivated our choice of a radial visualization, where links are typically drawn as arcs connecting related elements [89]. Our composite radial visualization supports such visual linking using an easy interaction — when the user hovers over a node of interest, visual links are immediately drawn as arcs to connect the node with the range in the distribution density map (as shown in Figure 4.6).

4.3.1.2 Multi-field Transfer Function Widget

Through visual exploration of RLFs in the RLT view and the RLF view, users can progressively understand the underlying probability distribution field when focusing on one RLF at a certain level in a RLT at a time. In addition, multiple RLFs can be selected to enable further analysis, as uncertain features may exist in multiple distinct ranges. As a result, multiple RLFs form a *multidimensional range likelihood field (MRLF)* where every distribution at a grid point in the underlying probability distribution field is transformed into a likelihood vector across the multi-field. To visualize these likelihood vectors, we employ

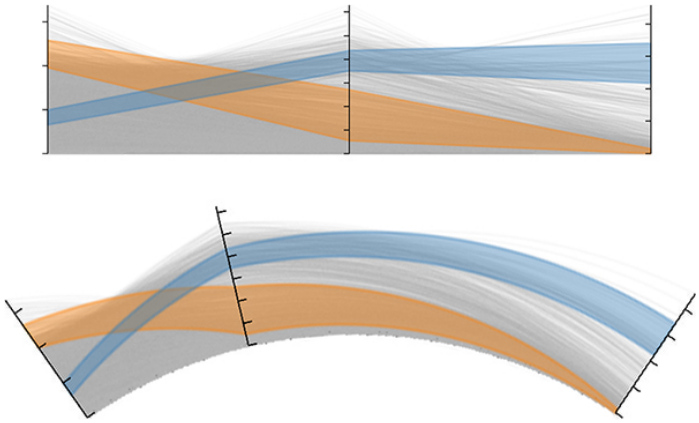


Figure 4.8: Multi-field transfer function widget using parallel coordinates axes (top) and flexible coordinates axes (bottom).

the parallel coordinates plot (PCP) [72], a popular choice in multivariate transfer function design that enables users to manipulate transfer functions through traditional brushing such as axis brushing [21]. In our approach, a *coordinates axis* represents the scale of likelihoods for one selected RLF, and a polyline that connects multiple coordinates axes represents a multi-field likelihood vector.

Inspired by flexible linked axes [35], which extend the conventional parallel coordinates design (Figure 4.8(top)) by flexibly positioning coordinates axes (Figure 4.8(bottom)), we orient the coordinates axes so that they are aligned with the distribution density map (in the RLT view) to form an integrated radial visualization. In this way, a coordinates axis can be placed near the corresponding subrange in the RLT visualization, which preserves the user's mental map during exploration.

As coordinates axes are positioned in the polar coordinate system, we encode likelihood vectors as polycurves across the coordinates axes. To overcome the overplotting problem of drawing polycurves of all likelihood vectors, we employ adaptive sampling to draw a set

of likelihood vectors that is statistically similar to the overall population. We start with a set of likelihood vectors which are sampled from the field randomly and keep adding more samples until their distribution is similar to the overall distribution. Overlaying the sampled polycurves, transfer functions are highlighted using ribbon metaphors across coordinates axes in an illustrative manner [95].

4.3.1.3 Spatial View

The spatial view has two sub-views: the range likelihood field (RLF) view and the multi-field classification view.

The Range Likelihood Field View is used to visualize a RLF of interest. After the user selects one node in the RLT, a RLF is constructed on the fly based on equation (4.4) and the RLF view is updated interactively with direct volume rendering. We adopt a perception-aware color map [133] to color the RLF in a blue-green-yellow manner with a linear opacity scale. This view is linked with the RLT view — when the user hovers over a node of interest, the RLF view is simultaneously updated to visualize the RLF within the selected subrange.

The Multi-Field Classification View mainly shows the probabilistic classification results of multiple RLFs. The user first brushes on the coordinates axes of PCP to select a set of multi-field likelihood vectors. Then Gaussian transfer functions are automatically generated based on the user selected likelihood vectors based on equation (4.6) and (4.7). The classification results of the MRLF is rendered in this view based on these Gaussian transfer functions.

4.3.2 Exploration Guidelines

After transforming a given probability distribution field into a RLT, the RLT serves as the exploration space where users can search for features of interest and perform further

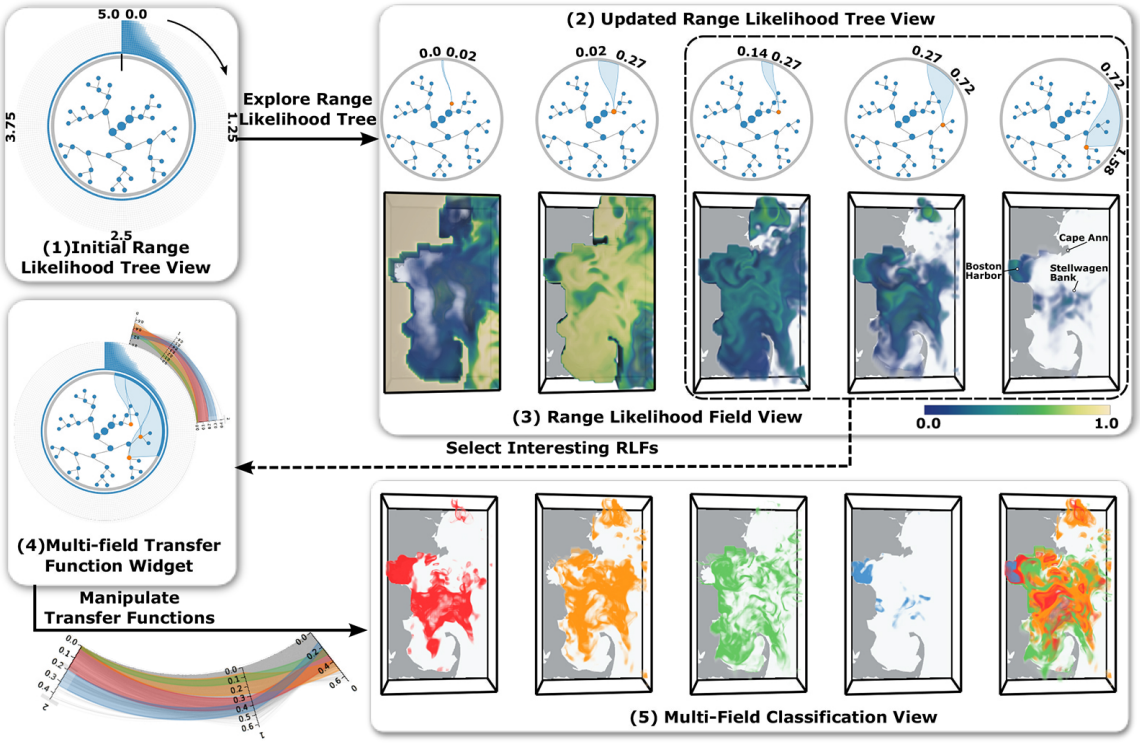


Figure 4.9: An illustration of our range likelihood tree guided exploration framework using the Massachusetts Bay Sea Trial Ensemble dataset. Starting from the initial range likelihood tree view (1), users can hover on different nodes in the tree to see their corresponding subranges (2), and examine the likelihoods of associated grid points in the range likelihood field view (3); after selecting a few subranges of interest, a multi-field transfer function widget (4) is created, which can be manipulated by the user to explore and classify grid points based on their likelihoods in different subranges in the multi-field classification view (5).

analysis such as probabilistic classification. We follow the visual information seeking mantra "Overview first, zoom and filter, then details-on-demand" by Ben Shneiderman [141] to guide the exploration process:

Overview First. The RLT view (Figure 4.9(1)) provides an overview of the multi-level structure of RLFs as well as the distributions in the underlying probability distribution field. By selecting a node in the RLT, its corresponding RLF is constructed (based on equation

(4.4)) to obtain an overview of how likely the uncertain values at each grid point falls within this selected range.

Zoom and Filter. To understand how much the distributions fall within a RLF and where they are spatially located, users can hover a node in the RLT view (Figure 4.9(2)) and examine the likelihoods of associated grid points in the RLF view (Figure 4.9(3)). When no prior knowledge is available about the uncertain data set, it is natural to start from the root and follow a binary-split path. As the user zooms into a deeper level of the hierarchy, the RLF in focus becomes smaller and more distributions will be filtered out.

Details on Demand. When users select a set of representative RLFs of interest, a multi-field transfer function widget (Figure 4.9(4)) is created. users can manipulate the transfer function widget to explore and classify grid points based on their likelihoods in different subranges in the multi-field classification view (Figure 4.9(5)).

4.4 Results

We demonstrate the effectiveness of our methods through experiments on three uncertain data sets in different applications: Massachusetts Bay Sea Trial (MBST-98), D-FTLE of temporal down-sampled Hurricane Isabel, and High-Resolution Rapid Refresh (HRRR) ensemble dataset. All the experiments were performed on a desktop computer with an Intel(R) Core(TM) i7-4790K CPU 4.0GHz processor, 16GB memory, and an NVIDIA GTX 970 GPU. In our approach, clustering of RLFs is the most time consuming operation, and Table 4.1 reports the computational performance of the clustering algorithm. In particular, we verified the visualization results (in Section 4.4.2 and Section 4.4.3) with two domain scientists, who are experts in environment science with over ten years of experience.

Table 4.1: Datasets and timings: t_c is the runtime (in seconds) for the clustering algorithm.

Data Set	Resolution	Runs	Partitions	Size	t_c
MBST-98	$106 \times 180 \times 32$	600	256	0.58GB	151.71s
D-FTLE (Isabel)	$250 \times 250 \times 50$	200	256	2.98GB	852.28s
HRRR	$449 \times 264 \times 40$	10	128	2.26GB	346.56s

4.4.1 Massachusetts Bay Sea Trial Ensemble Dataset

The Massachusetts Bay Sea Trial (MBST-98) was an interdisciplinary forecast simulation [87, 128] based on the Littoral Ocean Observing and Predicting System (LOOPS). The MBST-98 took place in Massachusetts Bay from 17 August to 5 October 1998 with 600

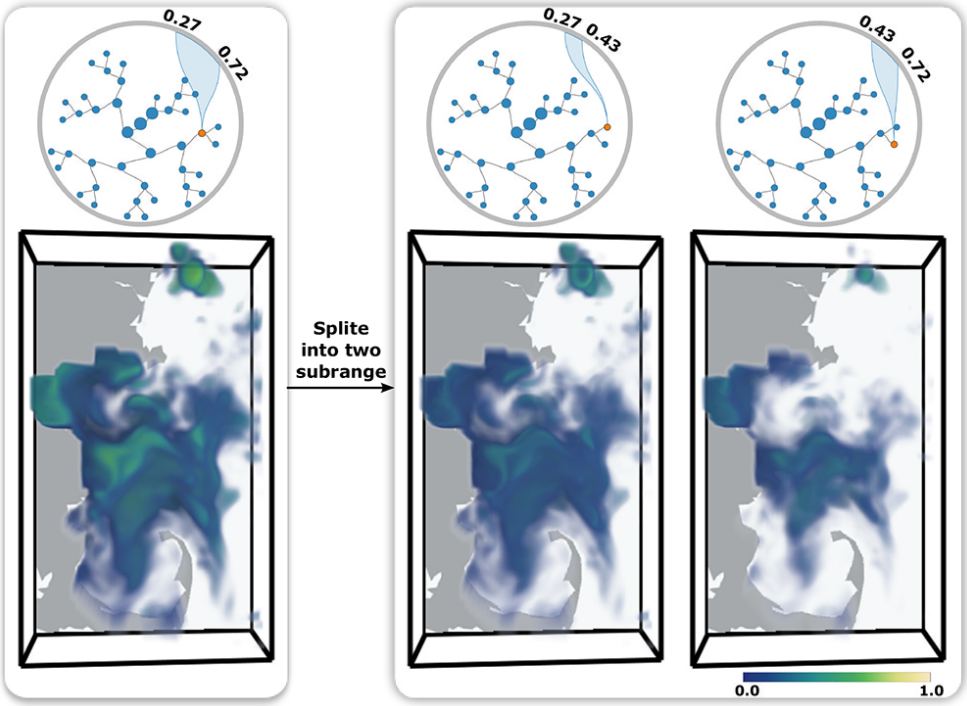


Figure 4.10: Exploring ranges at different levels in the RLT of the Massachusetts Bay Sea Trial Ensemble dataset.

ensemble runs. The scientific focus of this simulation was phytoplankton and zooplankton patchiness which were encoded in variables *chlorophyll-a concentration* and *zooplankton concentration*. In order to investigate the chlorophyll-a concentration, we selected the CHL variable at the time step representing September 2, 1998 and perform kernel density estimation based on all 600 ensembles at every grid point to model the uncertainty as probability density functions.

From the different components provided by our system, users can better explore the distribution field. Figure 4.9(1) presents an overview of the multi-level structure of the RLFs as well as the distributions in the underlying probability distribution field. Based on the distribution density map, we can see that the underlying probability distributions have high probabilities when chlorophyll-a concentration is low and have decreasing probability when chlorophyll-a concentration increases. By selecting ranges in the RLT view, their corresponding RLFs are shown in the RLF view, which help users comprehend the cumulative probabilities at every grid point in the selected ranges. Figure 4.9(2) and 4.9(3) show five chlorophyll-a concentration value subranges as well as their corresponding RLFs, which highlight distinct features. We can see that the land regions are highlighted with range likelihoods aroud 1.0 with chlorophyll-a concentration near 0.0 mg/m^3 (milligram per cubic meter), and the ocean regions are highlighted with different range likelihoods when chlorophyll-a concentration is greater than $0.0mg/m^3$. For the subrange $0.27 \sim 0.72mg/m^3$, chlorophyll-a concentration has a higher likelihood at northeast of Cape Ann compared with other regions, and for the subrange $0.72 \sim 1.58mg/m^3$ chlorophyll-a concentration has a higher likelihood near Boston Harbor. Besides these highlighted regions, there are a large number of transparent regions in these two RLFs, which represents the likelihoods in those regions are zero. This is consistent with the fact that nutrients advected over Stellwagen

Bank (due to tidal mixing) as well as along the coastline (due to wind driven upwelling and episodic wind mixing) [128]. We also observed that in some regions such as Stellwagen Bank, chlorophyll-a concentration has a non-zero likelihood in the selected value ranges, which indicates that chlorophyll-a concentration at those regions could fall within any of those ranges. With the RLT, the user can explore the probability distribution field with different levels of detail. For example, in Figure 4.10 the value range $0.27 \sim 0.72 mg/m^3$ is further divided into two subranges. We observed that the feature at northeast of Cape Ann splits into two features.

By selecting multiple RLFs with respect to the different chlorophyll-a concentration value ranges, the user can view the correlation of the multi-field likelihoods in the transfer function widget (Figure 4.9(4)) and manipulate the transfer function for visual classification of the distributions in the multi-field classification view (Figure 4.9(5)). With this multi-field classification, insightful visualization results can be obtained. For instance, in Figure 4.9(4), four Gaussian transfer functions are constructed and in Figure 4.9(5) the classification results are shown. The blue part of the result is mainly at the region of the Boston Harbor, where the cumulative probability in range $0.72 \sim 1.58 mg/m^3$ is around 0.27 and the cumulative probability in range $0.14 \sim 0.27 mg/m^3$ is around 0.17; The red feature surrounding the blue feature at the Boston Harbor has a higher cumulative probability in range $0.14 \sim 0.27 mg/m^3$ but lower cumulative probability in range $0.72 \sim 1.58 mg/m^3$. Many other features can also be identified from the visualization results.

4.4.2 Temporally Down-Sampled Hurricane Isabel Dataset

Finite-time Lyapunov exponent (FTLE) is becoming a popular method in fluid dynamics for transport behavior analysis of unsteady flows. The higher FTLE value a region has,

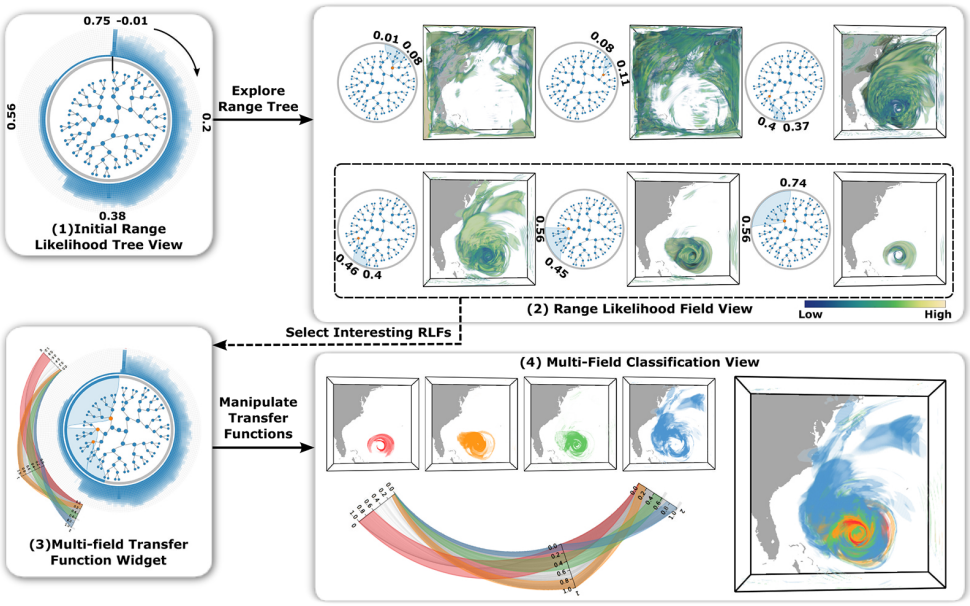


Figure 4.11: Experiments on the Temporally Down-Sampled Hurricane Isabel dataset. See Section 4.4.2 for details about the exploration process.

the more divergent flows are in the region. Guo et al. [56] extended the concept of FTLE to D-FTLE, which computes a probability density function of FTLE values for every grid point. In this experiment, we study unsteady flows using D-FTLE on the Hurricane Isabel dataset. The uncertainty is introduced through temporal down-sampling using the method proposed by Chen et al. [33]. The Hurricane Isabel dataset is an atmospheric simulation data created by the Weather Research and Forecast model (WRF) with 48 time steps (hourly average), which models a strong hurricane in the West Atlantic region in September 2003. We down-sampled the Hurricane Isabel dataset by aggregating every 12 time steps into one. Following the method in [33], for every 12 time steps, we fit a quadratic Bezier curve and store the quadratic Bezier curve parameters and the interpolation error distributions at every grid point. The variables U, V, and W which correspond to the wind vector directions are

used in this experiment. A D-FTLE distribution field is then generated starting from time step 0 and advecting 6 hours.

An overview of the multi-level structure of the RLFs and a density map of the underlying distributions are shown in Figure 4.11(1). We can see that most distribution of the FTLE has low probability when the FTLE value is high, suggesting that high FTLE values only exist in a small part of the spatial domain. Selecting ranges of FTLE values from low to high, their corresponding RLFs are visualized in the RLF view (Figure 4.11(2)). We found that high FTLE values have higher likelihoods in the hurricane eyewall, elevated in the surrounding rainbands and can seperate the spiral bands, especially the extended northerly band. Meteorologists with whom we discussed the visualization results confirmed that the structure of the hurricane eyewall and rainbands can be highlighted by selecting RLFs corresponding to high FTLE values. In particular, as the FTLE value increases, the RLF represents the structure closer to the hurricane eye.

To further investigate the distributions around the hurricane eye, we select three RLFs with respect to three FTLE value ranges $0.56 \sim 0.74$, $0.46 \sim 0.56$, and $0.4 \sim 0.46$. The probabilistic classification results of the underlying probability distribution of FTLE values are shown in Figure 4.11(3) and Figure 4.11(4). The red part of the result is within the hurricane eye, with high cumulative probability in the highest FTLE value range and low cumulative probability in the lowest FTLE value range. The outside feature with blue color has low cumulative probability in the highest FTLE value range and high cumulative probability in the lowest FTLE value range. Feedback from the scientists confirms that these classifications well separate the outflow cloud shield (blue region) from the hurricane eye and eyewall.

4.4.3 Ensemble HRRR Simulation Dataset

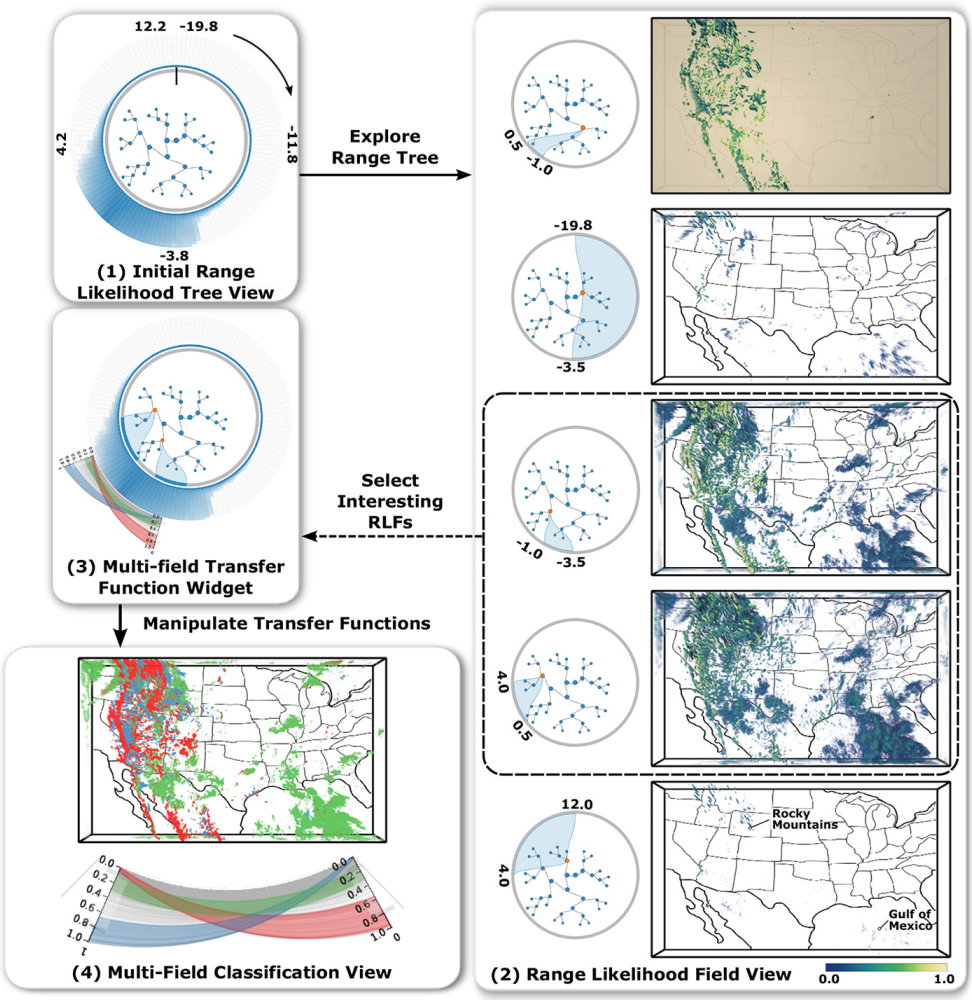


Figure 4.12: Experiments on the Ensemble HRRR Simulation dataset. See Section 4.4.3 for details about the exploration process.

The High-Resolution Rapid Refresh (HRRR) [6] is an National Oceanic and Atmospheric Administration (NOAA) real-time atmospheric model based on the Weather Research and Forecasting (WRF) model. The HRRR is hourly updated and available to the public through the Unidata's THREDDS Data Server2. We use an ensemble of the HRRR simulation output

at time step representing 22:00:00 UTC, August 29, 2015, which consists of more than 20 variables and starts from 10 different hours. Based on the domain experts' focus of interest, we select the variable vertical velocity for analysis. We perform kernel density estimation at every grid point to model the uncertainty as probability density functions.

The RLT, RLFs, and the probabilistic classification results are shown in Figure 4.12. In Figure 4.12(2), we selected several representative value ranges and visualize their corresponding RLFs. It can be seen that vertical flows have high likelihood to occur near the Rocky Mountains region due to the topography and median likelihood to occur near the Gulf of Mexico compared with other regions. To further analyze the upward and downward motion of the wind, we select two RLFs with respect to the upward flow and downward flow for classification.

The domain scientists helped us explain the classification results (in Figure 4.12(4)): near the Rocky Mountains the wind directions are with either upward motion (color in red) or downward motion (color in blue) due to orographic lift which occurs when an air mass is moving over rising terrain and the foehn winds which occurs in the downwind side of a mountain. The red and blue regions are interweaving since winds are moving across mountains. Near the Gulf of Mexico the wind directions have around 0.2 likelihood in either direction.

4.5 Discussion and Future Work

The range likelihood tree representation brings a novel perspective to visual summarization and exploration of probability distribution fields. Unlike previous statistical summary and dissimilarity based methods, our method summarizes complex probability distributions with the range likelihood fields over a few representative subranges by considering the

different roles that different subranges may play in understanding probability distributions. Our data model enables effective classification through user query and exploration.

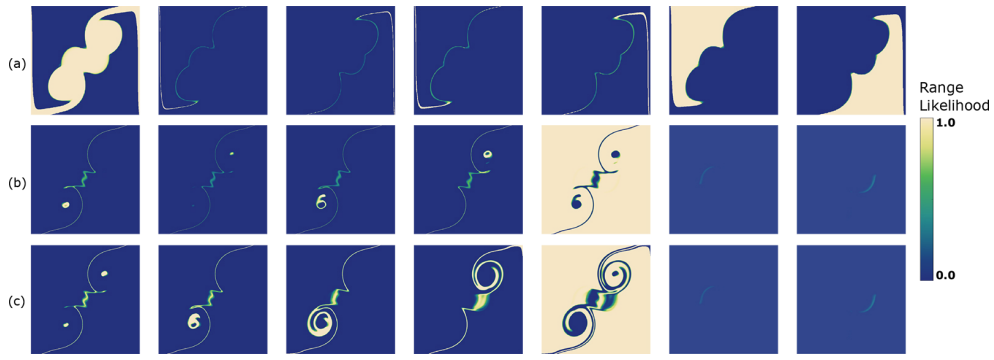


Figure 4.13: Comparison of different clustering approaches using the 2D material ensemble data set. Each image represents a RLF with respect to one cluster. (a): Hierarchical clustering based on the Euclidean distances between range likelihood fields. (b): Hierarchical clustering applied on likelihood distributions (scaled range likelihood fields) using the Euclidean distance. (c): K-means clustering using the Euclidean distances between likelihood distributions.

In our approach, clustering of RLFs was done in the preprocessing stage prior to the exploration process. While we used Jensen–Shannon divergence (JSD) and hierarchical clustering in our approach, we also experimented with alternative dissimilarity measures (e.g., Euclidean distance) and clustering methods (e.g., k-means clustering). Figure 4.13 shows the clustering results with alternative clustering approaches and dissimilarity measures using the 2D material ensemble dataset. Compared with these alternative methods, the clustering results of JSD and hierarchical clustering (Figure 4.4(b)) highlights more distinct regions with respect to different features. We also experimented with different numbers of initial subrange partitioning, and found the resulting leaf nodes in the pruned trees were

roughly the same with minor variations in all four datasets. Therefore, we used $N = 256$ as the initial number of partitions in our experiments.

We showed our system to the domain scientists to assess the potential of our work applied to weather research. The domain experts were able to understand the concept of range likelihood fields, and found the range likelihood tree effective as a compact representation of a probability distribution field. The composite visualization of the range likelihood tree and the distribution density map are intuitive for them to understand, and the transfer function widget is easy to use. Overall, the positive feedback we received from the domain scientists encouraged us to further apply the exploration framework to solve more real-world scientific problems in our future work. In particular, we plan to combine our approach with statistical summary of distributions such as entropy to analyze overall uncertainty at spacial locations. We would also like to investigate multivariate distributions and high-dimensional value ranges in analyzing multivariate uncertain data. We would also like to extend our work to time-varying probability distribution field visualization and exploration.

4.6 Conclusion

In this work, we present a compact and effective representation, called range likelihood tree (RLT), to summarize and explore probability distribution fields. The RLT representation decomposes and summarizes complex probability distributions with the range likelihood fields over a few representative subranges, which allows effective classification and identification of features through user query and exploration. We present an exploration framework with multiple interactive views to explore probability distribution fields through range likelihood fields, and provide guidelines for visual exploration using our framework. We demonstrated the effectiveness and usefulness of our approach in exploratory analysis using

several representative uncertain data sets, and verified the visualization results with domain scientists in environment science.

Chapter 5: InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations

As the computational power of modern supercomputers continues to grow, ensemble simulations are more often conducted with a large number of parameter settings in high spatial and/or temporal resolutions. Despite the advances in accuracy and reliability of simulation results, however, two challenges have emerged: (1) I/O bottleneck for the movement of the large-scale simulation data and (2) effective exploration and analysis of the simulation parameters. In situ visualization [13, 93], which generates visualization at simulation time and stores only the visualization results (that are much smaller than the raw simulation data [3, 4]) for post-hoc analysis, addresses the first challenge to some extent. However, it also limits the flexibility of post-hoc exploration and analysis, because the raw simulation data are no longer available.

This study focuses on improving scientists' ability in exploring the in situ visualization results of ensemble simulations and extending their capability in investigating the influence of different simulation parameters. Several pioneering works have been proposed to facilitate post-hoc exploration of in situ visualization results. For example, the Cinema framework [3, 4] visualized the simulation data from different viewpoints in situ and collected images to support post-hoc exploration. The volumetric depth images [46, 51] stored ray segments with composited color and opacity values to enable post-hoc exploration of arbitrary viewpoints

for volume rendering. However, these approaches focus more on extending the capability to explore the visual mapping parameters (e.g., transfer functions) and view parameters (e.g., view angles) and have little consideration of the simulation parameters, which are important in studying ensemble simulations.

Simulation parameter space exploration is not trivial, because the relationship between the simulation parameters and outputs is often highly complex. The majority of existing simulation parameter space exploration approaches [29, 155] resorted to visualizing a set of simulation parameters and outputs simultaneously and revealing the correspondence between the parameters and outputs through visual linkings. However, these approaches often depend on the raw simulation data that might not be available for large-scale ensemble simulations. Moreover, these approaches have limited ability in inferring simulation outputs with respect to new parameters. Hence, extra simulations have to be conducted for new parameters, which cost enormous computational resources for most scientific simulations.

In this chapter, we propose InSituNet, a deep learning based surrogate model to support parameter space exploration for ensemble simulations that are visualized in situ. Our work is based on the observation that images of high accuracy and fidelity can be generated with deep neural networks for various image synthesis applications, such as super-resolution [41, 73, 86], inpainting [111, 162], texture synthesis [53, 168], and rendering [15, 43]. Specifically, we train InSituNet to learn the end-to-end mapping from the simulation, visual mapping, and view parameters to visualization images. The trained model enables scientists to interactively explore synthesized visualization images for different simulation parameters under various visualization settings without actually executing the expensive simulations. Our approach consists of three major steps.

1. **In situ training data collection from ensemble simulations** Given ensemble simulations conducted with different simulation parameters, we visualize the generated simulation data in situ with various visual mapping and view parameters. The resulting visualization images and the corresponding parameters are collected and used for the offline training of InSituNet.
2. **Offline training of InSituNet** Given the parameters and image pairs, we train InSituNet (i.e., a convolutional regression model) with cutting-edge deep learning techniques on image synthesis to map simulation, visual mapping, and view parameters to visualization images directly.
3. **Interactive post-hoc exploration and analysis** With the trained InSituNet, we build an interactive visual interface that enables scientists to explore and analyze the simulation from two perspectives: (1) inferring visualization results for arbitrary parameter settings within the parameter space with InSituNet's forward propagations and (2) analyzing the sensitivity of different parameters with InSituNet's backward propagations.

We demonstrate the effectiveness of the proposed approach in combustion, cosmology, and ocean simulations, and compare the predicted images of InSituNet with the ground truth and alternative methods. In addition, we evaluate the influence of different hyperparameters of InSituNet (e.g., the choice of loss functions and the network architectures) and provide guidance in configuring the hyperparameters. In summary, the contributions of this study are threefold:

- A deep image synthesis model (i.e., InSituNet) that enables post-hoc parameter space exploration of ensemble simulations

- An interactive visual interface to explore and analyze the parameters of ensemble simulations with the trained InSituNet
- A comprehensive study revealing the effects of different hyperparameters of InSituNet and providing guidance for applying InSituNet to other simulations

5.1 Overview

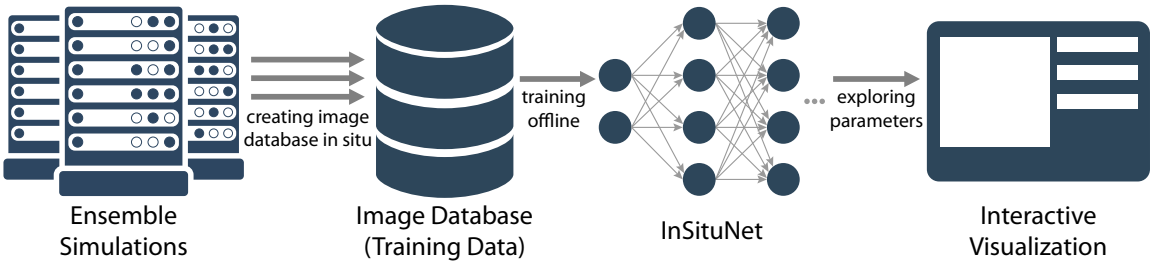


Figure 5.1: Workflow of our approach. Ensemble simulations are conducted with different simulation parameters on supercomputers, and visualization images are generated in situ for different visual mapping and view parameters. The generated images and the parameters are collected into an image database. A deep image synthesis model (i.e., InSituNet) is then trained offline based on the collected data, which is later used for parameter space exploration through an interactive visual interface.

Figure 5.1 provides the workflow of our approach, which consists of three major components. First, given ensemble simulations conducted with different simulation parameters, we visualize the generated simulation outputs in situ with different visual mapping and view parameters on supercomputers. The three groups of parameters—simulation, visual mapping, and view parameters—along with the corresponding visualization results (i.e., images) are collected to constitute an image database (Section 5.2). Second, with the collected data pairs between parameters and the corresponding images, we train InSituNet to learn the end-to-end mapping from the simulation inputs to the visualization outputs

(Section 5.3). To improve the accuracy and fidelity of the generated images, we use and combine different state-of-the-art deep learning techniques on image synthesis. Third, with the trained InSituNet, we build an interactive visual interface (Section 5.4) to explore and analyze the parameters from two aspects: (1) predicting visualization images interactively for arbitrary simulation, visual mapping, and view parameters within the parameter space and (2) investigating the sensitivity of different input parameters to the visualization results.

5.2 In Situ Training Data Collection

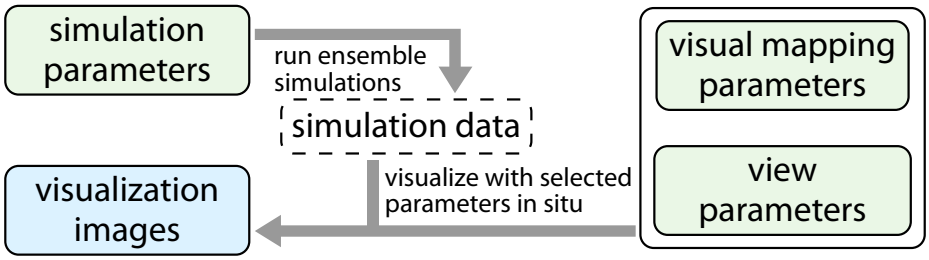


Figure 5.2: Our in situ training data collection pipeline. Simulation data, generated with different simulation parameters, are visualized in situ with different visual mapping and view parameters. The in situ visualization generates a large number of images, which are collected along with the corresponding parameters for the training of InSituNet offline.

Figure 5.2 illustrates our in situ training data collection pipeline. Given ensemble simulations conducted with different simulation parameters, we perform in situ visualization with a desired set of visual mapping parameters (e.g., isosurfaces extraction with a set of isovalues) and different view parameters (e.g., viewpoints). We denote an instance of simulation, visual mapping, and view parameters as P_{sim}, P_{vis}, and P_{view}, respectively, which corresponds to a visualization image I. The parameters (highlighted in green in Figure 5.2) and the corresponding visualization images (highlighted in blue in Figure 5.2) constitute

data pairs, which will be stored and used to train InSituNet. InSituNet learns a function $\mathscr{F}$ that maps the three groups of parameters to the corresponding visualization image, which can be defined as

$$\mathscr{F}(P_{sim}, P_{vis}, P_{view}) \rightarrow I, \tag{5.1}$$

so that it can predict visualization images with unseen parameters for parameter space exploration. In the following, we discuss the three groups of parameters in detail.

Simulation parameters P_{sim} are represented as a vector with one or more dimensions, and the value range of each dimension is defined by scientists. By sweeping the parameters within the defined ranges, ensemble simulations are conducted to generate the ensemble data.

Visual mapping parameters P_{vis} are predefined operations to visualize the generated simulation data, such as pseudo-coloring with predefined color schemes. Note that we limit the users' ability in selecting arbitrary visual mappings to produce and store fewer images.

View parameters P_{view} are used to control the viewpoints that the images are created from. In this work, we define the viewpoints by a camera rotating around the simulation data, which is controlled by azimuth $\theta \in [0, 360]$ and elevation $\phi \in [-90, 90]$. For panning and zooming, we resort to image-based operations (i.e., panning and resizing the images) as proposed in [4]). To train a deep learning model that can predict visualization images for arbitrary viewpoints, we sample the azimuth and elevation and generate images from the sampled viewpoints. Based on our study, we found that taking 100 viewpoints for each ensemble member is sufficient to train InSituNet.

With the specified values for the three groups of parameters, we generate the corresponding visualization images. Our work uses RGB images compressed to the portable network graphics (PNG) format instead of more sophisticated image formats, such as volumetric

depth images [46, 51] or explorable images [149–151], for two reasons. First, the benefits of using those sophisticated image formats, such as supporting changing of viewpoints, can be achieved by InSituNet trained on the RGB images. Second, RGB images are more generally applicable for various visualizations and more easily to be handled by neural networks compared with those sophisticated image formats.

5.3 InSituNet Architecture and Training

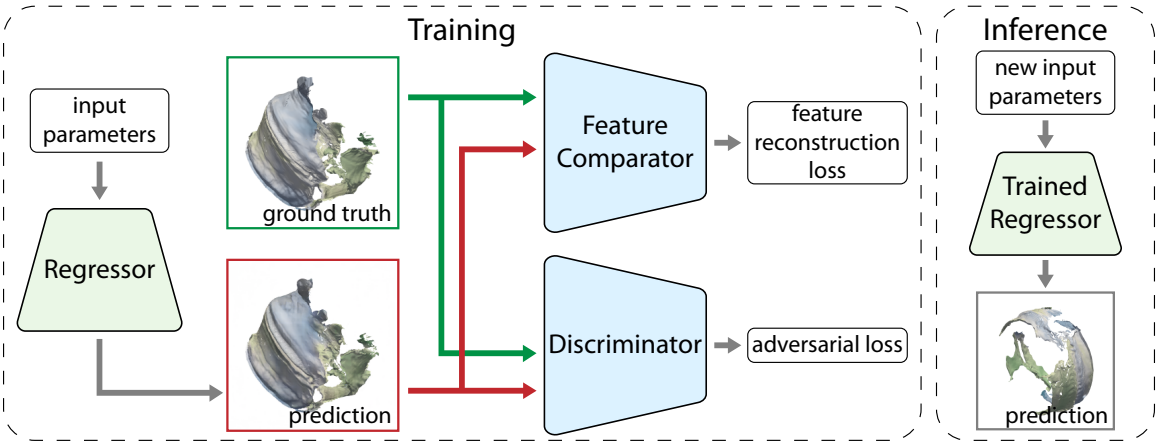


Figure 5.3: Overview of InSituNet, which is a convolutional regression model that predicts visualization images from input parameters. During training, the regression model is trained based on the losses computed with the assist of a pretrained feature comparator and a discriminator.

Figure 5.3 illustrates both training and inference pipelines of InSituNet. In the training stage, InSituNet consists of three subnetworks: a regressor, a feature comparator, and a discriminator. The regressor R_ω is a deep neural network (defined by a set of weights ω) modeling the function that maps input parameters to visualization images as defined in Equation 5.1. To train a regressor that can generate images of high fidelity and accuracy,

we introduced the feature comparator F and the discriminator D_υ to compute losses by comparing the predicted and the ground truth images. The feature comparator is a pretrained neural network whose convolutional kernels are used to extract and compare image features (e.g., edges, shapes) between the predicted and the ground truth images to obtain a feature reconstruction loss. The discriminator D_υ is a deep neural network whose weights υ are updated during training to estimate the divergence between the distributions of the predicted and the ground truth images. The divergence is known as the adversarial loss [54], which is combined with the feature reconstruction loss to train the regressor R_ω. In the inference stage, we need only the trained regressor R_ω, which can predict visualization images for new parameters that are not in the training data.

In the following, we first describe the architecture of the three subnetworks. Next, we explain the loss functions used to train InSituNet, along with the techniques used to stabilize the training. Last, details of our training process are connected and presented in Algorithm 1.

5.3.1 Network Architecture

Three subnetworks are involved during training: the regressor R_ω, feature comparator F, and discriminator D_υ. The regressor R_ω and discriminator D_υ are two deep residual convolutional neural networks [62] parameterized by the weights ω and υ, respectively. The architectures of R_ω and D_υ are designed by following the network architecture proposed by [81, 98], because the scale of our image synthesis problem is similar to theirs. For the feature comparator F, we use the pretrained VGG-19 model [144], which has been widely used in many deep image synthesis approaches [70, 73, 86].

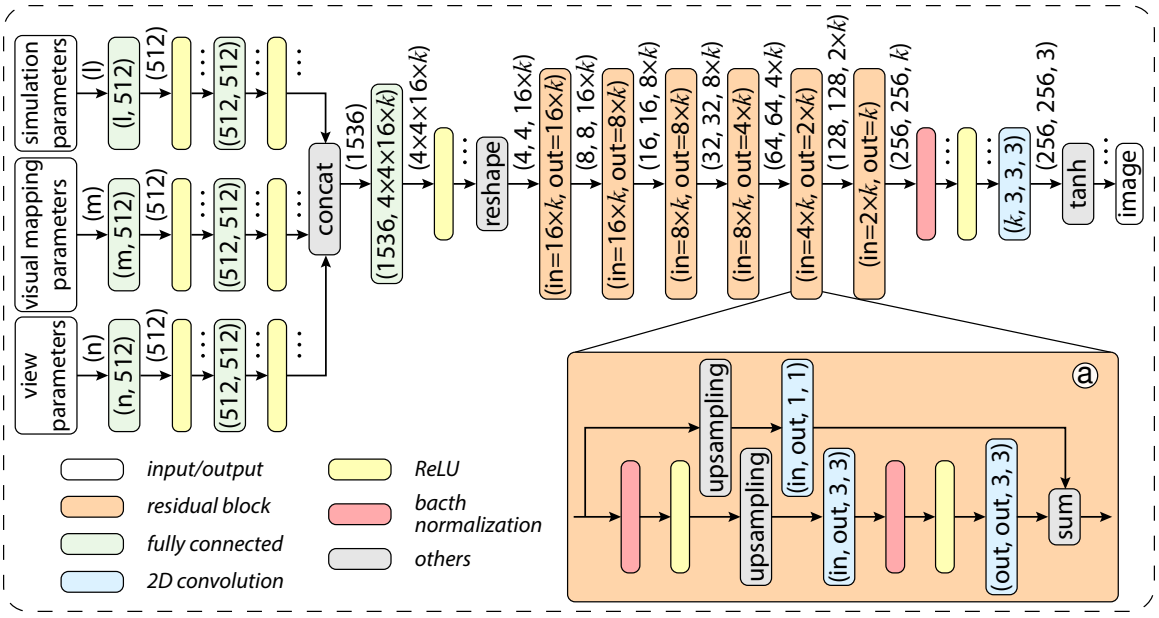


Figure 5.4: Architecture of R_ω, which encodes input parameters into a latent vector with fully connected layers and maps the latent vector into an output image with residual blocks (a). The size of R_ω is defined by k, which controls the number of convolutional kernels in the intermediate layers.

5.3.1.1 Regressor R_ω

The architecture of R_ω is shown in Figure 5.4, which takes the P_{sim}, P_{vis}, and P_{view} as inputs and outputs a predicted image I. The three types of parameters are first fed into three groups of fully connected layers separately, and the outputs are then concatenated and fed into another fully connected layer to encode them into a latent vector. Note that the parameters could also be concatenated first and then fed into fully connected layers. However, as each parameter is fully connected with all neurons in the next layer, more weights will be introduced in the network and the network size will be increased. Hence, we handle the parameters with separate fully connected layers. Next, the latent vector is reshaped into a low-resolution image, which is mapped to a high-resolution output

image through residual blocks performing 2D convolutions and upsamplings. Following the commonly used architecture [81, 98], we use the rectified linear unit (`ReLU`) activation function [103] in all layers except the output layer. For the output layer, we use the `tanh` function to normalize each pixel into $[-1, 1]$.

Note that we introduce a constant k in the network architecture to control the number of convolutional kernels in the intermediate layers. The constant k is used to balance the expressive power and the size and training time of R_ω to cope with datasets in different complexities.

Residual Blocks R_ω consists of several residual blocks (Figure 5.4(a)), which are proposed in [62] to improve the performance of neural networks with increasing depth. We adopted the residual blocks here because R_ω often needs to be very deep (i.e., more than 10 convolutional layers) to synthesize images with high-resolutions. Inside each residual block, the input image is first upsampled by using nearest neighbor upsampling. The upsampled image is then fed into two convolutional layers with kernel size 3×3. In the end, the original input image is added to the output, and the result is sent to the next layer. Batch normalizations are performed on the output of each convolutional layer to stabilize the training. Note that if the resolution or the channel number of the input image is not the same as the output, we perform the upsampling and convolution operations on the input image to transform it into the size of the output.

5.3.1.2 Discriminator D_υ

The architecture of D_υ is shown in Figure 5.5, which takes a predicted/ground truth image and the corresponding parameters as inputs and produces a likelihood value indicating how likely the input image is a ground truth image conditioning on the given parameters. With the likelihood values, an adversarial loss can be defined to update D_υ and R_ω (details

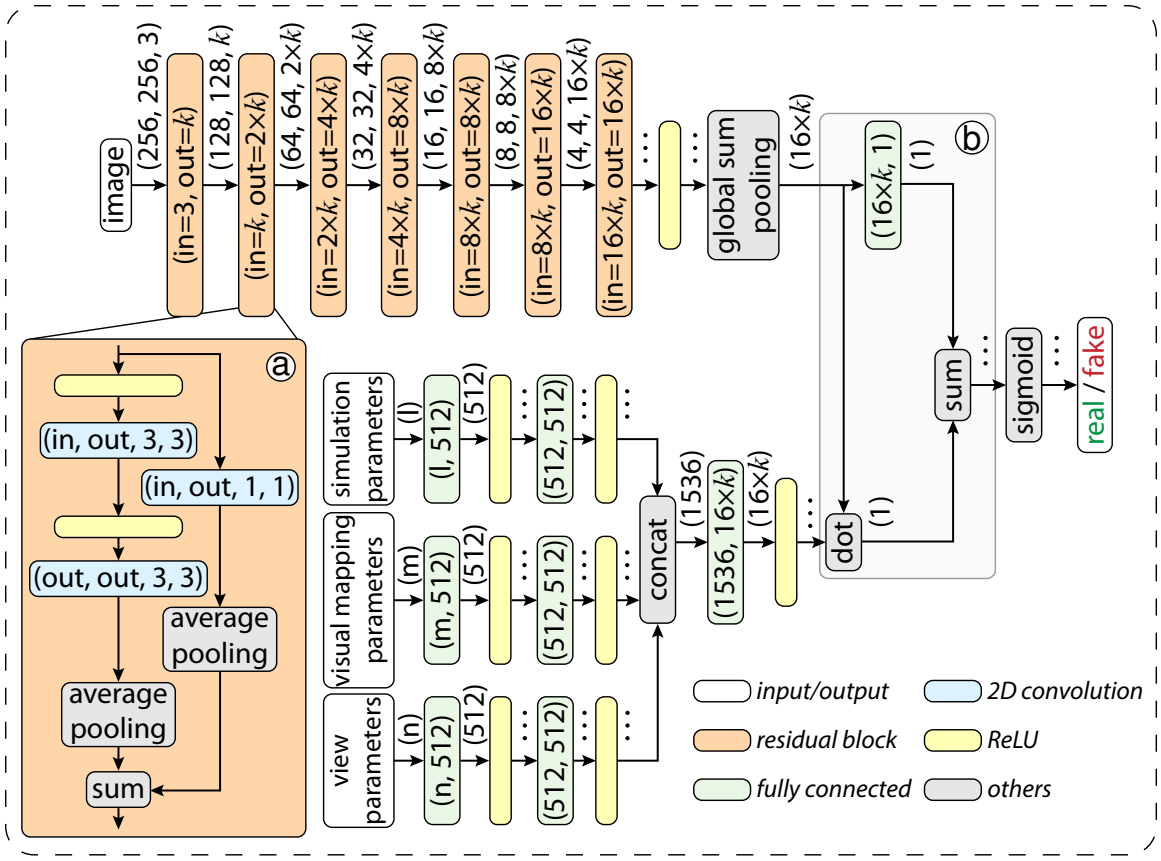


Figure 5.5: Architecture of D_υ. Input parameters and the predicted/ground truth image are transformed into latent vectors with fully connected layers and residual blocks (a), respectively. The latent vectors are then incorporated by using the projection-based method [99] (b) to predict how likely the image is a ground truth image conditioning on the given parameters. Similar to R_ω, the size of D_υ is controlled by the constant k.

in Section 5.3.2.2). Similar to R_ω, the three types of parameters are encoded into a latent vector in D_υ through fully connected layers. Meanwhile, the input image is fed through several residual blocks to derive its intermediate representation that is a latent vector. The latent vectors are then incorporated to obtain the likelihood value conditioning on the three groups of parameters. `ReLU` activations are used in all layers except the output layer, which instead uses the `sigmoid` function to derive a likelihood value within $[0, 1]$.

Residual blocks The architecture of the residual blocks in D_v (Figure 5.5(a)) is similar to that in R_ω except that downsampling (average pooling in this work) is performed instead of upsampling to transform images into low-resolution representations and no batch normalization is performed, because it often hurts the performance of D_v [81, 92].

Projection-based condition incorporation We employed the projection-based method [99] to incorporate the conditional information (i.e., the three groups of parameters) with the input image. This method computes a dot product between the data to be incorporated, which in our work is the latent vector of the input parameters and the latent vector of the image (Figure 5.5(b)). Compared with other condition incorporation methods, such as vector concatenation [15, 96], the projection-based method improves the quality of conditional image synthesis results, as demonstrated by Miyato and Koyama [99].

5.3.1.3 Feature Comparator F

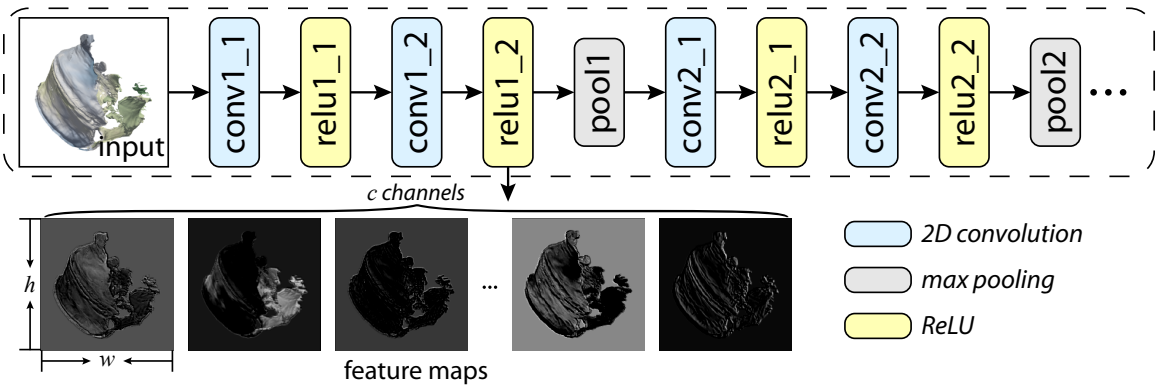


Figure 5.6: Architecture of F (i.e., VGG-19 network), where each layer is labeled with its name. Feature maps are extracted through convolutional layers (e.g., `relu1_2`) for feature-level comparisons.

To produce high quality image synthesis results, we also strive to minimize the feature-level difference between the generated and the ground truth image by employing a commonly used feature comparator F, namely the pretrained VGG-19 model [144] shown in Figure 5.6. F is a convolutional neural network, and the convolutional kernels on each layer have been pretrained to extract certain types of image features, such as edges and shapes. With it, we extract the features from a generated image, as well as its corresponding ground truth image, and minimize the difference between those features to improve the quality of the generated image (see details in Section 5.3.2.1). Specifically, we use the layer `relu1_2` to extract feature maps for comparison based on two observations. First, early layers such as the layer `relu1_2` of the VGG-19 network focus on low-level features such as edges and basic shapes, which commonly exist in scientific visualization images. Second, through our experiments we found that artifacts are introduced into the generated image by pooling layers (e.g., `pool1` in Figure 5.6). Hence, we use the layer `relu1_2` that is before the first pooling layer.

5.3.2 Loss Function

Given an image $\hat{I}$ generated using R_ω and the corresponding ground truth image I, a loss function $\mathscr{L}$ is defined by measuring the difference between them. Minimizing $\mathscr{L}$ can, therefore, be conducted by updating the parameters ω of R_ω over an iterative training process. The most straightforward choice for $\mathscr{L}$ is the average of the pixel wise distance between $\hat{I}$ and I, such as the mean squared error. As shown in earlier works [70, 73, 86], however, the average pixel wise distance often produces over-smoothed images, lacking high-frequency features.

In this work, we define $\mathscr{L}$ by combining two advanced loss functions: a feature reconstruction loss [73] $\mathscr{L}_{feat}^{F,l}$ and an adversarial loss [54] $\mathscr{L}_{adv\_R}$, namely

$$\mathscr{L} = \mathscr{L}_{feat}^{F,l} + \lambda \mathscr{L}_{adv\_R}, \tag{5.2}$$

where λ is the coefficient between them. $\mathscr{L}_{feat}^{F,l}$ measures the difference between features extracted from the feature comparator F using its convolutional layer l, whereas $\mathscr{L}_{adv\_R}$ quantifies how easily the discriminator D_υ can differentiate the generated images from real ones. As can be seen, minimizing $\mathscr{L}_{adv\_R}$ requires training R_ω and D_υ together in an adversarial manner (i.e., the adversarial theory of GANs [54]). In order to train D_υ, an adversarial loss $\mathscr{L}_{adv\_D}$ is used.

5.3.2.1 Feature Reconstruction Loss

The feature reconstruction loss between image $\hat{I}$ and I is defined by measuring the difference between their extracted features [70, 73, 86]. Specifically, for a given image I, our feature comparator F (the pretrained VGG-19) is applied on I and extracts a set of feature maps, denoted as $F^l(I)$. Here l indicates which layer the feature maps are from (e.g., the `relu1_2` of F). The extracted feature maps can be considered as a 3D matrix of dimension $h \times w \times c$, where h, w, and c are the height, width, and number of channels, respectively, as shown in Figure 5.6. The feature reconstruction loss between $\hat{I}$ and I can, therefore, be defined as the pixel wise mean squared error between $F^l(\hat{I})$ and $F^l(I)$. Extending this definition to a batch of images, the feature reconstruction loss between $\hat{I}_{0:b-1}$ and $I_{0:b-1}$ (b is the batch size) is

$$\mathscr{L}_{feat}^{F,l} = \frac{1}{hwcb} \sum_{i=0}^{b-1} \|F^l(I_i) - F^l(\hat{I}_i)\|_2^2. \tag{5.3}$$

Using the feature reconstruction loss enables our regressor to produce images sharing similar feature maps with the corresponding ground truth images, which lead to images with sharper features.

5.3.2.2 Adversarial Loss

In addition to the feature reconstruction loss described above, we add an adversarial loss $\mathscr{L}_{adv\_R}$ into the loss function. Unlike the feature reconstruction loss, which measures the difference between each pair of images, the adversarial loss focuses on identifying and minimizing the divergence between two image distributions following the adversarial theory of GANs. Specifically, our discriminator D_υ is trained along with the regressor R_ω to differentiate images generated by R_ω with ground truth images. As the regressor R_ω becomes stronger over the training, the discriminator D_υ is forced to identify more subtle differences between the generated images and the ground truth.

The adversarial loss can be used as complementary to the feature reconstruction loss for two reasons. First, the feature reconstruction loss focuses on the average difference between images, and the adversarial loss focuses on local features that are the most important to differentiate the predicted and ground truth images. Second, the feature reconstruction loss compares the difference between each pair of the generated and ground truth images, and the adversarial loss measures divergence between two image distributions.

In this work, we use the standard adversarial loss presented in [54], which uses different loss functions for the generator and discriminator. For the generator (i.e., our regressor R_ω), the adversarial loss is

$$\mathscr{L}_{adv\_R} = -\frac{1}{b}\sum_{i=0}^{b-1} \log D_\upsilon(\hat{I}_i), \tag{5.4}$$

which reaches the minimum when the discriminator cannot differentiate the generated images from the ground truth images. This loss is combined with the feature reconstruction loss to update our regressor (Equation 5.2). The adversarial loss of the discriminator is defined as

$$\mathscr{L}_{adv\_D} = -\frac{1}{b}\sum_{i=0}^{b-1}(\log D_{\upsilon}(I_i) + \log(1 - D_{\upsilon}(\hat{I}_i))), \tag{5.5}$$

which estimates the divergence between the distribution of the generated images and the ground truth images.

5.3.3 Techniques to Stabilize Training

We use several techniques to stabilize the adversarial training of R_{ω} and D_{υ}. The instability of adversarial trainings is a well-known problem [54], especially when the resolution of synthesized images is high [15]. The previous work [15] divided the training into two stages for stabilization. In the first stage, the opacity GAN that produces 64×64 opacity images is trained, whereas the opacity-to-color translation GAN is trained in the second stage to produce 256×256 color images, conditioning on the 64×64 opacity images. In this work, we train a single pair of adversarial networks (i.e., R_{ω} and D_{υ}) that directly produces 256×256 color images with the help of recent techniques in stabilizing the adversarial training, including the spectral normalization [98] and the two time-scale update rule (TTUR) [64].

5.3.3.1 Spectral Normalization

Spectral normalization [98] is used to mitigate the instability of the discriminator, which is a major challenge in stabilizing the adversarial training. Spectral normalization is a weight normalization technique, which outperforms other weight normalization techniques in many image synthesis tasks as shown in [81]. Spectral normalization normalizes the weight matrix

of each layer based on the first singular value of the matrix. With spectral normalization, the discriminator is enforced to be Lipschitz continuous, such that the discriminator is constrained and stabilized to some extent. Spectral normalization is applied on each layer of the discriminator without changing the network architecture; hence spectral normalization is not labeled in Figure 5.5.

5.3.3.2 Learning Rate

The learning rates of the regressor and the discriminator are critical for the stability of the adversarial training. This work uses the Adam optimizer [76] that changes the learning rate of each weight dynamically during training with respect to the momentum of the weight gradients. In detail, the learning rate in the Adam optimizer is influenced by three hyperparmeters: the initial learning rate α, the first-order momentum β_1, and the second-order momentum β_2. In order to stabilize the training, a small α is often preferred; and we found that 5×10^{-5} stabilized the training for our cases. In addition, we found that a bigger β_1 often cripples the training and set β_1 to 0 as suggested in [27,165]. Compared with β_1, β_2 has less influence on the stability of the training, which is set to 0.999 in experiments.

In previous works on training GANs, we found that people often update the discriminator more frequently than the generator, because they do not want to update the generator based on a discriminator that is not strong enough. Doing so, however, leads to a longer training time. Our work uses the same update frequency for the regressor and discriminator but with different learning rates α_D and α_R (i.e., the TTUR technique [64]). Based on the empirical results shown in [27,165], we set the learning rate of the discriminator to be 4 times that of the regressor, that is, $\alpha_D = 2 \times 10^{-4}$ and $\alpha_R = 5 \times 10^{-5}$.

Algorithm 1 Training process of InSituNet.

Require: Training data includes parameters $\{P_{sim}, P_{vis}, P_{view}\}_{0:N-1}$ and the corresponding images $I_{0:N-1}$. Initial weights ω and υ of R_ω and D_υ, respectively. The feature comparator F.

Ensure: Optimized weights ω and υ

1: Repeat:
2: $\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1}, I_{0:b-1}$ sampled from training data
3: $\hat{I}_{0:b-1} \leftarrow R_\omega(\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1})$
4: $\upsilon \leftarrow \text{Adam}(\nabla_\upsilon \mathscr{L}_{adv\_D}(I_{0:b-1}, \hat{I}_{0:b-1}; \upsilon), \upsilon, \alpha_D, \beta_1, \beta_2)$
5: $\omega \leftarrow \text{Adam}(\nabla_\omega \mathscr{L}(I_{0:b-1}, \hat{I}_{0:b-1}; \omega), \omega, \alpha_R, \beta_1, \beta_2)$
6: Until exit criterion is satisfied

5.3.4 Training Process

The process of training our regressor and discriminator is shown in Algorithm 1. Given the training data collected in situ, namely, N pairs of paramters $\{P_{sim}, P_{vis}, P_{view}\}_{0:N-1}$ and the corresponding images $I_{0:N-1}$, we first initialize the network weights ω and υ using the orthogonal initialization [135]. Then, the discriminator and regressor are updated alternatively by using the stochastic gradient descent until the exit criterion is satisfied. The exit criterion used in this work is the maximum number of iterations, which is set to 125,000 because the loss converged in our cases after 125,000 iterations.

In each iteration, a batch of parameters $\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1}$ and the corresponding images $I_{0:b-1}$ are sampled from the training data (line 2), where b is the batch size. Next, the current R_ω takes $\{P_{sim}, P_{vis}, P_{view}\}_{0:b-1}$ as inputs and produces $\hat{I}_{0:b-1}$ (line 3). According to the loss $\mathscr{L}_{adv\_D}$ defined on $I_{0:b-1}$ and $\hat{I}_{0:b-1}$ in Equation 5.5, the weights of the discriminator are updated (line 4). Similarly, the weights of the regressor are updated as well, according to the loss function $\mathscr{L}$ (defined in Equations 5.2, 5.3, and 5.4), which is computed using the feature comparator F and the updated discriminator D_υ (line 5). When updating the weights υ and ω, the gradients ∇_υ and ∇_ω of the loss functions $\mathscr{L}_{adv\_D}$ and $\mathscr{L}$ are computed,

respectively. With ∇_{υ} and ∇_{ω}, the weights υ and ω are updated through two Adam optimizers using the learning rates discussed in the preceding section.

5.4 Parameter Space Exploration with InSituNet

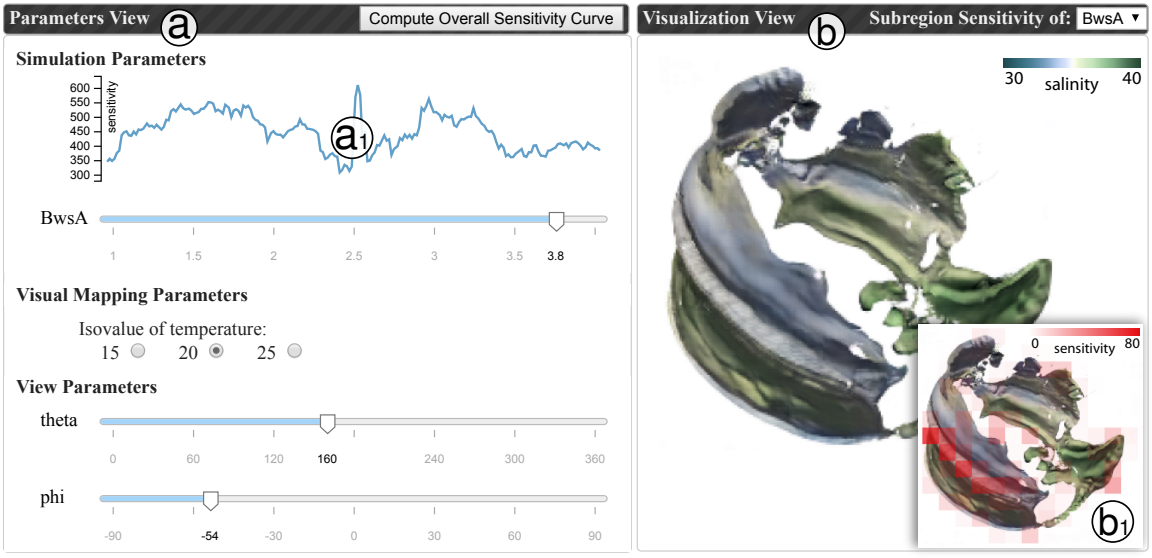


Figure 5.7: Visual interface developed for parameter space exploration. (a) The three groups of parameters: simulation, visual mapping, and view parameters. (b) The predicted visualization images and the sensitivity analysis results.

With the trained InSituNet, users can perform parameter space exploration of ensemble simulations from two perspectives. First, with InSituNet's forward propagations, users can interactively infer the visualization results for arbitrary parameters within the parameter space. Second, using InSituNet's backward propagations, users can investigate the sensitivity of different parameters and thus have better understanding on parameter selections. To support the parameter space exploration, we built an interactive visual interface as shown in Figure 5.7, which contains two views: Parameters View (Figure 5.7(a)) and Visualization

View (Figure 5.7(b)). In the following, we explain how users can perform parameter space exploration with this visual interface.

5.4.1 Inference of Visualization Results

InSituNet is able to interactively infer the visualization results for any user-selected parameter values. As shown in Figure 5.7(a), the three groups of input parameters for the InSituNet are visualized by using different GUI widgets. For the simulation and view parameters, because their values are usually in continuous ranges, we visualize them using slider bars whose ranges are clipped to the corresponding parameters' predefined value ranges. Users are able to select arbitrary parameter values by interacting with those sliders. For the visual mapping parameters, users can switch among a set of predetermined options using the ratio buttons, for example, selecting different isovalues for isosurface visualizations, as shown in Figure 5.7(a).

The selected values for the three groups of parameters are fed into the trained InSituNet. Through a forward propagation of the network, which takes around 30 ms, the corresponding visualization image for the given set of parameters is generated and visualized in the Visualization View, as shown in Figure 5.7(b).

5.4.2 Sensitivity Analysis on Simulation Parameters

Because InSituNet is differentiable, users can perform sensitivity analysis for the simulation parameters using the network's backward propagations. Specifically, users can compute the derivative of a scalar value derived from the generated image (e.g., L_1 norm of the pixel values) with respect to a selected simulation parameter. The absolute value of the derivative can be treated as the sensitivity of the parameter, which indicates how much the generated image will change if the parameter gets changed. Note that the sensitivity analysis in this

work is used to reflect the changes (with respect to the parameters) in the image space rather than the data space. Inspired by [15], our analysis includes overall sensitivity analysis and subregion sensitivity analysis.

In overall sensitivity analysis, we focus on analyzing the sensitivity of the entire image with respect to each simulation parameter across its value range. To this end, we sweep each parameter across its value range while fixing the values of other parameters. Images are then generated from the selected parameter values and aggregated into a scalar (i.e., the L_1 norm of the pixel values). The aggregated scalar values are then back propagated through the InSituNet to obtain the sensitivity of the selected parameter values. In the end, a list of sensitivity values is returned for each parameter and visualized as a line chart on top of the slider bar corresponding to the parameter (Figure 5.7(a1)) to indicate how sensitive the parameter is across its value range.

In subregion sensitivity analysis, we analyze the sensitivity of a selected parameter for different subregions of the generated image. This analysis is done by partitioning the visualization image into blocks and computing the sensitive of the parameter for the L_1 norm of the pixel values in each block. The computed sensitivity values are then color coded from white to red and overlaid on top of the visualization image to indicate what regions are more sensitive with respect to the selected parameter (red blocks in Figure 5.7(b1)).

5.5 Results

We evaluated InSituNet using combustion, cosmology, and ocean simulations (Section 5.5.1) from four aspects: (1) providing implementation details and analyzing performance (Section 5.5.2); (2) evaluating the influence of different hyperparameters (Section 5.5.3); (3) comparing with alternative methods (Section 5.5.4); and (4) performing parameter space exploration and analysis with case studies (Section 5.5.5).

5.5.1 Ensemble Simulations

Table 5.1: Datasets: k controls the size of InSituNet to cope with datasets in different complexities; diversity [153] measures how diverse the generated images are.

Simulation	P_{sim}		P_{vis}	P_{view}		k	Diversity
	Name	Number		Name	Number		
SmallPoolFire	Ck, C	4,000	pseudo-coloring with 5 color schemes	N/A	N/A	32	2.72
Nyx	OmM, OmB, h	500	volume rendering with a transfer function	θ, ϕ	100	48	1.72
MPAS-Ocean	$BwsA$	300	isosurface visualization with 3 isovalues	θ, ϕ	100	48	1.75

We evaluated the proposed approach using three ensemble simulations: SmallPoolFire [158], Nyx [7], and MPAS-Ocean [127]. They are summarized in Table 5.1 and detailed below.

SmallPoolFire is a 2D combustion simulation from the OpenFOAM simulation package [158]. We used it as a test case to evaluate InSituNet by studying two parameters: a turbulence parameter $Ck \in [0.0925, 0.0975]$ and a combustion parameter $C \in [4.99, 5.01]$. We sampled 4,000 parameter settings from the parameter space: 3,900 for training and 100 for testing. Images were generated for the temperature field by using pseudo-coloring with five predefined color schemes. To study how diverse the generated images are, we use the method proposed by Wang et al. [153], which measures the diversity as the reciprocal of

the average structural similarity (SSIM) between every pair of images. The diversity of the images in this dataset is 2.72, which means the average SSIM is smaller than 0.4.

Nyx is a cosmological simulation developed by Lawrence Berkeley National Laboratory. Based on the scientists' suggestion, we studied three parameters: the total matter density ($OmM \in [0.12, 0.155]$), the total density of baryons ($OmB \in [0.0215, 0.0235]$), and the Hubble constant ($h \in [0.55, 0.85]$). We sampled 500 parameter settings from the parameter space: 400 for training and 100 for testing. The simulation was conducted with each parameter setting and generated a $256 \times 256 \times 256$ volume representing the log density of the dark matters. The volume was visualized in situ by using volume rendering with a predefined transfer function of the wave colormap[1] and from 100 different viewpoints. The diversity of the generated images is 1.72.

MPAS-Ocean is a global ocean simulation developed by Los Alamos National Laboratory. Based on the domain scientists' interest, we studied the parameter that controls the bulk wind stress amplification ($BwsA \in [1, 4]$). We generated 300 ensemble members with different *BwsA* values. We used 270 of them for training and the rest for testing. The isosufaces of the temperature field (with isovalue={15, 20, 25}) were extracted and visualized from 100 different viewpoints for each ensemble member. The isosurfaces were colored based on salinity, using the colormap suggested by Samsel et al. [132]. The diversity of the generated images is 1.75.

5.5.2 Implementation and Performance

The proposed approach consists of three components: the in situ data collection, the training of InSituNet, and the visual exploration and analysis component. We discuss the implementation details and performance of the three components in the following.

[1] https://sciviscolor.org

Table 5.2: Timings: t_{sim}, t_{vis}, and t_{tr} are timings for running ensemble simulations, visualizing data in situ, and training InSituNet, respectively; t_{fp} and t_{bp} are timings for a forward and backward propagation of the trained InSituNet, respectively.

Simulation	Size (GB)			Performance				
	Raw	Image	Network	t_{sim} (hr)	t_{vis} (hr)	t_{tr} (hr)	t_{fp} (s)	t_{bp} (s)
SmallPoolFire	≈25.0	0.43	0.06	1,420.0	6.45	16.40	0.031	0.19
Nyx	≈30.0	3.92	0.12	537.5	8.47	18.02	0.033	0.22
MPAS-Ocean	≈300.0	3.46	0.15	229.5	10.73	18.13	0.033	0.23

The in situ visualization was implemented by using ParaView Catalyst[2] following the Cinema framework [3, 4]. The simulations and in situ visualization were conducted on a supercomputer of 648 computation nodes. Each node contains an Intel Xeon E5-2680 CPU with 14 cores and 128 GB of main memory. We used 1, 28, and 128 processes, respectively, for the SmallPoolFire, Nyx, and MPAS-Ocean simulations. InSituNet was implemented in PyTorch[3] and trained with an NVIDIA DGX-1 system, which contains 8 NVIDIA V100 GPUs with NVlink. The visual interface was implemented based on a web server/client framework. The interface was implemented with D3.js on the client side, and the images were generated from a Python server (with the assist of the trained InSituNet) and sent to the client for visualization. The visual exploration and analysis were tested on a desktop with an Intel Core i7-4770 CPU and an NVIDIA 980Ti GPU.

The space and computation costs using the proposed approach for the three different datasets are listed in Table 5.2. The size of InSituNet is less than 1% and 15% of the raw simulation data and the image data, respectively. The training of InSituNet generally takes more than 10 hours, but the time is much less than actually running the ensemble

[2]https://www.paraview.org/in-situ

[3]https://pytorch.org

simulations with extra parameter settings. After training, a forward or backward propagation of InSituNet takes less than one second on a single NVIDIA 980Ti GPU.

5.5.3 Model Evaluation for Different Hyperparameters

We evaluated InSituNet trained with different hyperparameters (i.e., loss functions, network architectures, and numbers of training samples) qualitatively and quantitatively using the data that were excluded from the training to study two questions: (1) Is InSituNet able to generate images that are close to the ground truth images? (2) How do the choices of hyperparameters influence the training results?

For quantitative evaluations, we used four metrics that focus on different aspects to compare the predicted images with the ground truth images, including peak signal-to-noise ratio (PSNR), SSIM, earth mover's distance (EMD) between color histograms [15], and Fréchet inception distance (FID) [64].

PSNR measures the pixel-level difference between two images using the aggregated mean squared error between image pixels. A higher PSNR indicates that the compared images are more similar pixel wise.

SSIM compares two images based on the regional aggregated statistical information (e.g., mean and standard deviation of small patches) between them. A higher SSIM means the compared images are more similar from a structural point of view.

EMD is used in [15] to quantify the distance between the color histograms of two images. A lower EMD means the compared images are more similar according to their color distributions.

FID approximates the distance between two distributions of images, which is widely used in recent image synthesis works [27, 165] as a complementary to other metrics. A lower FID suggests the two image collections are more similar statistically.

5.5.3.1 Loss Functions

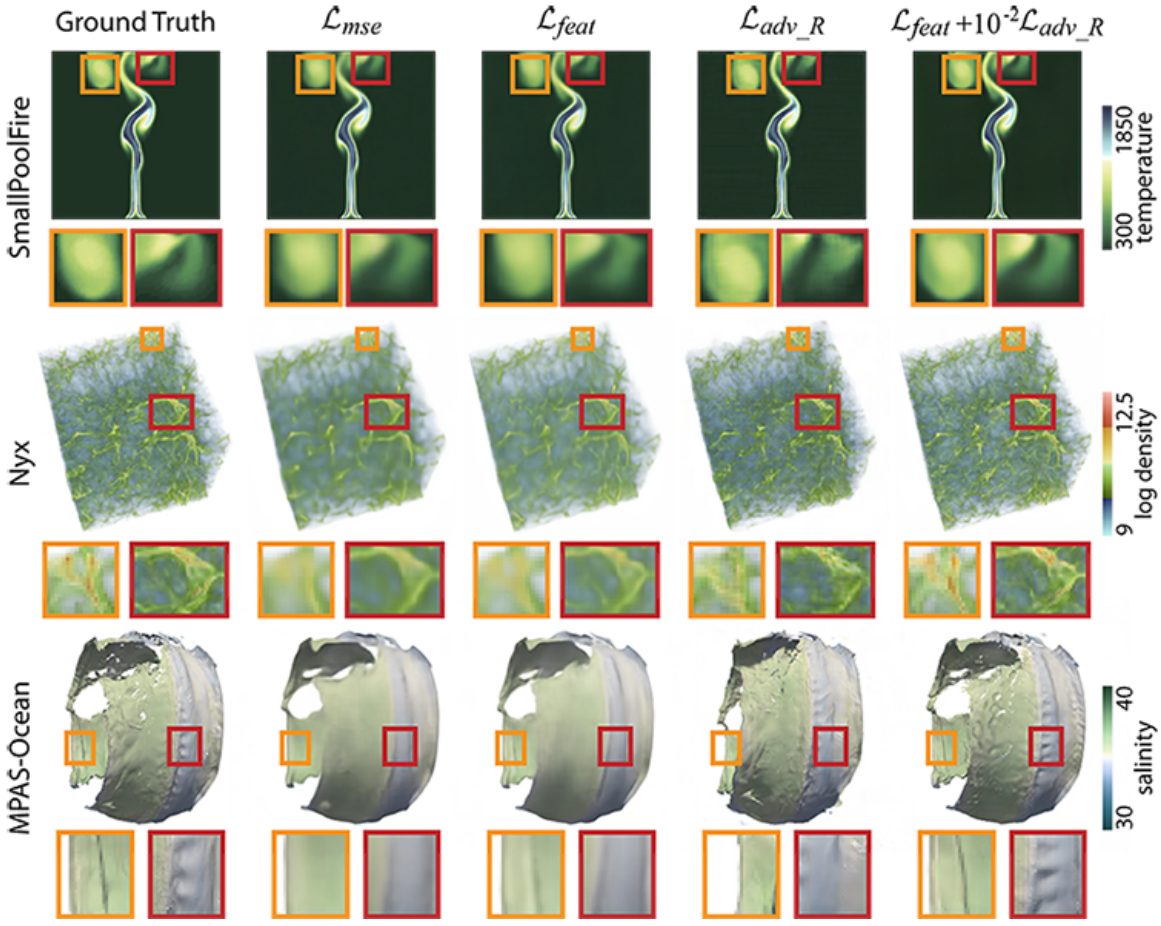


Figure 5.8: Qualitative comparison of InSituNet trained with different loss functions. Combining $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$ gives the results of high quality.

We evaluated InSituNet trained with different loss functions including the mean squared error $\mathscr{L}_{mse}$, the feature reconstruction loss $\mathscr{L}_{feat}$, the adversarial loss $\mathscr{L}_{adv\_R}$, and the combination of $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$.

Figure 5.8 compares the images generated by InSituNet trained with different loss functions with the ground truth. We can see that using $\mathscr{L}_{mse}$ often generates over-smoothed images lacking high-frequency features, whereas using $\mathscr{L}_{feat}$ can mitigate the problem to some extent. Using $\mathscr{L}_{adv\_R}$ can generate images that are as sharp as the ground truth, but the features are often not introduced in the desired positions. By combining $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$, we are able to generate images with sharp features, and those images are also similar to the ground truth.

Table 5.3: Quantitative evaluation of InSituNet trained with different loss functions. The model trained with the combination of $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$ generates images with the best EMD and FID and only a slightly lower PSNR and SSIM compared with the model trained with $\mathscr{L}_{mse}$ or $\mathscr{L}_{feat}$.

		$\mathscr{L}_{mse}$	$\mathscr{L}_{feat}$	$\mathscr{L}_{adv\_R}$	$\mathscr{L}_{feat} + 10^{-2}\mathscr{L}_{adv\_R}$
SmallPoolFire	PSNR	**25.090**	24.937	20.184	24.288
	SSIM	0.9333	**0.9390**	0.8163	0.9006
	EMD	0.0051	0.0064	0.0056	**0.0037**
	FID	21.063	15.881	12.859	**9.4747**
Nyx	PSNR	**31.893**	29.055	24.592	29.366
	SSIM	0.8684	**0.8698**	0.7081	0.8336
	EMD	0.0037	0.0083	0.0064	**0.0022**
	FID	60.825	54.670	24.036	**6.2694**
MPAS-Ocean	PSNR	**26.944**	26.267	17.099	24.791
	SSIM	**0.8908**	0.8885	0.7055	0.8655
	EMD	0.0025	0.0044	0.0036	**0.0017**
	FID	115.74	120.37	28.927	**21.395**

Table 5.3 reports the quantitative results from using different loss functions. We found that using $\mathscr{L}_{mse}$ gives the best PSNR, because the network using $\mathscr{L}_{mse}$ is trained to minimize

the mean squared error (i.e., maximize the PSNR). Using $\mathscr{L}_{feat}$ gives the best SSIM in some cases, because it focuses more on the structure of the images. However, using $\mathscr{L}_{mse}$ or $\mathscr{L}_{feat}$ often results in poor performance regarding EMD and FID. When training InSituNet with $\mathscr{L}_{adv\_R}$, the FID value can be improved, but the values of PSNR and SSIM drop a lot. By combining $\mathscr{L}_{feat}$ and $\mathscr{L}_{adv\_R}$, both EMD and FID improved a lot, though the PSNR and SSIM got slightly worse than using $\mathscr{L}_{mse}$ or $\mathscr{L}_{feat}$.

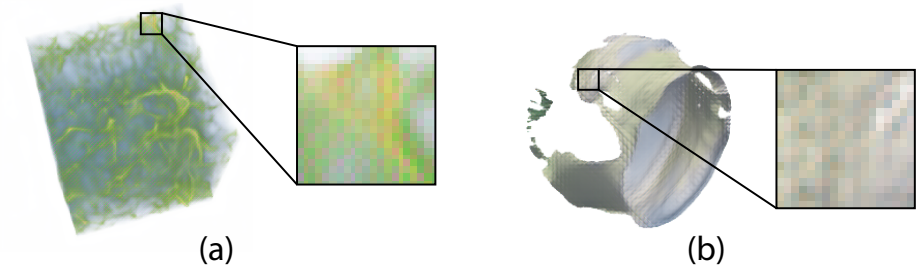


Figure 5.9: Images generated by InSituNet trained with $\mathscr{L}_{feat}$ that uses different layers after the first pooling layer of VGG-19: (a) `relu2_1` and (b) `relu3_1`. Checkerboard artifacts are introduced.

For $\mathscr{L}_{feat}$, using which layer of the pretrained VGG-19 to extract features from images can affect the image synthesis results. Through empirical studies, we found that using any layers after the first pooling layer of VGG-19 will introduce undesired checkerboard artifacts, because of the "inhomogeneous gradient update" of the pooling layer [5], as shown in Figure 5.9. Hence, we use the last layer right before the first pooling layer, which is the layer `relu1_2`.

We also evaluated the influence of the weight λ for $\mathscr{L}_{adv\_R}$ when combining it with $\mathscr{L}_{feat}$ (defined in Equation 5.2), and the results are shown in Table 5.4. We found that increasing λ over 0.01 cannot improve the accuracy of the generated images any further.

Table 5.4: Evaluating the weight λ of $\mathscr{L}_{adv\_R}$: $\lambda = 0.01$ provides the results that balance the PSNR, SSIM, EMD, and FID.

	$\lambda = 0.005$	$\lambda = 0.01$	$\lambda = 0.02$	$\lambda = 0.04$
PSNR	**30.043**	29.366	29.040	27.232
SSIM	**0.8619**	0.8336	0.8253	0.7680
EMD	0.0041	**0.0022**	0.0023	0.0025
FID	21.267	**6.2694**	6.6819	9.8992

In addition, a small λ (i.e., 0.005) will hurt the image accuracy in terms of EMD and FID, although the value of PSNR and SSIM can be improved slightly. We thereby set λ to 0.01 to balance its effects on the four metrics.

5.5.3.2 Network Architectures

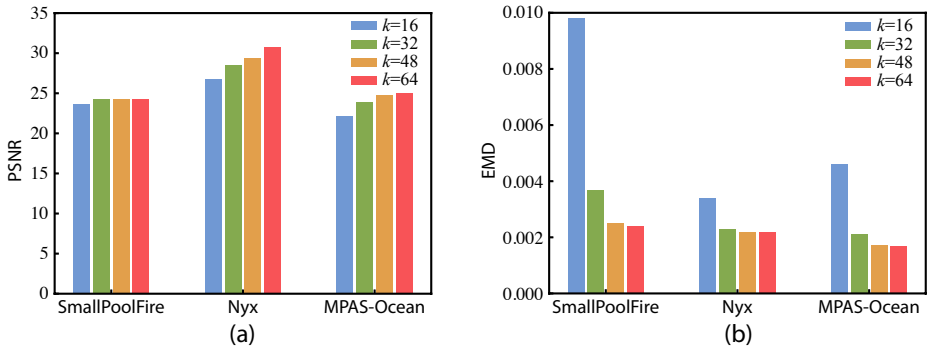


Figure 5.10: Quantitative evaluation of different network architectures controlled by k with (a) PSNR and (b) EMD.

We evaluated InSituNet with different network architectures in terms of the accuracy of predicted images, the network size, and the training time. As mentioned in Section 5.3.1, the architecture of our network is controlled by a constant k, which controls the number of

Table 5.5: Size and training time of different network architectures controlled by k for the Nyx dataset.

	$k = 16$	$k = 32$	$k = 48$	$k = 64$
Network Size (MB)	26.4	67.4	125.2	199.6
Training Time (hr)	13.73	16.42	18.02	20.17

convolutional kernels in the intermediate layers. In this experiment, we evaluated four k values: 16, 32, 48, and 64.

Figure 5.10 shows the PSNR and EMD of images generated by InSituNet with the four k values. We can see that InSituNet with larger k values can generate more (or at least equally) accurate images, because a larger k gives more expressive power to the neural network. On the other hand, training InSituNet with a larger k also costs more time, and more storage will be needed to store the networks, as shown in Table 5.5 using the Nyx dataset as an example. Hence, to balance the accuracy of the generated images and the cost from both computation and storage, we set k to 32, 48, and 48 for the SmallPoolFire, Nyx, and MPAS-Ocean, respectively.

5.5.3.3 Number of Ensemble Runs used for Training

We compared InSituNet trained using different numbers of ensemble runs (Table 5.6) to study how many ensemble runs will be needed to train a good model for the three simulations. We found that this number is different in different simulations, depending on the complexity of the mapping between simulation parameters and visualization results. Experiment results show that the accuracy of generated images becomes stable when the number of ensemble runs is greater than 2,900, 200, and 210 for the SmallPoolFire, Nyx, and MPAS-Ocean

Table 5.6: Evaluation of the number of ensemble runs used for training.

Simulation	# Ensemble Runs	PSNR	SSIM	EMD	FID
SmallPoolFire	900	21.842	0.8714	0.0040	14.398
	1900	23.192	0.9016	0.0036	11.732
	2900	23.932	0.9018	0.0037	9.5813
	3900	24.288	0.9006	0.0037	9.4747
Nyx	100	28.108	0.7951	0.0025	9.8818
	200	29.404	0.8319	0.0022	6.5481
	300	29.398	0.8326	0.0023	6.4239
	400	29.366	0.8336	0.0022	6.2694
MPAS-Ocean	70	24.347	0.8554	0.0016	37.229
	140	24.593	0.8607	0.0017	28.380
	210	24.732	0.8643	0.0017	22.794
	270	24.791	0.8655	0.0017	21.395

simulation, respectively. As a result, we used 3,900, 400, and 270 runs from the three simulations to train InSituNet for the rest of the study.

5.5.4 Comparison with Alternative Methods

Table 5.7: Quantitative comparison of images generated with interpolation, GAN-VR, and InSituNet.

	Network Size	PSNR	SSIM	EMD	FID
Interpolation	N/A	23.932	0.6985	0.0070	58.571
GAN-VR	80.5 MB	20.670	0.6274	0.0056	38.355
InSituNet	**67.5 MB**	**28.471**	**0.8034**	**0.0023**	**9.0152**

We compared our method with two alternative methods including interpolating images from the training data that close to the target image and the GAN-based volume rendering method (GAN-VR) [15] using the Nyx dataset. For the interpolation method, we sample g images from the training data whose parameter settings are the top g closest to the parameter

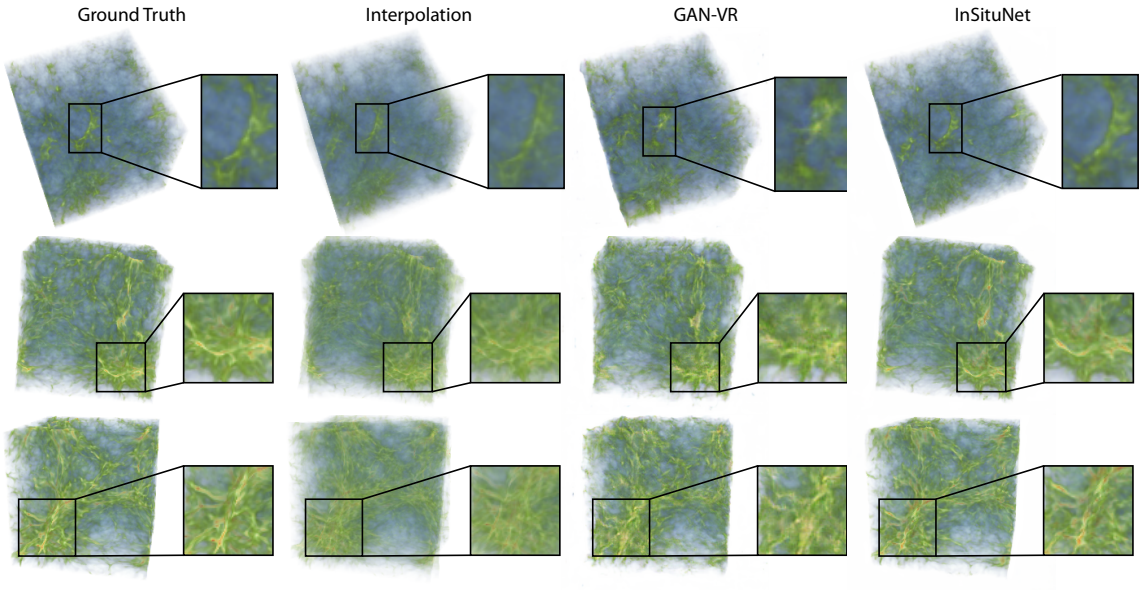


Figure 5.11: Comparison of the images generated using interpolation, GAN-VR, and InSituNet with the ground truth images.

setting of the test image and interpolate the sampled images using inverse distance weighting interpolation [140]. We use $g = 3$ for this experiment. For GAN-VR, we incorporated the simulation parameters into both the opacity GAN and the opacity-to-color translation GAN and removed the transfer function related parameters because we used a fixed transfer function for this dataset. For InSituNet, we selected a network architecture whose size is not greater than the size of GAN-VR network, for a fair comparison.

Figure 5.11 compares the ground truth images with the images generated by using interpolation, GAN-VR, and InSituNet. With the new network architecture (e.g., the projection-based condition incorporation method in Section 5.3.1), loss functions (e.g., the feature reconstruction loss in Section 5.3.2), and training strategies (e.g., the spectral normalization in Section 5.3.3), InSituNet can generate results that better preserve features compared with the other two methods. The quantitative comparisons between the three

methods are shown in Table 5.7. InSituNet outperforms the other two methods in all four metrics.

5.5.5 Parameter Space Exploration

This section demonstrates the effectiveness of our deep image synthesis driven parameter space exploration through case studies on the Nyx and MPAS-Ocean simulations.

5.5.5.1 Case Study with the Nyx Simulation

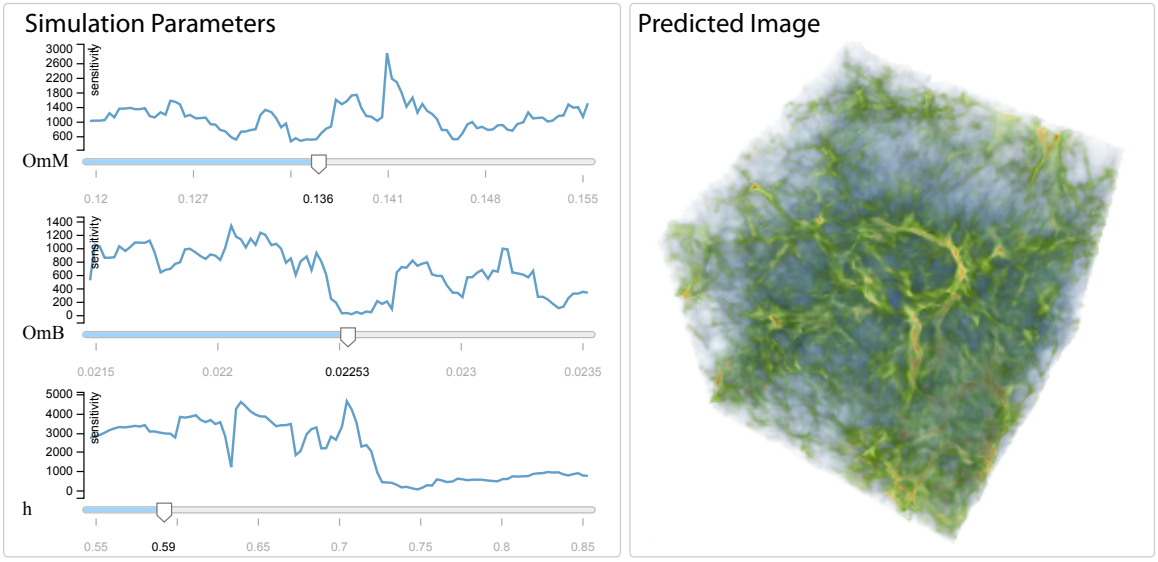


Figure 5.12: Parameter space exploration with the Nyx simulation. For the selected parameter values, the sensitivity of different parameters is estimated and visualized as line charts on the left, whereas the predicted image is visualized on the right.

Our first case study is focused on investigating the influence of different simulation parameters (i.e., *OmM*, *OmB*, and *h*) on the Nyx simulation.

Figure 5.12 shows a selected parameter setting with the predicted visualization image. To understand the influence of each parameter, we computed the sensitivity of the three

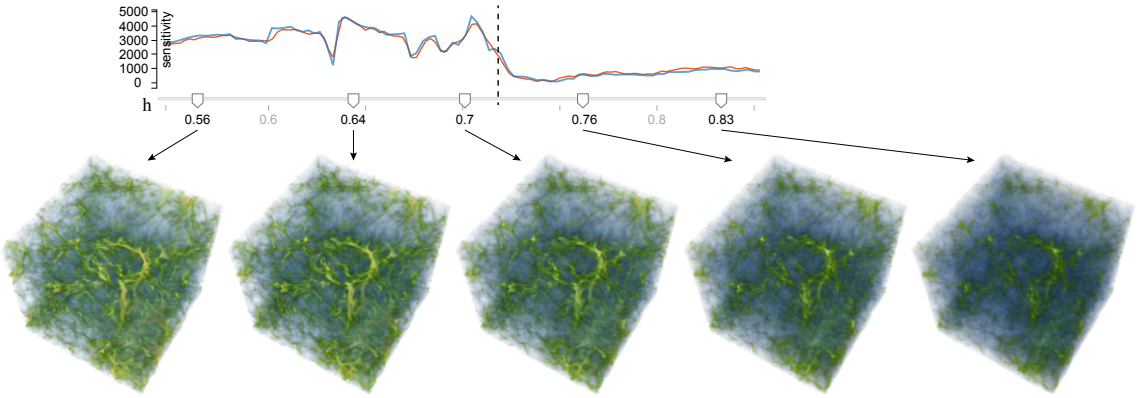


Figure 5.13: Comparison of the visual appearance of the predicted images using different h values to see the effect of this simulation parameter.

parameters with respect to the L_1 norm of the predicted image, shown as the three line charts in Figure 5.12. From the scale of the three charts (i.e., the values along the vertical axes), we see that parameter h is more sensitive to parameter *OmM* and parameter *OmM* is more sensitive to parameter *OmB*.

Focusing on the most sensitive parameter, namely, parameter h, we explored how it affects the visual appearance of the predicted images. Figure 5.13 shows five images predicted by using five different h values, while parameter *OmM* and *OmB* are fixed at the values shown on the two corresponding slider bars in Figure 5.12. We first evaluate the accuracy of the sensitivity curve (blue curve in Figure 5.13) computed by backpropagation with the central difference method. To this end, we first regularly sample the simulation parameters along the curve (128 samples are drawn) and then generate the visualization images with respect to the sampled simulation parameters. The L1 norm of the generated images is then computed and used to compute the sensitive curve using the central difference method. The result is shown as the orange curve in Figure 5.13. We can see that the sensitivity curves generated with the two methods are similar. From the guidance provided

by the line chart in Figure 5.13, we see that parameter h is more sensitive in the first half of its range (i.e., the left side of the dashed line). The three images generated using h values from this range demonstrate a bigger variance compared with the two images shown on the right (which are images generated by using h values from the second half of its range).

5.5.5.2 Case Study with the MPAS-Ocean Simulation

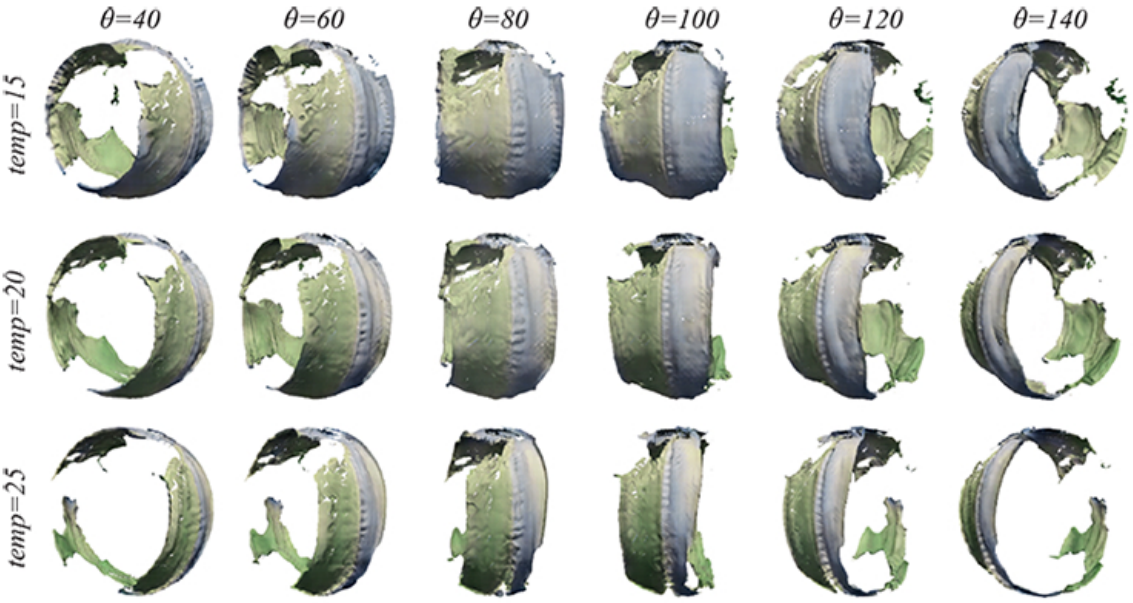


Figure 5.14: Predicted images of the MPAS-Ocean dataset for different isosurfaces and viewpoints, which reasonably reflect the change of view projections and shading effects.

Our next case study explores different parameter settings for the MPAS-Ocean simulation and demonstrates the subregion sensitivity analysis for the simulation parameter *BwsA*, which characterizes the bulk wind stress. Note that here we focus only on exploration and analysis of new parameter settings, the comparison between the predicted and ground truth images is discussed in the previous sections.

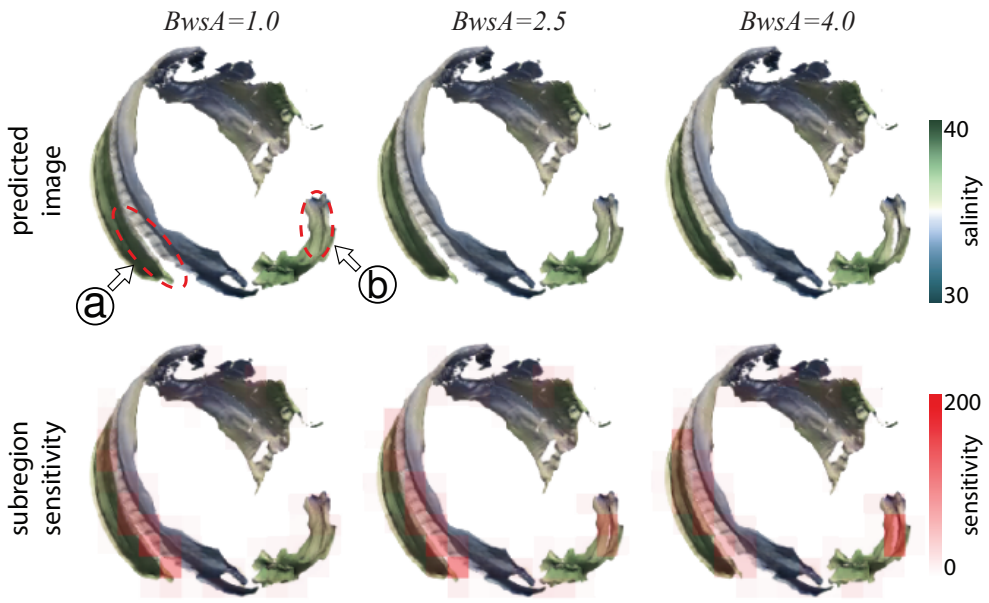


Figure 5.15: Forward prediction (top row) and backward subregion sensitivity analysis (bottom row) for different *BwsA*. Regions that influenced by *BwsA* (i.e., regions a and b) are highlighted by the sensitivity map.

Figure 5.14 shows isosurface visualizations of the temperature field with three different isovalues from six different viewpoints. The value of parameter *BwsA* is fixed at 1 in this study. The images reasonably reflect the change of view projections and shading effects.

Figure 5.15 shows the predicted images when using different *BwsA* values. All images are generated from the temperature field (isovalue=15) of MPAS-Ocean and from the same viewpoint. The first row of images shows the result of forward inference with different *BwsA* values, whereas the second row of images overlays the subregion sensitivity maps onto the corresponding images of the first row. The labeled regions (i.e., Figure 5.15(a) and (b)) change the most when adjusting the value of *BwsA*, and the subregion sensitivity maps on the second row echo these bigger changes, as indicated by the darker red color.

5.6 Limitations, Discussion, and Future Work

This section discusses several directions that we would like to explore in the future: (1) improving the flexibility in exploring arbitrary visual mapping parameters; (2) increasing the accuracy of predicted images; and (3) increasing the resolution of predicted images.

One limitation of our approach is that we restricted the users' ability in exploring arbitrary visual mapping parameters, for example, exhausting all possible transfer functions for volume rendering. Instead, we allow users to switch only among several predefined visual mappings, for example, the three isovalues when exploring the MPAS-Ocean data. Theoretically, training a deep learning model to predict visualization images for arbitrary simulation and visualization parameters is feasible. However, it will require a large number of training images to cover the joint space of all possible simulation and visualization parameters. For example, in order to train a model that can predict volume rendering results of a single volume data for arbitrary transfer functions, 200,000 training images are required, as shown in [15]. Consequently, the size of the training data may even exceed the size of the raw simulation data, which offsets the benefit of in situ visualization. Considering this issue, we would like to explore deep learning techniques that do not require a large number of training samples, such as one- or zero-shot learning, to improve the flexibility of exploration.

Similar to most other machine learning techniques, generating prediction results that are exactly the same as the ground truth is extraordinary difficult. By taking advantage of recent advances in deep learning for image synthesis, the proposed approach has already outperformed other image synthesis based visualization techniques in terms of the fidelity and accuracy of the generated images (see the comparison in Section 5.5). However, we believe further improvement is still possible, and we would like explore other network architectures and/or other loss functions to improve our deep image synthesis model.

Our network architecture limits the resolution of output images to 256×256, which might not be sufficient for some high-resolution simulation data. We believe that our network architecture has the potential to generate images with higher resolutions by adding more residual blocks, and we will investigate this approach in the future.

5.7 Conclusion

In this work, we propose InSituNet, a deep learning based image synthesis model supporting the parameter space exploration of large-scale ensemble simulations visualized in situ. The model is trained to learn the mapping from ensemble simulation parameters to visualizations of the corresponding simulation outputs, conditioned on different visualization settings (i.e., visual mapping and view parameters). With a trained InSituNet, users can generate visualizations of simulation outputs with different simulation parameters without actually running the expensive simulation, as well as synthesize new visualizations with different visualization settings that are not used during the runs. Additionally, an interactive visual interface is developed to explore the space of different parameters and investigate their sensitivity using the trained InSituNet. Through both quantitative and qualitative evaluations, we validated the effectiveness of InSituNet in analyzing ensemble simulations that model different physical phenomena.

Chapter 6: CECAV-DNN: Collective Ensemble Comparison and Visualization using Deep Neural Networks

This chapter focuses on the *collective comparison* of ensemble members. Instead of making comparisons between individual members, we aim at collectively compare different groups of members in this study. The demand for collective ensemble comparison arises from many real-world applications. In climate research, scientists routinely use and compare different simulation models, such as the community atmosphere model (CAM) and the rapid radiative transfer model for general circulation models (RRTMG) [163]. To understand the differences between CAM and RRTMG, scientists conduct ensemble simulations for each model and collectively compare the two ensembles. In the field of fluid dynamics, scientists need to compare simulation models with different spatial resolutions through ensemble runs [1].

Based on our discussion with domain scientists, the purpose of collective comparison between any two ensembles is to answer three specific questions: (1) How to measure the overall dissimilarity between the two ensembles? In other words, what is the dissimilarity between the two distributions of simulation outputs? (2) Which members of the two ensembles agree or disagree with each other? The agreement or disagreement between members depends not only on the similarities between them but also on their probabilities

of occurrence in the two distributions of simulation outputs. (3) What spatial regions are the most important to differentiate the two ensembles?

The collective comparison between ensembles has not been well developed. Although ensemble visualization has been extensively studied recently, the majority of previous ensemble visualization techniques focused on comparing individual members within an ensemble. Several pioneering works in comparing multiple ensembles include: (1) the analysis of simulation parameters for ensembles with different spatial resolutions [20, 155], where the focus is on parameter analysis; (2) visual comparison of multiple collections/ensembles of scalar values [68] or 2D isocontours [49] via juxtaposition or superimposition, where the focus is on qualitative comparison of particular features. However, effective visualization and comprehensive analysis of variability between ensembles of simulation outputs remain challenging.

The key research challenge of collective ensemble comparison is to measure the divergence of two given ensembles. The metric must incorporate all values of the simulation solution, such as values on every grid points, which resides in a high dimensional space. A straightforward approach is to model the probability density function (PDF) of each ensemble, and then measure the divergence between the PDFs. However, the dimensionality of the simulation solution space is usually prohibitively high for modeling and comparing PDFs. Even if we measured the divergence of two PDFs, it is still non-trivial to visualize and analyze where the two PDFs are different (questions (2) and (3)).

We explore to use deep neural networks (DNNs) to tackle the challenge of measuring the divergence of high-dimensional distributions for collective ensemble comparison. DNN is a emerging technology that makes it possible to model and compare distributions of images (e.g. generative models [10, 54, 55, 105, 124]). Without modeling a PDF for a distribution of

images defined on high dimensional space, DNN approaches can differentiate images of one distribution (e.g., fake images) from images of the other distribution (e.g., true images), and also measure the dissimilarity between the two distributions. Similarly, we believe DNN approaches are also promising to visualize and analyze the difference between two ensembles.

In this study, we propose an approach to perform Collective Ensemble Comparison And Visualization using Deep discriminative Neural Networks. A discriminative network is a specific type of DNN, which can compare two probability distributions represented by two sets of samples. In our context, a discriminative network is trained to differentiate members of one ensemble from members of the other. Through training, each individual member (from both ensembles) will be assigned a numerical score by the discriminative network, indicating the likelihood that the member is from one ensemble rather than the other. By analyzing and visualizing the outputs of the discriminative network, our approach provides three-level comparative analysis of the two ensembles to answer the aforementioned questions: (1) We analyze the loss value of the discriminative network to measure the overall dissimilarity between the two ensembles. (2) By visualizing and analyzing the distributions of the two collections of likelihood scores, the agreement and disagreement between individual members of the two ensembles can be studied. (3) Using the internal parameters of the discriminative network, the importance of different spatial locations in differentiating the two ensembles can be characterized. Furthermore, to support the three-level comparative analysis mentioned above, we design and develop a visualization system based on the outputs of the discriminative network. Our system supports not only the three-level comparison of a single pair of ensembles, but also the comparison among multiple pairs of ensembles simultaneously (e.g., a temporal sequence of ensemble pairs).

We demonstrate the effectiveness and usefulness of the proposed approach using two real-world use-cases, and verify the results with a domain scientist from environmental sciences. In the first case, we compare ensembles generated using different weather forecast models in climate research [163]. In the second case, we study the influence of spatial resolutions on the outputs of ensemble simulations in the field of fluid dynamics [1]. With the proposed approach, the difference between ensembles can be identified and visualized effectively. In summary, the contributions of this study are twofold:

- A deep learning approach (CECAV-DNN) for collective comparison and visualization of multiple ensembles. Discriminative networks are trained to identify how and where the ensembles are different
- A visualization system to explore and analyze the outputs of discriminative networks with respect to the difference between ensembles

6.1 Overview

Figure 6.1 shows the workflow of CECAV-DNN to collectively compare multiple ensembles. The input of our approach is a sequence of ensemble pairs (e.g., a temporal sequence of ensemble pairs generated with different simulation models), where each ensemble is a collection of members (i.e. scalar fields). We first train a sequence of discriminative networks to perform comparative analysis on the ensemble pairs. Then, we design and develop a visualization system to facilitate the comparative analysis based on the trained discriminative networks.

We train a discriminative network to differentiate each pair of ensembles. After training, the discriminative network assigns a likelihood score to each member, which indicates the likelihood that the member is from one ensemble rather than the other. Based on

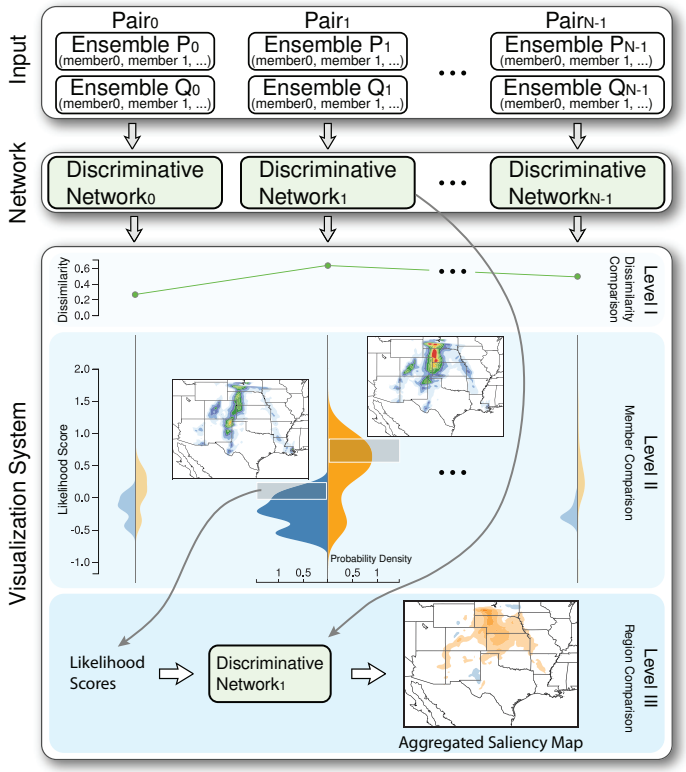


Figure 6.1: Workflow of CECAV-DNN, which takes a sequence of ensemble pairs as input, and trains a discriminative network for each ensemble pair. After training, three levels of comparative analysis are provided with an interactive visualization system to compare the ensemble pairs.

the outputs of the trained discriminative network, our approach provides three levels of comparative analysis. First, we measure the dissimilarity between the two ensembles based on the loss value of the discriminative network. Second, we compare the distributions of the two collections of likelihood scores to identify members in which the two ensembles agree or disagree with each other. Third, by taking advantage of the back-propagation algorithm [130], the spatial regions that are the most sensitive in differentiating the two ensembles are identified.

We design and develop an interactive visualization system to facilitate the analysis on the trained discriminative networks, which empowers the three-level comparative analysis mentioned above (details in Section 6.3). For the first level analysis, the overall dissimilarities are visualized with line charts, which provides informative hints to help users focus on a particular pair in the sequence for further exploration (e.g., the pair of ensembles at a timestep with the maximum dissimilarity). A sequence of violin plots are used to encode and compare the distributions of likelihood scores for the second level analysis. For each violin plot, users can explore different sub-ranges of the PDF via brushing, and visualize the corresponding members as well as the sensitive spatial regions for the third level analysis.

6.2 Collective Ensemble Comparison

Our method starts with a pair of ensembles $P = p_{0:n-1}$ and $Q = q_{0:m-1}$. Each member in the ensembles, p_i or q_i, is a scalar field, and values at different locations of the field denote the simulation output of a certain variable (e.g. temperature, precipitation) from the corresponding ensemble run. Members in individual ensembles share the same spatial region (i.e. same mesh discretization). Mathematically, we consider each member p_i as a high-dimensional vector in $\mathbb{R}^M$, where M is the number of grid points in the spatial field. Then, the two ensembles P and Q are considered as two sets of samples that are sampled from two probability distributions defined on $\mathbb{R}^M$.

Inspired by the recent advances of generative models [10, 54, 55, 105, 124], which use a discriminative network to compare two probability distributions represented by two sets of samples, we train a discriminative network to quantify the dissimilarity between the two ensembles and identify members (or spatial regions of those members) in which the two ensembles are different. Specifically, our discriminative network differentiates members of

one ensemble (P) from members of the other (Q) by assigning each member a likelihood score, such that the difference between the two collections of likelihood scores for members in P and Q is maximized. In this way, we transform the problem of comparing two ensembles of vectors in $\mathbb{R}^M$ into comparing two collections of likelihood scores in $\mathbb{R}$.

In the rest of this section, we first provide foundations for our neural network based approach, and then elaborate the three-level comparative analysis between a single pair of ensembles. We discuss how our approach can be extended to compare multiple pairs of ensembles, which are often required in real-world applications.

6.2.1 Discriminative Networks

In this section, we first describe the basic concepts in the architecture of discriminative networks, and then discuss the objective function and the training process of discriminative networks.

6.2.1.1 Architecture

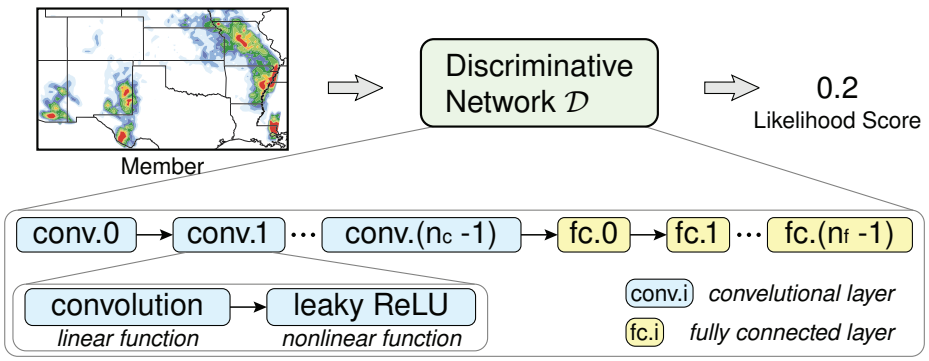


Figure 6.2: Architecture of discriminative network $\mathscr{D}$, which begins with a sequence of convolutional layers, followed by a few fully connected layers.

A discriminative network $\mathscr{D}$ (shown in Figure 6.2) is a nonlinear function. This function maps a member, i.e., a high-dimensional vector, in $\mathbb{R}^M$ to a likelihood score in $\mathbb{R}$. Specifically, $\mathscr{D}$ is implemented through an alternating sequence of linear and nonlinear functions (i.e. activation functions), where each pair of the linear and nonlinear function is commonly referred to as a *layer* of the network. Based on the linear computations in the function pair, two types of layers are commonly used in $\mathscr{D}$: convolutional layer and fully connected layer.

Convolutional Layers Convolutional layers perform linear convolutions to extract features from the input. Because we work with scientific data that has spatial continuities, our $\mathscr{D}$ starts with a sequence of such computations to elicit features layer by layer. Each convolutional layer consists of many trainable *filters* (i.e. convolutional kernels), and each filter can extract a specific type of spatial features from the input. Also, because the convolution operations are performed with a fixed stride, the output of each convolutional layer is usually a down-sampled version of the input, in which certain features are highlighted.

Fully Connected Layers The fully connected layers of our discriminative network reduce the features maps from the last convolutional layer to a numerical value. Each fully connected layer performs a matrix multiplication to assign weights to different elements of the input and aggregate those elements. To the last fully connected layer, the original input (outputs from the last convolutional layer) is reduced to a numerical value (i.e. a likelihood score).

Activation Function The output from both convolutional and fully connected layers will be fed into a nonlinear activation function to filter out inactivate elements from the outputs. Same as the generative models [10, 55, 124], we use Leaky Rectified Linear Units (ReLUs) as the activation function for all layers of $\mathscr{D}$ to accelerate the convergence of the

training process, which is defined as

$$a(x) = \begin{cases} x, & \text{if } x > 0 \\ -cx, & \text{otherwise} \end{cases}, \tag{6.1}$$

where c is a constant value, and typically set to 0.2 in $\mathscr{D}$ [124].

The trainable parameters of $\mathscr{D}$ are the filters in convolutional layers and the weights in fully connected layers. In this work, we denote a discriminative network defined by a collection of parameters ϕ as $\mathscr{D}_\phi$. To train a discriminative network $\mathscr{D}_\phi$ that can differentiate two given ensembles, the parameters ϕ need to be optimized based on an objective function through an iterative training process, which are detailed in the following sections.

6.2.1.2 Objective Function

Given two ensembles P and Q, we map them into two collections of likelihood scores through $\mathscr{D}_\phi$, and use an objective function to optimize the mapping, so that the difference between the two collections of likelihood scores is maximized. More importantly, from the resulting collections of likelihood scores and the optimized parameters ϕ, we are able to detect members in which, and spatial regions where, the two ensembles are different.

Various objective functions [10, 54, 55, 105] have been proposed for the training of a discriminative network to differentiate two probability distributions represented by two sets of samples. The maximum value of a specific objective function usually reflects the distance or divergence between the two distributions. In this work, we demonstrate our approach using the objective function proposed in [10, 55], which has a maximum value corresponds to the Wasserstein distance (i.e. earth mover's distance) [102]. By treating two ensembles P and Q as two sets of samples from two probability distributions, the Wasserstein distance $W(P,Q)$ between the two distribution is defined as:

$$W(P,Q) = \sup_{f \in \mathscr{F}} \left(\mathbb{E}_{p \sim P}[\,(f(p))\,] - \mathbb{E}_{q \sim Q}[\,(f(q))\,] \right), \tag{6.2}$$

where p and q are members sampled from P and Q, respectively, $\mathbb{E}(\cdot)$ represents expectation, f is a 1-Lipschitz function (i.e. a function that satisfies $|f(x)-f(y)| \leq |x-y|$ for all x and y) mapping from $\mathbb{R}^M$ to $\mathbb{R}$, $\mathscr{F}$ is a class of functions that are all 1-Lipschitz. By limiting $\mathscr{D}_\phi$ as a parameterized family of functions that are all 1-Lipschitz (details in [10]), the objective function L is defined as:

$$L(P,Q;\phi) = \mathbb{E}_{p\sim P}[\mathscr{D}_\phi(p)] - \mathbb{E}_{q\sim Q}[\mathscr{D}_\phi(q)]. \tag{6.3}$$

Through iteratively maximizing the objective function, the Wasserstein distance between P and Q can be measured.

Although we focus on the objective function corresponding to the Wasserstein distance in our study, the proposed method is flexible to use objective functions corresponding to other distances or divergences, which will be discussed in Section 6.6.

6.2.1.3 Training Process

As shown in Algorithm 2, we discuss the training process used to optimize the parameters ϕ of a discriminative network $\mathscr{D}_\phi$ with respect to the objective function $L(P,Q;\phi)$. We use stochastic gradient descent [24] to iteratively optimize the parameters ϕ, as shown in Algorithm 2. In each iteration, a batch of members are sampled from each of the two ensembles randomly and fed into the neural network (lines 2–3). Then, the gradient of the objective function is computed with respect to the current parameters ϕ using backpropagation [130], which computes gradients from output to input layer by layer (i.e., starting from the last layer and propagating back to the first layer). Based on the resulting gradients, the parameters ϕ are updated (line 4) using the Adam optimizer with default settings (i.e. $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) [76], the one that has been widely used in the training of

discriminative networks. We keep executing the loop (lines 1–5) of updating the parameters ϕ until the exit criteria is satisfied (e.g. reaching the maximum number of iterations).

Algorithm 2 Training process of the discriminative network $\mathscr{D}_\phi$. Parameters ϕ are initialized by sampling from a Gaussian distribution randomly. b is the bath size. The function `random_sample`(P,b) randomly samples b members from the input ensemble P. L is the objective function. α, β_1, β_2 are the parameters of the Adam optimizer [76]. $\nabla_\phi L(P', Q'; \phi)$ is the gradient of L with respect to ϕ.

Require: Initial parameters ϕ of the discriminative network, ensembles P and Q
Ensure: Optimized parameters ϕ
1: Repeat:
2: $P' \leftarrow$ `random_sample`(P,b)
3: $Q' \leftarrow$ `random_sample`(Q,b)
4: $\phi \leftarrow \text{Adam}(\nabla_\phi L(P', Q'; \phi), \phi, \alpha, \beta_1, \beta_2)$
5: Until exit criteria is satisfied

6.2.2 Three-Level Comparative Analysis of Two Ensembles

In this section, we explain how we use the trained discriminative network $\mathscr{D}_\phi$ to collectively compare a pair of ensembles. By analyzing and visualizing the result of the objective function, the two collections of likelihood scores produced by $\mathscr{D}_\phi$, and the network parameters ϕ, our approach provides three levels of comparative analysis between the two ensembles, which are detailed as follows.

Level I: Dissimilarity Comparison This level measures the overall dissimilarity between two given ensembles. It is conducted by optimizing the objective function (Equation 6.3) of $\mathscr{D}_\phi$ for the two ensembles. To the end, the maximum value of the objective function can measure the distance between the two probability distributions represented by the two sets of samples (i.e., the two input ensembles).

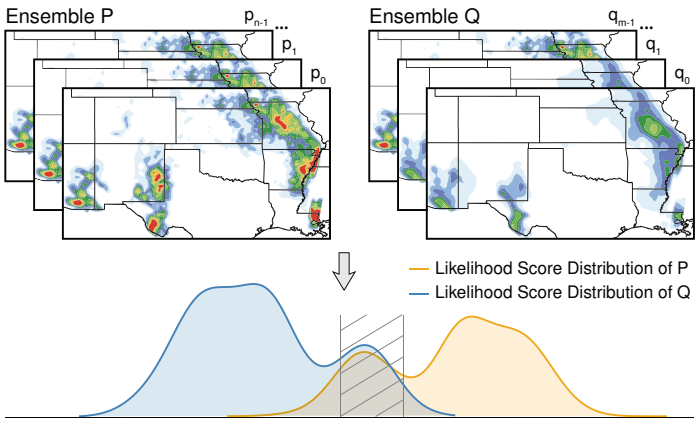


Figure 6.3: Comparing two ensembles using the two distributions of likelihood scores produced by the trained discriminative network $\mathscr{D}_\phi$.

Level II: Member Comparison This level identifies the members in which the two ensembles are more (or less) similar with each other. In other words, it compares the probabilities of occurrence for individual members in the two ensembles. The output of $\mathscr{D}_\phi$, i.e., a likelihood score, indicates the likelihood that the input member is from one ensemble rather than the other. By mapping the two ensembles to two collections of likelihood scores through $\mathscr{D}_\phi$ (Figure 6.3) and modeling them to two distributions, we can more effectively compare the original high-dimensional members in $\mathbb{R}^M$. For example, for the ranges (of the likelihood scores) where the two distributions have similar heights (the shaded area in Figure 6.3), their corresponding members have similar probability of occurrence in both ensembles.

Level III: Region Comparison This level extracts spatial regions, which are sensitive to differentiate the two ensembles (Figure 6.4). It is conducted by computing how strong each spatial location of each member is affecting the likelihood score from $\mathscr{D}_\phi$. Specifically, one member $p_i \in \mathbb{R}^M$ can be denoted as $p_i = (s_0, s_1, \ldots s_{M-1})$, where s_j $(j \in [0, M-1])$

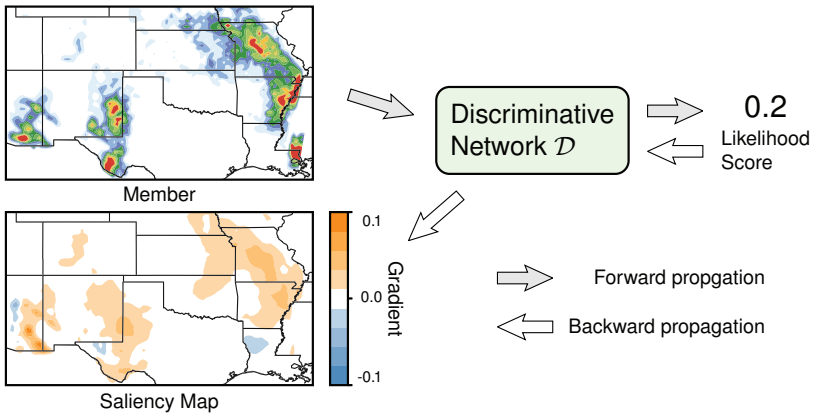


Figure 6.4: Saliency map of a member computed using backpropagation.

is the value associated with the jth spatial location (grid point) of the member; and the likelihood score for this member is $\mathscr{D}_\phi(p_i)$. By computing the gradient of $\mathscr{D}_\phi(p_i)$ with respect to each s_j, i.e. $\frac{\partial \mathscr{D}_\phi}{\partial s_j}$, using backpropagation [130], we can disclose the influence of s_j on the likelihood score of p_i from two aspects:

- The magnitude of the gradient measures the sensitivity of s_j with respect to the likelihood score. When the gradient is high, it means a little change of s_j affects the likelihood score of p_i strongly;
- The sign of the gradient indicates that when s_j increases, the member p_i is more likely to come from one ensemble or the other.

Extending the computation of the gradient to all $s_j \in p_i$, we derive a saliency map [143] for p_i, which has the same size with p_i and highlights the spatial locations whose values are the most important for differentiating the two ensembles, as shown in Figure 6.4.

6.2.3 Extending to Multiple Pairs of Ensembles

We noticed that comparisons are often needed among multiple pairs of ensembles in real-world applications. For example, in order to analyze two spatiotemporal climate ensembles, scientists often need to compare the two ensembles at individual timesteps (i.e., a sequence of ensemble pairs). Our proposed three-level comparative analysis between a single pair of ensembles can easily be extended to multiple pairs. For example, the single numerical value in level one will become a sequence of numerical values and the two probability distributions in level two will become a sequence of distribution pairs.

As we extending the analysis scope (from one pair to multiple pairs of ensembles) to cope with real-world applications, we realize the need of a visualization system. The visualization system can help us better organize different facets of the complicated ensemble data and facilitate our three-level comparative analysis. Moreover, a process of visual exploration emerges to be necessary to avoid information overload. For example, when facing with two large temporal sequence of ensembles, users should be able to get an effective overview of them first, and the overview should provide informative guidance to users' further detailed investigations. Friendly user interactions to help flexibly switch between the overview and details would significantly improve the exploration experience. In an attempt to address these requirements, we design a visualization system to support our three-level comparative analysis for multiple pairs of ensembles. The details of the system is further explained in Section 6.3.

6.3 Visualization System for Collective Comparison

We design and develop an visualization system (Figure 6.5) to help domain scientists compare ensembles collectively. The visualization system support the three-level comparative analysis by visualizing and analyzing the outputs of the trained discriminative networks. In this section, we describe the design considerations and choices of our interface, and provide guidelines for visual exploration using our system.

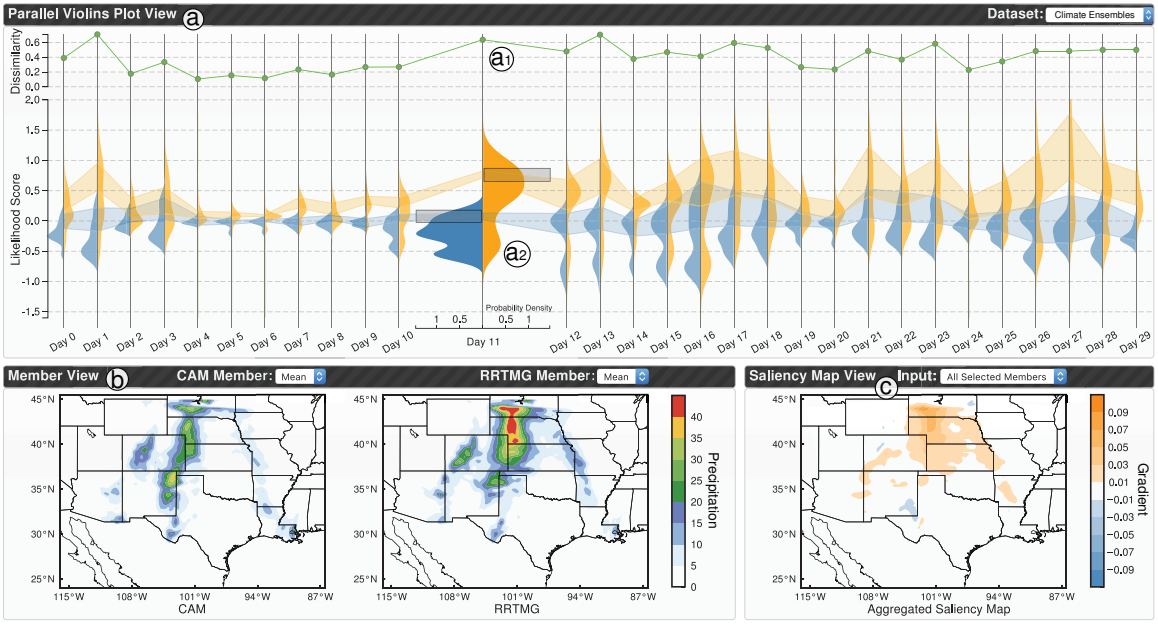


Figure 6.5: User interface of CECAV-DNN demonstrated by the climate ensembles generated with two different simulation models: (a) the parallel violins plot view presents the overall dissimilarities (a1) as well as the likelihood score distributions (a2) of the ensemble pairs in comparing; (b) after selecting an ensemble pair of interest and brushing on the corresponding violin plot, members whose likelihood score are within the brushed ranges are visualized in the member view; (c) by feeding the selected members into the trained discriminative network, saliency maps are generated using backprogation and visualized in the saliency map view.

6.3.1 User Interface

The proposed visualization system is constituted of three coordinated views: the parallel violins plot (PVP) view, the member view, and the saliency map view. Details for each view are discussed as follows.

6.3.1.1 Parallel Violins Plot (PVP) View

The PVP view (Figure 6.5(a)) consists of a dissimilarity chart and a sequence of violin plots, which presents the overall dissimilarities and the distributions of likelihood scores for a sequence of ensemble pairs that users are interested in. The PVP view is used as user interface to support our level I and level II comparative analysis.

Dissimilarity Chart A dissimilarity chart is a line chart to present the sequence of dissimilarities for the sequence of ensemble pairs, as shown in Figure 6.5(a1). This straightforward visualization enables users to quickly identify interesting pairs (e.g. ensemble pairs with larger dissimilarities) for further detailed investigations.

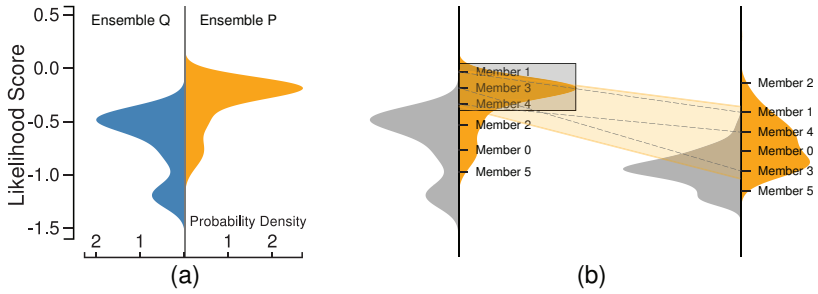


Figure 6.6: (a) Jointing two PDFs as side-by-side views for comparison. (b) A band is created after brushing on a violin plot, which reveals the distribution of the currently selected members in other ensembles.

Violin Plots We use the violin plots [65, 68] to encode and compare the distributions of likelihood scores. For each pair of ensembles, we first transform them into two collections of likelihood scores using the trained discriminative network. Then, we model a 1D PDF for each collection of likelihood scores. In order to visually compare the two PDFs, we joint them into a violin plot by putting them as side-by-side views for juxtaposed comparison, as shown in Figure 6.6(a). For a sequence of ensemble pairs, we arrange their corresponding violin plots in parallel to compare them and track the trend encoded in them.

The violin plots support two types of interactions. First, users can click on a violin plot to select an ensemble pair for further exploration. To focus on the selected ensemble pair, we enlarge the corresponding violin plot, meanwhile, compress and push other violin plots aside. When focusing on a violin plot of interest, users can explore different sub-ranges of the two PDFs via brushing. The brushed region in the current violin plot will be propagated to other violin plots to form a selection band, which reveals the distribution of the currently selected members in other ensembles. Figure 6.6(b) illustrates how we compute the band. The left PDF is in focus now and the brushed region selects members 1, 3, and 4. We compute the mean and standard deviation of the three members' likelihood scores in the current violin plot, as well as all other violin plots (e.g., the right violin plot in Figure 6.6(b)). Connecting the one standard deviation range of the mean likelihood scores across all violin plots forms a selection band. The two selections in the violin plot in Figure 6.5(a2) (i.e., the two brushed regions) lead to two bands. Comparing the width of the bands and the overlap between the bands provides useful insight in understanding the two selected sets of members across all ensemble pairs (details in Section 6.4.1).

6.3.1.2 Member View

The member view (Figure 6.5(b)) is used to visualize and compare the selected members (i.e., members in the brushed region of the violin plot). After users select members of interest in both ensembles, the selected members in each ensemble are aggregated into a mean field. The two mean fields are then visualized in two juxtaposed views as two heat maps (for 2D fields) or two volume renderings (for 3D fields) for comparison. Users can also check individual selected members by selecting them from the drop-down list shown in the header of this view (Figure 6.5(b), by default, the view presents the mean field). When users update their selections in the violin plot, the content of the member view is updated accordingly to synchronize the selection.

6.3.1.3 Saliency Map View

The saliency map view (Figure 6.5(c)) targets to demonstrate the result from our third level comparative analysis for the selected members, i.e., the spatial regions that are the most important for differentiating the two ensembles. Specifically, each selected member is fed into the trained discriminative network and the gradient of the likelihood score with respect to this member is computed using backpropagation to generate a saliency map (details in Section 6.2.2). The saliency maps from different selected members are then aggregated into a mean saliency map, and visualized using heat map (for 2D fields) or volume rendering (for 3D fields). To differentiate the positive gradients and negative gradients in the aggregated saliency map, a diverging color map (blue-white-orange) is used in this view. Same as the member view, the saliency map view is also linked with the violin plot view, i.e. any updates in the violin plot will trigger the updates in this view as well.

6.3.2 Exploration Guidelines

In exploring and comparing the ensembles using our visualization system, a top-down exploration strategy is recommended, which follows Shneiderman's information seeking mantra, i.e., *"Overview first, zoom and filter, then details-on-demand"* [141].

Overview First The PVP view (Figure 6.5(a)) provides the overall dissimilarities and the likelihood score distributions of the ensemble pairs in comparing. By comparing the dissimilarities and likelihood score distributions, users can obtain an overview of which pairs of ensembles are more similar/dissimilar with each other. From there, they can identify ensemble pairs of interest for further exploration.

Zoom and Filter After identifying an ensemble pair of interest (e.g. an ensemble pair with a large dissimilarity value), users can focus on this pair by clicking on the corresponding violin plot. This interaction will highlight the current violin plot by zooming into it, meanwhile, compressing and pushing other violin plots aside, as shown in Figure 6.5(b). From the currently focused violin plot, users can brush on either (or both) side(s) of the violin plot (representing the two ensembles) to select subsets of members for further exploration and analysis.

Details-on-Demand When users select a subset of members from each ensemble, the selected members will be visualized and compared in the member view (Figure 6.5(b)). In addition, the selected members are also fed into the trained discriminative network to generate their saliency maps, which will be aggregated and visualized in (Figure 6.5(c)). From the saliency map view, important spatial regions in differentiating the two ensembles can be located.

6.4 Case Studies

We demonstrate the effectiveness of CECAV-DNN with two real-world applications: one compares climate ensembles generated with different simulation models (Section 6.4.1), the other compares CFD ensembles generated with different spatial resolutions (Section 6.4.2).

6.4.1 Climate Ensembles with Different Simulation Models

The climate ensembles were generated using the weather research and forecasting (WRF) regional climate model with two different physical sub-models: CAM vs. RRTMG. The simulation domain was located over the southern great plains region (latitude: $25°N \sim 44°N$, longitude: $112°W \sim 90°W$) with the grid spacing of 25 km, i.e., the spatial resolution is 87 (latitude) $\times$ 89 (longitude). The simulations generated an ensemble for each sub-model using 150 different parameter settings to predict the precipitation over the 30 days of June 2007. As a result, for each sub-model, an ensemble of 150 members were produced for each day of June 2007. Using our three-level collective comparison approach, we compare the two ensembles in our visualization system.

6.4.1.1 Level I: Dissimilarity Comparison

The overall dissimilarities for the ensemble pairs of the 30 days are visualized as a line chart in Figure 6.5(a1). From the line chart, days that the two models highly agree (e.g. day 4, 5, and 6) or disagree (e.g. day 2, 11, and 13) with each other can be easily identified. With the guidance from the overall dissimilarities, representative days can be selected for further exploration, such as the days with a higher dissimilarity between the two models.

6.4.1.2 Level II: Member Comparison

From the first level comparison, day 11 attracted our attention the most, because the two ensembles in that day have a large dissimilarity value. After selecting day 11, the likelihood score distributions of the two models are highlighted in Figure 6.5(a2). By comparing the two likelihood score distributions, we can see that the two distributions share some common ranges with non-zero probabilities, indicating the results generated from the two models agree with each other in the corresponding members. We can also see that the distribution of the RRTMG model (i.e. the orange PDF) covers some ranges in which the CAM model (i.e. the blue PDF) has zero probabilities. These ranges indicate that the corresponding members in the RRTMG model are dramatically different from members in the CAM model. By brushing on the two distributions, users can select members whose likelihood score falls in a certain range. For example, in Figure 6.5(a2), two sets of members from the two ensembles are selected. With these selections, two bands (one for each sets of members) are also generated. From the two bands, we found that the mean likelihood scores of the selected members are consistent across different days, i.e., the mean likelihood score of members from the orange ensemble is always higher than that of the members from the blue ensemble across all 30 days. The mean fields of the two sets of selected members in Day 11 are visualized and compared in Figure 6.5(b). We observed that the precipitation is higher in the Central United States than other regions in both sets of the selected members. However, the selected members from the RRTMG model (which have higher likelihood scores) have higher precipitation in the Central United States regions than the selected members from the CAM model (which have lower likelihood scores).

The ensembles for Day 13 also have very high dissimilarity (as shown in the line chart in Figure 6.5(a1)). To explore the members from the two ensembles, four ranges were selected

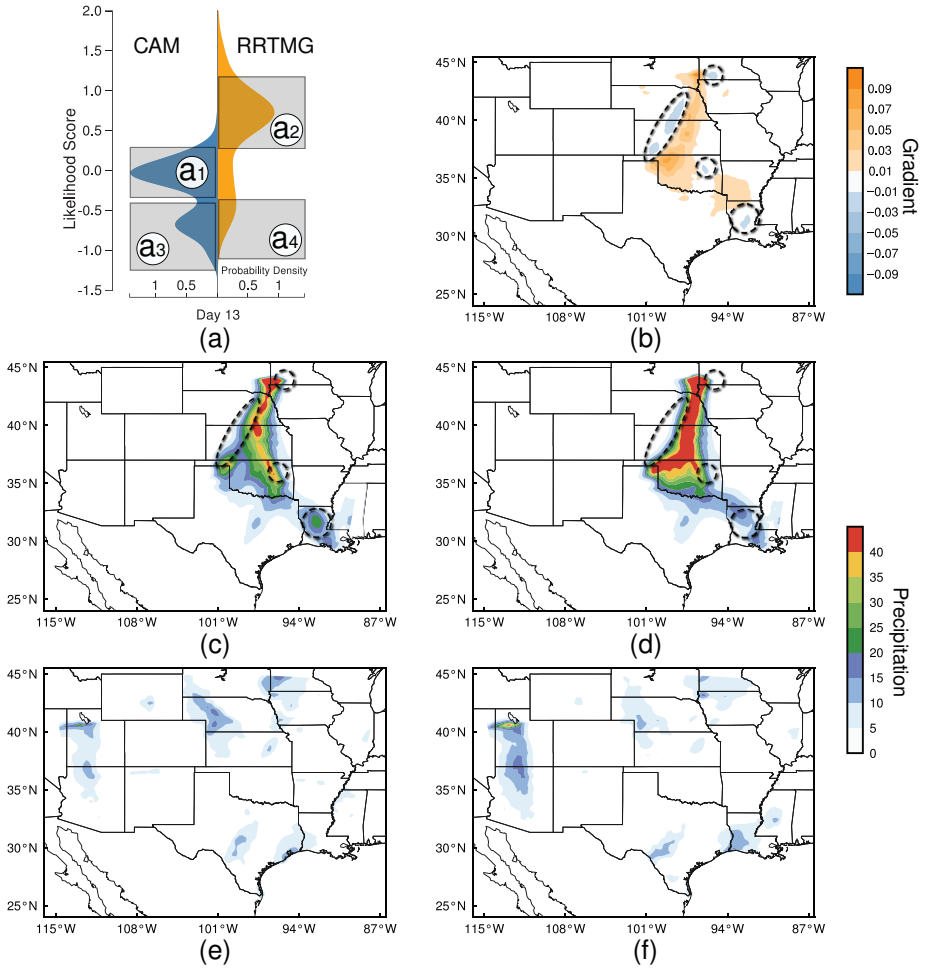


Figure 6.7: Comparison between two models in day 13: (a) likelihood score distributions and selected ranges; (b) aggregated saliency map of the selected members; (c), (d), (e), and (f) mean of the selected members within the ranges (a1), (a2), (a3), and (a4), respectively.

by brushing on the two likelihood distributions, as shown in Figures 6.7(a1−a4). The mean fields of the members in the ranges indicated by Figures 6.7(a1) and 6.7(a2) are shown in Figures 6.7(c) and 6.7(d), respectively. Similar to day 11, we found that the two models predict high precipitation in the Central United States, and the precipitation predicted by the RRTMG model is higher than that of the CAM model in general. Figures 6.7(e) and 6.7(f) show the mean fields of the members in the ranges indicated by 6.7(a3) and 6.7(a4), which

share similar predicted precipitation. However, by comparing the two distributions in Figure 6.7(a) (the size of the two selected areas), we can see that the CAM model generated more members within the range indicated by 6.7(a3).

By exploring and comparing the members of the two simulation models, we found that: (1) in general the two simulation models predict high precipitation in similar spatial regions; (2) the RRTMG model tends to produce members with higher precipitation than the CAM model in those regions.

6.4.1.3 Level III: Region Comparison

Focusing on the selected members of the two ensembles in day 13 (Figure 6.7), we drill down to the third level comparative analysis to further investigate what spatial regions are the most important for differentiating the two ensembles. The mean of the saliency maps for the selected members is visualized in 6.7(b). We found that the sensitive spatial regions (i.e. regions with high gradient magnitude) are located at regions with high precipitation in both models. We can also see that there are two large spatial regions that have positive gradients, which means if the precipitation in these regions increases, the corresponding members are more likely to be generated by the RRTMG model. There are also several small regions with negative gradients (marked by circles in Figure 6.7), which indicates that increasing precipitation in these regions will decrease the likelihood score (i.e. make the member more likely to be generated by the CAM model).

We also performed similar explorations on the other days to fully understand the two ensembles. Figure 6.8 shows the aggregated saliency maps of all members across the 30 days. We can find that in most of the spatial regions, the gradients are zero (i.e. white space in Figure 6.8). In these spatial regions, the value of the precipitation is not affecting the likelihood score of the members. Positive gradients cover larger spatial regions than negative

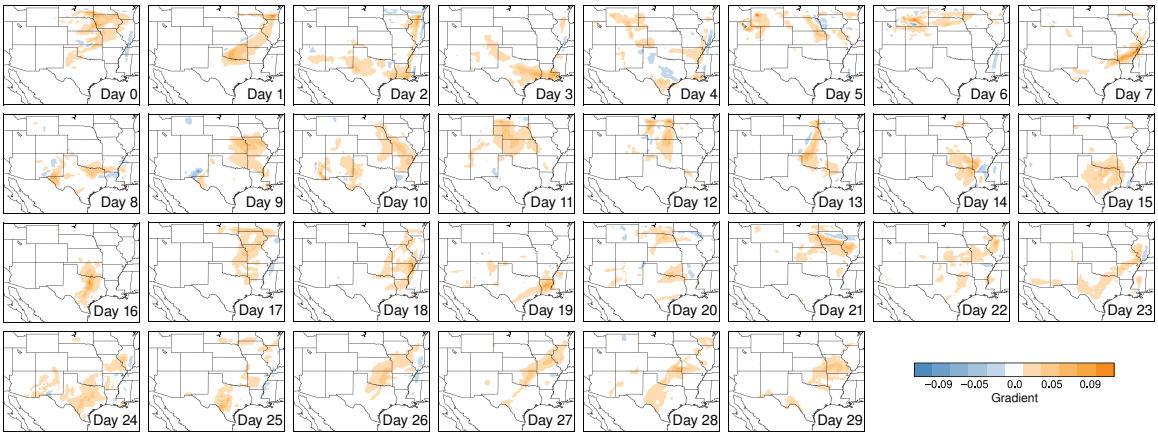


Figure 6.8: Aggregated saliency map of all members for each day. Positive gradients cover larger spatial regions than negative gradients.

gradients across all 30 days. Hence, compared with the CAM model, the RRTMG model has higher probability to produce higher precipitation in these regions.

6.4.2 CFD Ensembles with Different Spatial Resolutions

The CDF particle ensembles we studied in this section is from the 2016 IEEE SciVis Contest [1], which were generated with ensemble simulations using the finite pointset method. The simulations modeled the diffusion of salt from the top surface of a cylindrical flow domain that is filled with pure water. The simulation domain was represented by a cloud of discrete particles, each of which was endowed with the relevant local properties of the data, such as velocity and salt concentration. Three ensembles with different spatial resolutions (i.e. different numbers of particles) were generated, and we use E_l (around $250,000$ particles), E_m (around $650,000$ particles), and E_h (around $1,900,000$ particles) to index them. These three ensembles have 48, 23, and 22 members, respectively. Each member is a sequence of particle clouds, which record the temporal evolution of particles.

Because adaptive time stepping is used in the simulation, differed members may record the state of the particles at different sets of timesteps.

The goal of this simulation is to compare the salt concentration at different spatial locations over time. In this application, we first converted the particles in individual member from all three ensembles into a density field of $64 \times 64 \times 64$ grid points using density estimation. Second, we resampled the resulting density fields in the temporal domain using linear interpolation to make the ensembles share the same timesteps. There are 40 timesteps in total after the resampling. Finally, we applied our collective comparison approach to compare ensembles at individual timesteps, and analyzed the results from all 40 timesteps.

6.4.2.1 Level I: Dissimilarity Comparison

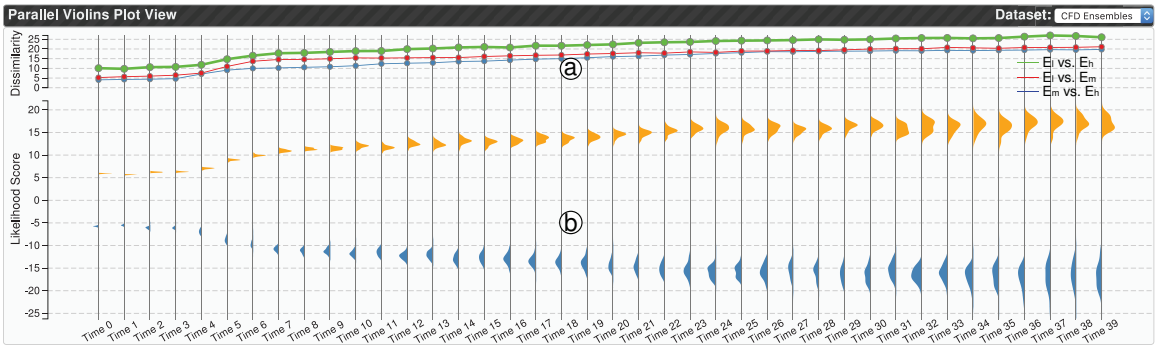


Figure 6.9: The PVP for the CFD ensembles: (a) overall dissimilarities across three resolutions; (b) likelihood score distributions of E_l and E_h.

Figure 6.9(a) shows the overall dissimilarities for the three ensemble pairs of the 40 timesteps. We found that the dissimilarities between E_l and E_h is the largest, and the dissimilarities between E_m and E_h is the smallest across all timesteps. We also found that

the dissimilarity increases over time for all three ensemble pairs. For further exploration we focused on the comparison between E_l and E_h.

6.4.2.2 Level II: Member Comparison

Figure 6.9(b) shows the likelihood score distributions of E_l and E_h. We can see that across all timesteps, none of the two distributions have overlapped ranges, which means the two ensembles do not share any similar members. We can also see that distance between the likelihood score distributions of the two ensembles increases over time.

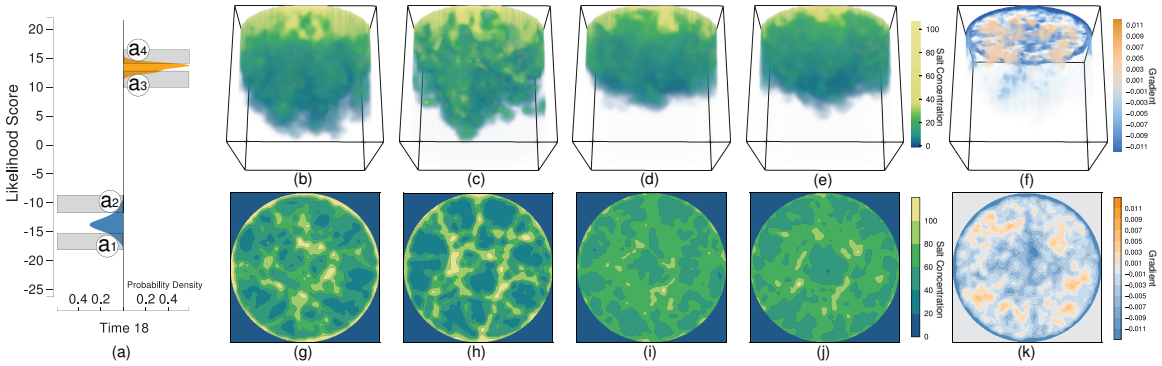


Figure 6.10: Visualization of the selected members in time 18: (a) the likelihood distributions of E_l and E_h as well as the selected ranges; (b)−(e) mean fields of the selected members within the ranges shown in (a1)−(a4), respectively; (g)−(j) slice on the top of the domain for the four mean fields; (f) and (k) mean saliency map of selected members and the slice of the saliency map.

To analyze how the members of the two ensembles are different, we selected timestep 18 for further exploration. Figure 6.10 shows the violin plot of two distributions at timestep 18, and 4 ranges are selected as shown in Figures 6.10(a1−a4). We compared E_l and E_h by visualizing and analyzing the members of the four selected ranges. The mean of the members within the ranges indicated by Figures 6.10(a1−a4) are shown in Figures 6.10(b−e),

respectively. We observed that the members of E_l have higher concentration at the lower half region of the domain than the members of E_h. By further comparing a slice on the top of the domain for the four mean fields in Figures 6.10(g−j), we can see that the members of E_l typically have higher concentration at the boundary of the cylindrical domain and a few small regions in near the center of the domain. We can also see that the members of E_l have large number of regions with concentration smaller than 20. Compared with E_l, the members of E_h do not have regions with extreme high or low concentration in the slice, which means the concentration are nearly evenly distributed in the slice. Compared with Figures 6.10(g) and (h), we found that the members with higher likelihood scores (i.e. members within the range of Figure 6.7(a2)) have lower concentration in the boundary than the members with lower likelihood scores (i.e. members within the range of Figure 6.7(a1)). Hence, members with higher concentration in the boundary are considered more likely to be sampled from E_l.

6.4.2.3 Level III: Region Comparison

We further visualized and analyzed the sensitivity of spatial regions for differentiating the two ensembles based on the third level comparative analysis. Figure 6.10(f) shows the mean saliency map of the selected members, and Figure 6.10(k) shows a slice on the top of the domain of the mean saliency map. Regions with low gradient magnitude are rendered with low transparency in the volume visualization. We found that regions with high gradient magnitude are located at the top of the domain, which indicates that in those regions the value of the salt concentration is important for differentiating the two ensembles.

High negative gradients are concentrated at the boundary of the cylindrical domain on the top, which means if the salt concentration at these regions increases, the member is more likely to come from E_l. Negative gradients can also be found in the lower half of the

domain, which is because the members of E_l have higher concentration at the lower half of the domain than the members of E_h. We also observed that on the top of the domain, there are several regions with positive gradient. This is because the members of E_l have regions with extreme low concentration on the top, and the concentration of members of E_h at those regions are higher.

6.5 Implementation and Performance

Table 6.1: Architecture details and training time of the discriminative networks used in the two case studies.

Dataset	Convolutional Layers				Fully Connected Layers		# Pairs	Batch Size	# Iterations	# Nodes	Time (mins)
	# Layers	# Filters	Filter Size	Strides	# Layers	# Neurons					
Climate Ensembles	3	$[8,16,32]$	5×5	2×2	1	4224	30	50	4000	30	1.92
CFD Ensembles	3	$[16,32,64]$	$3\times3\times3$	$2\times2\times2$	1	32768	120	20	4000	40	17.95

The proposed approach consists of two modules: the training of the discriminative networks, and the visualization system to support the collective comparison of ensemble pairs based on the trained discriminative networks. The training of discriminative networks is implemented using TensorFlow [2]. The architecture details of the discriminative networks used in the two applications are listed in Table 6.1. The visualization system is implemented based on a web server/client framework. The PVP view is implemented using D3.js on the client side. The members and the saliency maps are rendered using OpenGL on the server side and sent to the client to visualize.

The most compute-intensive part of our method is the training of the discriminative networks. Because the training of the discriminative network for each ensemble pair is independent from other discriminative networks, we parallelized the training of the discriminative networks over the ensemble pairs. In our implementation, the ensemble pairs

are assigned to multiple processes, and each process trains a subset of the ensemble pairs. We tested the performance of the training on Owens at the Ohio Supercomputer Center, which contains 160 nodes. Each node has an Intel Xeon E5-2680 CPU, an NVIDIA Pascal P100 GPU, 128 GB main memory, and 16 GB GPU memory. Our benchmark used up to 40 nodes with GPUs, and the performances of the two applications are listed in Table 6.1. From our observation, in most cases, the loss converged after 2000 iterations. In the experiments, the maximum number of iterations was set to 4000 to ensure the convergence of loss for all cases.

6.6 Discussion and Future Work

In this section, we first discuss our choice of hyperparameters (i.e. parameters whose value are fixed during training) for the discriminative networks. Then other divergences between distributions and their corresponding objective functions are discussed. Then we discuss the feedback we got from the domain expert and several directions we would like to explore in the future based on the feedback.

Hyperparameters of Discriminative Networks The most important hyperparameters of a discriminative network include number of layers, number of neurons per layer, and the size of the convolutional filters. Few routine approaches or practical recommendations have been reported to guide the selection of those hyperparameters, and the values of them are often data dependent. In this work, we resort to the following heuristic, which is commonly adopted by deep learning experts, to select the value for those hyperparameters. First, we start with an estimated size of the neural network, e.g., 3 convolutional layers, 32 filters per layer, and the size of each filter is 5×5. Second, by observing the training losses, we gradually increase/decrease those values, so that the network could accomplish

our training goals with a rather smaller size (to reduce the training time). The final values of those hyperparameters used in this work are reported in Table 6.1. They may not be the optimal selections, but the successful training results reported in this chapter have proved that they are proper choices, and deriving the optimal hyperparameters is out of the scope of this chapter.

Divergences and Objective Functions The proposed approach is flexible to use various objective functions, each of which corresponds to a specific type of divergence between two distributions. These objective functions can be written in a general form as:

$$L(P,Q;\phi) = \mathbb{E}_{p\sim P}[\,\mu(\mathscr{D}_\phi(p))\,] - \mathbb{E}_{q\sim Q}[\,\upsilon(\mathscr{D}_\phi(q))\,], \tag{6.4}$$

where μ and υ are two specific functions that control the type of the divergence related to the objective function. Table 6.2 shows various divergences between distributions (e.g. Kullback-Leibler divergence) and their corresponding functions μ and υ. More divergences and detailed mathematical proof can be found in [104, 105, 146].

Table 6.2: Defined μ and υ functions for different types of divergences between distributions. $\mathscr{F}$ is a class of real-value bounded functions, and $\|\mathscr{D}_\phi\|_1$ stands for 1-Lipschitz functions.

Divergence	μ	υ	Function Class of $\mathscr{D}_\phi$
Kullback-Leibler	$\mathscr{D}_\phi$	$e^{\mathscr{D}_\phi}$	$\mathscr{F}$
Jensen-Shannon	$\log\frac{2}{1+e^{-\mathscr{D}_\phi}}$	$-\log\frac{2}{1+e^{\mathscr{D}_\phi}}$	$\mathscr{F}$
Pearson χ^2	$\mathscr{D}_\phi$	$\frac{1}{4}\mathscr{D}_\phi^2+\mathscr{D}_\phi$	$\mathscr{F}$
Squared Hellinger	$1-e^{\mathscr{D}_\phi}$	$e^{-\mathscr{D}_\phi}-1$	$\mathscr{F}$
Total Variation	$\frac{1}{2}\tanh(\mathscr{D}_\phi)$	$\frac{1}{2}\tanh(\mathscr{D}_\phi)$	$\mathscr{F}$
Wasserstein	$\mathscr{D}_\phi$	$\mathscr{D}_\phi$	$\mathscr{F}\colon \|\mathscr{D}_\phi\|_1$

Domain Expert Feedback and Future Work We verified the effectiveness and usefulness of the proposed approach with one domain expert, who is from environmental

sciences and working with weather models. Overall, the expert commented that our approach provided a good guidance to explore and compare multiple ensembles at different levels, and the visualization interface is intuitive and friendly to him. Specifically, the expert commented that reducing members into simple representations (i.e. likelihood scores) is helpful for him to visualize and analyze the differences between complicated ensembles, and the saliency maps are useful to identify important spatial regions. He also mentioned that our approach is an important step toward revealing the complex input/output relations driven by dynamical and physical sensitivities in weather and climate models. Based on our discussion with the expert, we would like to further incorporate multiple variables in the climate ensembles to understand mechanisms leading to variation in precipitation duration, intensity, and distribution. We also plan to parallelize the proposed method on supercomputers to help the expert study large datasets.

6.7 Conclusion

In this chapter, we present a method to collectively compare and visualize ensembles using deep discriminative neural networks. Our method compares a pair of ensembles by training a discriminative neural network, which takes the two ensembles as input. The output from the network enables to compare the pair of ensembles from three different levels (in a top-down order): overall dissimilarity level; member level; and spatial region level. We further extend our method to the comparison of multiple ensemble pairs to copy with real-world ensemble data. To present our comparison results and facilitate the visual analytics of complex ensembles, we design and develop a visualization system. The system demonstrates our three-level comparative analysis results via multiple coordinated views (i.e., line chart, violin plots, and multiple juxtaposed spatial views). Through case studies on

real-world ensembles from different disciplines, we validate the effectiveness and usefulness of our approach with domain scientists.

Chapter 7: Conclusion and Future Work

7.1 Conclusion

In this dissertation, we proposed four approaches for exploration and analysis of ensemble datasets from different perspectives. In Chapter 3, we presented SDE to model the variability of surface features (e.g., isosurfaces, ridge surfaces, and streamsurfaces) for 3D ensemble datasets. Given an ensemble of surfaces represented as triangular meshes, we compute SDE as the KDE over the infinite set of points on the surfaces. We proposed a method to compute the SDE of each triangular patch analytically, and the resulting densities of the triangular patches are accumulated to approximate the SDE of the surfaces. As our method is directly applied on the surfaces, no field datasets are required. For SDE computation, we also proposed a bandwidth selection method that relies on the concentration of the surfaces. To explore the major trends and outliers of ensemble surfaces, we clustered the surfaces based on their contributions to the SDE and developed an interactive visual interface based on the clusters. We evaluated the accuracy of SDE computation by comparing with ground truth density fields and alternative methods. In addition, we demonstrated the usefulness and effectiveness of the proposed method on ensemble surfaces of different applications. In Chapter 4, we focused on modeling and visualizing the uncertainty of scalar values at each spatial location for ensemble scalar fields. We model the uncertainty of scalar values

as a probability distribution and transform ensemble scalar fields into distribution fields. To explore and analyze the distribution fields, we proposed a compact and effective representation, namely RLT, which decomposes and summarizes probability distributions with the range likelihood fields over a few representative subranges. Based on the RLT, we developed a visual interface to explore probability distribution fields through range likelihood fields. We demonstrated the usefulness and effectiveness of the proposed approach with ensemble datasets of different applications and verified the visualization results with two domain scientists. In Chapter 5, we focused on parameter space exploration of extreme-scale ensemble simulations. Due to the IO and storage bottleneck, the simulation outputs are visualized in situ and only the visualization results (i.e., images) are stored on disk for post-hoc parameter space exploration. We proposed InSituNet, which is a deep learning based image synthesis model, to learn the end-to-end mapping from the simulation parameters to the visualization images, conditioned on different visualization settings. The InSituNet is trained on the images generated in situ with a limited amount of simulation and visualization parameters. After training, the InSituNet can be used to generate visualization images of simulation outputs with novel simulation and visualization parameters without actually running the expensive simulation. Additionally, the sensitivity of the parameters can also be derived from the trained InSituNet. We demonstrated the accuracy and effectiveness of InSituNet on various ensemble simulations through both quantitative and qualitative evaluations. In Chapter 6, we focused on collective comparisons of ensembles generated with different phenomenological models and/or spatial resolutions to help scientists understand the differences between the underlying models. We trained a discriminative neural network to differentiate the members of one ensemble from members of another ensemble. Based on the trained discriminative neural network, we were able to approximate the overall difference between

the ensembles and identified members and spatial regions that the ensembles agree/disagree with each other. In addition, we designed and developed a visualization system to facilitate the visual comparison of complex ensembles. To demonstrate the effectiveness and usefulness of our approach, we performed case studies on two real-world simulations with domain scientists.

7.2 Future Work

In this section, we discuss several directions we would like to explore in the future for ensemble data visualization and analysis. First, as most of the existing ensemble visualization approaches were limited to analyze a single variable, we would like to study ensemble datasets with multiple variables. Specifically, we would like to focus on the correlation between multiple variables, for which modeling and analyzing the correlations between variables with the awareness of uncertainty play a key role. Second, we would like to model the uncertainty of the ensemble data at each spatial location by taking the correlation of the data values at different spatial locations under consideration. To this end, we need to model multidimensional distributions across different spatial locations and visualize such complex distributions to understand the correlations. Third, as the spatial and/or temporal resolutions of ensemble simulations are getting finer, we would like to focus more on in situ data analysis and visualization for ensemble simulations. For example, we would like to extend InsituNet to predict visualization images for ensemble simulations at different time steps with the assist of sequential models such as long short-term memory (LSTM). In addition, we would like to improve the resolution and quality of the prediction image by exploring different network designs, such as self-attention mechanisms. Moreover, we would like to explore different in situ visualization techniques such as distribution-based

and feature-based approaches other than just image-based approaches, for which we will extend InsituNet from handling 2D images to 3D volumes or geometries (e.g., point clouds). Fourth, as the existing ensemble visualization approaches focused more on comparing simulation results across ensemble members, we would like to study more on the collective comparison between ensembles. Although one approach has been proposed to perform the collective comparison between ensembles in this dissertation, the approach is limited to scalar fields with a single variable. In the future, we would like to extend to technique to study differences between ensembles of flow fields, which plays an important role in understanding the difference between different CFD models.

Appendix A: Bivariate Normal Integral Calculation

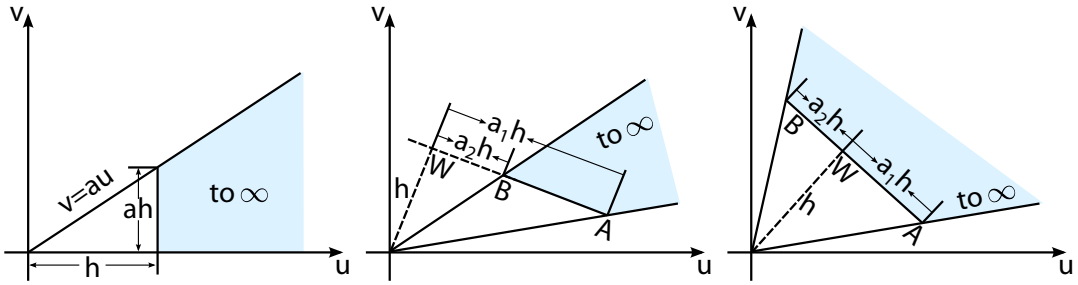


Figure A.1: (a) Region over which $\Gamma(h,a)$ gives the integral of the standard bivariate normal distribution. (b) and (c) Bivariate normal integral over areas defined by an edge of a given triangle.

This appendix provides the method proposed by Owen [109] for calculating the bivariate normal integral over an area defined by an edge of a given triangle. He first defines a function $\Gamma(h,a)$ that gives the integral of the standard bivariate normal distribution over the area between line $v = au$ and $v = 0$ and to the right of line $u = h$, as shown in Figure A.1(a). $\Gamma(h,a)$ is then used to calculate the bivariate normal integral over an area defined by an edge of a given triangle, as the shaded areas shown in Figures A.1(b) and (c).

The function $\Gamma(h,a)$ is defined as

$$\Gamma(h,a) = \frac{arctan(a)}{2\pi} - \frac{1}{2\pi}\sum_{i=0}^{\infty} c_i a^{2i+1}, \tag{A.1}$$

where

$$c_i = (-1)^i \frac{1}{2i+1}(1 - e^{-\frac{1}{2}h^2 \sum_{j=0}^{i} \frac{h^{2j}}{2^j j!}}), \tag{A.2}$$

which converges rapidly for small a and h. To reduce run-time computations, we generate a lookup table for the Γ-function for h from 0 to 4.76, a from 0 to 1, and $a = \infty$, with all 0.01 in between. We combine the lookup table with bilinear interpolation to obtain the $\Gamma(h,a)$ for given a and h. For negative h or a, $\Gamma(-h,a) = \Gamma(h,a)$ and $\Gamma(h,-a) = -\Gamma(h,a)$. For $1 < a < \infty$, $\Gamma(h,a)$ is obtained by using

$$\Gamma(h,a) = \frac{1}{2}\Phi(h) + \frac{1}{2}\Phi(ah) - \Phi(h)\Phi(ah) - \Gamma(ah, \frac{1}{a}), \tag{A.3}$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. For $h > 4.76$, $\Gamma(h,a) = 0$.

With the Γ-function, the integral of the standard bivariate normal distribution over an area defined by an edge of a given triangle can be obtained. For example, in Figure A.1(b), let AB be an edge of a triangle. The integral of the standard bivariate normal distribution over the shaded area, denoted as α_{AB}, can be obtained by using $\Gamma(h,a_1) - \Gamma(h,a_2)$ for $a_1 > a_2$, where h is the length between the origin and its projection W on the line defined by AB, which is

$$h = \frac{|u_1 v_2 - u_2 v_1|}{\sqrt{(u_2 - u_1)^2 + ((v_2 - v_1)^2}}, \tag{A.4}$$

and the slopes a_1 and a_2 can be computed with

$$a_1 = \frac{|u_1(u_2 - u_1) + v_1(v_2 - v_1)|}{|u_1 v_2 - u_2 v_1|} \tag{A.5}$$

$$a_2 = \frac{|u_2(u_2 - u_1) + v_2(v_2 - v_1)|}{|u_1 v_2 - u_2 v_1|}, \tag{A.6}$$

where $(u_1, v_1)^T$ and $(u_2, v_2)^T$ are the coordinates of the vertices A and B. If W lies between A and B, then $\alpha_{AB} = \Gamma(h,a_1) + \Gamma(h,a_2)$, as shown in Figure A.1(c).

Bibliography

[1] IEEE SciVis Contest 2016. `http://www.uni-kl.de/sciviscontest/`.

[2] TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015.

[3] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, P. Fasel, A. Bauer, M. Petersen, F. Samsel, and B. Boeckel. In situ MPAS-Ocean image-based visualization. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Visualization & Data Analytics Showcase*, 2014.

[4] J. Ahrens, S. Jourdain, P. O'Leary, J. Patchett, D. H. Rogers, and M. Petersen. An image-based approach to extreme scale in situ visualization and analysis. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 424–434, 2014.

[5] A. Aitken, C. Ledig, L. Theis, J. Caballero, Z. Wang, and W. Shi. Checkerboard artifact free sub-pixel convolution: A note on sub-pixel convolution, resize convolution and convolution resize. *arXiv preprint arXiv:1707.02937*, 2017.

[6] C. Alexander, S. S. Weygandt, D. C. D. S. Benjamin, T. G. Smirnova, E. P. James, M. H. P. Hofmann, J. Olson, and J. M. Brown. The high-resolution rapid refresh: Recent model and data assimilation development towards an operational implementation in 2014. In *Proceedings of 26th Conference on Weather Analysis and Forecasting / 22nd Conference on Numerical Weather Prediction*, 2014.

[7] A. S. Almgren, J. B. Bell, M. J. Lijewski, Z. Lukić, and E. V. Andel. Nyx: A massively parallel AMR code for computational cosmology. *The Astrophysical Journal*, 765(1), 2013.

[8] J. C. Anderson, L. Gosink, M. A. Duchaineau, and K. Joy. Feature identification and extraction in function fields. In *Proceedings of Eurographics / IEEE-VGTC Symposium on Visualization*, pages 195–201, 2007.

[9] J. C. Anderson, L. Gosink, M. A. Duchaineau, and K. Joy. Interactive visualization of function fields by range-space segmentation. *Computer Graphics Forum*, 28(3):727–734, 2009.

[10] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[11] T. Athawale and A. Entezari. Uncertainty quantification in linear interpolation for isosurface extraction. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2723–2732, 2013.

[12] T. Athawale, E. Sakhaee, and A. Entezari. Isosurface visualization of data with nonparametric models for uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):777–786, 2016.

[13] A. C. Bauer, H. Abbasi, J. Ahrens, H. Childs, B. Geveci, S. Klasky, K. Moreland, P. O'Leary, V. Vishwanath, B. Whitlock, and E. W. Bethel. In situ methods, infrastructures, and applications on high performance computing platforms. *Computer Graphics Forum*, 35(3):577–597, 2016.

[14] K. Bensema, L. Gosink, H. Obermaier, and K. Joy. Modality-driven classification and visualization of ensemble variance. *IEEE Transactions on Visualization and Computer Graphics*, 22(10):2289–2299, 2016.

[15] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019.

[16] H. Bhatia, S. Jadhav, P. T. Bremer, G. Chen, J. A. Levine, L. G. Nonato, and V. Pascucci. Flow visualization with quantified spatial and temporal errors using edge maps. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1383–1396, 2012.

[17] T. Biedert and C. Garth. Contour tree depth images for large data visualization. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, pages 77–86, 2015.

[18] C. K. Birdsall and D. Fuss. Cloud-in-cell computer experiments in two and three dimensions. Technical report, University of California, Livermore. Lawrence Radiation Laboratory, 1969.

[19] C. K. Birdsall and D. Fuss. Clouds-in-clouds, clouds-in-cells physics for many-body plasma simulation. *Journal of Computational Physics*, 3(4):494–511, 1969.

[20] A. Biswas, G. Lin, X. Liu, and H.-W. Shen. Visualization of time-varying weather ensembles across multiple resolutions. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):841–850, 2017.

[21] J. Blaas, C. P. Botha, and F. H. Post. Extensions of parallel coordinates for interactive exploration of large multi-timepoint data sets. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1436–1451, 2008.

[22] A. Bock, A. Pembroke, M. L. Mays, L. Rastaetter, T. Ropinski, and A. Ynnerman. Visual verification of space weather ensemble simulations. In *Proceedings of 2015 IEEE Scientific Visualization Conference*, pages 17–24, 2015.

[23] R. P. Botchen, D. Weiskopf, and T. Ertl. Texture-based visualization of uncertainty in flow fields. In *Proceedings of IEEE Visualization*, pages 647–654, 2005.

[24] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186, 2010.

[25] A. W. Bowman. An alternative method of cross-validation for the smoothing of density estimates. *Biometrika*, 71(2):353–360, 1984.

[26] P.-T. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 17(9):1307–1324, 2011.

[27] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of International Conference on Learning Representations*, 2019.

[28] K. Brodlie, R. Allendes Osorio, and A. Lopes. A review of uncertainty in data visualization. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong, editors, *Expanding the Frontiers of Visual Analytics and Visualization*, pages 81–109. Springer London, 2012.

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

[30] C. Buchheim, M. Jünger, and S. Leipert. Improving walker's algorithm to run in linear time. In *Proceedings of Graph Drawing*, pages 344–353, 2002.

[31] S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo. Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata*, pages 1000–1012, 2005.

[32] S.-H. Cha. Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 2007.

[33] C.-M. Chen, A. Biswas, and H.-W. Shen. Uncertainty modeling and error reduction for pathline computation in time-varying flow fields. In *Proceedings of 2015 IEEE Pacific Visualization Symposium*, pages 215–222, 2015.

[34] H. Chen, S. Zhang, W. Chen, H. Mei, J. Zhang, A. Mercer, R. Liang, and H. Qu. Uncertainty-aware multidimensional ensemble data visualization and exploration. *IEEE Transactions on Visualization and Computer Graphics*, 21(9):1072–1086, 2015.

[35] J. H. Claessen and J. J. Van Wijk. Flexible linked axes for multivariate data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2310–2316, 2011.

[36] D. Coffey, C.-L. Lin, A. G. Erdman, and D. F. Keefe. Design by dragging: An interface for creative forward and inverse design with simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2783–2791, 2013.

[37] I. Demir, C. Dick, and R. Westermann. Multi-charts for comparative 3d ensemble visualization. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2694–2703, 2014.

[38] I. Demir, J. Kehrer, and R. Westermann. Screen-space silhouettes for visualizing ensembles of 3D isosurfaces. In *Proceedings of 2016 IEEE Pacific Visualization Symposium*, pages 204–208, 2016.

[39] S. Di and F. Cappello. Fast error-bounded lossy HPC data compression with SZ. In *Proceedings of International Parallel and Distributed Processing Symposium*, pages 730–739, 2016.

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

[41] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[42] T. G. Donnelly. Algorithm 462: Bivariate normal distribution. *Commun. ACM*, 16(10):629–640, 1973.

[43] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[44] S. Dutta, C.-M. Chen, G. Heinlein, H.-W. Shen, and J.-P. Chen. In situ distribution guided analysis and visualization of transonic jet engine simulations. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):811–820, 2017.

[45] C. Ericson. *Real-Time Collision Detection*. CRC Press, Inc., Boca Raton, 2004.

[46] O. Fernandes, S. Frey, F. Sadlo, and T. Ertl. Space-time volumetric depth images for in-situ visualization. In *Proceedings of 2014 IEEE Symposium on Large Data Analysis and Visualization*, pages 59–65, 2014.

[47] F. Ferstl, K. Bürger, and R. Westermann. Streamline variability plots for characterizing the uncertainty in vector field ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):767–776, 2016.

[48] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Visual analysis of spatial variability and global correlations in ensembles of iso-contours. *Computer Graphics Forum*, 35(3):221–230, 2016.

[49] F. Ferstl, M. Kanzler, M. Rautenhaus, and R. Westermann. Time-hierarchical clustering and visualization of weather forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):831–840, 2017.

[50] N. Fout and K.-L. Ma. Fuzzy volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2335–2344, 2012.

[51] S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting. In *Proceedings of 2013 XXVI Conference on Graphics, Patterns and Images*, pages 123–130, 2013.

[52] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[53] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 262–270, 2015.

[54] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[55] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[56] H. Guo, W. He, T. Peterka, H.-W. Shen, S. M. Collis, and J. J. Helmus. Finite-time lyapunov exponents and Lagrangian coherent structures in uncertain unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 22(6):1672–1682, 2016.

[57] H. Guo, H. Xiao, and X. Yuan. Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates. In *Proceedings of 2011 IEEE Pacific Visualization Symposium*, pages 19–26, 2011.

[58] G. Haller. Lagrangian coherent structures from approximate velocity data. *Physics of Fluids*, 14(6):1851–1861, 2002.

[59] G. Haller. A variational theory of hyperbolic Lagrangian coherent structures. *Physica D: Nonlinear Phenomena*, 240(7):547–598, 2011.

[60] J. Han, J. Tao, and C. Wang. FlowNet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2018, Early Access.

[61] S. Hazarika, A. Biswas, and H.-W. Shen. Uncertainty visualization using copula-based analysis in mixed distribution models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):934–943, 2018.

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[63] J. Heinrich and D. Weiskopf. Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1531–1538, 2009.

[64] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proceedings of Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[65] J. L. Hintze and R. D. Nelson. Violin plots: A box plot-density trace synergism. *The American Statistician*, 52(2):181–184, 1998.

[66] M. Hlawatsch, P. Leube, W. Nowak, and D. Weiskopf. Flow radar glyphs & static visualization of unsteady flow with uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1949–1958, 2011.

[67] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[68] T. Höllt, A. Magdy, P. Zhan, G. Chen, G. Gopalakrishnan, I. Hoteit, C. D. Hansen, and M. Hadwiger. Ovis: A framework for visual analysisof ocean forecast ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1114–1126, 2014.

[69] F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. In *Proceedings of 2018 IEEE Pacific Visualization Symposium*, pages 76–85, 2018.

[70] X. Hou, L. Shen, K. Sun, and G. Qiu. Deep feature consistent variational autoencoder. In *Proceedings of 2017 IEEE Winter Conference on Applications of Computer Vision*, pages 1133–1141, 2017.

[71] C. Hurter, O. Ersoy, and A. Telea. Graph bundling by kernel density estimation. *Computer Graphics Forum*, 31(3pt1):865–874, 2012.

[72] A. Inselberg. The plane with parallel coordinates. *The visual computer*, 1(2):69–91, 1985.

[73] J. Johnson, A. Alahi, and F.-F. Li. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision*, pages 694–711, 2016.

[74] J.-M. Jolion, P. Meer, and S. Bataouche. Robust clustering with applications in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(8):791–802, 1991.

[75] J. Kiefer. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society*, 4(3):502–506, 1953.

[76] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations*, 2015.

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

[78] A. Knoll, Y. Hijazi, A. Kensler, M. Schott, C. Hansen, and H. Hagen. Fast ray tracing of arbitrary implicit surfaces with interval and affine arithmetic. *Computer Graphics Forum*, 28(1):26–40, 2009.

[79] P. Köthur, C. Witt, M. Sips, N. Marwan, S. Schinkel, and D. Dransch. Visual analytics for correlation-based comparison of time series ensembles. *Computer Graphics Forum*, 34(3):411–420, 2015.

[80] A. Kumpf, B. Tost, M. Baumgart, M. Riemer, R. Westermann, and M. Rautenhaus. Visualizing confidence in cluster-based ensemble weather forecast analyses. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):109–119, 2018.

[81] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly. The GAN landscape: Losses, architectures, regularization, and normalization. *arXiv preprint arXiv:1807.04720*, 2018.

[82] S. Lakshminarasimhan, N. Shah, S. Ethier, S.-H. Ku, C. S. Chang, S. Klasky, R. Latham, R. Ross, and N. F. Samatova. ISABELA for effective in situ compression of scientific data. *Concurrency and Computation: Practice and Experience*, 25(4):524–540, 2013.

[83] O. D. Lampe and H. Hauser. Curve density estimates. *Computer Graphics Forum*, 30(3):633–642, 2011.

[84] O. D. Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *Proceedings of 2011 IEEE Pacific Visualization Symposium*, pages 171–178, 2011.

[85] O. D. Lampe, J. Kehrer, and H. Hauser. Visual analysis of multivariate movement data using interactive difference views. In *VMV 2010: Proceedings of Vision, Modeling, and Visualization*, pages 315–322, 2010.

[86] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 105–114, 2017.

[87] P. F. J. Lermusiaux. Uncertainty estimation and prediction for interdisciplinary ocean dynamics. *J. Comput. Phys.*, 217(1):176–199, 2006.

[88] S. Liu, J. A. Levine, P. Bremer, and V. Pascucci. Gaussian mixture model based volume visualization. In *Proceedings of IEEE Symposium on Large Data Analysis and Visualization*, pages 73–77, 2012.

[89] X. Liu and H.-W. Shen. Association analysis for visual exploration of multivariate scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):955–964, 2016.

[90] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. UFLOW: Visualizing uncertainty in fluid flow. In *Proceedings of IEEE Visualization 1996*, pages 249–254, 1996.

[91] A. Luo, D. Kao, and A. Pang. Visualizing spatial distribution data sets. In *Proceedings of VisSym*, 2003.

[92] M. Lučić, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? A large-scale study. In *Proceedings of Advances in Neural Information Processing Systems*, pages 700–709, 2018.

[93] K.-L. Ma. In situ visualization at extreme scale: Challenges and opportunities. *IEEE Computer Graphics and Applications*, 29(6):14–19, 2009.

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

[95] K. T. McDonnell and K. Mueller. Illustrative parallel coordinates. 27:1031–1038, 2008.

[96] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[97] M. Mirzargar, R. T. Whitaker, and R. M. Kirby. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2654–2663, 2014.

[98] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of International Conference on Learning Representations*, 2018.

[99] T. Miyato and M. Koyama. cGANs with projection discriminator. In *Proceedings of International Conference on Learning Representations*, 2018.

[100] J. J. Monaghan. Smoothed particle hydrodynamics. *Annual Review of Astronomy and Astrophysics*, 30(1):543–574, 1992.

[101] P. Muigg, M. Hadwiger, H. Doleisch, and M. E. Gröller. Visual coherence for large-scale line-plot visualizations. *Computer Graphics Forum*, 30(3):643–652, 2011.

[102] A. Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429—-443, 1997.

[103] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of International Conference on Machine Learning*, pages 807–814, 2010.

[104] X. Nguyen, M. J. Wainwright, and M. I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, 2010.

[105] S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Proceedings of Advances in Neural Information Processing Systems*, pages 271–279, 2016.

[106] H. Obermaier, K. Bensema, and K. I. Joy. Visual trends analysis in time-varying ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(10):2331–2342, 2016.

[107] H. Obermaier and K. I. Joy. Future challenges for ensemble visualization. *IEEE Computer Graphics and Applications*, 34(3):8–11, 2014.

[108] D. Orban, D. F. Keefe, A. Biswas, J. Ahrens, and D. Rogers. Drag and track: A direct manipulation interface for contextualizing data instances within a continuous parameter space. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):256–266, 2019.

[109] D. B. Owen. Tables for computing bivariate normal probabilities. *The Annals of Mathematical Statistics*, 27(4):1075–1090, 1956.

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

[111] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[112] T. Peterka, H. Croubois, N. Li, S. Rangel, and F. Cappello. Self-adaptive density estimation of particle data. *SIAM Journal on Scientific Computing*, 38(5):S646–S666, 2015.

[113] C. Petz, K. Pöthkow, and H.-C. Hege. Probabilistic local features in uncertain vector fields with spatial correlation. *Computer Graphics Forum*, 31(3pt2):1045–1054, 2012.

[114] T. Pfaffelmoser, M. Mihai, and R. Westermann. Visualizing the variability of gradients in uncertain 2D scalar fields. *IEEE Transactions on Visualization and Computer Graphics*, 19(11):1948–1961, 2013.

[115] T. Pfaffelmoser, M. Reitinger, and R. Westermann. Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields. In *Proceedings of the 13th Eurographics*, pages 951–960, 2011.

[116] M. N. Phadke, L. Pinto, O. Alabi, J. Harter, R. M. Taylor II, X. Wu, H. Petersen, S. A. Bass, and C. G. Healey. Exploring ensemble visualization. *Proc. SPIE*, 8294(82940B):1–12, 2012.

[117] J. Poco, A. Dasgupta, Y. Wei, W. Hargrove, C. R. Schwalm, D. N. Huntzinger, R. Cook, E. Bertini, and C. T. Silva. Visual reconciliation of alternative similarity spaces in climate modeling. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1923–1932, 2014.

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

[119] K. Pöthkow and H.-C. Hege. Nonparametric models for uncertainty visualization. *Computer Graphics Forum*, 32(3):131–140, 2013.

[120] K. Pöthkow, C. Petz, and H.-C. Hege. Approximate level-crossing probabilities for interactive visualization of uncertain isocontours. *International Journal for Uncertainty Quantification*, 3(2):101–117, 2013.

[121] K. Pöthkow, B. Weber, and H.-C. Hege. Probabilistic marching cubes. *Computer Graphics Forum*, 30(3):931–940, 2011.

[122] K. Potter, P. Rosen, and C. R. Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In *Uncertainty Quantification in Scientific Computing*, pages 226–249. 2012.

[123] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johhson. Ensemble-vis: A framework for the statistical visualization of ensemble data. In *Proceedings of IEEE Workshop on Knowledge Discovery from Climate Data: Prediction, Extremes.*, pages 233–240, 2009.

[124] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[125] M. Raj, M. Mirzargar, J. S. Preston, R. M. Kirby, and R. T. Whitaker. Evaluating shape alignment via ensemble visualization. *IEEE Computer Graphics and Applications*, 36(3):60–71, 2016.

[126] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. Generative adversarial text to image synthesis. In *Proceedings of International Conference on Machine Learning*, pages 1060–1069, 2016.

[127] T. Ringler, M. Petersen, R. L. Higdon, D. Jacobsen, P. W. Jones, and M. Maltrud. A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69:211–232, 2013.

[128] A. R. Robinson. Realtime forecasting of the multidisciplinary coastal ocean with the littoral ocean observing and predicting system (loops). In *Proceedings of Preprint Volume of the Third Conference on Coastal Atmospheric and Oceanic Prediction and Processes*, pages 3–5, 1999.

[129] M. Rudemo. Empirical choice of histograms and kernel density estimators. *Scandinavian Journal of Statistics*, 9(2):65–78, 1982.

[130] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[131] E. Sakhaee and A. Entezari. A statistical direct volume rendering framework for visualization of uncertain data. *IEEE Transactions on Visualization and Computer Graphics*, 23(12):2509–2520, 2017.

[132] F. Samsel, M. Petersen, G. Abram, T. L. Turton, D. Rogers, and J. Ahrens. Visualization of ocean currents and eddies in a high-resolution global ocean-climate model. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, Visualization & Data Analytics Showcase*, 2015.

[133] F. Samsel, M. Petersen, T. Geld, G. Abram, J. Wendelberger, and J. Ahrens. Colormaps that improve perception of high-resolution ocean data. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, 2015.

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

[135] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[136] W. E. Schaap. *DTFE: the Delaunay Tessellation Field Estimator*. PhD thesis, University of Groningen, The Netherlands, 2007.

[137] W. E. Schaap and R. Van De Weygaert. Continuous fields and discrete samples: reconstruction through delaunay tessellations. *Astronomy and Astrophysics*, 363:L29–L32, 2000.

[138] R. Scheepens, N. Willems, H. van de Wetering, G. Andrienko, N. Andrienko, and J. J. van Wijk. Composite density maps for multivariate trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2518–2527, 2011.

[139] R. Scheepens, N. Willems, H. van de Wetering, and J. J. van Wijk. Interactive visualization of multivariate trajectory data with density maps. In *Proceedings of 2011 IEEE Pacific Visualization Symposium*, pages 147–154, 2011.

[140] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, pages 517–524, 1968.

[141] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of IEEE Symposium on Visual Languages*, pages 336–343, 1996.

[142] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

[143] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[144] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[145] R. Splechtna, K. Matković, D. Gračanin, M. Jelović, and H. Hauser. Interactive visual steering of hierarchical simulation ensembles. In *Proceedings of 2015 IEEE Conference on Visual Analytics Science and Technology*, pages 89–96, 2015.

[146] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet. Non-parametric estimation of integral probability metrics. In *Proceedings of 2010 IEEE International Symposium on Information Theory*, pages 1428–1432, 2010.

[147] M. Steinberger, M. Waldner, M. Streit, A. Lex, and D. Schmalstieg. Context-preserving visual links. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2249–2258, 2011.

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

[149] A. Tikhonova, C. D. Correa, and K.-L. Ma. Explorable images for visualizing volume data. In *Proceedings of 2010 IEEE Pacific Visualization Symposium*, pages 177–184, 2010.

[150] A. Tikhonova, C. D. Correa, and K.-L. Ma. An exploratory technique for coherent visualization of time-varying volume data. *Computer Graphics Forum*, 29(3):783–792, 2010.

[151] A. Tikhonova, C. D. Correa, and K.-L. Ma. Visualization by proxy: A novel framework for deferred interaction with volume data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1551–1559, 2010.

[152] M. Wand and M. Jones. *Kernel Smoothing*. Monographs on statistics and applied probability. Chapman and Hall, 1995.

[153] J. Wang, L. Gou, H. Yang, and H.-W. Shen. GANViz: A visual analytics approach to understand the adversarial game. *IEEE Transactions on Visualization and Computer Graphics*, 24(6):1905–1917, 2018.

[154] J. Wang, S. Hazarika, C. Li, and H.-W. Shen. Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 2018, Early Access.

[155] J. Wang, X. Liu, H.-W. Shen, and G. Lin. Multi-resolution climate ensemble parameter analysis with nested parallel coordinates plots. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):81–90, 2017.

[156] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi. Efficient volume exploration using the gaussian mixture model. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1560–1573, 2011.

[157] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963.

[158] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics*, 12(6):620–631, 1998.

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

[160] C. M. Wittenbrink, A. T. Pang, and S. K. Lodha. Glyphs for visualizing uncertainty in vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 2(3):266–279, 1996.

[161] C. Xie, W. Xu, and K. Mueller. A visual analytics framework for the detection of anomalous call stack trees in high performance computing applications. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):215–224, 2019.

[162] J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 341–349, 2012.

[163] B. Yang, Y. Qian, G. Lin, R. Leung, and Y. Zhang. Some issues in uncertainty quantification and parameter tuning: A case study of convective parameterization scheme in the WRF regional climate model. *Atmospheric Chemistry and Physics*, 12(5):2409–2427, 2012.

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

[165] H. Zhang, I. J. Goodfellow, D. N. Metaxas, and A. Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[166] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[167] H. Zhou, W. Cui, H. Qu, Y. Wu, X. Yuan, and W. Zhuo. Splatting the lines in parallel coordinates. In *Proceedings of the 11th Eurographics*, pages 759–766, 2009.

[168] Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, and H. Huang. Non-stationary texture synthesis by adversarial expansion. *ACM Transactions on Graphics*, 37(4):49:1–49:13, 2018.

[169] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of 2017 International Conference on Computer Vision*, pages 2242–2251, 2017.

[170] M. Zinsmaier, U. Brandes, O. Deussen, and H. Strobelt. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.