Analysis and Visualization of Vortices: Topological, Physics-Based and Deep Learning Techniques

by
Marzieh Berenjkoub

A dissertation submitted to the Department of Computer Science,

College of Natural Sciences and Mathematics

in partial fulfillment of the requirements for the degree of

Doctor of Philosphy
in Computer Science

Chair of Committee: Dr. Guoning Chen

Committee Member: Dr. Edgar Gabriel

Committee Member: Dr. Thamar Solorio

Committee Member: Dr. Rodolfo Ostila Monico

University of Houston
December 2019

ACKNOWLEDGMENTS

I would like to sincerely thank my advisor Dr. Guoning Chen for his patience, guidance, and support during my studies at the University of Houston. His knowledge, insights, and generous help gave me the confidence to pursue the challenging research projects that this dissertation presents. I am very grateful to Dr. Edgar Gabriel, Dr. Thamar Solorio, and Dr. Rodolfo Ostilla Monico for sharing their knowledge and experience and for taking the time to serve on my dissertation committee. It was my privilege to work with a distinguished researcher, Dr. Tobias Günther. I want to thank him for all his guidance in the last part of my research. I am also very appreciated that Dr. Guoning Chen took so much time in proofreading and revising my dissertation so that my dissertation has better readability and quality. I would like to thank my labmates, Doung Nguyen and Dr. Lieyu Shi for all their supports and helps during the past five years. Last but not least, I want to thank my parents, my sister and my friends Dr. Razieh Mehri and Dr. Faranak Shamsafar for their unconditional love and support that I am so blessed to have through my graduate study at the University of Houston.

ABSTRACT

Vector fields, especially flow, and their analysis are of paramount importance to a wide variety of scientific and engineering applications. Despite significant advances in the analysis and visualization of vector fields, the interpretation of their behavior still remains a challenge. There are different flow features that are of interest to different applications. Among them, vortices are the main subject for the study of turbulence flows. Vortices characterize the rotational behavior of the flow; they are responsible for the transportation of materials and energy. Despite their importance, there currently does not exist an effective way to identify and characterize vortices due to the lack of a unified definition of vortices. To address that, this research proposes three different approaches to analyze and visualize vortices based on their geometric and physical characteristics. The first approach characterizes the circulating pattern of vortices using a topology-based analysis, while the second approach identifies vortices based on their unique physics. The third approach uses a deep learning method to identify the boundaries of vortices, a traditionally difficult and unsolved problem. This dissertation provides detailed description and discussion on each of these three approaches. The proposed methods and their visualization have been applied to a number of 2D and 3D steady and unsteady flows to evaluate their effectiveness. Important flow characteristics that cannot be revealed with the previous methods are captured by the proposed methods.

TABLE OF CONTENTS

ACKNOWLEDGMENTS iii

ABSTRACT iv

LIST OF TABLES vii

LIST OF FIGURES xi

1 INTRODUCTION 1
- 1.1 My Contributions . . . 1
  - 1.1.1 Detect 3D Vortices via a Topology-based Analysis . . . 1
  - 1.1.2 Vortex Characterization via Quantifying and Visualizing Attribute Relations 2
  - 1.1.3 Vortex Boundary Identification Using Convolutional Neural Network . . . 3
- 1.2 Structure of the Dissertation . . . 5

2 Vector Fields and their Processing 6
- 2.1 Topology-based Approach . . . 6
  - 2.1.1 Background . . . 6
  - 2.1.2 Related Work in Topology-based Approach . . . 6
- 2.2 Physics-based Approach . . . 8
  - 2.2.1 Related Work in Physics-based Approach . . . 10
- 2.3 Deep Learning . . . 14
  - 2.3.1 Related Work in Vortex Extraction . . . 14
  - 2.3.2 Related Work in Using Deep Learning in Visualization . . . 16

3 Locate 3D Vortices via Morse Decomposition 17
- 3.1 Flow Combinatorialization . . . 18
- 3.2 Adaptive Sampling for Faces . . . 19
- 3.3 Hierarchical Refinement . . . 23
- 3.4 CUDA Implementation . . . 26
- 3.5 Results . . . 30
- 3.6 Summary and Future Work . . . 34

4 Physics-based Analysis of Vortices in Unsteady Flow 38
- 4.1 Correlation between Time-varying Attributes . . . 38
  - 4.1.1 Spatio-temporal Local Correlation Coefficient (LCC) . . . 39
- 4.2 Vector-based Correlation . . . 41
- 4.3 Measuring Attribute Dependency Using MI . . . 43
  - 4.3.1 Spatio-temporal Mutual Information . . . 43
- 4.4 Correlation Visualization . . . 45
- 4.5 Results . . . 49
  - 4.5.1 Performance Analysis . . . 60
  - 4.5.2 Parameter Study . . . 61

5 Deep Learning of Vortex Boundary Extraction 63

5.1 Synthetic Generation of Training Vector Fields 64
5.2 Pipeline 66
5.3 Generating Training Data 67
5.4 Architectures 68
5.5 Dataset 71
5.6 Results 72
5.6.1 Testing on Synthetic Data 73
5.6.2 Validation of Network on Numerical Data 73
5.7 Future Work 74

6 Conclusions and Future Work **76**

Bibliography **78**

LIST OF TABLES

1 Performance of the Morse decomposition for all four data sets with different levels of adaptive sampling. Both τ and resolution are fixed for the individual data set. Specifically, for Lorenz: τ=200 and res is $32 \times 32 \times 32$; for Tornado: τ=100 and res is $32 \times 32 \times 32$; for Bernard: τ=1200 and res is $64 \times 16 \times 32$; for cylinder: τ=200 and res is $96 \times 32 \times 24$. 30

2 Performance (in seconds) and memory usage (in MB) of the Morse decomposition of the Lorenz data with different integration steps and different resolution ($k = 2$). . 32

3 The comparison of the performance (in terms of the computation time (in seconds) and memory consumption (in MB)) of a GPU implementation versus a CPU implementation of the proposed pipeline with the Lorenz, Tornado, and Bernard data sets. The last column is the speed-up gained from the GPU implementation. 36

4 Performance result for different datasets. 61

LIST OF FIGURES

1 Sampled pathlines colored based on our proposed correlation method. 3
2 Visualizing boundary of vortices using deep learning (Unet) (a) vs. physics-based (b). 4
3 This shows various topological analysis results for an experimental vector field of: (a) ECG and (b) MCG(geometry-based), (c) MCG ($\tau = 6$) and (d) MCG ($\tau = 24$). [9] 7
4 The Morse decomposition computation pipeline. 9
5 Illustration of the adaptive face sampling strategy on regular grid. Top images show an example of face subdivision based on the connectivity of the image of face f_1. The bottom images illustrate an outer approximation (e.g., the shaded region) of a face (i.e., the red quad on the left). 20
6 Morse decomposition results of the Bernard data with $64 \times 16 \times 32$ cubic cells, using $\tau = 1200$. Different colors indicate different Morse sets. The Morse sets computed without using adaptive face sampling (top) are typically disconnected, while they are well connected with the proposed adaptive sampling strategy (bottom). This result highlights eight regions that correspond to the eight vortex bundles of the Bernard data. 22
7 The pipeline of the proposed hierarchical refinement framework of Morse decompositions of vector fields. 25
8 Results for the Tornado data set using the proposed hierarchical framework. The results shown in the first three images are generated with $k = 3$ and $\tau = 150$, 300, and 600, respectively. We see the Morse set become disconnected with $\tau = 600$. We then apply binary search for the ideal τ which turns out to be 450 (the right image). The computation times are 19.715, 58.419, 86.427, and 98.761 seconds, respectively. 26
9 The Morse set of the Lorenz system with resolution of $32 \times 32 \times 32$. The results are obtained using $\tau = 100$, 200, 400, 800 and 1600, respectively. The computation times are 24, 45, 84, 153 and 193 seconds, respectively. 30
10 Morse sets (magenta) of the Lorenz system in three different resolutions. In this case, we started with a regular $32 \times 32 \times 32$ cubical grid (left) on the domain $[-30, 30] \times [-30, 30] \times [-10, 50]$, and gradually doubled the resolution (middle and right). $\tau =$ 400, 800 and 1600, respectively. 31
11 Results for the Bernard-Rayleigh convection data set using the sub-sampled data set. The result on the top left is generated using $\tau = 1200$ and $k = 1$. The top right result is obtained using $\tau = 2400$ and $k = 2$. The bottom left image is generated using $\tau = 4800$ and $k = 3$, which contains a disconnected Morse set. We then apply binary search and determine the ideal τ for those local regions that may contain flow recurrent dynamics. The bottom right image shows the converged result with eight isolated Morse sets (in different colors). The τ used for these Morse sets ranges from 2400 to 3600. The computation times are 143.2, 287.34, 976.53 and 1032.51 seconds, respectively. 35

12 Results of the flow behind a square cylinder data set. The images in the left column show the Morse decompositions of the flow with increasing resolutions (from top to bottom). Specifically, the upper-left image visualizes the Morse set (in red) obtained with resolution $96 \times 32 \times 24$, $\tau = 900$ and $k = 4$; the middle-left image shows the Morse set with resolution $192 \times 64 \times 48$, $\tau = 1800$ and $k = 2$; the bottom-left image is the Morse set with resolution $384 \times 128 \times 96$, $\tau = 3600$ and $k = 1$. The computation times for these three results are 1150.98, 1621.2, and 2456.24 seconds, respectively. The images in the right column show the Morse decomposition results of the cylinder flow at time step 2028 with $\tau = 200$, 400, and 800, respectively. The resolution of the data for these results is $96 \times 32 \times 24$ and $k = 1, 2$ and 4, respectively. The computation times are 161.31, 326.4 and 519.574 seconds, respectively. Note that for all these results, only one Morse set is identified. 37
13 Overview of the two approaches to vortex tracking: subtracting a suitable ambient flow. Images are from [30]. 39
14 Different correlation measurements between norm (A_7) and shear rate (A_5) of a flow past a square cylinder. Kernel size $r = 3$ and $h = 250$ with $\tau = 0.008s$. For ST_LCC results, blue indicates negatively correlated and red indicates positively correlated. For MI results blue indicates low dependency and red indicates high dependency. . 41
15 The relationship between two associated information measurements for MI computation. 44
16 Sampled pathlines colored based on the ST_LCC (a), ST_VEC_LCC (b) and ST_MI (c) of acceleration and Q, respectively. Each top plot in (a) and (b) shows the ST_LCC/ST_VEC_LCC values along the respective pathlines indicated by the arrows. The two sub-plots beneath each large plot show the trends of the two given attributes within a kernel. For the MI result (c), the left two histograms show the distributions of the attribute values of Q within kernels on two pathlines (indicated by the arrows), while the right two show the distributions of the values of acceleration within the same kernels. Each histogram shows two distributions of the values of the respective attribute within two different kernels sampled at t=0 (blue) and t=50 (orange) on the corresponding pathline, respectively. Since we used equal-width bins, the number of bins is different for different attributes. 47
17 Spatial LCC of acceleration and Q (a) vs acceleration and shearing (b) and the segmentation result for three pairs of these attributes (c). In (c), red regions correspond to places where the correlation between acceleration and Q is the strongest. White regions visualize the places where acceleration and shearing has the highest value of correlation, and purple segments correspond to the regions where correlation between shearing and Q is the strongest. 49
18 ST_LCC results of curl and Q (a) and curl and shearing (c); ST_MI results of curl and Q (b) and curl and shearing (d). All results are computed based on the Lagrangian attributes using a spatial kernel r = 3 and temporal kernel h = 100 with $\tau = 0.01$ . . 50

19 Different correlation measurements of a region of the ocean simulation that contains eddies. (a) Spatial LCC, (b) Spatio-temporal (Lagrangian) LCC, and (c) mutual information (MI) (Lagrangian) between the *Acceleration* and *Determinant* of Jacobian matrix, respectively. For the first two results, the color shows the correlation characteristics, i.e., blue indicates negatively correlated and red positively correlated. For the MI result (c) blue indicates lower dependency and red indicates higher dependency. 52
20 Different correlation measurements between λ_2 and acceleration of the HCCI data set. Kernel size $r = 3$ and $h = 50$ (with $\tau = 0.05$)). 53
21 The comparison of conditional entropy for norm of the Jacobian (A_7) vs. λ_2 (A_3). Blue-white-red color coding is applied to (a) and (b), while for (c), light green indicates $H(A_7|A_3) < H(A_3|A_7)$ and blue means $H(A_3|A_7) < H(A_7|A_3)$. 54
22 The ST_LCC linear correlation (c) and ST_MI dependency (d) of the acceleration (a) and Q (b) of the large ocean simulation data with a spatio-temporal kernel size 5×50. The blue-white-red color scheme is used to highlight the different characteristics of attribute correlation and dependency. The ST_LCC result (c) highlights a string of strong vortices (or eddies) around the equator, while the MI result (d) emphasizes the strong offshore currents on the east coast of the individual continents. 55
23 The top row shows the ranking based segmentation for the spatial correlation between pressure and other attributes over time (from left to right). The green color indicates the regions where the correlation of pressure and vorticity is the highest, the pink color corresponds to the pair of pressure and kinetic energy, and blue for the pair of pressure and shearing. The second row shows the result of ST_LCC for acceleration and kinetic energy. The results highlight regions similar to the FTLE ridges. The spatial kernel size is $r = 3$ and the temporal kernel size is $h = 40$. 56
24 Spatial LCC of *acceleration* and vorticity for the 3D flow behind cylinder at time step 40. Note that the LCC result highlights the outer layer of the 3D vortices of the flow as expected. 57
25 Comparison of Lagrangian (L_) ST_MI and Eulerian (E_) ST_MI for the 3D cylinder data with different time windows h. Left column shows L_ST_MI for acceleration and Q, and the right shows E_ST_MI. We see that larger h makes both L_ST_MI and E_ST_MI smoother, but it does not change the structure of E_ST_MI much. 58
26 The spatial correlation of dye and vorticity for the elliptical instability simulation (a-f) and the Crow / reconnection simulation (g-l). In the top, we can see how the dye first tracks vorticity, until the non-linear elliptical instability kicks in and destroys the vortex, and the correlation is lost. In the bottom, we can see how the dye follows the vorticity much better even after the change in topology, and there is less mixture between red and blue regions. The spatial kernel size is 3. 59
27 The effect of different temporal kernel lengths, h, for ST_LCC (a-b) and ST_MI (c-d). The h values are 50 (left column) and 100 (right column), respectively, using $\tau = 0.05$. 60
28 Performance result for different metrics and different datasets. 61
29 The effect of spatial kernel size, r, on Lagrangian ST_LCC. From the left image to right, the r values for the results are 3, 5, and 10, respectively. 62

30 Overview of the deep learning reference problem. The input is vortex velocity patches and the output is a signed distance to the boundary of the vortex. The isoline for the boundary is with isovalue equal to 0. 63
31 Histograms of the individual model parameters, showing the near-Gaussian distribution of the individual parameter values [44]. 66
32 Examples of synthetically generated vector field patches. 67
33 Overview of our pipeline for optimizing the parameters. 68
34 Our CNN Architecture. The numbers below each box represent the dimensions of the feature maps. 69
35 One residual block in Resnet32 Architecture. 70
36 Our Unet Architecture. We used three layers of depth. The numbers above each box represent the dimension of feature maps. 71
37 Results of vortex extraction on test splits. The L2 errors are 0.12, 0.06 and 0.02 for the three architectures, respectively. 73
38 Comparison of boundary extraction with different architectures of neural network using the CYLINDER flow. The input to the networks is velocity patches. Our method shows Unet outperforms the other networks. 74

1 Introduction

Vector fields are one of the omnipotent data forms for the study of a wide range of continuous dynamical systems that describe the behaviors of gas and liquids under different circumstances. Applications that involve vector fields and their analysis include automobile and aircraft engineering, climate study, oceanography, combustion physics and chemistry, earthquake engineering, and medical practice, among others. Analyzing and visualizing vector fields, especially flow data, stemming from these applications to reveal features of interest is essential for the knowledge advances and technology innovation for these applications. There are different flow features that are of interest to different applications. Among them, vortices are one of the main features studied by different applications, as they are responsible for the transportation of material and energy. Despite its importance, there currently does not exist an effective way to visualize and characterize vortices due to the lack of a unified definition of vortices. My dissertation is dedicated to address this challenge.

1.1 My Contributions

To address the aforementioned challenge of vortex visualization and characterization, I introduce three different techniques that exploit the unique geometric and physical properties of vortices. These three techniques focus on the votices and their behaviors in steady and unsteady flows, respectively. Here, a steady flow is a vector field where the vector values at any positions of the domain do not change over time, while an unsteady flow is a vector field whose vector values change over time.

1.1.1 Detect 3D Vortices via a Topology-based Analysis

My first approach focuses on the vortices in the steady flow. It is inspired by an observation that the flow particles within a vortex area usually move around a common rotation center, namely the *vortex core*. This circulating flow pattern within a vortex is related to the so-called *flow recurrent dynamics*, which is used to define *vector field topology* (see Chapter 2 for a detailed definition). Based on this observation, my first approach resorts to the vector field topology and its computation

to identify the possible regions in the flow domain that may contain vortices.

To compute the vector field topology, I resort to the Morse decomposition of vector fields as it has been shown to be a reliable way to compute and represent vector field topology. Its computation first converts the original vector field into a directed graph representation, so that flow recurrent dynamics (i.e., Morse sets) can be identified as some strongly connected components of the graph. Since its introduction, Morse decomposition has been extended to analyze piecewise constant vector fields [83] and 3D vector fields defined on unstructured meshes [62]. However, there is still little work on the computation of Morse decomposition of vector fields defined on 3D regular grids.

To address this issue, I introduce an effective framework that enables the efficient computation of Morse decompositions of 3D *piecewise linear* vector fields that are defined on regular grids. Similar to the previous approaches, the first step of our framework is to convert the original vector field into a directed graph by tracking the image of each 3D cell of the regular grid based on the vector field defined on it (i.e., flow map estimation). To accurately estimate this image, we extend the adaptive edge sampling technique introduced in [9] and make use of the configuration of the regular grid to develop a 3D adaptive face sampling strategy. This leads to an accurate construction of the directed graph. After obtaining the graph, flow recurrent dynamic features can be identified as the strongly connected components of the graph.

1.1.2 Vortex Characterization via Quantifying and Visualizing Attribute Relations

While the above topology-based analysis may capture some regions with potential vortical flow behaviors, it also contains other non-vortex features (e.g., sinks, sources and saddles). In addition, in the fluid mechanic community, vortices are usually located via certain local physical relevant properties (or attributes), such as vorticity, Q-criterion, λ_2, and acceleration [**gunther2018state**]. Inspired by this common practice, in my second approach, I focus on characterizing vortices via their relevant local physical attributes.

Different from the existing approaches that usually concentrate on one local attribute at a time for vortex analysis, I propose to inspect two (or potentially more) relevant attributes together for

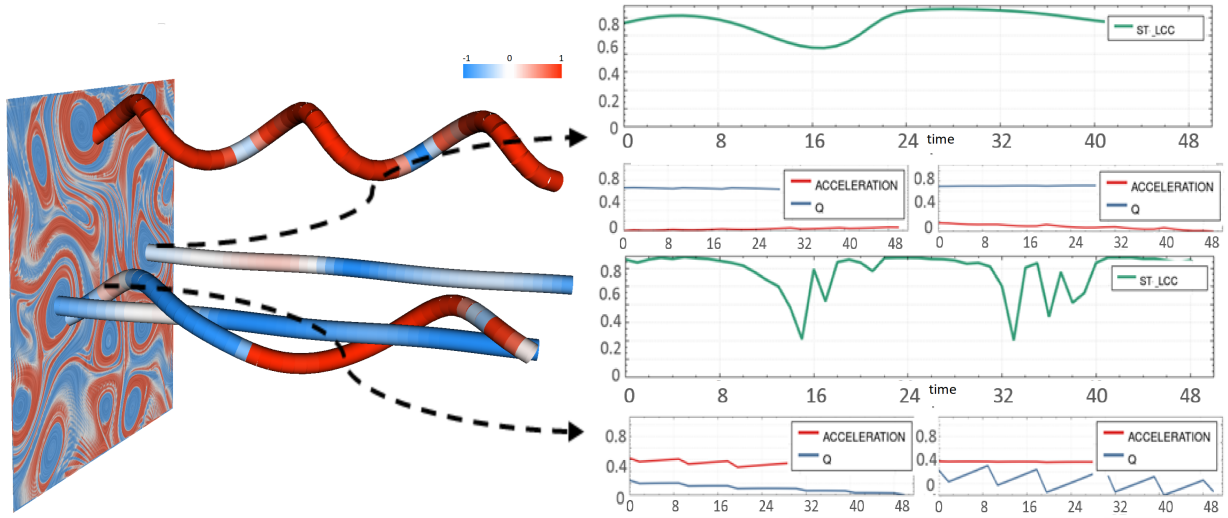
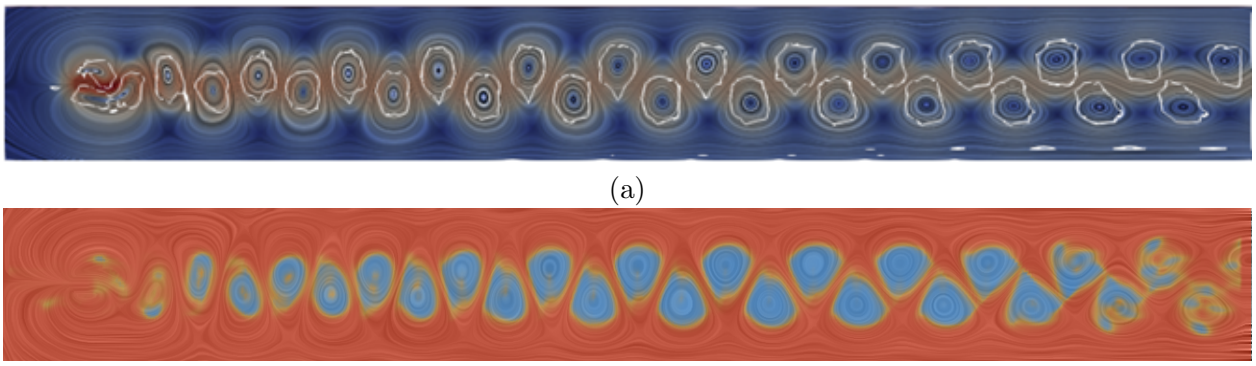


Figure 1: Sampled pathlines colored based on our proposed correlation method.

the characterization of vortices. In particular, my approach quantifies the co-varying behavior of pairs of relevant attributes over time in hope of finding vortex regions. This approach is based on the observation that the temporal behavior of certain attributes within a vortex region share some similarity. For instance, the vorticity value and the Q concentration remain relative high within the vortex core area during the transportation of the vortex, while their values exhibit some fluctuation at the boundary of the vortex as shown in Figure 1. We extend the existing spatial correlation quantification, i.e., the Local Correlation Coefficient (LCC), to the spatio-temporal domain to study the correlation of attribute-pairs from both the Eulerian and Lagrangian views.

1.1.3 Vortex Boundary Identification Using Convolutional Neural Network

The previous two approaches may capture the potential vortex regions. However, they cannot detect the boundaries of the individual vortices. Knowing the boundaries of vortices will provide additional information about vortex behaviors. For instance, if the boundary of a vortex is with a wiggly shape (maybe captured via a frequency analysis of the boundary curve), then it could be a sign of vortex breakdown. In addition, the extent and area of a vortex can be more accurately computed with the vortex boundary detected. Such information can be important in the study of



(a)

(b)

Figure 2: Visualizing boundary of vortices using deep learning (Unet) (a) vs. physics-based (b).

eddies of the oceans – a large ocean current structure for the transportation of nutrition and heat. Specifically, we can perform multiple simulations on ocean currents with different temperature increases (e.g., +1.5 degree Celsius or +2.0 degree Celsius set as a temperature increase bound in the Paris Climate Agreements). We then can quantitatively compare the impact of different temperature increases to the shape and extent of ocean eddies. Considering its importance, in the last part of my dissertation work, I am focused on the extraction of vortex boundaries.

The detection of vortex boundaries is challenging as there is not a unified definition of it. To address this challenge, I propose to use Deep Neural Network to detect vortex boundaries. In order to train a neural network, a large training set with example flows that contain vortices with different configurations that may be seen in the real-world flow is needed. To generate such a large training set, we employ a parametric flow model that allows us to precisely specify the configuration of a vortex, including its center, shape, orientation, and size. With this model, we generate thousands of synthesis flow patches with vortex boundaries labeled as the ground truth for the neural network to learn. We have used this set of synthetic flows to train different models with different architectures including a CNN, a Resnet, and a Unet with the velocity information as the input to these networks. After supervised learning, we apply these trained networks to real-world flow to demonstrate their effectiveness. Figure 2 (a) shows the detected vortex boundaries in a 2D flow behind a square cylinder using the trained Unet. As a comparison, Figure 2 (b) shows the

highlighted vortex regions using our second, physics-based approach.

1.2 Structure of the Dissertation

The rest of the dissertation is structured as follows. Chapter 2 provides a general background and reviews the work most relevant to the proposed techniques. Chapter 3 provides details of the proposed topology-based approach. Chapter 4 describes our second approach based on the analysis of the co-varying behaviors of pairs of physical attributes. Finally, Chapter 5 introduces our boundary extraction method based on deep learning, followed by a summary and discussion of future work, in Chapter 6.

2 Vector Fields and their Processing

In this chapter, we will briefly review some important concepts of vector fields that will be applied in the subsequent discussions. Consider a 3-manifold $\mathbb{M} \subset \mathbb{R}^3$, a general vector field $\mathbf{v}$ can be expressed as an ordinary differential equation $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t)$ or a map $\varphi : \mathbb{R} \times \mathbb{M} \to \mathbb{R}^3$, satisfying $\varphi_{t_0}^{t_0}(\mathbf{x}) = \mathbf{x_0}$ and $\varphi_{t_0}^{t+s}(\mathbf{x}) = \varphi_s^{t+s}(\varphi_{t_0}^s(\mathbf{x})) = \varphi_t^{t+s}(\varphi_{t_0}^t(\mathbf{x}))$. This flow map describes the spatial correlation of points through *trajectories* (or flow paths) starting at time t_0: $\varphi_{t_0}^t(\mathbf{x})$. In a steady flow, the trajectory $\mathbf{x}$ of a tracer particle is called a streamline. For a given seed point $\mathbf{x}$, the streamline is computed as the tangential curve one computes within a time step. While in an unsteady flow, the trajectory $\mathbf{x}$ of a tracer particle is called a pathline which is the tangential curve in the space and time domain.

2.1 Topology-based Approach

2.1.1 Background

In steady flows, a point $\mathbf{x}_0 \in \mathbb{M}$ is a *fixed point* if $\varphi(t, \mathbf{x}_0) = \mathbf{x}_0$ for all $t \in \mathbb{R}$. That is, $\mathbf{v}(\mathbf{x}_0, t) = 0$. $\mathbf{x}$ is a *periodic point* if there exists $T > 0$ such that $\varphi(T, \mathbf{x}) = \mathbf{x}$. The trajectory of a periodic point is called a *periodic orbit*. Both fixed points and periodic orbits are examples of *flow recurrent features*. The connectivity of these flow recurrent features represents the qualitative (or structural) information of the vector fields, which we refer to as *vector field topology*.

2.1.2 Related Work in Topology-based Approach

Helman and Hesselink were the first to introduce vector field topology to the visualization community [35]. Specifically, they defined a topological skeleton that is comprised of first-order fixed points and their connectivity. This topological skeleton has been extended to handle the boundary features and higher-order fixed points [71], respectively. Wischgoll and Scheuermann [99] introduced a technique for periodic orbit detection from 2D steady vector fields, which is later extended to 3D [98].

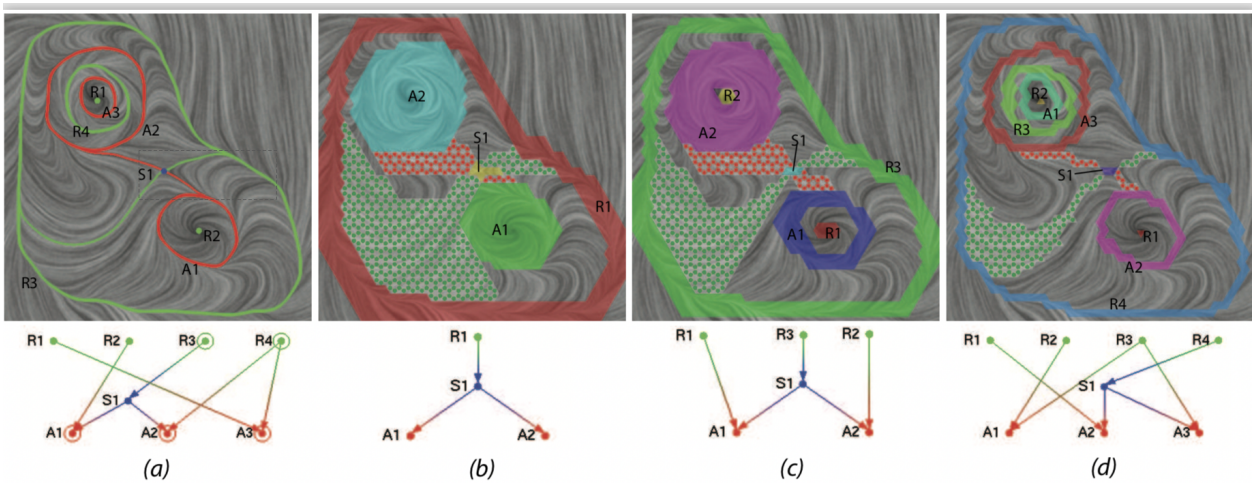


Figure 3: This shows various topological analysis results for an experimental vector field of: (a) ECG and (b) MCG(geometry-based), (c) MCG ($\tau = 6$) and (d) MCG ($\tau = 24$). [9]

Chen et al. introduced an efficient method for detecting periodic orbits from 2D/2.5D vector fields and defined a more complete vector field topology by including periodic orbits into the topology [10]. Theisel et al. proposed a saddle-saddle connector for 3D vector field topology visualization to reduce the occlusion issue [98]. Figure 3 shows their method results in comparing to the previous techniques. Extracting higher-order critical points has been introduced by Weinkauf et al. to assist the simplification of 3D vector field topological representation [99]. Considering the problem of instability in previous methods, Chen et al. introduced the first stable and discrete representation of vector field topology based on Morse decomposition [9]. This framework has been applied to 3D vector fields defined on unstructured grids [62] and extended to construct a hierarchical representation of the flow structure [12], respectively. Different from their work, we handle 3D vector fields defined on regular grids, which enables us to develop some efficient algorithms and implementations to compute a Morse decomposition for large scale data in pratice. Recently, Szymczak and Zhang proposed a stable Morse decomposition framework for piecewise constant vector fields on surfaces [84]. Later, Szymczak et al. presented an algorithm for computing nearly recurrent components for 3D piecewise constant (PC) vector fields defined on regular grids [83]. Note that piecewise constant vector fields assume a constant vector value within each cell, which changes the nature of the original vector fields. Different from their setting, we compute a

Morse decomposition directly based on the original piecewise linear vector fields.

Morse Decomposition A Morse decomposition is a collection of disjoint closed sets, called Morse sets. Together, they contain all the recurrent dynamics of the flow induced by the vector field. More precisely, sets M_i, $i \in \{1, 2, \ldots, N\}$, form a Morse decomposition if and only if the trajectory of any point is either (i) entirely contained in one of the Morse sets or (ii) is contained in some Morse set M_i for large enough negative times and in some other Morse set M_j, with $j > i$, for large enough positive times. Intuitively, (ii) means that the trajectory of any point outside the Morse sets can only move from a set with lower subscript to a set with a higher subscript.

Condition (ii) excludes any recurrent dynamics outside the Morse sets, making it gradient-like. In practice, the partial order between Morse sets can be represented as an acyclic directed graph called *Morse connection graph*, or MCG.

In [9], Chen et al. described a computation pipeline for Morse decompositions of the given 2D vector fields. In this pipeline, the input vector field is firstly converted into a directed graph, denoted by $\mathcal{F}$, through a numerical computation, called flow combinatorialization. The nodes of $\mathcal{F}$ are the individual triangles of the mesh where the vector field is defined. The directed edges indicate the flow mapping relations between triangles. For instance, if there is a directed edge $T_1 \to T_2$, the particles released at triangle T_1 can be advected by the flow and reach T_2 over certain time τ. In other words, $\mathcal{F}$ encodes the dynamics of the flow at a combinatorial level. From $\mathcal{F}$, Morse sets can be identified as the strongly connected components of $\mathcal{F}$ with non-trivial Conley index. The connectivity between the identified Morse sets can then be extracted by path searching between their corresponding strongly connected components in $\mathcal{F}$. It has been shown that this pipeline can be extended to 3D as illustrated in Figure 4.

2.2 Physics-based Approach

Consider an unsteady vector field $\mathbf{v}(\mathbf{x}(t))$ defined in a space-time domain $\Omega \subset \mathbb{R}^d \times \mathbb{R}$. Its dynamics are determined by the temporal evolution of densely placed particles, that is, for a particle $\mathbf{p}$, it satisfies $\frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \mathbf{v}(\mathbf{x}, t)$ ($\mathbf{x} = \mathbf{p}(t)$ is the position of the particle at time t).

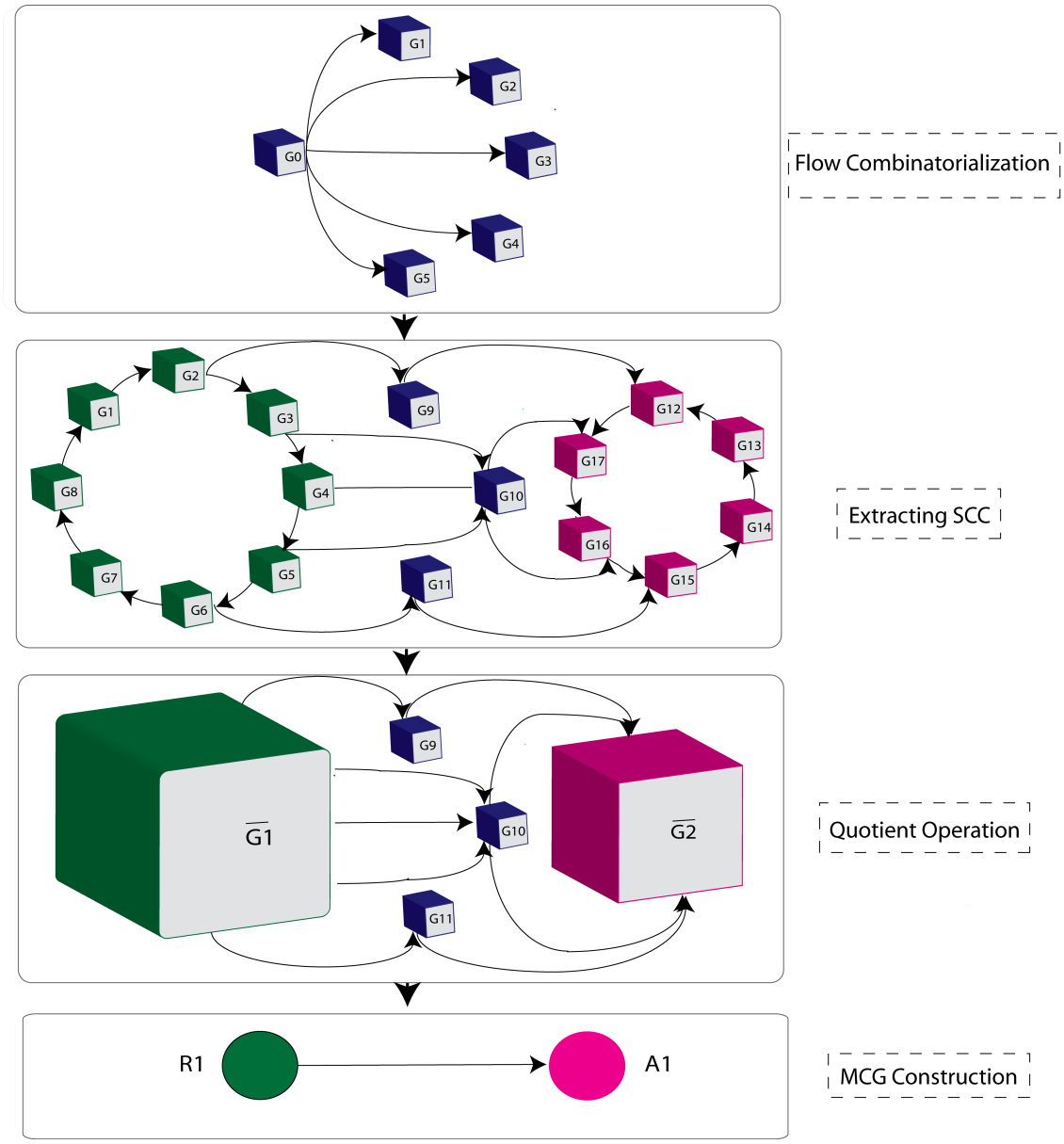


Figure 4: The Morse decomposition computation pipeline.

There are a number of local attributes for an unsteady vector field that are of interest to domain experts. For example, the *acceleration* of $\mathbf{v}$ is defined as $\mathbf{a}(\mathbf{x},t) = \frac{D\mathbf{v}}{Dt} = \frac{\partial \mathbf{v}(\mathbf{x},t)}{\partial t} + (\mathbf{v}(\mathbf{x},t)\cdot\nabla)\mathbf{v}(\mathbf{x},t)$, where $(\mathbf{v}(\mathbf{x},t)\cdot\nabla) = \nabla_{\mathbf{x}}\mathbf{v}$ represents the spatial gradient of $\mathbf{v}$, i.e., *Jacobian*, denoted by $\mathbf{J}$ for simplicity.

Important deformation modes of the flow can be obtained through the decomposition of $\mathbf{J}$. Specifically, the Jacobian matrix can be decomposed as $\mathbf{J} = \mathbf{S} + \mathbf{R}$, where $\mathbf{S} = \frac{1}{2}[\mathbf{J} + (\mathbf{J})^\top]$ and $\mathbf{R} = \frac{1}{2}[\mathbf{J} - (\mathbf{J})^\top]$ are the symmetric and antisymmetric components of $\mathbf{J}$, respectively. From this decomposition, the local shear rate is measured as the Frobenius norm of $\mathbf{S}$, the Q value at each point can be computed as $Q = \frac{1}{2}(\|\mathbf{R}\|^2 - \|\mathbf{S}\|^2)$ [30], and λ_2 is the second largest eigenvalue of the tensor $\mathbf{S}^2 + \mathbf{R}^2$ [40], respectively. They are typically used to characterize the vortical or rotational behavior of the flow. The attributes we consider in this dissertation work are denoted as follows:

- A_1: vorticity, $\langle \nabla \times \mathbf{v}, \mathbf{z} \rangle$ for 2D, and $||\nabla \times \mathbf{v}||$ for 3D.
- A_2: acceleration magnitude, $||\mathbf{a}(\mathbf{x},t)||$.
- A_3: λ_2 (see above definition).
- A_4: Q (see above definition).
- A_5: local shear rate (i.e., the Frobenius norm of $\mathbf{S}$).
- A_6: determinant of $\mathbf{J}$.
- A_7: norm of $\mathbf{J}$, defined as $\sqrt{\sum_{ij} J_{ij}^2}$.

Note that all the above attributes can be measured (or calculated) at a given position and time in the flow domain in parallel.

2.2.1 Related Work in Physics-based Approach

There exists a large number of algorithms and techniques for the visualization and analysis of vector-valued data. A series of survey papers [61, 48] provide a comprehensive review of this vigorous research area. In the following, we review only the work most related to the presented research.

Structural analysis of unsteady flow aims to study the transport behavior of the flow and identify the boundaries of different regions, such that the particles within each region exhibit similar temporal transport behavior [60]. There are various methods to define and compute the structure of unsteady flow, including topological feature tracking [89, 90, 87] based on bifurcation theory and pathline-based segmentation [88]. Nonetheless, the most successful and popular method is the computation of *Finite Time Lyapunov Exponent (FTLE)* fields [31, 51, 73, 27], whose ridges (i.e., *Lagrangian Coherent structures (LCS))* are extracted as the boundaries of different regions with distinct transport behavior. This method inspires a new direction of defining unsteady vector field topology based on streak lines/surfaces [67, 21, 25].

In addition to flow separation behavior in the transport of unsteady flow, the rotational behavior also leads to an important coherent structure, i.e., vortices [41, 24, 39]. In contrast to the transport structure, the vortical structure can be suppressed by a strong translational flow [85]. To reveal the vortex structure of interest, a certain reference frame needs to be extracted. Cucitore et al. [18] used a reference frame that moves with a particle. Haller [32] proposed M_z to detect vortices, which is both Galilean invariant and rotation invariant. Bhatia et al. [4] introduced an internal frame that is computed as the harmonic component of a natural Helmholtz-Hodge decomposition [3]. Günther et al. [29] showed how to construct a Galilean invariant and rotation invariant technique for vortex detection. Recently, Sauer et al. [69] introduced a novel method to explore spatio-temporal characteristics in both particle and volume data simultaneously. The transport and vortical behaviors in unsteady flow can also be studied via analyzing various attributes associated with pathlines [76, 59]. These attributes can be utilized for pathline and streak line placement [86] and glyph design [36].

Local analysis of unsteady flow focuses on small-scale behaviors, such as local volume dilation, compression, infinitesimal rotation, stretching, shearing, acceleration and momentum. Due to its locality, it is fast to compute and has been widely applied in the fluid mechanics community, for instance, the well-known λ_2 [40] and Q criterion [37] for the detection of vortices.

However, local analysis may not reveal the global flow structures associated with the transport behavior of the flow, and its visualization typically relies on the selection of a user-specified threshold to highlight the most salient regions, whose boundaries need not be aligned with the flow. Shi et al. [77] developed an advection filter along pathlines to study the transport behavior of various local properties. It is compared with the LCS structures of the corresponding FTLE fields. The set of local characteristics used in that work is adapted for our study. Fuchs et al. [26] introduced a local unsteadiness metric based on the material derivative. A global unsteadiness is then computed using a similar idea to the advection filter along pathlines. Critical points under the Lagrangian view are defined as the pathlines that have minimal unsteadiness.

Different from the above work, we concentrate on the pairwise relations of attributes based on their temporal behavior measured at both fixed locations and along particles.

Correlation analysis of multivariate data is one of the most important tasks in the study of multi-field data. Sauber et al. [68] analyzed correlations in 3D multi-field scalar data using a vector similarity metric derived from gradient similarity measures. Lee et al. [49] proposed an algorithm to describe the correlation among various attributes in multivariate time-varying data sets based on how the attribute values change over time. Wang et al. [28] studied hierarchical clustering of volumetric samples based on the similarity of their correlation for scientific volume data. Zhang et al. [104] introduced a correlation map, which was expressed as a 2D layout of variables encoding their pairwise correlations. This map is then employed for visual correlation analysis. Chen et al. [8] presented a sampling-based approach to classify correlation patterns, based on which a static volume classification was created to summarize the correlation connection in time-varying multivariate data sets. Recently, Zhang et al. [101] proposed a dissimilarity-preserving clustering algorithm and measured correlation connection in multi-variable and time-varying data sets. Their method can characterize time-varying patterns and spatial positions. The above works typically focus on linear correlation among attributes. In contrast, we study both linear and non-linear relation and/or dependency between attributes in unsteady flow.

Mutual Information (MI) quantifies the dependence between two random variables [15] based on information theory [75]. There are several papers that use information theory for various visualization problems. Xu et al. presented an information-theoretic framework to guide the selection and rendering of integral curves [100]. Biswas et al. proposed a new approach based on information theory through the multivariate data exploration process [6]. Wang et al. proposed a block-wise distribution based representation to preserve important features and alleviate uncertainty in large-scale scientific data visualization [92]. Zhang et al.[102] also analyzed the variable association in ensemble data. Analysis of dependent attributes values from a scalar-based view on multi-field data [20] and a brief overview of relation between the flow attributes [50] have been studied previously. Closely related to our work, Chen and Jaenicke were among the first to introduce mutual information to the visualization community in their seminal work [14]. In particular, they integrated the MI calculation in to their proposed framework to reduce the uncertainty in the generated visualization. Their work has a very different goal from our work. Specifically, we apply MI to understand the time-varying dependency of flow attributes. Dutta et al. [20] applied MI to the study of scalar fields. Their analysis is based on pointwise MI (PMI) comparison of time-varying scalar fields to identify salient features. Our work is a logical progression of this in a few ways. Firstly, we extend this idea based on comparison of pairwise integral curves (as opposed to points only). Secondly, we consider both Eulerian and Lagrangian settings and compare the two.

To our best knowledge, this work is the first that explores the spatio-temporal, linear and non-linear relation of unsteady flow attributes using various metrics from both the Eulerian and Lagrangian perspectives. We also compare the ability of various metrics in revealing time-varying relationships between attributes, which can offer guidance for selecting metrics to study and compare amongst the range of flow attributes.

2.3 Deep Learning

2.3.1 Related Work in Vortex Extraction

The foundations of many formal vortex definitions were laid out between the late 70s and early 90s. As an example, Lugt [55] defined vortices as follows: "Any mass of fluid moving around a common axis constitutes a vortex. Mathematically, such motion can be described by closed or spiraling streamlines (or pathlines) if a reference frame exists for which the flow field becomes steady." Another definition was proposed by Robinson [63]: "A vortex exists when instantaneous streamlines mapped onto a plane normal to the vortex core exhibit a roughly circular or spiral pattern, when viewed from a reference frame moving with the center of the vortex core." Both definitions include two concepts that are still actively researched to this day. Both indicate the essentiality of a vortex coreline, i.e., a line that particles rotate around, and they require to select an appropriate reference frame [**gunther2018state**]. Based on these requirements, there are several groups of methods that identify the vortex regions that we briefly explain in this section.

Region-based Methods Region-based methods identify a volume of vortex-like behavior based on some physical attributes such as pressure [37], vorticity, Hellicity, Q-criterion, Okubo-Weiss criterion and λ_2 criterion [30]. The main limitation of this group of methods is that most of them require the selection of certain threshold value to identify vortex regions, which may miss vortices with smaller values (i.e., weak vortices).

Line-based Methods These methods mostly focus on the coreline of vortices. In steady flow, they can be identified using a reduced velocity criterion, which considers the eigenvalues and eigenvectors of the Jacobian $\mathbf{J}$ [93]. In unsteady flow, there exist several methods such as in magnetic fields or when the vector field is transformed into a reference frame, in which the flow appears steady [**gunther2018state**]. However, many line-based methods are numerically instable and easily result in fragmented corelines.

Geometric Methods These methods mainly focus on constructing the skeleton of a vortex tube. For 2D flows, Sadarjoen et al. [66, 65] presented two geometric approaches that are based on the shape of streamlines. Their *curvature center method* computes the density of curvature centers for a given set of streamlines. Their second method called *winding angle* is based on the angle between two subsequent polyline segments. Streamlines that have winding angles larger than a threshold are considered part of vortices.

Integration-based Measures All the above methods are local. However, it was shown that there are classes of vortices that cannot be extracted by local methods, for instance attracting vortices that move on non-linear paths. As a solution, integration-based measures such as particle density estimation [96] and analyzing of Jacobian [93] were developed. They proposed to inject a number of particles and observe their attraction behavior over time.

Objective Methods Objectivity refers to the invariance of a measure under a change of the reference frame. There are several measurements that could be used to identify vortex objectivity for instance, Relative Vorticity Tensor-based measures, Strain Tensor-based measures and Vorticity-based measures[30].

Extraction of Vortex Boundaries In addition to finding vortex corelines, the size of vortices is also of interest. With region-based methods, the size is determined by a threshold, which is typically difficult to set. Line-based methods may serve as a starting point for region growing approaches. There exist a few research works that aim to estimate the size based on streamlines or corelines [30]. To the best of our knowledge, the only work that has focused on vortex boundary extraction was proposed by Haller et al. [33]. He proposed elliptic LCS, which preserve arc length and area in incompressible 2D flows, and considered the outermost elliptic LCS, of a family of nested elliptic LCSs, as the boundary of a coherent vortex. This work will bring a new perspective to solve this problem considering the power of feature extraction and deep learning.

2.3.2 Related Work in Using Deep Learning in Visualization

In recent years, several deep learning approaches appeared for vortex extraction. Bai et al. [2] introduced an RCNN to detect the ocean eddy which is a circular current water in the ocean. Kim and Günther proposed a robust reference frame extraction method [45] based on the Vatistas velocity profile. It uses neural networks to extract a steady reference frame from a given unsteady 2D vector field. Under increasing noise and resampling artifacts, this method is robust on real-world data.

The classification approach of Bin and Li [5] categorizes flow patches into rotating (cw/ccw), saddles and others. More specific features have been searched with the classification network by Ströfer et al. [82], looking for recirculation, boundary layers and a horse shoe vortex. Deng et al. [19] trained a convolutional neural network to extract the instantaneous vorticity scalar field of Haller et al. [33]. Given the sea surface height, Lguensat et al. [52] extracted ocean eddies. Franz et al. [23] described a CNN that receives the Okubo-Weiss criterion [57, 95] as input to detect ocean eddies and followed their paths using optical flow and a spatio-temporal recurrent neural network (RNN). Kim and Günther [45] developed a CNN that extracts a reference frame in which a given unsteady flow becomes steady, enabling vortex coreline extraction. The network was trained with noisy synthetic data to improve the robustness. Liu et al. [53] extracted shock waves using CNNs.

3 Locate 3D Vortices via Morse Decomposition

This chapter describes our topology-based approach, called Morse decomposition, for the locating of 3D vortices.

Conventional vector field topology, also referred to as the *differential topology*, is computed based on the characterization of streamlines – solutions to the ordinary differential equation that describes the vector field, which is numerically unstable. To address this, Chen et al. [9] introduced the Morse decomposition as a stable representation of the vector field topology. Different from the differential topology, Morse decomposition represents the flow recurrent dynamics by the individual disjoint sub-regions of the flow domain, called *Morse sets*, that enclose these flow recurrent features. This coarse representation offers additional room for topology to tolerate a certain amount of error or perturbation, resulting in a more stable representation compared to the differential topology.

One challenge for Morse decomposition computation is to determine an ideal integration time for the flow map estimation, which can accurately capture the flow behavior while achieving efficient calculation. To address that, we extend the hierarchical refinement framework for 2D vector fields [12] to 3D flows, which enables the Morse decomposition to start with a smaller integration time. This integration time will be increased gradually at regions that may contain flow recurrent dynamics to refine the obtained results. In addition, our refinement framework borrows ideas from the recently introduced image-space Morse decomposition for 2D vector fields [11] and procedurally sub-divides the 3D cells to refine the boundary of the obtained Morse sets. Figure 10 provides an example of such a refinement to the Morse set detected from the Lorenz system.

Since the input vector fields are defined on regular grids, a Compute Unified Device Architecture, (CUDA) implementation of our framework is possible, which largely improves the computation performance of our framework. This is much needed in practice when handling large scale 3D vector fields. To our best knowledge, this is the first work that addresses Morse decomposition of piecewise linear vector fields defined on 3D regular grids. It is also the first time that a CUDA implementation of Morse decomposition is presented. We have applied our framework to a number

of analytic and real-world 3D data sets to demonstrate its utility.

The rest of this chapter is organized as follows. Section 3.1 provides the detail of our 3D flow combinatorialization based on the adaptive face sampling. Section 3.2 describes our pipeline of hierarchical Morse decomposition. Section 3.4 details the implementation of our algorithm using CUDA. Experimental results for analytical and simulation data are provided in Section 3.5.

3.1 Flow Combinatorialization

As described earlier, the most important step in Morse decomposition computation is the flow combinatorialization. In this work, we employ the idea of τ-maps for the computation of flow combinatorialization. In the rest of our discussion, we assume that the underlying computation domain is represented by a regular grid. Vector values are defined at the vertices and tri-linear interpolation is used to obtain values on the edges and inside the grid. The purpose of flow combinatorilization is to construct the directed edges $e_k = (V_i \rightarrow V_j)$ (V_i and V_j are two voxels) in $\mathcal{F}_\tau$, which accurately encodes the flow map induced by the vector field.

As shown in previous work [9], the directed edges starting from each node in $\mathcal{F}_\tau$, i.e., corresponding to a 3D cell (or a voxel) V, can be obtained by locating all the 3D cells, $\{V_j\}$, that intersect with the image of V, denoted by $\varphi_{t_0}^{t_0+\tau}(V)$ or simply $\varphi_\tau(V)$. That is, $e_k = (V \rightarrow V_j) \in \mathcal{F}_\tau$ $\iff$ $V_j \cap \varphi_\tau(V) \neq \emptyset$.

This set of 3D cells constitutes the *outer approximation* of $\varphi_\tau(V)$. Hence, to obtain an accurate directed graph, $\mathcal{F}_\tau$, it is essential to compute an accurate outer approximation of the image of each 3D cell of the given grid. A naive approach for the outer approximation estimation is to advect the densely placed particles within each 3D cell over time τ and locate the set of cells that enclose the end positions of these particles. However, such a naive approach will usually result in disconnected Morse sets due to inaccurate estimation of the flow map (see an example shown in Figure 6 top). To address that, we extend the 2D adaptive edge sampling strategy [9] and propose an adaptive face sampling process described as follows.

3.2 Adaptive Sampling for Faces

Consider a voxel V and its boundary ∂V. Let $\varphi_\tau(\partial V)$ be the image of the ∂V under the flow φ over time τ. It suffices to have an accurate outer approximation of $\varphi_\tau(\partial V)$ to obtain the outer approximation of $\varphi_\tau(V)$. The interior of $\varphi_\tau(V)$ will be identified via a backward mapping [9]. To obtain the outer approximation of $\varphi_\tau(\partial V)$, we propose an adaptive face sampling strategy for each V. This adaptive sampling is based on the observation that the image of a connected object under a continuous map remains connected. Consequently, the basic idea of our strategy is to preserve the connectivity of $\varphi_\tau(\partial V)$ during mapping.

Specifically, we are interested in finding the set of connected voxels that contains the image of each face of a voxel. To compute this set of connected voxels, we first release particles from the four vertices (or corners) of each face f_i. If the voxels that contain the end positions of these four particles are not identical or neighboring to each other (i.e., they are not connecting), more seeds will be inserted between the previous seeds recursively. We achieve that by subdividing f_i into four equal quads. We consider each new quad as a new face $f_{ij}^{(k)}$ and apply the same process above, i.e., placing particles at the corners of each $f_{ij}^{(k)}$ (excluding those corners that have been considered in the previous iteration) and validating the connectivity of the voxels that enclose their images. This recursive process continues until we locate a set of connected voxels that completely encloses the image of f_i. In theory, this process will converge given a finite τ and a continuous flow. In practice, we set a maximum recursive level k to prevent the stack overflow exception because of the memory limitation of the GPU. In all our experiments, we set k up to 4. Algorithm 1 provides a pseudo-code of our adaptive face sampling. Figure 5 provides an example to illustrate our adaptive face sampling. For a face f_1, initially four particles are placed at the four vertices v_i and advected by the flow over time τ (left image of Figure 5(a)). If the two voxels containing the images of the two vertices v_1 and v_2 are neither the same nor neighbors, we then divide f_1 to four faces in a way that v_{cc} located at the center and v_{c1} to v_{c4} are located in the middle of each edge (middle image of Figure 5(a)). We trace streamlines from the new faces and determine whether the voxels containing the images of faces f_{11} and f_{12} are connected or not. If they are not, it means that we

need more samples on face (f_{11}). Therefore, we split the face segment (f_{11}) to four equal faces and recursively call the algorithm until a set of connected voxels is found.

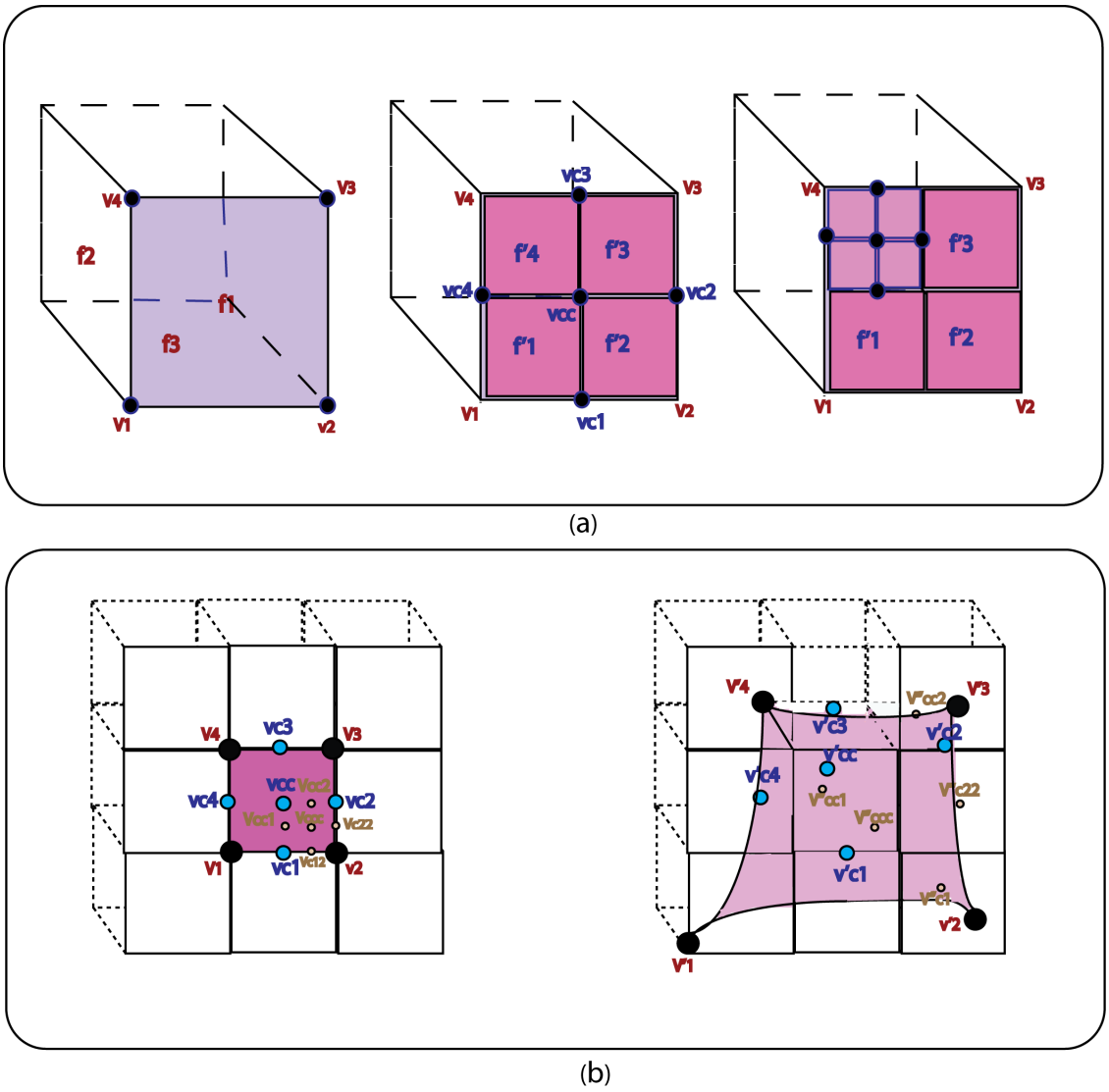


Figure 5: Illustration of the adaptive face sampling strategy on regular grid. Top images show an example of face subdivision based on the connectivity of the image of face f_1. The bottom images illustrate an outer approximation (e.g., the shaded region) of a face (i.e., the red quad on the left).

Figure 6 (bottom) shows the Morse decomposition result with the proposed adaptive face sampling. Note that in our experiments, τ is estimated by the number of integration steps.

Algorithm 1 Adaptive sampling on a face algorithm

procedure ADAPTIVE_ FACE_ SAMPLING(V,f, τ,k)

Input: V: vector field; f: current face; τ: user specified integral time; k: the level of adaptive face sampling.

Output: V_o:the index of edges that will be added to the graph $\mathcal{F}_\tau$ in Algorithm 2 (after all GPU computation is done).

Local variables:

f_1', f_2', f_3', f_4': the four sub-faces of f;
v_1, v_2, v_3, v_4: the four vertices of f;
e_1, e_2, e_3, e_4: the four end points of f;
L: the current level of adaptive face sampling.

Begin

$L = L + 1$

if (L > k ||
check_ neighborhood_ condition(e_1, e_2, e_3, e_4))
return;
else
$v_{c1} = add\_particle(v_1, v_2)$;
trace(v_{m1}, τ);
$v_{c2} = add\_particle(v_2, v_3)$;
trace(v_{m2}, τ);
$v_{c3} = add\_particle(v_3, v_4)$;
trace(v_{m3}, τ);
$v_{c4} = add\_particle(v_4, v_1)$;
trace(v_{m4}, τ);
$v_{cc} = add\_particle–center(f)$;
trace(v_{cc}, τ);
check_edge_end_points();
f_1' = build_ new_ face($v_1, v_{c1}, v_{cc}, v_{c4}$);
call Adaptive_ face_ sampling(V,f_1',τ,L);
f_2' = build_ new_ face($v_{c1}, v_2, v_{c2}, v_{cc}$);
call Adaptive_ face_ sampling(V,f_2',τ,L);
f_3' = build_ new_ face($v_{cc}, v_{c2}, v_3, v_{c3}$);
call Adaptive_ face_ sampling(V,f_3',τ,L);
f_4' = build_ new_ face($v_{c4}, v_{cc}, v_{c3}, v_4$);
call Adaptive_ face_ sampling(V,f_4',τ,L);

end procedure

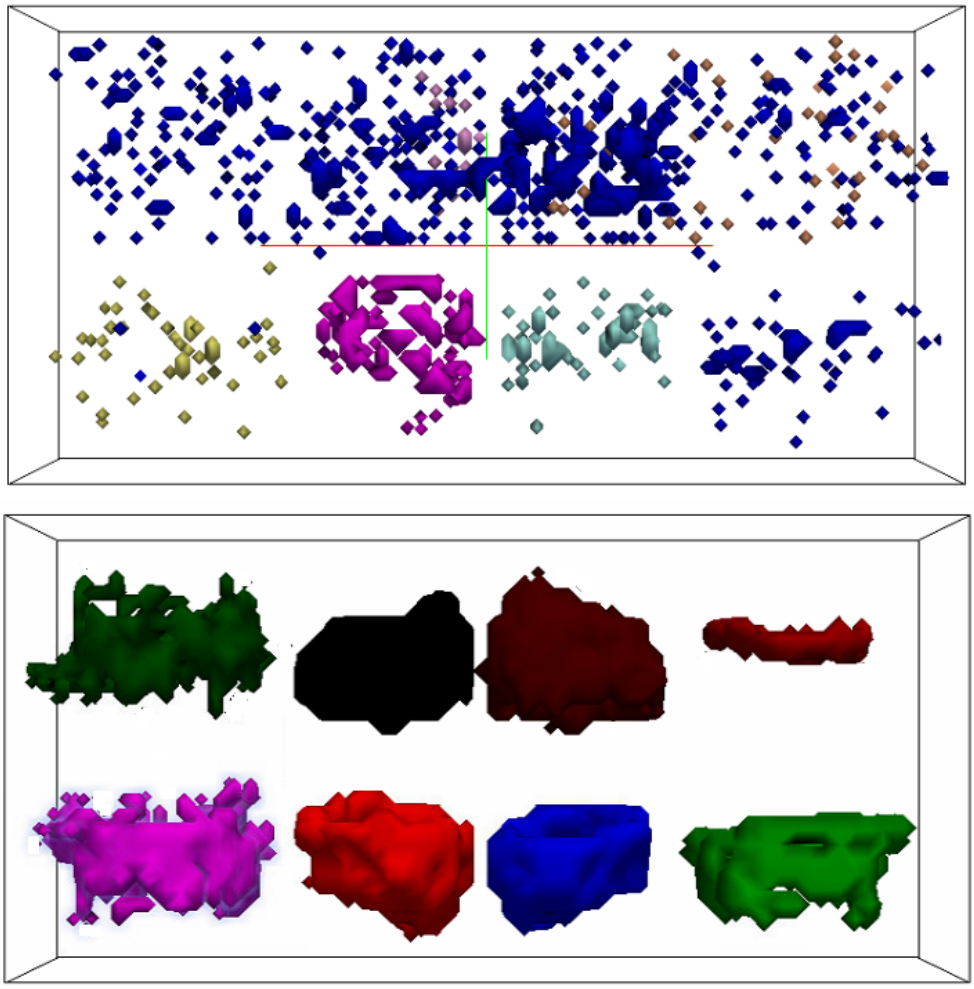


Figure 6: Morse decomposition results of the Bernard data with $64 \times 16 \times 32$ cubic cells, using $\tau = 1200$. Different colors indicate different Morse sets. The Morse sets computed without using adaptive face sampling (top) are typically disconnected, while they are well connected with the proposed adaptive sampling strategy (bottom). This result highlights eight regions that correspond to the eight vortex bundles of the Bernard data.

The complete flow combinatorialization is computed in both forward and backward flow directions. Particularly, we first estimate $\mathcal{F}_\tau$ in the forward flow direction using the above adaptive face sampling process and add the obtained directed edges accordingly. Then, we estimate $\mathcal{F}_\tau$ in the backward flow direction using the same adaptive face sampling process and add the corresponding edges by avoiding the repeated edges. In addition to the above boundary estimation, we also seed particles at the centers of the individual cells and compute their images and add the corresponding

edges while avoiding redundancy [9].

3.3 Hierarchical Refinement

Even with the above flow combinatorialization computation, it still remains a challenge to determine an ideal τ to compute the Morse decompositions with sufficiently fine Morse sets and fast computation. Chen et al. [9] have shown that in general, the larger the τ the finer the decomposition result will be but with slower computation and larger error (e.g., disconnected Morse sets). In practice, users have to select different τ's in order to get the desired result. To address that, Chen et al. [12] proposed a hierarchical refinement framework which enables to compute a Morse decomposition using a smaller τ at first, then gradually refines the results by using a larger τ within the Morse set regions detected in the previous computation. We adapt their framework and combine it with an additional cell subdivision process to refine the boundaries of the resulting Morse sets, whose smoothness is known constrained by the resolution of the underlying grid.

Specifically, our hierarchical refinement method consists of two stages. In the first stage, we gradually increase τ for the refinement of the detected Morse set. This is based on the work by Chen et al., which demonstrates that computing the flow combinatorialization does not require a constant τ value everywhere in the domain [12]. This refinement process enables us to semi-automatically determine an ideal τ for the extraction of finer Morse sets within a local region given its flow characteristics. In the second stage, we sub-divide the voxels that fall in the resulting Morse sets from the first stage and re-compute the local Morse decomposition within each Morse set region with a higher grid resolution. This enables us to partially improve the smoothness of the Morse set boundaries for visualization, which is in a much similar spirit of the image-space Morse decomposition framework [11]. In particular, our refinement framework can be described as follows.

1) An initial Morse decomposition is computed using a small τ. 2) For each obtained Morse set (we denote it as M_i), if its size (i.e., total number of voxels within it) is larger than a user-specified threshold, we increase τ_i as $\tau_i' = 2\tau_i$ and recompute the Morse decomposition within a sub-volume

corresponding to M_i. 3) If the newly obtained Morse sets are all well-connected individually, we put them into a queue Q, set $\tau_i = \tau_i'$, and repeat step 2) with a Morse set from Q. Otherwise, if the extracted Morse sets become disconnected or we reach the threshold for adaptive face sampling (i.e., $k > 4$), we go back to the previous level of this Morse set M_i and set $\tau_i' = (\tau_i + \tau_i')/2$, then go to step 2). 4) After all Morse sets converge using steps 2) and 3), we refine the grid within the obtained Morse sets by doubling their resolution. For each Morse set M_i and its corresponding τ_i, we use $2\tau_i$ to compute a new Morse decomposition within it. If the newly obtained Morse sets are all well-connected individually, we put them into Q, and repeat step 4) to increase its resolution unless the memory doesn't permit this. Otherwise, we set $\tau_i' = (\tau_i + \tau_i')/2$ and recompute the Morse decomposition within M_i and go to step 4).

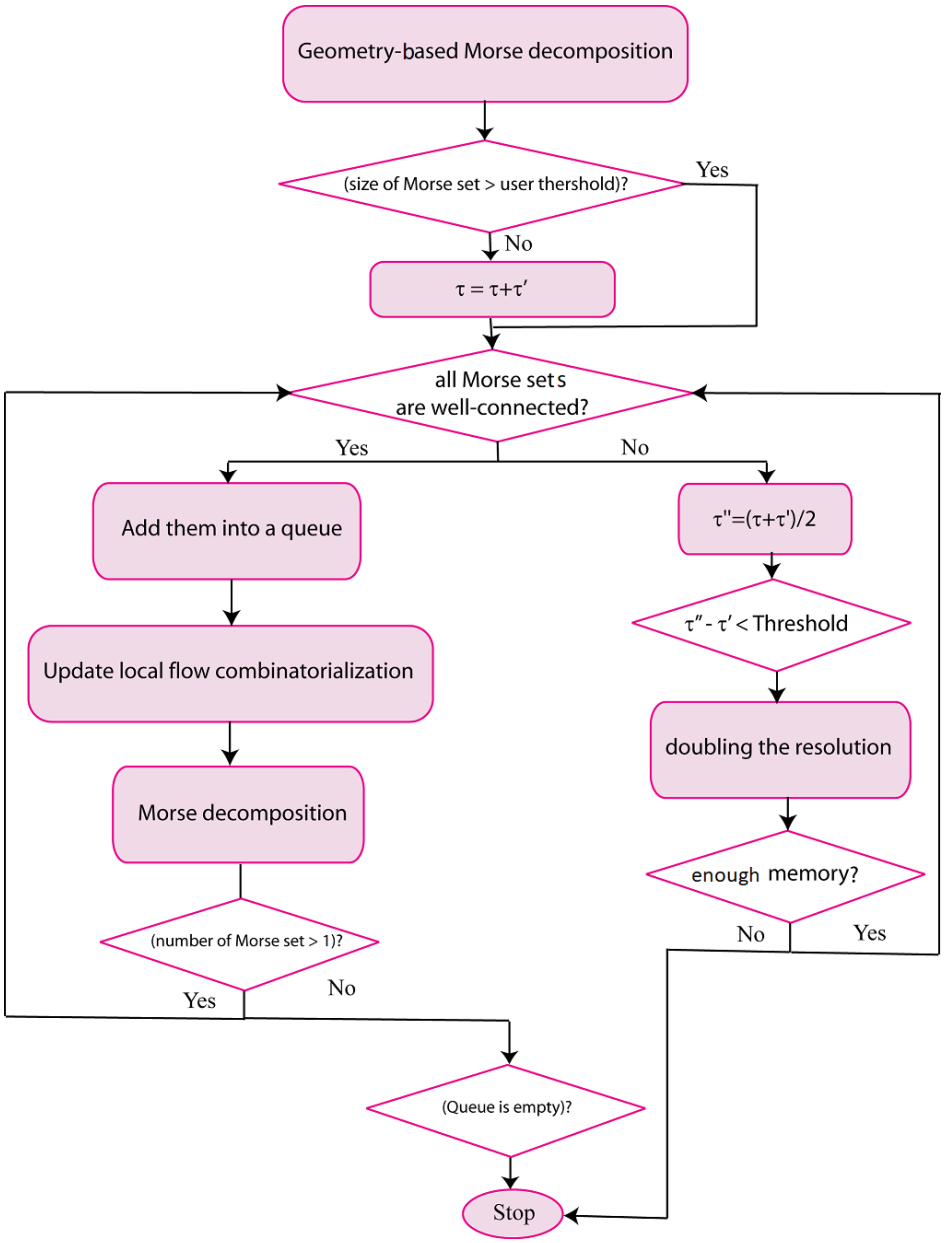


Figure 7: The pipeline of the proposed hierarchical refinement framework of Morse decompositions of vector fields.

Figure 7 shows the flowchart of our algorithm. To explain how the hierarchical framework searches for an ideal τ, we use the Tornado data as an example. Figure 8 shows the result of Tornado with different numbers of integration steps. We start from $\tau = 150$ and increase it by doubling it using the hierarchical refinement framework. The third image in Figure 8 shows the

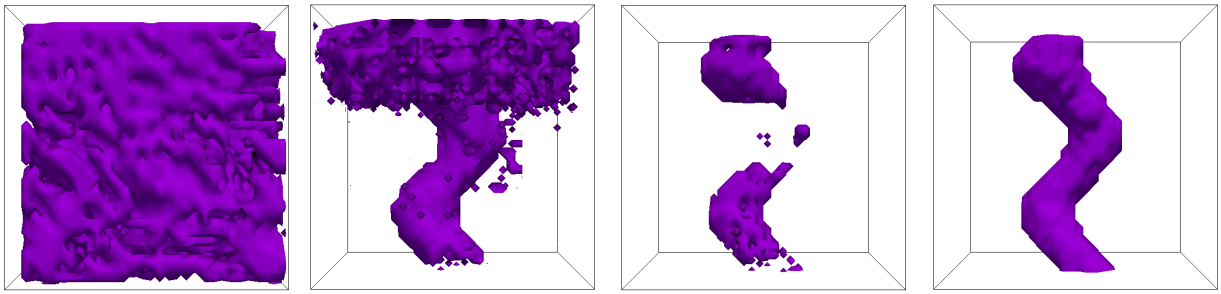


Figure 8: Results for the Tornado data set using the proposed hierarchical framework. The results shown in the first three images are generated with $k = 3$ and $\tau = 150$, 300, and 600, respectively. We see the Morse set become disconnected with $\tau = 600$. We then apply binary search for the ideal τ which turns out to be 450 (the right image). The computation times are 19.715, 58.419, 86.427, and 98.761 seconds, respectively.

result which is obtained by $\tau = 600$. The resulting Morse set is disconnected. We then perform a binary search for the ideal τ, as illustrated in the pipeline. The ideal τ for this data is 450. We then subdivide the cells falling in this Morse set in an attempt to further refine the Morse set. However, the resulting Morse set with a finer resolution does not improve much for this example.

3.4 CUDA Implementation

Even with the previously described hierarchical framework, the computation cost for the Morse decomposition of 3D vector fields can still be very high. To further accelerate the computation, we explore the GPU and CUDA implementation of our framework. As the estimation of the outer approximation of each 3D cell (or voxel) is independent of the others, it is natural to process them in parallel. In the higher-level, one can assign a thread for the outer approximation computation for each cell. In the lower-level, the outer approximation is extracted via the tracing of particles seeded at each cell. Ideally, if the advection of each particles is independent of the others, a thread can be assigned to it to achieve the maximum parallelization. However, considering the proposed adaptive face sampling strategy, not all the required particles and their initial positions are known at the beginning. This makes their parallel computation challenging. To address that, we proceed as follows (see Algorithm 2).

At first, streamlines are computed concurrently by executing a CUDA thread per vertex of the

grid for the initialization of flow combinatorialization. These streamlines and their end positions will be accessed multiple times in the subsequent face adaptive sampling and connectivity determination. Function *trace_Streamlines_GPU* in Algorithm 2 accomplishes this initialization. An important array for this initialization is h that contains the coordinates of the vertices. It is passed to the GPU memory.

Algorithm 2 An efficient GPU outer approximation computation

procedure CONSTRUCT_ EDGE_ MAP(V,B τ,k)

Input: V: vector field; B: boundary coordinates;

Output: f_τ: completed graph;

Local variables: h: array of vertex coordinates; e: array of end points coordinate; h_p: array of seed point coordinates; e_p: array of coordinates of ending positions of seed points; Q is a queue for saving Morse sets;

Begin

Generate_ Vertex_ Coordinates(h);

for all h $\in B$

do in parallel

$e \leftarrow Trace\_Streamlines\_GPU(h, \tau, v)$;

end in parallel

end for

Add_ Edges(h,e,f_τ);

Call Morse_ Decomposition(f_τ);

$Current\_Morse\_Sets$ = Save_ Morse_ Set();

for all h $\in Current\_Morse\_Sets$

do in parallel

$(h_p, e_p) \leftarrow Adaptive\_Face\_Sampling\_GPU(h, e, \tau, k)$;

end in parallel

end for

Add_ Edges(h_p,e_p,f_τ);

Call Morse_ Decomposition(f_τ);

$Current\_Morse\_Sets$ = Save_ Morse_ Set();

for all $x \in Current\_Morse\_$ Sets **do**

$if(!Check\_Connectivity(x))$; Add x to Q;

end for (!isEmpty(Q))

End

end procedure

After computing the streamlines starting from vertices, the ending positions of the streamlines will be saved at e. Based on these end positions, the corresponding directed edges are added to the graph. Specifically, since one vertex is shared by six voxels, six directed edges starting from these six voxels will be added for each streamline. In the next step, we perform the face sampling (i.e., the *Adaptive_Face_Sampling_GPU* function in Algorithm 2), in which we need the ending positions of the vertices of the six faces of each voxel which are computed in the last step. Again, since each vertex is shared by six faces, we access the ending position and starting position of this voxel using four threads. One thread will cover three faces and the other three faces will be covered by three other threads running for other vertices.

During the adaptive sampling process, the connectivity of the voxels that contain the end positions of the streamlines starting from a given sub-face can be achieved by checking the adjacency of the indices of these voxels. To reduce repetition in the computation, in the implementation of the *Adaptive_Face_Sampling_GPU* function, we only consider the three faces whose normals are in the forward flow direction, and the other three faces will be covered by other threads assigned to other adjacent voxels. Because in each level of face sampling we need to compute the previous level completely, each face must be split sequentially. The algorithm of this adaptive face sampling process has been described in Figure 5 and Algorithm 1.

Since it is impossible to run a particle per thread because the number of required levels of adaptive sampling as well as the required particles are unknown, we allocate the GPU memory to the size of maximum number of particles in all levels that the algorithm may need. As we add one particle to the middle point of each edge of every (sub-)face and one to its center, each (sub-)face will place totally five particles. If we set $k = 4$, the total number of all the (sub-)faces involved in this adaptive sampling process will be $341 = \sum_{k=0}^{4} 4^k$. Thus, the maximum number of particles needed for processing a face will be $5 \times 341 = 1705$.

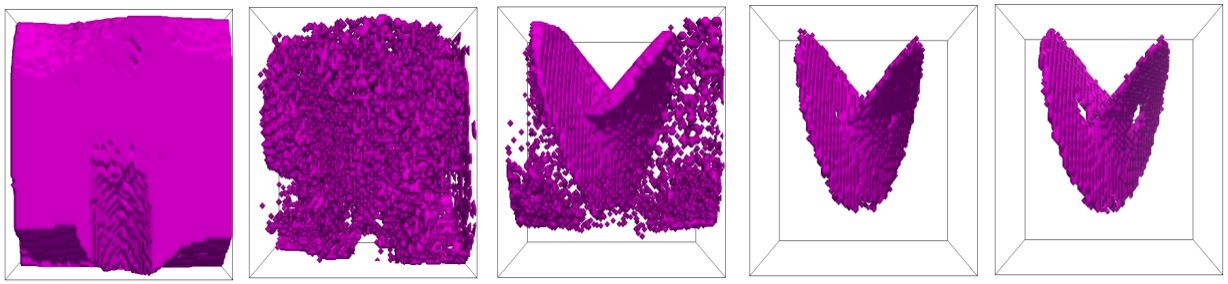


Figure 9: The Morse set of the Lorenz system with resolution of $32 \times 32 \times 32$. The results are obtained using $\tau = 100$, 200, 400, 800 and 1600, respectively. The computation times are 24, 45, 84, 153 and 193 seconds, respectively.

Table 1: Performance of the Morse decomposition for all four data sets with different levels of adaptive sampling. Both τ and resolution are fixed for the individual data set. Specifically, for Lorenz: τ=200 and *res* is $32 \times 32 \times 32$; for Tornado: τ=100 and *res* is $32 \times 32 \times 32$; for Bernard: τ=1200 and *res* is $64 \times 16 \times 32$; for cylinder: τ=200 and *res* is $96 \times 32 \times 24$.

Dataset	Measurements \ k	0	1	2	3	4
Lorenz	Time (s)	1.453	21.768	24.42	52.330	89.366
	Memory (MB)	398.0	1,156.3	1,156.2	1,159.1	1,162.5
Tornado	Time (s)	1.543	31.768	32.815	100.367	252.005
	Memory (MB)	398.6	1,154.3	1,154.4	1,155.1	1,155.6
Bernard	Time (s)	2.531	241.519	242.261	976.892	2996.884
	Memory (MB)	400.1	1,157.7	1,157.6	1,163.0	1,171.5
Cylinder	Time (s)	1.859	78.209	79.522	283.086	742.849
	Memory (MB)	187.7	1,886.5	1,886.5	1,889.1	1,900.2

3.5 Results

We have applied our 3D Morse decomposition framework to a number of analytic and real-world flow data. Figures 9 and 10 show the obtained Morse sets for the *Lorenz system*. This vector field is defined on a $32 \times 32 \times 32$ grid with $\sigma = 10$, $b = 8/3$ and $\rho = 28$ in the domain $[-30; 30] \times [-30; 30] \times [-10; 50]$, which is originally presented in [54]. From the result, we see our method successfully locates the Morse set that encloses a saddle-like periodic orbit.

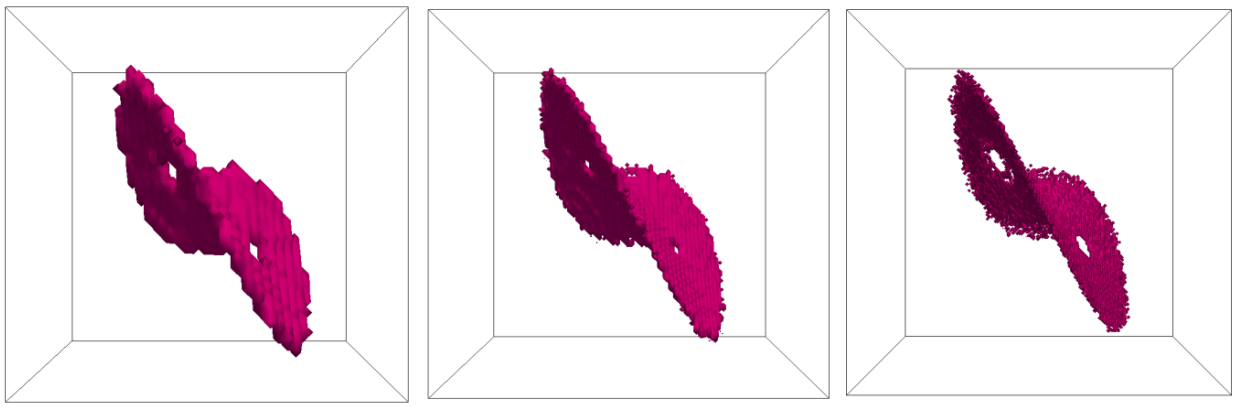


Figure 10: Morse sets (magenta) of the Lorenz system in three different resolutions. In this case, we started with a regular $32 \times 32 \times 32$ cubical grid (left) on the domain $[-30, 30] \times [-30, 30] \times [-10, 50]$, and gradually doubled the resolution (middle and right). $\tau = 400$, 800 and 1600, respectively.

Figures 6b and 11 show the result for the *Bernard-Rayleigh Convection* data [94] using one-level cell subdivision. We have subsampled the data set to $64 \times 16 \times 32$ within the domain $[-16; 16] \times [-4; 4] \times [-8; 8]$. Our result clearly shows the eight Morse sets that match the configuration of the Bernard flow. Figure 12 shows the flow behind a *Square-Cylinder* Data [80]. The dimension of this data is $96 \times 32 \times 24$ in the domain $[-20; 12] \times [-4; 4] \times [0; 6]$. By subtracting the average (or mean) velocity from the flow, we can observe some interesting swirling structure, which is highlighted by the identified Morse set. Figure 8 shows the flow of *Tornado* Data [22]. This vector field is defined on a $32 \times 32 \times 32$ grid and in the domain $[-30; 30] \times [-30; 30] \times [-10; 50]$. Our result identifies one connected Morse set that corresponds to the core of the tornado.

To study the performance of the proposed Morse decomposition, we conduct a number of experiments with the above data sets. There are a number of factors that may affect the performance of our pipeline including its computational time and memory consumption. In the following experiments, we particularly concentrate on four factors: 1) the resolution of the grid, *res* 2) number of integration steps τ, 3) number of levels of adaptive face sampling k and 4) number of levels of refinement L. Tables 1 and 2 provide the performance report of our experiments, respectively. All the performance information is measured on a PC with 2 Intel Xeron 2.6GHz processors, 128GB

Table 2: Performance (in seconds) and memory usage (in MB) of the Morse decomposition of the Lorenz data with different integration steps and different resolution ($k = 2$).

Res	32	64	128
τ	Memory / Time	Memory / Time	Memory / Time
100	1,162.7 / 24.611	3,463.2 / 148.981	23,199.0 / 1199.119
200	1,163.3 / 45.47	3,420.3 / 332.292	23,200.7 / 2,572.517
400	1,164.0 / 84.866	3,430.9 / 674.945	23,201.6 / 5,722.202
800	1,164.8 / 153.059	3,440.1 / 1,236.084	23,823.3 / 10,153.4
1600	1,165.0 / 193.171	3,452.9 / 1775.597	23,204.8 / 15,092.574

RAM, and a Quadro K4000 Graphic Card.

Different integration step τ and resolutions *res* To see how different resolutions affect the performance of our framework, we compute the Morse decompositions of the Lorenz system with three resolutions of 32, 64 and 128, respectively. We set the number of levels of adaptive sampling k to 2 and start the τ from 100 and increase it up to the $1,600$. Figure 9 shows the result of Lorenz with different integration steps. The computation times for $\tau = 200$, 400, 800 and 1600 are 45.47, 84.866, 153.059 and 193.171 seconds, respectively, and the memory foot-prints are at most 1,164.0MB for all of them. Figure 9 and Table 2 provide the results of Lorenz with different integration steps and resolutions.

Different numbers of levels of adaptive sampling k To see how different numbers of levels of the proposed adaptive sampling algorithm affect the performance of our framework, we compute the Morse decomposition of all four data sets with different levels of adaptive sampling. We use the sub-sampled data for our computation which is mentioned at the beginning of this section. As described earlier, due to the hardware constraint, we set the number of levels of adaptive sampling, k, up to 4. From these results, we see that the computation time is a few seconds for all data sets when $k = 0$. When k is increased, the computation times are increased accordingly. However, the increase of the computation time need not be linear with respect to k. This is in large measure due to the different flow configuration of different data. Among different data sets, the computation time is varying. This is mostly because of different resolutions of the data and different parameter settings for the computation. For example, the cylinder data has the highest resolution, therefore, its computation time is longer than the others when using a similar τ. Another example is the Bernard data which has a highly convoluted flow configuration. Therefore, a larger τ (i.e., $1,200$) is needed in order to isolate the eight vortex systems. Consequently, it has the longest computation time in general. In terms of the physical memory consumption, it is easy to understand that the larger the data set (with larger number of cells) the more memory it will consume. For instance, the cylinder data typically demands more memory space than the other data sets. In terms of the level of adaptive sampling, when $k = 0$, the memory use is relatively small, while it becomes much larger when $k > 0$ due to the enabling of the adaptive sampling. However, the memory consumption does not change much if k is increased. This can be explained by the fact that we initialize the memory for the storage of the graph by assuming an average 16 particles for each voxel in our implementation.

Another interesting observation from our experiments is that for some data sets, after k is larger than certain value, the improvement of the resulting Morse decomposition (i.e., the connectivity and smoothness of the Morse sets) is not obvious. For example, for the Tornado data, we found a connected Morse set consisting of $2,701$ voxels using $k = 1$. When we set $k = 2, 3, 4$, the obtained Morse sets contain $2,901$, $3,006$ and $3,091$ voxels, respectively.

Different levels of cell subdivision An important factor that affects the possible levels of cell subdivision is the GPU memory constraint. Since we have no a-prior information for the number of particles, we have to allocate the possible maximum number for the returning array of indexes. It's also dependent on the dimension of the data set, i.e., for the Lorenz data with the resolution of $64 \times 64 \times 64$, we can have adaptive sampling up to 3 levels, and for the Bernard data with the resolution of $128 \times 32 \times 64$, the maximum level of refinement is 2.

Figures 10 and 12 show the examples of the effect of cell subdivision to the decomposition. Generally, the deeper the level of refinement is used, the smoother and more coherent would the results become.

CPU vs. GPU There is a substantial performance gain with the GPU version as shown by the comparison in Table 3. The GPU version is 7.30x to 12.87x faster than the single threaded CPU implementation. The reductions in the run-time are from 843.274 to 115.436 seconds for the Lorenz, 13,295.7 to 1,032.509 seconds for the Bernard, 1,096.402 to 98.761 seconds for the Tornado and 2,193.303 to 881.951 seconds for the Square-Cylinder. Those correspond to 7.8, 12.8, 35.2 and 4.49 times speed-up of the GPU over the CPU, respectively. The smallest speed up is for Cylinder and the largest one is for Bernard. The reason is because of the longer integration step and also the complexity of the Bernard flow, which needs to go through deeper levels of adaptive sampling in order for the extracted Morse sets to converge.

3.6 Summary and Future Work

In this chapter, we have presented a new framework for Morse decomposition of piecewise linear vector field defined on 3D regular grids. Most vector fields of interest in science and engineering are three-dimensional. Our approach allows one to analyze them directly, without being restricted to slices or other two-dimensional domains. Specifically, we present a hierarchical Morse decomposition framework that enables the automatic selection of a proper τ for the flow map estimation at different flow regions in order to generate a Morse decomposition with the desired fineness. We also introduce

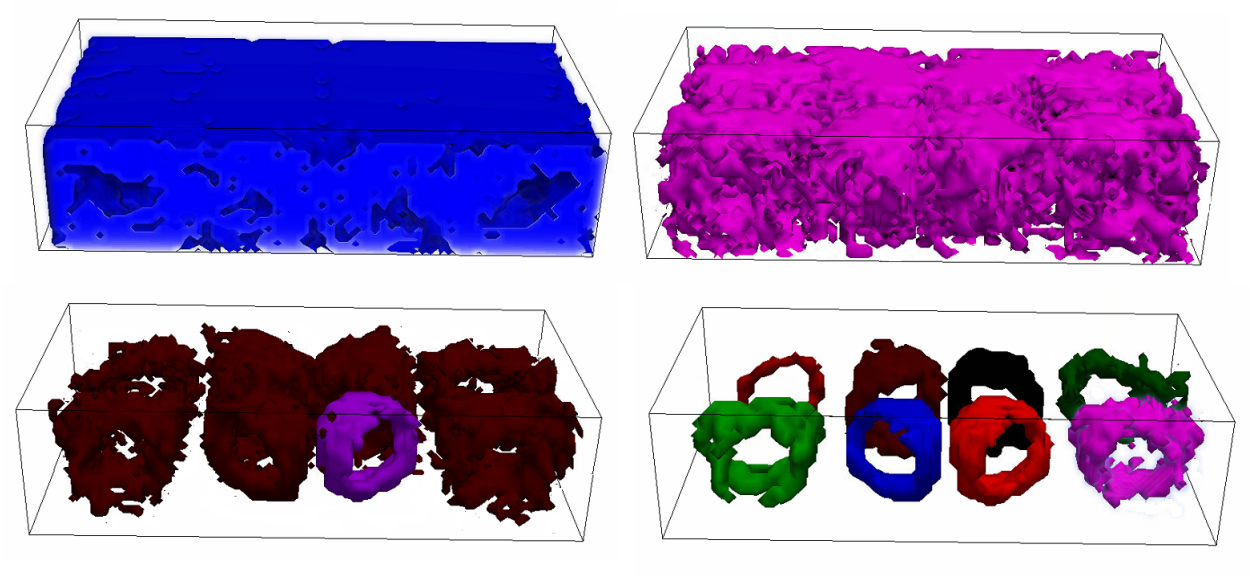


Figure 11: Results for the Bernard-Rayleigh convection data set using the sub-sampled data set. The result on the top left is generated using $\tau = 1200$ and $k = 1$. The top right result is obtained using $\tau = 2400$ and $k = 2$. The bottom left image is generated using $\tau = 4800$ and $k = 3$, which contains a disconnected Morse set. We then apply binary search and determine the ideal τ for those local regions that may contain flow recurrent dynamics. The bottom right image shows the converged result with eight isolated Morse sets (in different colors). The τ used for these Morse sets ranges from 2400 to 3600. The computation times are 143.2, 287.34, 976.53 and 1032.51 seconds, respectively.

Table 3: The comparison of the performance (in terms of the computation time (in seconds) and memory consumption (in MB)) of a GPU implementation versus a CPU implementation of the proposed pipeline with the Lorenz, Tornado, and Bernard data sets. The last column is the speed-up gained from the GPU implementation.

	GPU	CPU	
Dataset	Memory / Time	Memory / Time	**Speed-up**
Lorenz	1,233.8 / 115.436	1,341.9 / 843.274	**7.30x**
Bernard	1,367.8 / 1,032.509	1,481.8 / 13,295.7	**12.87x**
Tornado	1,328.7 / 98.761	1,419.7 / 1,096.402	**11.10x**
Cylinder	9,288.9 / 881.951	9,858.1 / 2,193.303	**4.49x**

an adaptive face sampling strategy for the accurate estimation of the outer approximation of each cell during the flow combinatorialization computation. Finally, we integrate the cell subdivision strategy into our computation pipeline to refine the boundaries of the obtained Morse sets. Our pipeline has been implemented using CUDA and applied to a number of 3D data sets to demonstrate its efficacy.

There are some limitations of the current framework. In particular, for the flow behind a square cylinder data set, our method cannot sufficiently refine the obtained Morse set due to the limited level of adaptive face sampling (i.e., k). In addition, the proposed CUDA implementation may not be optimal, as it does not utilize the advantage of shared memory. Furthermore, the memory allocation for particles is performed at the beginning of the algorithm and remains unchanged,

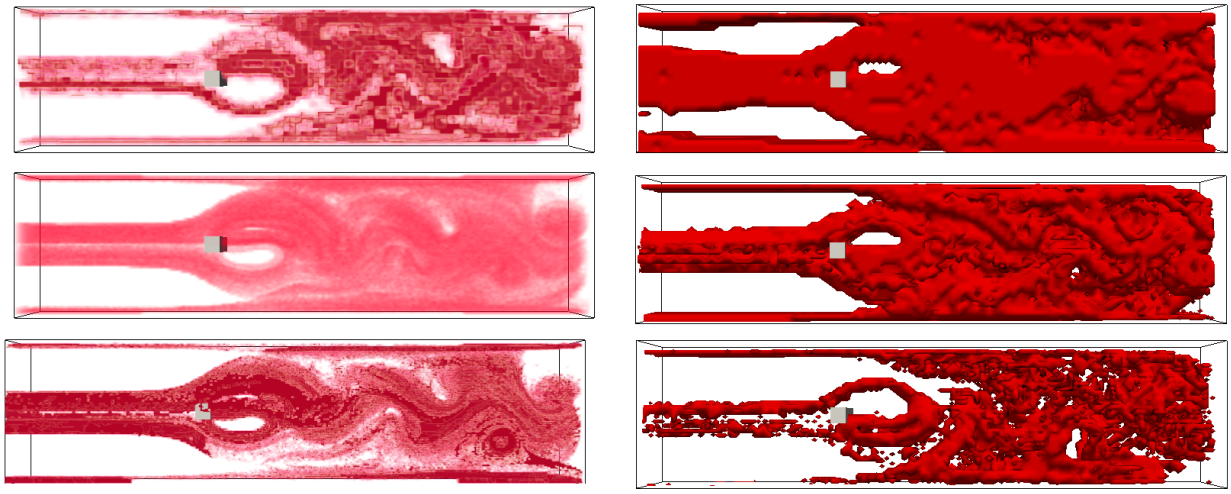


Figure 12: Results of the flow behind a square cylinder data set. The images in the left column show the Morse decompositions of the flow with increasing resolutions (from top to bottom). Specifically, the upper-left image visualizes the Morse set (in red) obtained with resolution $96 \times 32 \times 24$, $\tau = 900$ and $k = 4$; the middle-left image shows the Morse set with resolution $192 \times 64 \times 48$, $\tau = 1800$ and $k = 2$; the bottom-left image is the Morse set with resolution $384 \times 128 \times 96$, $\tau = 3600$ and $k = 1$. The computation times for these three results are 1150.98, 1621.2, and 2456.24 seconds, respectively. The images in the right column show the Morse decomposition results of the cylinder flow at time step 2028 with $\tau = 200$, 400, and 800, respectively. The resolution of the data for these results is $96 \times 32 \times 24$ and $k = 1, 2$ and 4, respectively. The computation times are 161.31, 326.4 and 519.574 seconds, respectively. Note that for all these results, only one Morse set is identified.

which is sub-optimal. In the future, we plan to address all these issues.

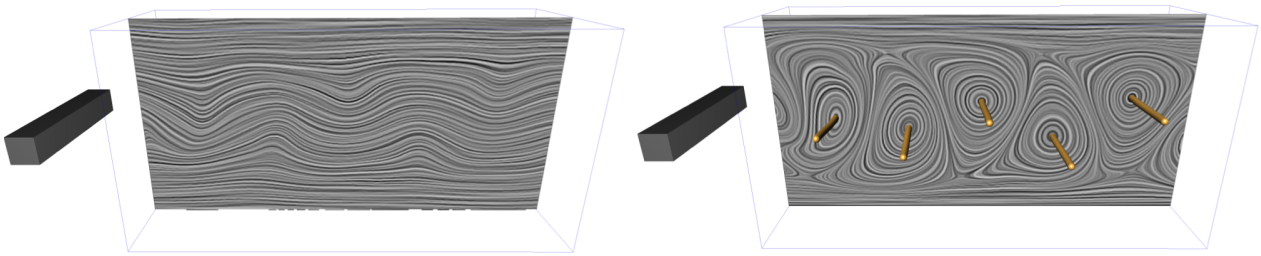
4 Physics-based Analysis of Vortices in Unsteady Flow

The Morse decomposition computation described in Chapter 3 utilizes the geometry characteristics of the flow within a vortex region (i.e., with strong rotational flow) to help locate candidate regions with vortices. However, not all vortices exhibit strong rotational behavior due to the laminar component in the flow motion. See an example shown in Figure 13. This example shows a flow injected from the left boundary colliding with a square cylinder and forming a vortex shedding (i.e., a series of vortices) behind the cylinder. The visualization on the left shows the original flow where the vortex street is hidden due to the strong translational component in the flow. After removing this translational component (i.e., minus the average of the velocity field in this case), the vortices behind the cylinder are better revealed (on the right). That said, the topology-based method may not be able to locate the regions with vortices in this case, as the topology definition and computation is based on the given vector field, and thus, not Galilean invariant (i.e., not invariant under the addition of a constant flow).

On the other hand, vortices are Galilean invariant features. Thus, instead of inspecting the geometry patterns in the flow, fluid mechanic experts usually resort to certain physical properties (or attributes) and thresholding technique to locate vortices (e.g., regions with certain attribute values above a specific threshold are considered within vortices). The relevant physical attributes that experts usually consider for vortex study include vorticity, Q concentration, and λ_2. However, with different attributes, the identified vortex regions may be different, making a deterministic analysis challenging. To address this, we propose to study the co-varying behaviors of pairs of attributes relevant to vortices over time. In this chapter, I will provide a detailed description of this new analysis framework.

4.1 Correlation between Time-varying Attributes

The aforementioned attributes of unsteady flow in Chapter 2 are time-dependent. To study their pairwise attributes and behavior over time, we first briefly describe how we obtain sequences of



(a) Unsteady vector field u(x,t). (b) Subtraction of the right ambient flow.

Figure 13: Overview of the two approaches to vortex tracking: subtracting a suitable ambient flow. Images are from [30].

attribute values in the Eulerian or Lagrangian sense. Given a specific local attribute $\mathbf{A}$, its value at a spatial position $\mathbf{x} \in$ at time $t \in$ can be denoted as $\mathbf{A}(\mathbf{x}, t)$. Computing $\mathbf{A}(\mathbf{x}, t)$ at a location $\mathbf{x}$ over time or along a pathline gives rise to a series of attribute values, which we denote as the attribute value sequence (AVS). The former sequence is obtained in the Eulerian fashion (i.e., measured at fixed location) and the latter is in the Lagrangian fashion (i.e., moving with a particle). With these attribute value sequences, we now describe our metrics to quantify the linear correlation and dependency among attributes over time.

4.1.1 Spatio-temporal Local Correlation Coefficient (LCC)

We extend the LCC [68] for steady scalar field to our unsteady setting, which is the Pearson Product-Moment Correlation Coefficient (PPMCC) [81] extended to the spatio-temporal domain. Specifically, for a local position $\mathbf{p} = (\mathbf{x}, t) \in \Omega$ and a window function $G_{(\mathbf{x},t)}$ positioned at $\mathbf{p}$, the correlation value at $\mathbf{p}$ is defined by:

$$\text{ST\_LCC}_{A_i,A_j}(\mathbf{p}) = \rho_{A_i,A_j}(\mathbf{p}) = \frac{\text{cov}_{A_i,A_j}(\mathbf{p})}{\sigma_{A_i}(\mathbf{p})\sigma_{A_j}(\mathbf{p})} \tag{1}$$

$$\text{cov}_{A_i,A_j}(\mathbf{p}) = \int_{\Omega} G_{(\mathbf{x},t)}(\mathbf{q})(A_i(\mathbf{q}) - \bar{A}_i)(A_j(\mathbf{q}) - \bar{A}_j)d\mathbf{q} \tag{2}$$

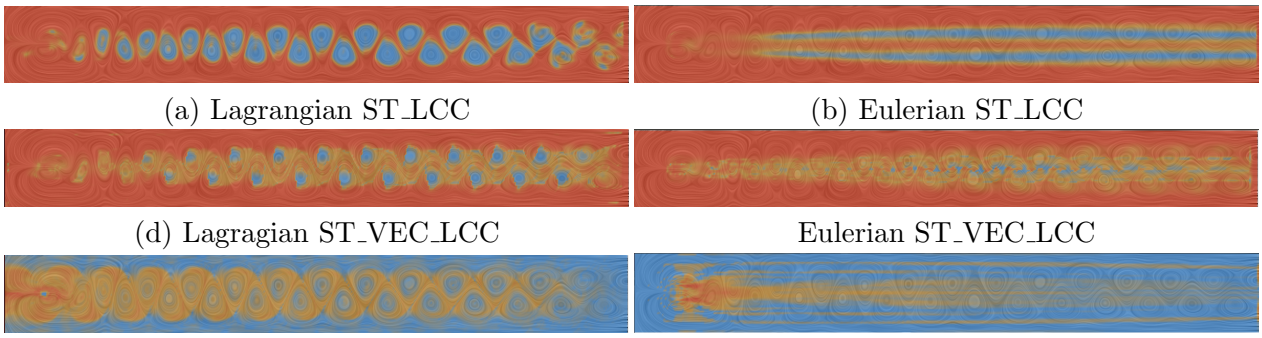
$$\sigma_{A_i}(\mathbf{p}) = \sqrt{\text{cov}_{A_i,A_i}(\mathbf{p})}$$

$$\bar{A}_i = \int_{\Omega} G_{(\mathbf{x},t)}(\mathbf{q})A_i(\mathbf{q})d\mathbf{q}$$

$\text{ST\_LCC}_{A_i,A_j}(\mathbf{p})$ returns a real number between -1 and 1. The closer to 0, the weaker the correlation between A_i and A_j. A_i and A_j are positively correlated if $\text{ST\_LCC}_{A_i,A_j}(\mathbf{p}) > 0$; otherwise, they are negatively correlated. We use a cylinder filter for 2D unsteady flows (i.e., a disk in space and a deformed cylinder or tube in space-time along the pathline) and a hyper-cylinder for 3D data as the window function $G_{(\mathbf{x},t)}$. We set $G_{(\mathbf{x},t)}(\mathbf{q}) = \frac{1}{n}$ if $\mathbf{q}$ falls in the local region (or kernel) defined around $\mathbf{p}$ and n is the number of points within the region, and $G_{(\mathbf{x},t)}(\mathbf{q}) = 0$ otherwise. Other proper window functions can also be used here. The radius (r) and height (h) of the cylinder determine the kernel size in the spatio-temporal domain Ω. Both r and h can be interactively adjusted by the user. Note that h is defined as the total number of time steps in the kernel, which corresponds to a time range $T = h\tau$ with τ being the time interval between two neighboring frames for a dataset. When $h = 0$, ST_LCC is reduced to the conventional LCC.

One characteristic of the above ST_LCC metric for scalar attributes is that it is scale invariant. That is, $\text{ST\_LCC}_{A_i,A_j} = \text{ST\_LCC}_{kA_i,lA_j}$ where $k, l \in R^+$. This may return very large ST_LCC values for cases where both attributes have small values, which can be altered by multiplying the above result with the absolute value of the normalized attribute values. We provide it as an option for the user to control. By default, this multiplication is disabled.

The above ST_LCC can be applied to both a fixed location (Eulerian) and a particle over time (Lagrangian). Figure 14 compares the ST_LCC fields computed in the Eulerian (b) and the Lagrangian (a) views, respectively, for the 2D flow past a cylinder. The comparison shows that the time-varying attribute behaviors in the Eulerian view highlight the overall (or average) movement of the vortices in the vortex street region behind the cylinder, while its behaviors in the Lagrangian view emphasize the individual vortices. This observation is similar to what is reported in [103] which compares the behavior of the accumulated attributes in the Eulerian and Lagrangian views, respectively.



(a) Lagrangian ST_LCC (b) Eulerian ST_LCC

(d) Lagragian ST_VEC_LCC Eulerian ST_VEC_LCC

(g) Lagrangian ST_MI (h) Eulerian ST_MI

Figure 14: Different correlation measurements between norm (A_7) and shear rate (A_5) of a flow past a square cylinder. Kernel size $r = 3$ and $h = 250$ with $\tau = 0.008s$. For ST_LCC results, blue indicates negatively correlated and red indicates positively correlated. For MI results blue indicates low dependency and red indicates high dependency.

4.2 Vector-based Correlation

In this section, we propose a correlation measurement based on the gradient vector field of the attribute field. Given an attribute field A_i, we define its gradient field as $\mathbf{g}_i = \nabla A_i = (\frac{\partial A_i}{\partial x}, \frac{\partial A_i}{\partial y}, \frac{\partial A_i}{\partial z}, \frac{\partial A_i}{\partial t})$. $\mathbf{g}_i$ points to the direction that the value of A_i increases the fastest, and its magnitude $||\mathbf{g}_i||$ represents how quickly the value changes. Therefore, studying the correlation of the gradient vector field of two attributes may further reveal their similarity in behavior or variation in space and time. First, we define the gradient similarity measure (GC) of two attributes A_i and A_j at position $\mathbf{p} = (\mathbf{x}, t)$ as:

$$
\begin{aligned}
GC(\mathbf{g}_i, \mathbf{g}_j) &= GC_d(\mathbf{g}_i, \mathbf{g}_j) GC_m(\mathbf{g}_i, \mathbf{g}_j) \\
GC_d(\mathbf{g}_i, \mathbf{g}_j) &= \frac{\langle \mathbf{g}_i, \mathbf{g}_j \rangle}{||\mathbf{g}_i||\ ||\mathbf{g}_j||} \\
GC_m(\mathbf{g}_i, \mathbf{g}_j) &= 4 \frac{||\mathbf{g}_i||\ ||\mathbf{g}_j||}{(||\mathbf{g}_i|| + ||\mathbf{g}_j||)^2}
\end{aligned} \tag{3}
$$

$GC_d(\mathbf{g}_i, \mathbf{g}_j)$ represents the direction similarity between the two gradient vectors, and $GC_m(\mathbf{g}_i, \mathbf{g}_j)$ measures their magnitude similarity. We use multiplication to combine the two terms, as they may have different value ranges (i.e., g_i is not normalized). $GC(\mathbf{g}_i, \mathbf{g}_j)$ is 0 when the two gradients are orthogonal. When the two gradient vectors point in the same direction and have the same

magnitude, $GC(\mathbf{g}_i, \mathbf{g}_j)$ returns 1 (i.e., positively correlated). If the two gradient vectors have the same magnitude but point in opposite directions, $GC(\mathbf{g}_i, \mathbf{g}_j)$ returns -1 (i.e., negatively correlated). As opposed to the previous vector similarity metrics that always return positive values [68], our metric can now measure both the positive and negative similarity of two vectors. Compared with the vector similarity metric introduced by Crouzil et al. [17], which relies on the magnitude of the gradient vectors, our metric is able to capture the correlation even when the magnitude of the gradient vectors is small.

Since $GC(\mathbf{g}_i, \mathbf{g}_j)$ is a scalar field in the range of $[-1, 1]$, we can now define a vector-based LCC based on this vector similarity metric. First, the mean gradient field with a given window function $G_{(\mathbf{x},t)}$ is computed as $\bar{\mathbf{g}}_i = \int_{\mathbf{q}\in\Omega} G_{(\mathbf{x},t)}(\mathbf{q})\mathbf{g}_i(\mathbf{q})d\mathbf{q}$. Then the co-variance of the two vector-valued attributes (i.e., the gradient vector fields here) can be defined as:

$$\mathrm{cov}_{\mathbf{g}_i,\mathbf{g}_j}(\mathbf{p}) = GC(\bar{\mathbf{g}}_i, \bar{\mathbf{g}}_j) \int_{\Omega} G_{(\mathbf{x},t)}(\mathbf{q}) GC(\mathbf{g}_i(\mathbf{q}), \bar{\mathbf{g}}_i) GC(\mathbf{g}_j(\mathbf{q}), \bar{\mathbf{g}}_j) d\mathbf{q} \tag{4}$$

Note that multiplying $GC(\bar{\mathbf{g}}_i, \bar{\mathbf{g}}_j)$ is necessary in order to preserve the sign of the correlation. Thus, the vector LCC is defined as:

$$vecLCC_{\mathbf{g}_i,\mathbf{g}_j}(\mathbf{p}) = \frac{\mathrm{cov}_{\mathbf{g}_i,\mathbf{g}_j}(\mathbf{p})}{\sigma_{\mathbf{g}_i}(\mathbf{p})\sigma_{\mathbf{g}_j}(\mathbf{p})}, \quad \text{where } \sigma_{\mathbf{g}_i}(\mathbf{p}) = \sqrt{\mathrm{cov}_{\mathbf{g}_i,\mathbf{g}_i}(\mathbf{p})} \tag{5}$$

Compared to the above scalar LCC, our proposed vector based VEC_LCC is scale-sensitive. Figures 14c and 14d show the results for vector-based ST_LCC in Lagrangian and Eulerian domains. The results show that in the core regions of the vortices, a minimal variation of shear rate is exhibited while the norm does not need that variation, resulting in negative correlation. In the meantime, the boundary layers of the vortices exhibit stronger variation of shear rate, which is aligned with the behavior of the norm. Therefore, we observe positive correlation values there. This matches previous results [72].

4.3 Measuring Attribute Dependency Using MI

The above correlation computation is limited to the linear co-varying behavior between two attributes. Another metric is needed to capture non-linear relations between attributes. Among many non-linear relations, the dependency between attributes is of interest to fluid experts, which will facilitate understanding the causal relations between attributes. To study the dependency between attributes, we adapt the mutual information metric. In this section, first we review the definition of mutual information, then introduce the spatio-temporal mutual information metrics.

4.3.1 Spatio-temporal Mutual Information

In this section, we start with a brief introduction of mutual information based on a book by Chen et al. [15]. Consider X to be a discrete random variable with alphabet $\mathbb{X}$ and probability distribution $p(x)$, where $p(x) = Pr[X = x]$ and $x \in \mathbb{X}$. The *entropy* $H(x)$ of a discrete random variable X is defined by:

$$H(X) = -\sum_{x \in \mathbb{X}} p(x) \log_2 p(x)$$

for a pair of discrete random variables X and Y with a joint probability distribution $p(X, Y) = \{p(x, y)\}$, the *joint entropy* $H(X, Y)$ is defined by:

$$H(X, Y) = -\sum_{y \in \mathbb{Y}} \sum_{x \in \mathbb{X}} p(x, y) \log_2 p(x, y)$$

where $p(x, y) = Pr[X = x, Y = y]$ is the joint probability of x and y. The *conditional entropy* $H(Y|X)$ of a random variable Y given a random variable X is defined by:

$$\begin{aligned} H(Y|X) &= \sum_{x \in \mathbb{X}} p(x) H(Y|X = x) = \sum_{x \in \mathbb{X}} p(x)(-\sum_{y \in \mathbb{Y}} p(y|x) \log_2 p(y|x)) \\ &= -\sum_{x \in \mathbb{X}} \sum_{y \in \mathbb{Y}} p(x, y) \log_2 p(y|x) \end{aligned}$$

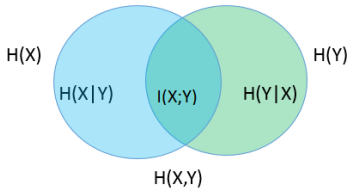


Figure 15: The relationship between two associated information measurements for MI computation.

where $p(y|x) = Pr[Y = y|X = x]$ is the conditional probability of y given x. The entropy measures the average amount of information or uncertainty in a random variable X. We want to quantify how much uncertainty the realization of a random variable X has if the outcome of another random variable Y is known. Figure 15 illustrates the additive and subtracting relationships for different measurements associated with two correlated variables X and Y. The area contained by both circles is the joint entropy $H(X, Y)$ and the circle on the left (blue and cyan) is the individual entropy $H(X)$, with the blue being the conditional entropy $H(X|Y)$. The circle on the right (green and cyan) is $H(Y)$, with the green being $H(Y|X)$. The cyan overlap represents the mutual information $I(X;Y)$ which measures the common uncertainty between two random variables X and Y and is defined as:

$$\begin{aligned} I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) \\ -H(X,Y) = H(X,Y) - H(X|Y) - H(Y|X) \end{aligned} \tag{6}$$

where $H(X)$ and $H(Y)$ are the marginal entropies, $H(X|Y)$ and $H(Y|X)$ are the conditional entropies, and $H(X, Y)$ is the joint entropy of X and Y. In our case, X and Y are two attributes and $I_{\mathbf{p}}(X;Y)$ (i.e., ST_MI) is computed within a spatio-temporal kernel located at $\mathbf{p}$.

Using the MI metric to study attribute dependency Figures 14g and 14h show the ST_MI fields computed based on the Lagrangian and the Eulerian attribute sequences of the Frobenius norm of the Jacobian and local shearing, respectively. The Lagrangian ST_MI field (Figures 14g) exhibits a similar vortex street pattern to its ST_LCC counterpart (Figure 14a), while the ST_MI in the Eulerian view reveals the average movement of the vortices. However, compared to the result

of ST_LCC, ST_MI has large values at the vortex regions behind the cylinder. This is because the MI values are always non-negative, therefore, MI essentially reveals the strength of the dependency of the attributes, ignoring whether the attributes are positively or negatively correlated. Also, there are relations, e.g., the non-linear relations, between attributes that cannot be captured by ST_LCC, but can be captured by ST_MI. See the HCCI results for an example, in which the non-linear dependency of the two attributes within the vortex regions is revealed by ST_MI (Figure 20d, h), but ST_LCC only highlights regions where the two attributes have strong co-varying behavior (Figure 20a, e). Note that the MI and entropy computation is dependent on the number of bins for distribution estimation.

As most other papers, we used equal-width bins for all our results to automatically choose the optimal bin size [78].

4.4 Correlation Visualization

To visualize the scalar fields computed using either the ST_LCC (Section 4.1.1) or the ST_MI (Section 4.3.1), we utilize a blue-white-red (BWR) color coding scheme. For the ST_LCC correlation fields, since the range of the correlation value is [-1, 1], the BWR color scheme is particularly useful to highlight the different characteristics of correlation, i.e., blue for negative values, white for zero and red for positive values. For ST_MI fields, while the values are always non-negative, we still opt for the BWR scheme to highlight places with large correlation values (i.e., in red). Both the 2D color plots and 3D volume renderings shown in this dissertation work are the results of the S_LCC, ST_LCC, ST_VEC_LCC or ST_MI in spatio-temporal domain starting from a given time.

Pathline rendering with attribute relation information To reveal more detailed attribute relation during particle advection, we visualize the seeded pathlines using color coding based on the quantified relation of the pairwise attributes associated with the particles over time. Figures 16a and 16b provide a few examples of pathlines that are color-mapped based on the ST_LCC and ST_VEC_LCC computed along them, respectively. The two attributes used for these experiments

are acceleration and Q.

Specifically, we seed these pathlines at the outer layers of two vortices and near a vortex core, as indicated by the LIC patterns, respectively. Interestingly, the colors along the pathline seeded near the vortex core exhibit little variation, while those pathlines seeded at the boundaries of vortices or near the saddle-like regions exhibit large fluctuation, especially when gradient similarity (i.e, vector-based ST_LCC) (Eq.(3)) is employed for the color coding.

In contrast, at the boundaries of vortices and near the saddle-like regions where the flow exhibits large shearing and stretching behavior, the co-varying behavior of the pairwise attributes may not be similar over time. These correlations can be better understood by inspecting the accompanying plots of attributes. The same scenario occurs with the mutual information based result (Figure 16c). As shown in the accompanying histograms in Figure 16c, we see a wide range and non-uniform (or fluctuating) distribution of the values for attributes associated with pathlines seeded at the outer layers of the vortices, indicating the high entropy (i.e., high uncertainty) and joint probability of the two attributes, which result in high MI values. In contrast, the distributions of values for attributes associated with pathlines seeded at the vortex center are very concentrated, leading to low entropy (i.e., low uncertainty) and low MI value. Such an interpretation is not easy to obtain without the proposed visualization.

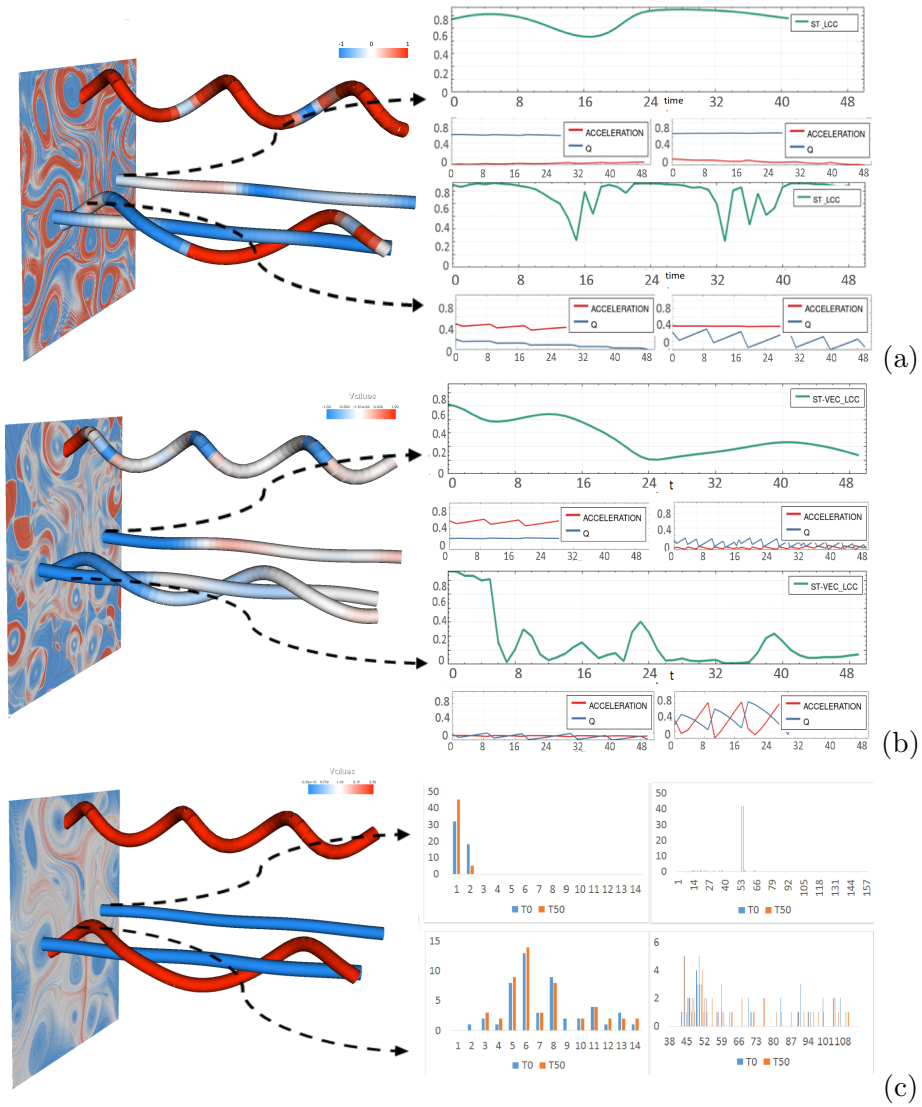
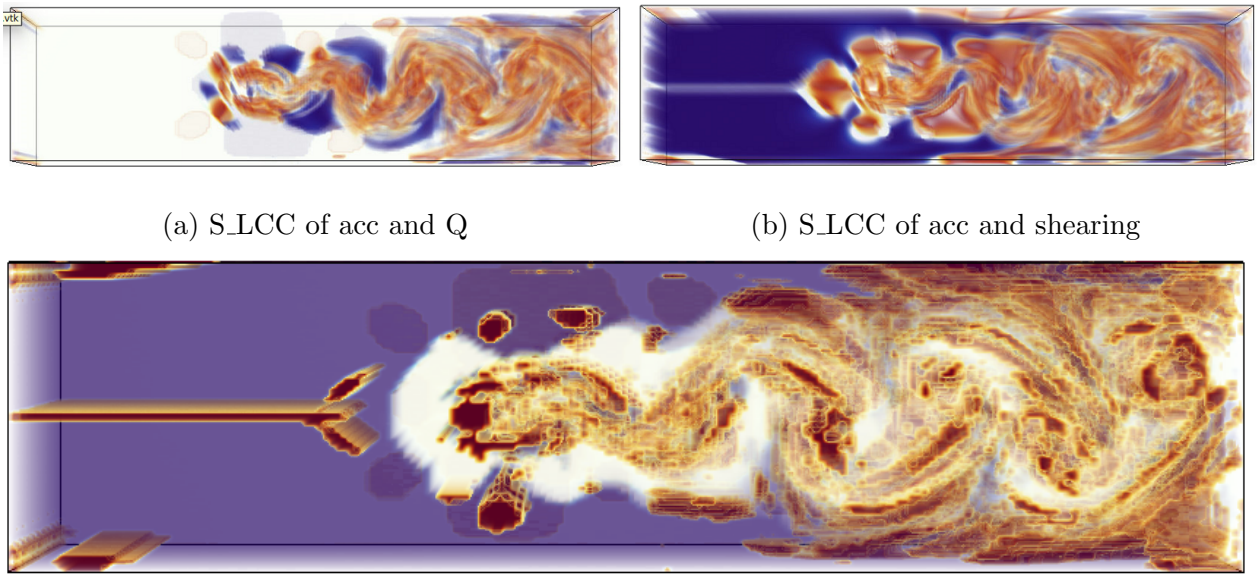


Figure 16: Sampled pathlines colored based on the ST_LCC (a), ST_VEC_LCC (b) and ST_MI (c) of acceleration and Q, respectively. Each top plot in (a) and (b) shows the ST_LCC/ST_VEC_LCC values along the respective pathlines indicated by the arrows. The two sub-plots beneath each large plot show the trends of the two given attributes within a kernel. For the MI result (c), the left two histograms show the distributions of the attribute values of Q within kernels on two pathlines (indicated by the arrows), while the right two show the distributions of the values of acceleration within the same kernels. Each histogram shows two distributions of the values of the respective attribute within two different kernels sampled at t=0 (blue) and t=50 (orange) on the corresponding pathline, respectively. Since we used equal-width bins, the number of bins is different for different attributes.

Ranking strategy One interesting question that experts may ask in practice is which pairs of attributes are more related in certain areas (e.g., having large absolute correlation value or having high dependency), and how this information is related to certain flow characteristics (e.g., vortex cores and outer layers of vortices). Knowing this will in turn help develop an effective way to characterize regions with distinct flow behavior.

To achieve this, for each spatial position, we compute the ST_LCC (or MI) of various attribute-pairs, and sort their ST_LCC values in descending order. We then color this position based on the top-ranked pair of attributes. This gives rise to a domain segmentation, which we hope enables us to identify the most dominant behaviors (characterized by the most correlated attributes) in the individual flow regions. Here, let us take the 3D flow past a cylinder as an example. As we already know the dominant characteristic of this flow is the von Kármán vortex street, we concentrate on the three relevant attributes, i.e., acceleration, Q and local shear rate for the ranking and perform the segmentation based on the ranking. Figure 17a and 17b show the ST_LCC results between acceleration and Q vs. acceleration and shearing, respectively, while the ranking based segmentation is visualized in Figure 17c. Different colors correspond to different top-ranked pairs of attributes. From this result, we see that at the boundary layers of the vortex street, the attribute pair of acceleration and shearing is ranked top (colored by white), while in the vortex core areas the attribute pair of Q and accelerating dominates (colored in red). The rest of the domain (in purple) is dominated by the pair of shearing and Q. This result matches the knowledge in fluid mechanics [97]. This example demonstrates that our simple ranking-based segmentation can provide an overview of the dominant flow dynamics (e.g., rotational versus stretching or shearing) in different flow regions for an effective unsteady flow exploration.



(a) S_LCC of acc and Q

(b) S_LCC of acc and shearing

(c) segmentation

Figure 17: Spatial LCC of acceleration and Q (a) vs acceleration and shearing (b) and the segmentation result for three pairs of these attributes (c). In (c), red regions correspond to places where the correlation between acceleration and Q is the strongest. White regions visualize the places where acceleration and shearing has the highest value of correlation, and purple segments correspond to the regions where correlation between shearing and Q is the strongest.

4.5 Results

We have applied our correlation analysis and visualization to a number of 2D and 3D analytic and real-world simulated flow. In the following, we provide some detailed discussion on some of the results.

Double gyre Figure 18 shows the ST_LCC correlation (6a and 6c) and ST_MI (6b and 6d) of the vorticity (A_1) vs. Q ((A_4)) (top row) and the vorticity vs. shearing (A_5) (bottom row), respectively, of the double gyre flow [74] starting from $t = 0$ and with time window $T = 100 \times 0.01$. Lagrangian attribute value sequences are used. The spatial sample resolution is 256×128. Figures 18a and 18c convey similar structure, but the colors are reversed. This is because in regions where flow exhibits strong vortical (or rotational) behavior (i.e., with large Q values), its shear characteristic is low (i.e., with low shear values). Similarly, when the shearing motion dominates

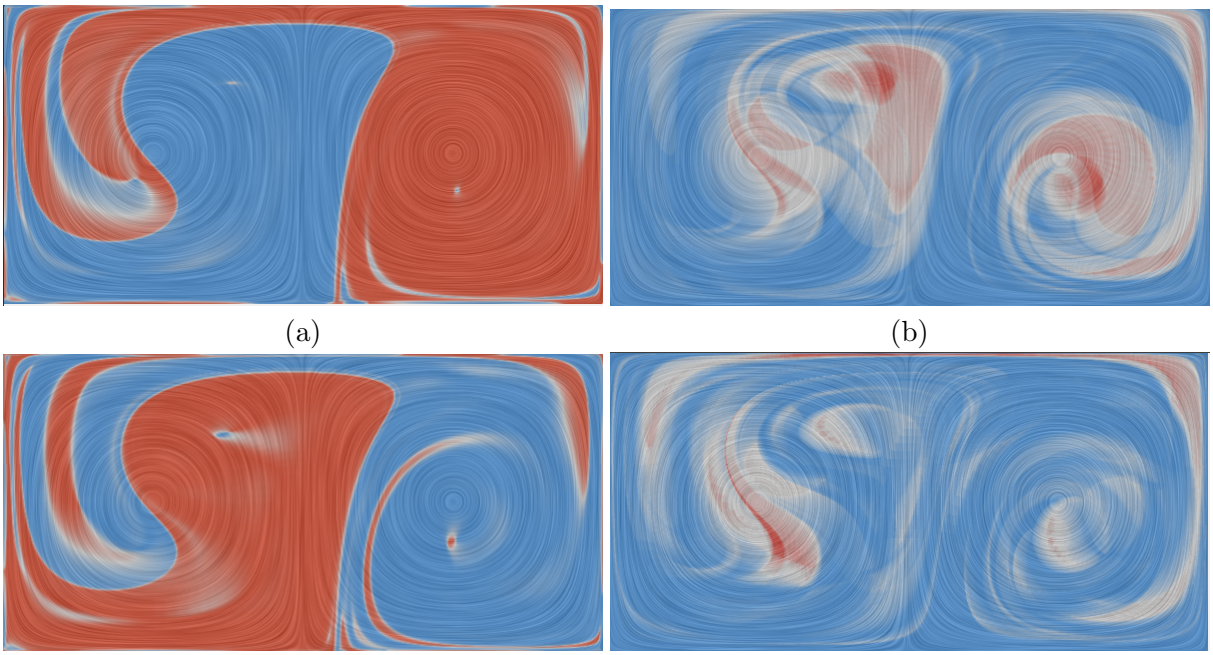


Figure 18: ST_LCC results of curl and Q (a) and curl and shearing (c); ST_MI results of curl and Q (b) and curl and shearing (d). All results are computed based on the Lagrangian attributes using a spatial kernel r = 3 and temporal kernel h = 100 with $\tau = 0.01$

.

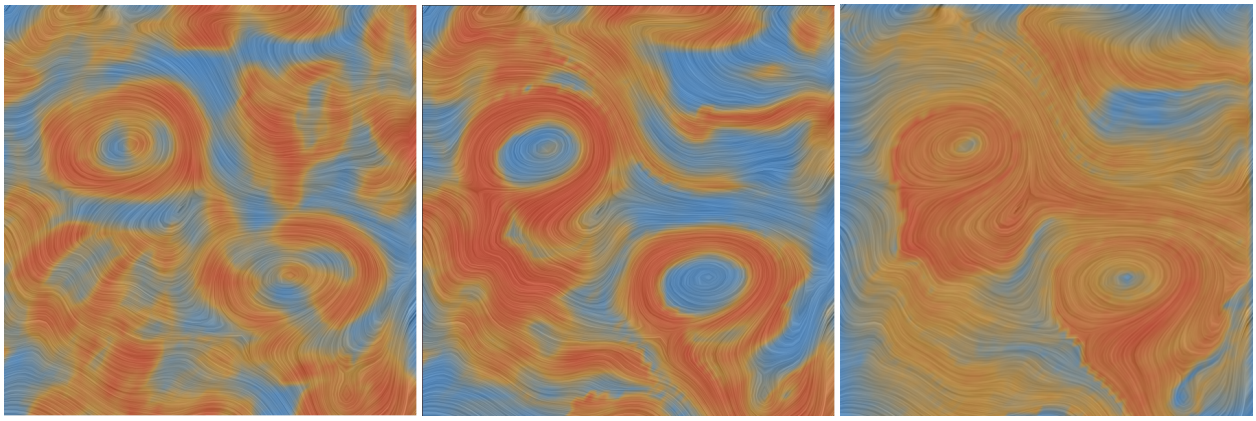
the flow behavior, its rotational behavior is weak. This can also be derived from the definition of $Q = \frac{1}{2}(\|\mathbf{R}\|^2 - \|\mathbf{S}\|^2)$[30], where $\|\mathbf{S}\|^2$ denotes the square of the local shear rate. In this definition, Q and shearing have relatively negative (but non-linear) correlation. From the ST_MI results shown in Figures 18b and 18d, we find that both of them exhibit a similar pattern; however, the result between vorticity and Q highlights the LCS structure (not shown here) that is stronger due to the high vorticity in that region. Also, the interleaving layout of the negative (blue) and positive (red) regions are aligned with the LCS structure, indicating the partitioning of regions with different dominant rotational behaviors.

2D flow past a cylinder This simulation covers a subset of the spatio-temporal domain, i.e., $[-0.5, 7.5] \times [-0.5, 0.5] \times [15, 23]$. The time range is chosen such that the von Kármán vortex street is fully developed behind the cylinder. The resolution of the data set is $400 \times 50 \times 1001$.

Total time is 8 seconds for all 1001 frames. We measured the time from when a vortex is created close to the cylinder until it leaves the boundary. The lifespan of a vortex is 925 frames. We use the same spatial resolution, i.e., 400×50 to compute the attribute fields and the subsequent correlation fields. Figure 14 provides the correlation results for the local shear rate (A_5) and norm of Jacobian (A_7) . Based on their definitions, in 2D vector fields, $A_7 = \sqrt{J_{11}^2 + J_{12}^2 + J_{21}^2 + J_{22}^2}$, and $A_5 = \sqrt{J_{11}^2 + J_{22}^2 + \frac{(J_{12}+J_{21})^2}{2}}$. The difference of the squares of these two norms is $\sigma^2 = \frac{(J_{12}-J_{21})^2}{2}$. That said, in regions that possess strong rotational characteristics (i.e., $|(J_{12} - J_{21})|$ is large), the two norms have larger difference, while in regions that are close to curl-free (i.e., $|(J_{12} - J_{21})| \approx 0$), the two norms are almost identical. Using this observation, we find that the correlation results shown in Figure 14 match our expectation. That is, the correlation of these two attributes in the vortex regions should be small, while in other places it should be relatively large. One interesting point is that due to the dissipation of energy, attributes are going to be depreciated at the end (far right) of the flow. This results in lower correlation values close to the outlet on the right of the cylinder flow.

Figures 14a, 14c and 14g show the ST_LCC, ST_VEC_LCC, and ST_MI results in the Lagrangian view, which highlight the structure of vortex street. In contrast, the results shown in Figures 14b, 14d and 14h, using a spatio-temporal kernel in the Eulerian view, reveals that the individual vortex structure is merged into a corridor, which corresponds to the area that the vortex street sweeps through. Comparing the results of ST_MI and ST_LCC, we see that ST_MI reveals the boundaries of the vortex structure clearer than ST_LCC. This is due to the fluctuating pattern of the attribute values in the boundary of the vortex structure which indicates a higher dependency between these attributes in the boundary. Looking closer at the ST_LCC field computed in the Lagrangian view, we see that the two attributes involved, i.e., acceleration and Q, exhibit a positive correlation along the boundaries of the individual vortices, while having negative correlation near the center of the vortices. This also matches the observation in previous literature [42]. That is, the acceleration magnitude at the centers of vortices are minimal, while the Q values there are large.

Ocean data The next data set is taken from the top layer of a 3D simulation of global oceanic eddies for 350 days of the year 2002 [79]. Each time step corresponds to one day. The 2D vector field has a spatial resolution of 3600×2400. We extract tiles from the central Atlantic Ocean (600×600). Figure 19 shows the results of the acceleration and the determinant of Jacobian matrix (i.e., Q in 2D [30]). As we see, these two attributes are positively correlated in the boundary of the vortices, while negatively correlated at the vortex center. Figure 19c shows the Lagrangian MI computed for these two attributes. Compared to ST_LCC the entire vortex regions are pronounced except at the vortex cores due to the high dependency of these two attributes at vortex regions whether they are positively or negatively correlated.



(a) (b) (c)

Figure 19: Different correlation measurements of a region of the ocean simulation that contains eddies. (a) Spatial LCC, (b) Spatio-temporal (Lagrangian) LCC, and (c) mutual information (MI) (Lagrangian) between the *Acceleration* and *Determinant* of Jacobian matrix, respectively. For the first two results, the color shows the correlation characteristics, i.e., blue indicates negatively correlated and red positively correlated. For the MI result (c) blue indicates lower dependency and red indicates higher dependency.

HCCI data This data simulates a homogeneous charge compression ignition (HCCI) engine combustion [13], which has a spatial resolution of 640×640 and 299 time steps. Figure 20 shows a number of correlation fields for acceleration (A_2) and λ_2 (A_3). Similar to the results of the cylinder data, the ST_LCC computed using the Lagrangian attribute sequences highlights a more complete structure of vortices, while the Eulerian ST_LCC emphasizes the areas the vortex cores

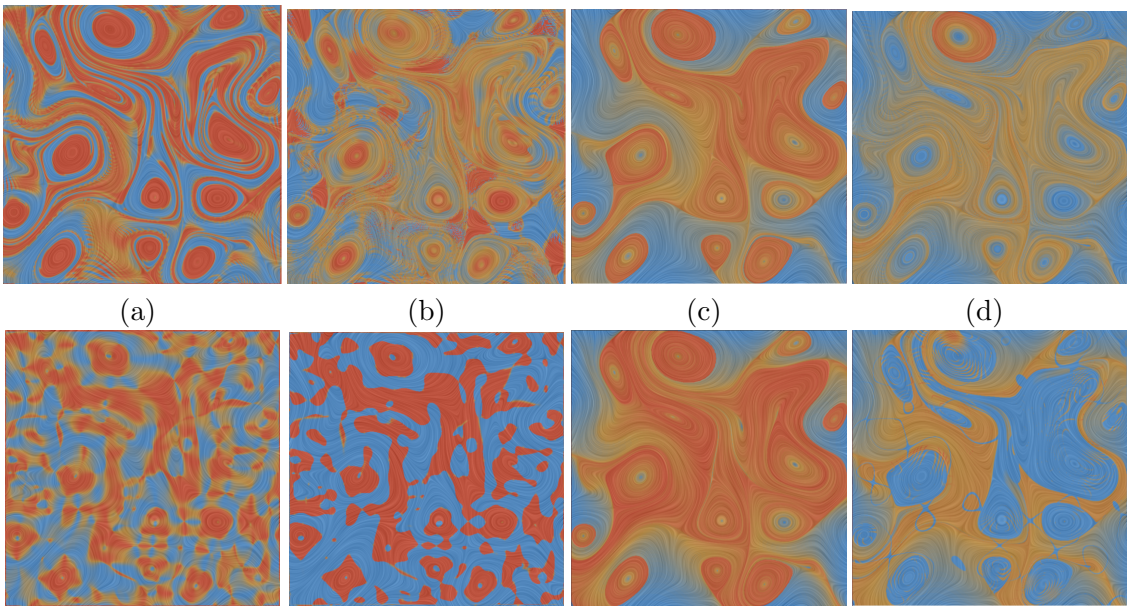


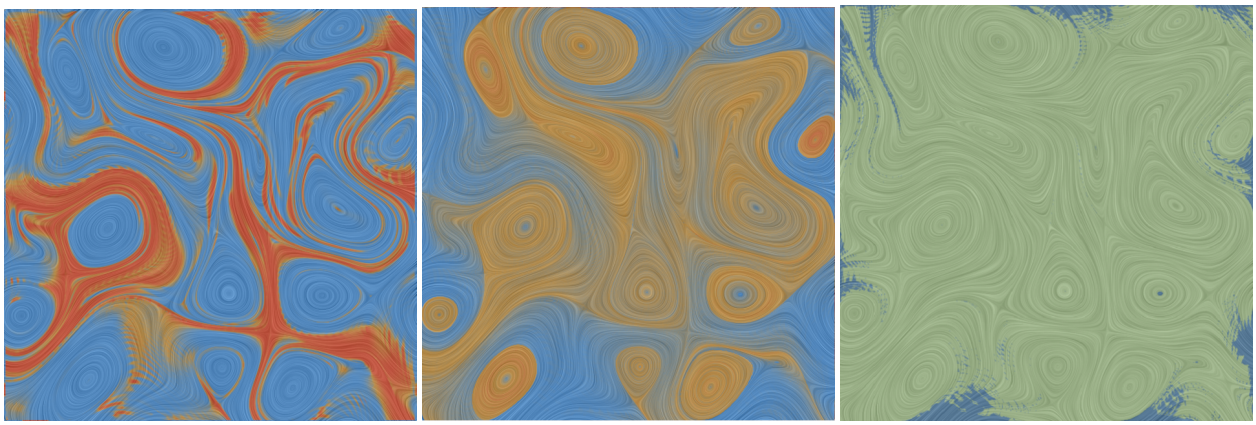
Figure 20: Different correlation measurements between λ_2 and acceleration of the HCCI data set. Kernel size $r = 3$ and $h = 50$ (with $\tau = 0.05$)).

sweep through over time. Also, we find that acceleration and λ_2 are positively correlated at vortex regions which is expected.

Figures 20b and 20f show the ST_VEC_LCC results using the Lagrangian and Eulerian attribute sequences for acceleration and λ_2, respectively. The Lagrangian ST_VEC_LCC highlights the vortex centers and places with strong stretching flow (i.e., places with saddle-like patterns), indicating the gradients of the two attributes have similar time-varying behavior. Similarly to its ST_LCC counter part, the Eulerian ST_VEC_LCC does not show much meaningful structure. Figures 20c and 20g show the Lagrangian and Eulerian joint entropy of these two attributes, respectively. The joint entropy highlights regions where both attributes have strong relation, either dependency or independency. In this comparison, Lagrangian joint entropy is more focused on vortex regions than the Eulerian result, as the Eulerian result tends to highlight regions where those vortices sweep through, similar to the Eulerian ST_LCC we have seen earlier (Figure 14b).

Figures 21a and 21b show the conditional entropy of two attributes λ_2 (A_3) and the norm of the Jacobian (A_7), for the HCCI data. The conditional entropy $H(A_3|A_7)$ measures the amount

of dependency of A_3 on A_7, and usually, $H(A_7|A_3) \neq H(A_3|A_7)$. We compare the two conditional entropies for the two attributes and visualize it in Figure 21c. We see that in most regions, $Norm$ is dependent on λ_2, while λ_2 is less dependent on $Norm$, since $H(A_7|A_3) < H(A_3|A_7)$. This is expected, as regions with strong vortical flow result in a larger norm of the Jacobian, while the inverse is not always true.



(a) (b) (c)

Figure 21: The comparison of conditional entropy for norm of the Jacobian (A_7) vs. λ_2 (A_3). Blue-white-red color coding is applied to (a) and (b), while for (c), light green indicates $H(A_7|A_3) < H(A_3|A_7)$ and blue means $H(A_3|A_7) < H(A_7|A_3)$.

Ocean circulation This dataset is a model of oceanic circulation, provided by *the Estimating the Circulation and Climate of the Ocean, Phase II (ECCO2): High-Resolution Global-Ocean and Sea-Ice Data Synthesis project* [56]. This data provides us the opportunity to examine the evolution of ocean eddies and circulation through time. The data has a spatial resolution of 1440×720 with 50 time steps (equals 150 days). Figure 22 shows the Lagrangian ST_LCC and ST_MI for acceleration and Q. The spatial kernel window size is 5 and the temporal kernel size is 50 ($\tau = 0.004$). The outline of the African Continent is visible at the far left of that slice, with South America visible at the far right. The result shows that around the equator, there exists a series of strong vortices (i.e., eddies).

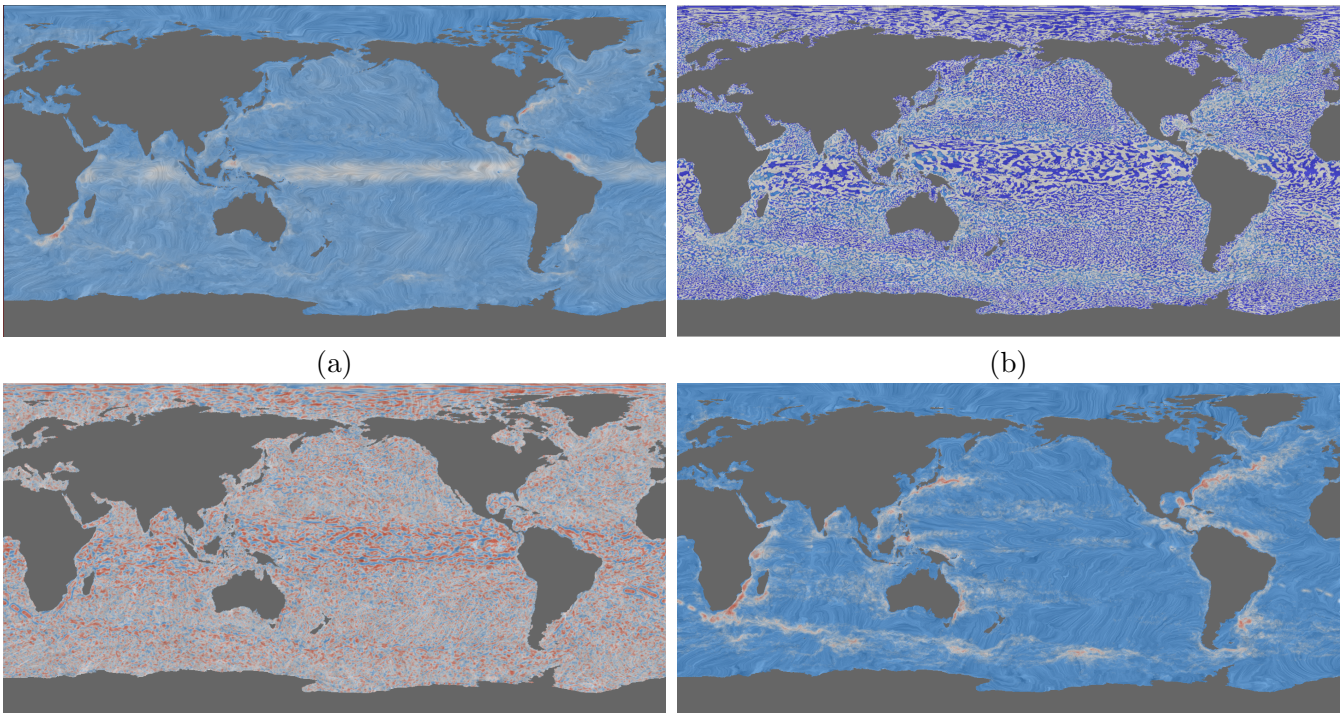


Figure 22: The ST_LCC linear correlation (c) and ST_MI dependency (d) of the acceleration (a) and Q (b) of the large ocean simulation data with a spatio-temporal kernel size 5×50. The blue-white-red color scheme is used to highlight the different characteristics of attribute correlation and dependency. The ST_LCC result (c) highlights a string of strong vortices (or eddies) around the equator, while the MI result (d) emphasizes the strong offshore currents on the east coast of the individual continents.

Axisymmetric vortex ring impact This flow is an axisymmetric simulation of a vortex ring hitting a no-slip wall. During the interaction process, the vortex ring approaches the wall and causes a boundary layer to appear. As the vortex slides against the wall, the boundary layer is lifted up as a secondary vortex, which in turn lifts up the primary vortex. This dataset helps us analyze the role of coherent structures and their interaction with walls, as well as the generation of turbulence in wall-bounded flows [58]. Figure 23 shows the segmentation results for pairs of attributes involving pressure (i.e., the pressure versus the other attributes) over time. From the result, we see that at the vortex center region, the correlation between pressure and vorticity is the highest (i.e., the dark green areas), while there is a layer around the vortex center (i.e., the pink region) in which the correlation between pressure and kinetic energy is the strongest. Outside the

vortex area (i.e., the blue regions), the correlation between pressure and shearing dominates.

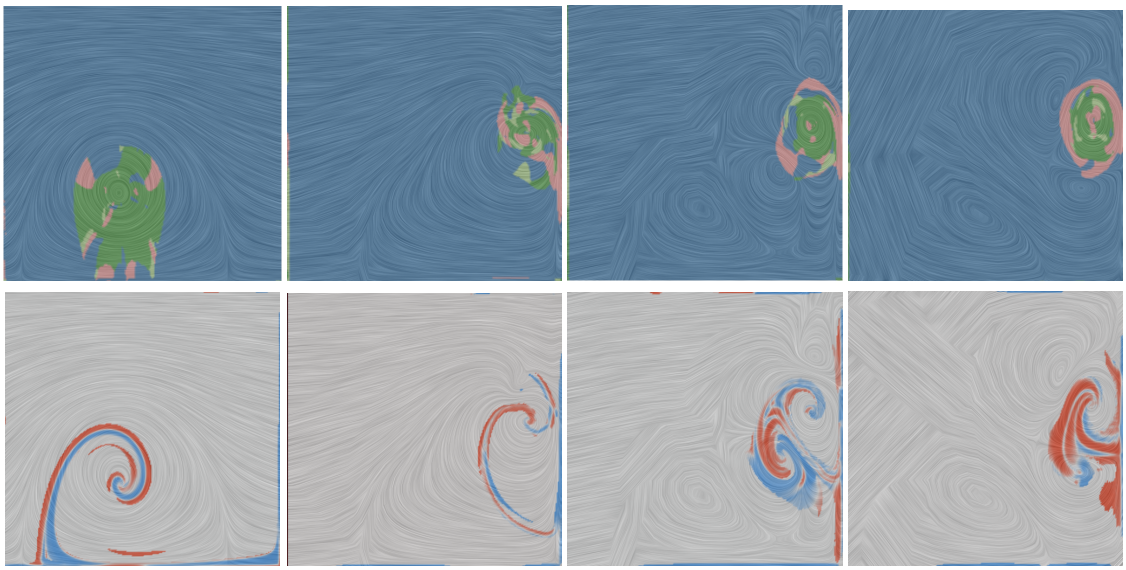
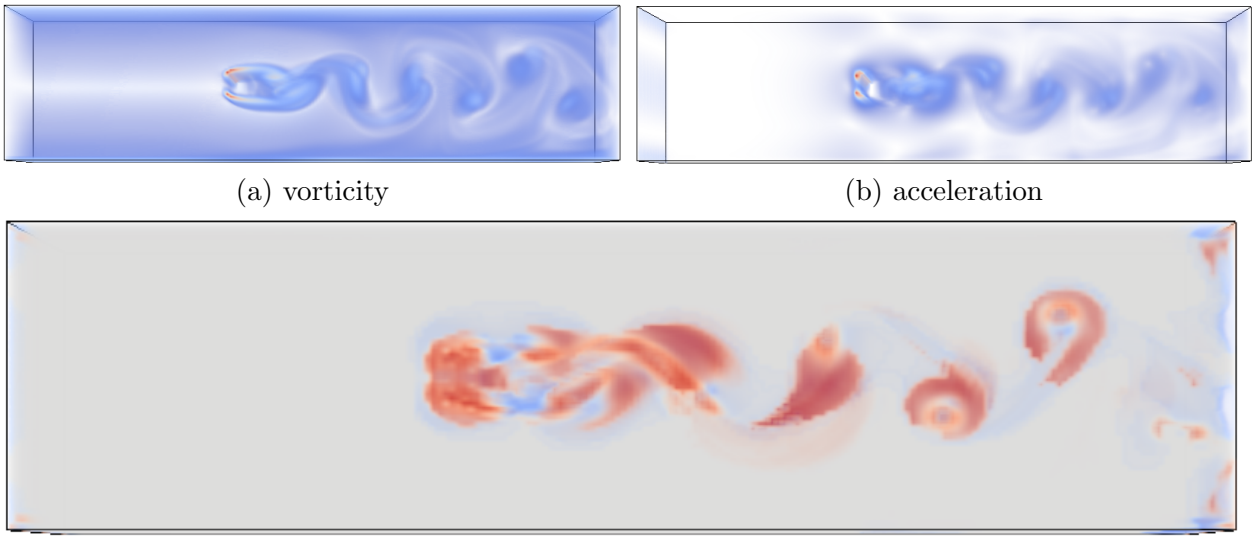


Figure 23: The top row shows the ranking based segmentation for the spatial correlation between pressure and other attributes over time (from left to right). The green color indicates the regions where the correlation of pressure and vorticity is the highest, the pink color corresponds to the pair of pressure and kinetic energy, and blue for the pair of pressure and shearing. The second row shows the result of ST_LCC for acceleration and kinetic energy. The results highlight regions similar to the FTLE ridges. The spatial kernel size is $r = 3$ and the temporal kernel size is $h = 40$.

3D flow behind a square cylinder This flow was simulated by placing a square cylinder in a fluid flow. By subtracting the average velocity from the flow, we see interesting swirling structures [7]. The dimensions are $192 \times 64 \times 48 \times 101$ in the volume of $[-12, 20] \times [-4, 4] \times [0, 6]$. Figures 24a and 24b show the vorticity and acceleration of the flow, respectively, and Figure 24c shows the S_LCC of these two attributes. From this result, we see that acceleration and vorticity are positively correlated in the outer layer of vortices, which matches our previous observation in the 2D flow past a cylinder. Figure 25 compares the results of Eulerian and Lagrangian ST_MI between acceleration and Q with different kernel sizes.

In Section 4.1 we showed that for 2D datasets, larger temporal kernel sizes reveal the flow structure clearer. However, this need not be the case for some 3D data. This is because for most 2D data we have experimented with (e.g., Double Gyre, 2D cylinder flow, HCCI etc.), the



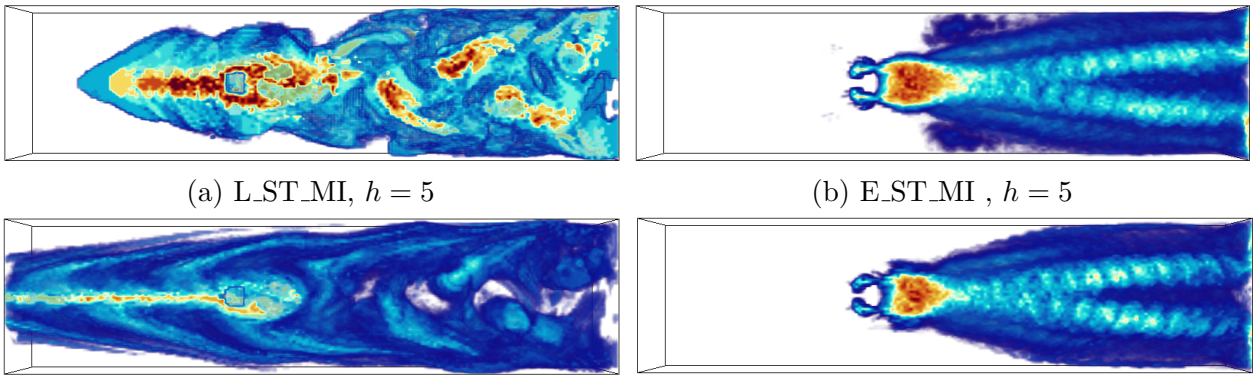
(a) vorticity (b) acceleration

(c) spatial LCC

Figure 24: Spatial LCC of *acceleration* and vorticity for the 3D flow behind cylinder at time step 40. Note that the LCC result highlights the outer layer of the 3D vortices of the flow as expected.

vortex structures are relatively stable over time, while in some 3D flows the vortices undergo large changes. For instance, the 3D cylinder flow contains the state before the vortex street is formed; thus, the pathlines started in early time may exhibit large change in shape during the transition from the laminar flow to turbulence flow. In that case, our proposed metrics work well for a small temporal kernel size in order to be able to capture the behavior of pathlines in a specific region. Larger temporal kernels will average (or smooth) the values of correlation or MI in different regions. Hence, we see Figure 25c does not reveal the vortices better than Figure 25a using ST_MI. In contrast, the ST_MI result in the Eulerian view is not affected much by the kernel size, since particle movement is not considered there.

3D vortex tube data This dataset simulates two separate cases of parallel, counter-rotating vortex tubes at a circulation-based flow with Re=3500 and a distance of 2.5 radii apart, which undergo either an elliptical instability [70] that ends with a vortex disintegration, or a Crow instability which ends with a vortex reconnection [38] depending on the initial conditions. The two vortices interact mainly through strain. The datasets have dimensions of $360 \times 360 \times 360 \times 120$. A



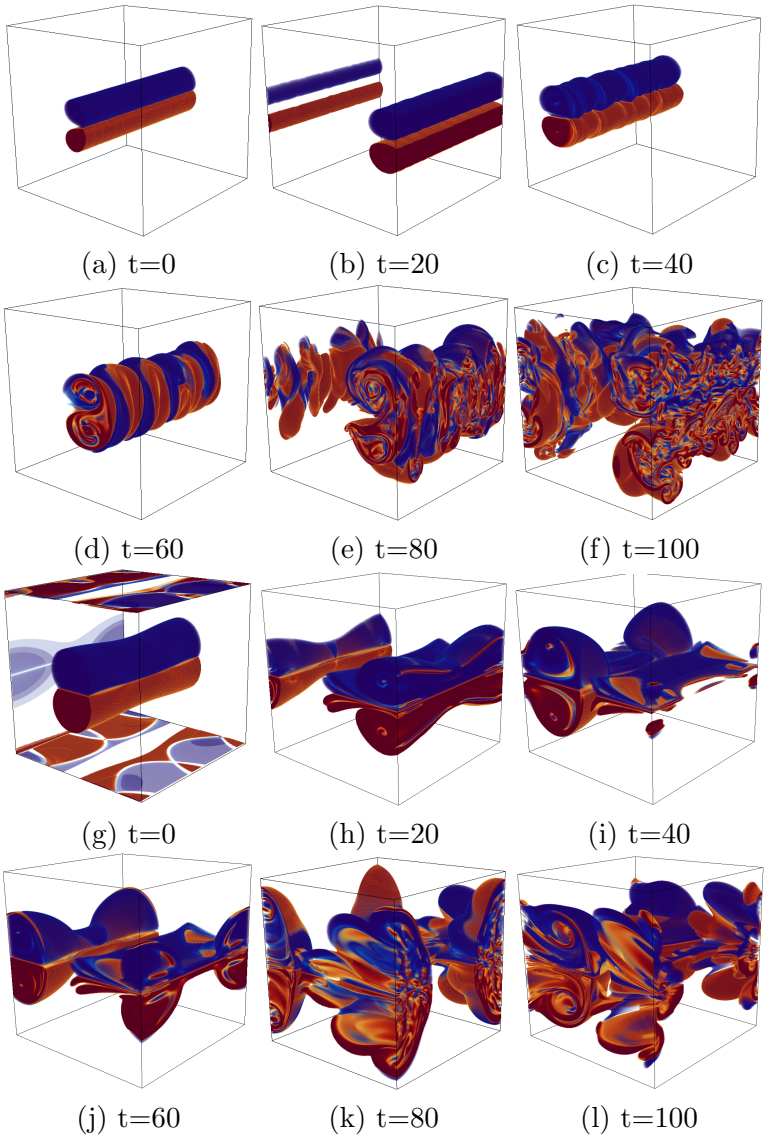
(a) L_ST_MI, $h = 5$ (b) E_ST_MI , $h = 5$

(c) L_ST_MI, $h = 20$ (d) E_ST_MI , $h = 20$

Figure 25: Comparison of Lagrangian (L_) ST_MI and Eulerian (E_) ST_MI for the 3D cylinder data with different time windows h. Left column shows L_ST_MI for acceleration and Q, and the right shows E_ST_MI. We see that larger h makes both L_ST_MI and E_ST_MI smoother, but it does not change the structure of E_ST_MI much.

comparison between the two instabilities is made by adding a tracer, or dye, simulated as a passive scalar with a Schmidt number of unity. This scalar will track the fluid where it was originally injected. For the reconnection simulation, there are two counter-rotating vortex tubes, which after the reconnection changes topology, half of each tube reconnects with the half of the other tube, to form two tubes. For this case, the dye follows the vorticity from each tube, while for the elliptical case, the instability generates perpendicular filaments of vorticity that have a lower dye intensity, so the correlation between dye and vorticity will be weaker. Figure 26 compares the spatial LCC of vorticity and dye over time for the two simulations. From this comparison, we see that in both flows, the two attributes have high correlation at vortex regions, indicating similar co-variance behavior of the two attributes over time despite the flow in the elliptical instability decorrelating dye and vorticity more than the other one. This discrepancy from the expectation is interesting and asks for an in-depth investigation.

Conclusive remarks: Based on the above experiments, we can summarize that for finding the center region of vortices, computing ST_LCC and/or ST_VEC_LCC of acceleration with another relevant attribute (e.g., Q or λ_2) in a Lagrangian view, works well (e.g., Figures 20a-b and 17c). Computing ST_LCC of the norm of the Jacobian and local shearing or the ST_MI of various



(a) t=0 (b) t=20 (c) t=40

(d) t=60 (e) t=80 (f) t=100

(g) t=0 (h) t=20 (i) t=40

(j) t=60 (k) t=80 (l) t=100

Figure 26: The spatial correlation of dye and vorticity for the elliptical instability simulation (a-f) and the Crow / reconnection simulation (g-l). In the top, we can see how the dye first tracks vorticity, until the non-linear elliptical instability kicks in and destroys the vortex, and the correlation is lost. In the bottom, we can see how the dye follows the vorticity much better even after the change in topology, and there is less mixture between red and blue regions. The spatial kernel size is 3.

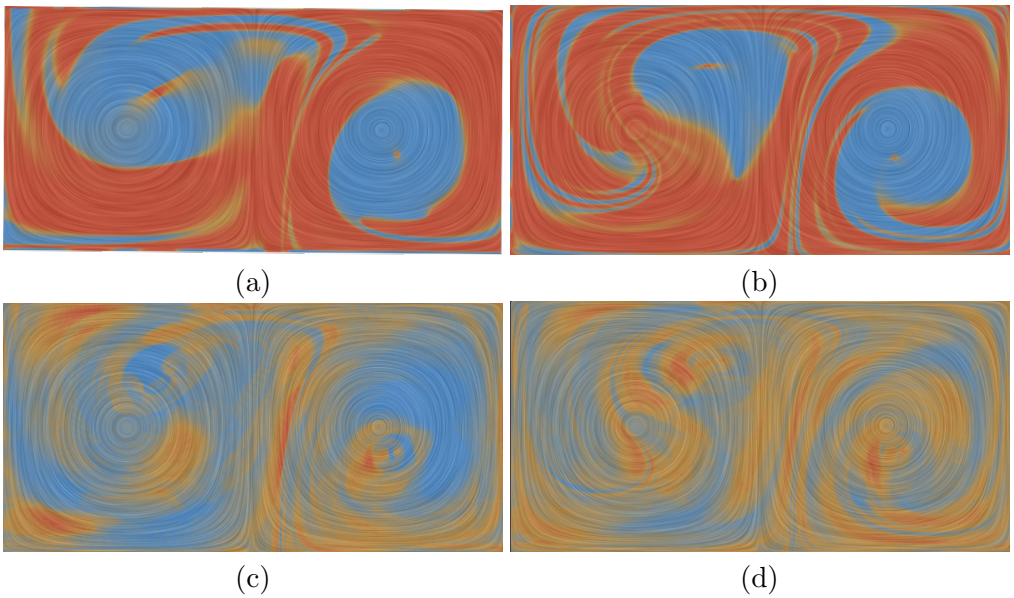


Figure 27: The effect of different temporal kernel lengths, h, for ST_LCC (a-b) and ST_MI (c-d). The h values are 50 (left column) and 100 (right column), respectively, using $\tau = 0.05$.

attribute pairs better highlights the outer layer of vortices (Figures 14g and 20d). For the general understanding of attribute relation and the overview of the flow structure, ST_LCC (and S_LCC for 3D data) should be used. For highlighting the full vortex regions as well as more complicated types of dependency, ST_MI should be applied. Also, our ranking based segmentation can help identify different layers of vortex structure (Figures 17 and 23). Furthermore, joint/conditional entropy of attributes may provide explanation on the dependency of two attributes (Figure 21).

4.5.1 Performance Analysis

We conducted a performance analysis on a number of datasets for the three metrics (i.e., ST_LCC, ST_VEC_LCC, and ST_MI) to demonstrate the performance of our method. We used a workstation with an Intel® Xeon(R) CPU E5-2640 v3 @ 2.60GHz, 128GB RAM and an Nvidia Quadro M4000 graphics card. Using the GPU accelerated implementation of our measurements, we are able to generate the flow movement animation from large sets of particles even for Isabel and large scale ocean in real time. Figure 28 and Table 4 provide the timing results for the correlation computation with respect to the given resolutions, kernel sizes and datasets. The performance gain obtained

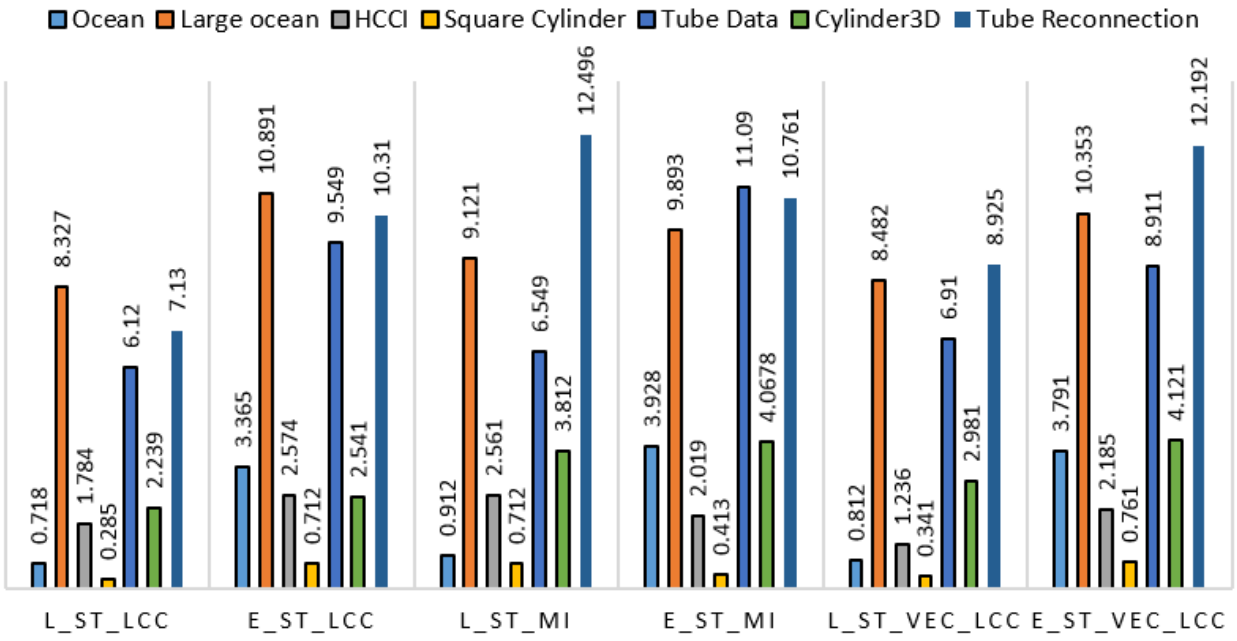


Figure 28: Performance result for different metrics and different datasets.

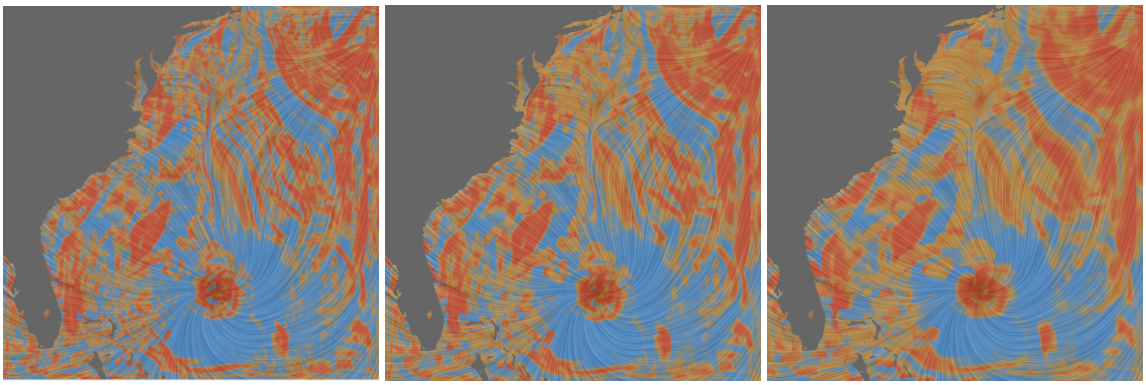
Table 4: Performance result for different datasets.

Dataset/Time(s)	cylinder		HCCI		VortexRing		Large Ocean		Cylinder3D		Tube Reconnection	
	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU	GPU	CPU
ST_LCC	0.285	117	1.784	363	0.679	91	8.327	2670	2.239	671	7.13	1902
ST_VEC_LCC	1.219	687	4.019	1093	2.76	310	20.341	3100	2.981	1589	12.496	3045
ST_MI	0.341	132	2.01	540	1.102	129	12.41	2540	3.812	1890	8.925	2013

using a GPU accelerated version of our method is between 20-50 times, compared to a CPU implementation. This does not include data loading time. The kernel sizes for different datasets were selected empirically to ensure a clear representation of flow features (e.g., vortex structures).

4.5.2 Parameter Study

The spatial size, r, of the kernel and the temporal length, h, are two parameters that the user may need to explore in order to obtain optimal results. In particular, a small spatial kernel size may lose some large features such as large vortices or LCS structures. Figure 22c shows that we only capture many small size vortices although some of them may be artifacts due to a relatively small r. Increasing r may not address this limitation, as some small but important vortices could be smoothed out. Figure 29 shows the results of different spatial kernel sizes on the hurricane simulation. As we can see, using a larger kernel size removes some shearing effect in the upper



(a) (b) (c)

Figure 29: The effect of spatial kernel size, r, on Lagrangian ST_LCC. From the left image to right, the r values for the results are 3, 5, and 10, respectively.

right of the images. The temporal length may affect the completeness of the captured features. The larger the h is, the more pronounced the structure will be. Figure 27 shows the result of a double gyre using various temporal kernel lengths. Comparing these results we can see that after increasing the temporal window size over a certain threshold, the results have somewhat converged.

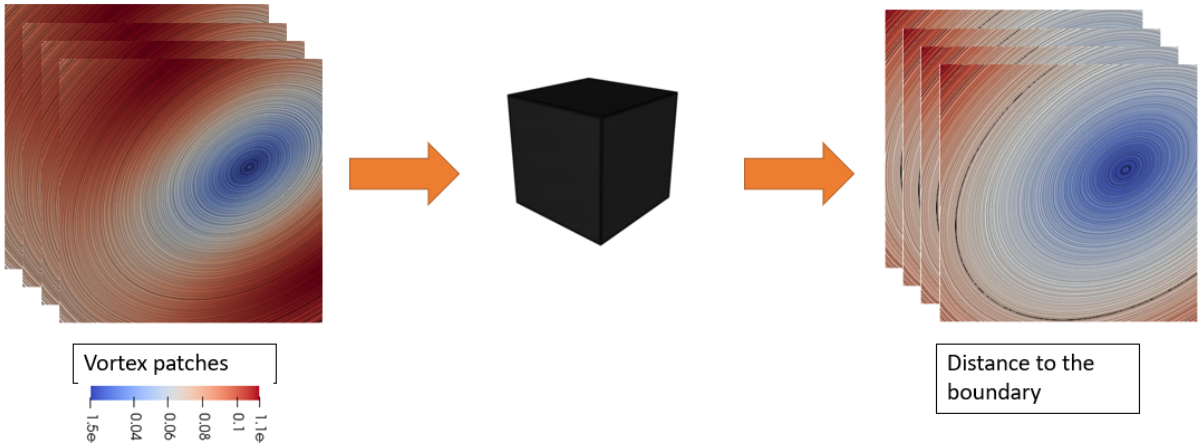


Figure 30: Overview of the deep learning reference problem. The input is vortex velocity patches and the output is a signed distance to the boundary of the vortex. The isoline for the boundary is with isovalue equal to 0.

5 Deep Learning of Vortex Boundary Extraction

Deep learning has been used recently as a valuable method to solve many unsolved and complicated problems in computer vision, robotics and natural language processing. Part of its success is attributed to its unmatched capability to discriminate among features. Extracting features is also an important component in scientific data visualization, and specifically in situations in which features are difficult to characterize accurately. Deep learning and in general machine learning provides a potential way to aid in this kind of analysis. In this part of my dissertation work, I develop a deep neural network that is capable of finding vortex boundaries. To the best of our knowledge, there is no unique definition for the boundary of a vortex other than outermost elliptic LCS introduced by Haller et al. [33], which makes the automatic and robust extraction of vortex boundaries difficult. In addition, another main challenge in extracting vortex boundaries using neural networks is to decide the network inputs. A combination of manifold learning and dimension reduction is one approach to create such a training data from real data. However, due to the limitation of the number of examples that describe the features in different configurations, it is not easy to generalize a model. On the other hand, there exist several parametric methods to

model a vortex synthetically [91, 43, 61]. In our research, we employ a parametric flow model that generates thousands of vector field patches with known groundtruth which is the boundary isoline we computed based on a velocity profile. In addition, the conventional Eulerian grid of velocities caters to a deep convolutional architecture. After supervised learning, we apply the trained model to numerical fluid flow simulations to demonstrate its applicability in practice. Figure 30 shows the pipeline of our approach.

5.1 Synthetic Generation of Training Vector Fields

In order for the convolutional neural network to learn the proper features for vortex boundary identification, a large set of sample fields with various configurations of vortices is needed. Different from face recognition and other similar image segmentation and object recognition problems where a large set of labelled images is available for training, it is not easy to obtain a proper training set for our problem. To address this challenge, we adapt the vector field synthesis strategy recently introduced by Kim and Günther [45].

Similar to Kim and Günther, we synthetically generate physically plausible vortices for training. Since we aim to train the neural network for vortex boundary identification in a supervised manner, the ground truth distance to the vortex boundary is required during training. In order to obtain the exact boundary of a vortex in our synthetic model, we modify the parametric model of Kim and Günther. In our model, the velocity at any point $\mathbf{x} = (x, y)$ is given by the following formula:

$$\mathbf{v}(\mathbf{x}) = \mathbf{S}_i \cdot \mathbf{x} \cdot \frac{v_0(\|\mathbf{x}\|)}{\|\mathbf{x}\|}, \quad \text{with} \quad v_0(r) = \frac{r}{2\pi r_c^2 \left((\frac{r}{r_c})^{2n} + 1 \right)^{\frac{1}{n}}} \tag{7}$$

where $v_0(r)$ is Vatistas's experimentally measured velocity profile [91], r_c is the radius with maximum velocity and n controls the shape of the velocity profile. I refer to Fig. 2 of the work [45] for more information about the effect of n. The matrix $\mathbf{S}_i$ with $i \in \{1, 2, 3\}$ defines one of the following

three base shapes:

$$\mathbf{S}_1 = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{\text{saddle}} \qquad \mathbf{S}_2 = \underbrace{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{\text{center (cw)}} \qquad \mathbf{S}_3 = \underbrace{\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}}_{\text{center (ccw)}} \tag{8}$$

Among these three base shapes, only $\mathbf{S}_2$ and $\mathbf{S}_3$ contain vortices, and the signed distance to the boundary of a vortex in each of these two cases is:

$$d(\mathbf{x}) = \|\mathbf{x}\| - r_c \tag{9}$$

A positive distance indicates locations outside of the vortex, while a negative distance means inside. Since $\mathbf{S}_1$ does not contain vortices, the signed distance in this example flow is set to a large value, e.g., 10 in our experiment.

To introduce variations to the location, orientation, and size of the vortex, we define a random linear transformation matrix $\mathbf{A}$ and a random translation vector $\mathbf{t}$ to transform the domain from $\mathbf{x}$ to $\mathbf{x}'$, and the velocity from $\mathbf{v}(\mathbf{x})$ to $\mathbf{v}'(\mathbf{x}')$:

$$\mathbf{x}' = \mathbf{A} \cdot \mathbf{x} + \mathbf{t} \tag{10}$$

$$\mathbf{v}'(\mathbf{x}') = \mathbf{A} \cdot \mathbf{v}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{v}(\mathbf{A}^{-1} \cdot (\mathbf{x}' - \mathbf{t})) \tag{11}$$

$$d'(\mathbf{x}') = d(\mathbf{A}^{-1} \cdot (\mathbf{x}' - \mathbf{t})) \tag{12}$$

To control the shape of the deformed vortices, we compose the linear transformation from a rotation θ and a non-uniform scaling (s_x, s_y):

$$\mathbf{A}(\theta, s_x, s_y) = \begin{pmatrix} s_x \cos(\theta) & -s_y \sin(\theta) \\ s_x \sin(\theta) & s_y \cos(\theta) \end{pmatrix} \tag{13}$$

In our experiments, θ and $\mathbf{t}$ are chosen from a uniform distribution, and s_x, s_y, r_c and n are chosen

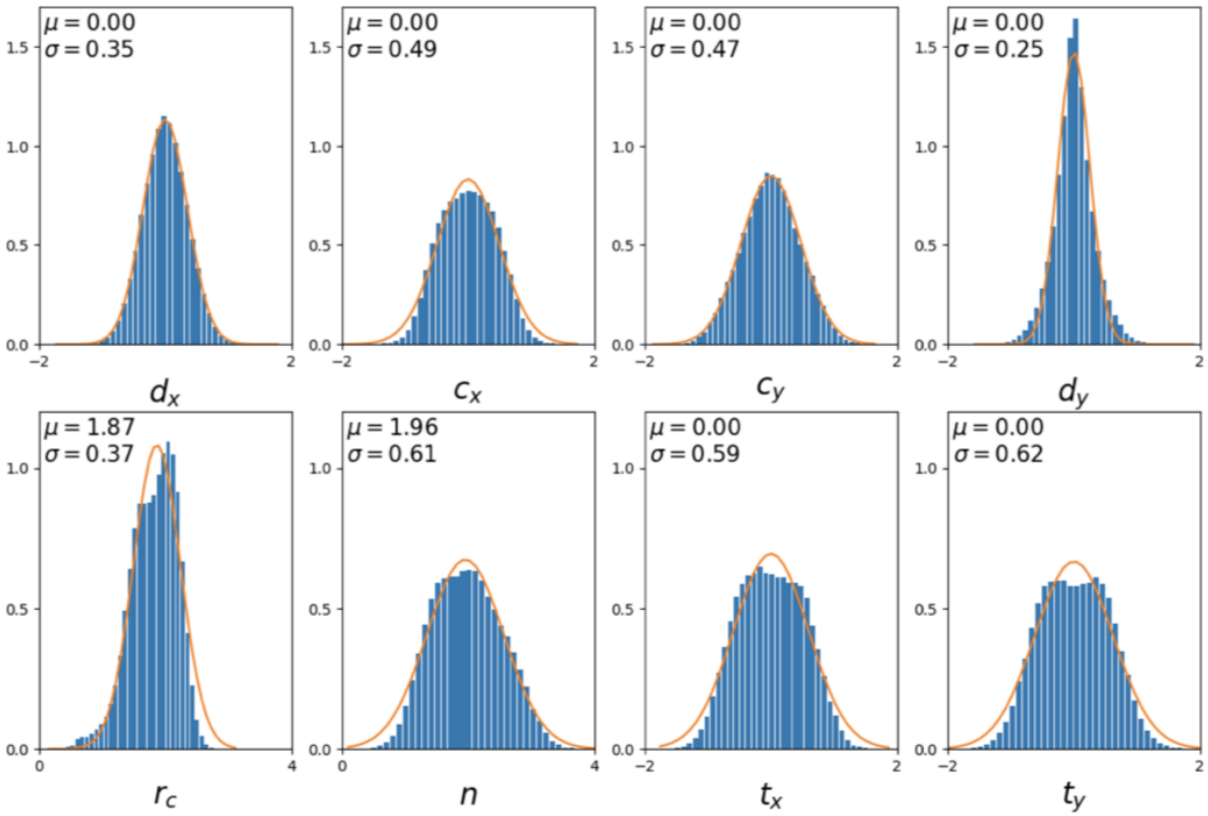


Figure 31: Histograms of the individual model parameters, showing the near-Gaussian distribution of the individual parameter values [44].

from a Gaussian distribution. Figure 31 illustrates the ranges and distributions of the values used by these parameters. The parameter space fitting was done by using the simulated annealing optimization suggested by Kim and Günther [44]. They extracted the optimal reference frame first using 200 iterations of simulated annealing followed by a further 200 iterations of gradient descent to locate the local minimum. The distance metric used between the synthesized vector field and the real-world vector field was the L1 distance with the gradient introduced by [46]. Figure 32 shows different generated patches including vortex(a), saddle(b) and fixed point(c).

5.2 Pipeline

In this section, we propose a convolutional neural network that spans two stages of the visualization pipeline: preprocessing stage (e.g., remove noise and outliers), and then feature extraction stage

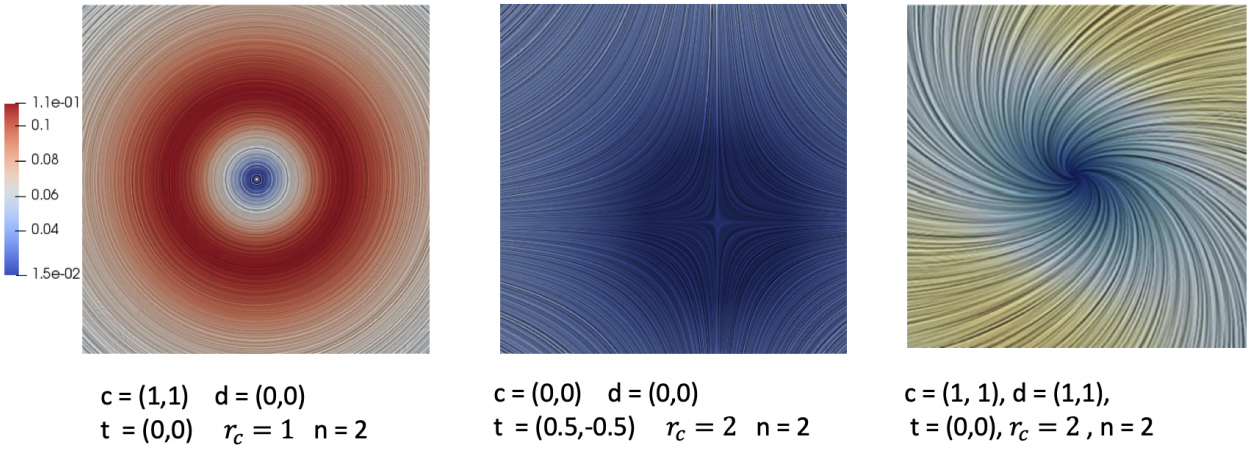


Figure 32: Examples of synthetically generated vector field patches.

(e.g., extract vortex boundary). By combining both in an end-to-end fashion, the extraction of the boundary can be done with high accuracy. In addition to CNN, we also experimented with Resnet [34] and Unet [64] to explore their effectiveness on extracting vortex boundaries. During our supervised learning, we teach the network pairs of steady vector fields and their corresponding vortex boundary. These training examples are generated by using the parametric vector field synthesis described in Section 5.1. In addition, we measure the smoothness as part of the loss function and penalize non-smooth results. The following sections introduce the network architecture and the generation of the training data in more detail.

5.3 Generating Training Data

A key ingredient to a successful training is the adequate preparation of the input data. Aside from the raw size of the (possibly augmented) training data, the shape of the data requires equally careful consideration. Since many of the more recent network architectures, such as CNNs, ResNets and Unets, utilize the spatial relationships of the input pixels, i.e., they operate on image data that is layed out on regular grids, it would make sense to similarly preserve the spatial embedding of the inputs. In its most direct way, however, this limits us to an Eulerian perspective: at each grid point, we would only encode the given velocity vector.

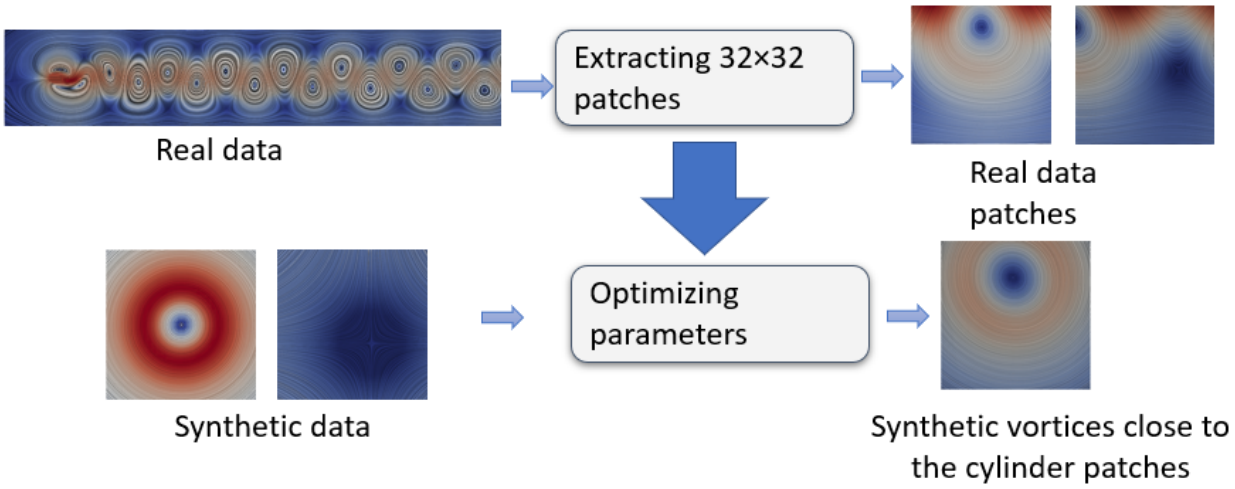


Figure 33: Overview of our pipeline for optimizing the parameters.

Velocity Input As baseline, we directly encode the velocity given at each grid point in each synthetically generated vortex patch and choose the signed distance to the vortex boundary as target. Figure 33 shows the pipeline for our patch optimization method.

5.4 Architectures

CNN We build our network as a form of typical CNN as seen in Figure 34. The first part of our CNN consists of 2D convolutional kernels.

In this part, the dimension reduction is performed for the feature extraction with a kernel size of 3×3 and strides of 2×2, followed by a batch normalization and a rectified linear unit (ReLU) layer. The size of the filter is doubled starting from 64 to maximally 1024. With 2D steady fields v, the number of convolutional layers is calculated by $n = log_2 max(H, W) - 2$, and the output dimension of the last convolutional layer would be 1024.

The second part of our network exploits fully connected layers instead of convolutional filters for the final inference from identified high level vortex features. Batch normalization and a ReLU layer are followed in the same fashion as convolutional layers, and a dropout with the probability of 0.1 is used to avoid overfitting. Note that we infer the full patch as the target of network, and we test either the center or the full patch as target.

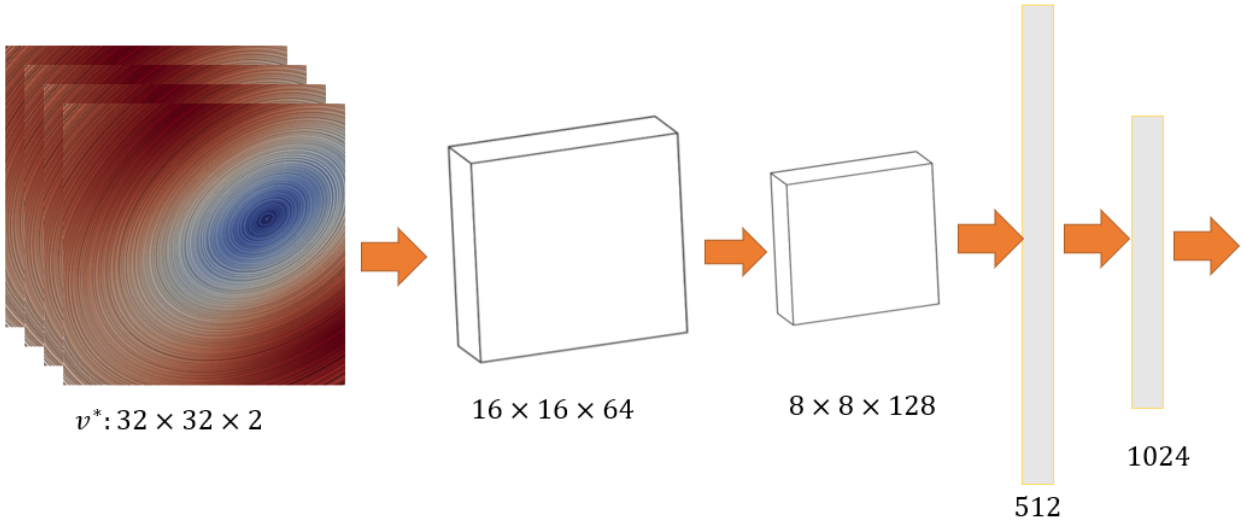


Figure 34: Our CNN Architecture. The numbers below each box represent the dimensions of the feature maps.

Resnet Our Resnet model is based on the Resnet20 architecture by He et al. [34]. We use a learning rate 0.001. There exists six residual blocks and each block includes three convolutional layers. Figure 35 shows the structure of one residual block. The core idea of ResNet is introducing an identity shortcut connection or skip connection that skips one or more layers, as shown in that figure. The motivation behind that is the deeper model should not produce a training error higher than its shallower counterparts. The authors in [34] hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping. And the residual block above explicitly allows it to do precisely that.

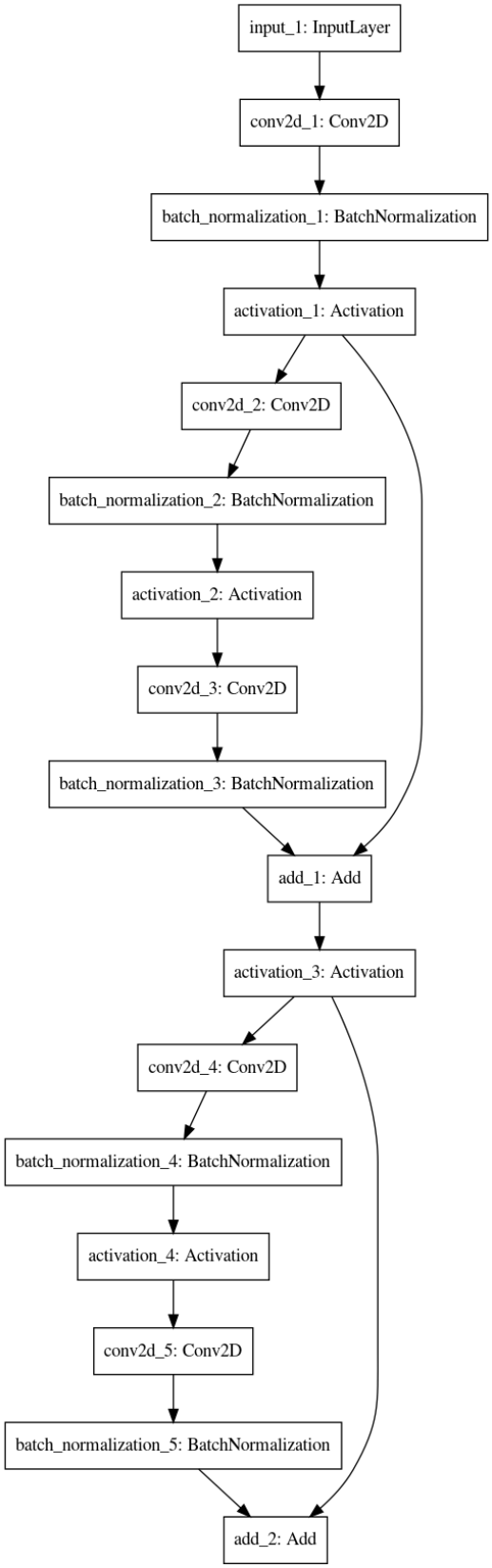


Figure 35: One residual block in Resnet32 Architecture.

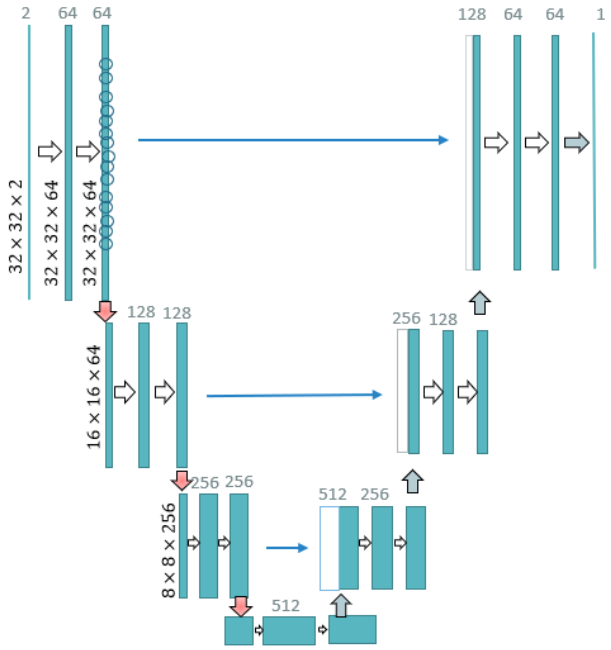


Figure 36: Our Unet Architecture. We used three layers of depth. The numbers above each box represent the dimension of feature maps.

Unet Our Unet model is based on the original model proposed by Ronneberger et al. [64]. We use a depth of three and a dropout of 0.001. The activation function is set to ReLu and we use Max Pooling between the layers. Figure 36 shows the structure of our model. In addition to a skip connection, it also includes a concatenation with the correspondingly cropped feature map from the contracting path. In the end, the 1×1 Conv layer is used to make the number of feature maps the same as the number of segments which are desired in the output. Unet uses a loss function for each pixel of the image. This helps in easy identification of individual cells within the segmentation map and it could generate the results with higher accuracy in comparison to ResNet.

5.5 Dataset

We synthesize our data set based on the Vatistas velocity profile as described in Section 5.1. Once a 2D steady field is generated, we apply linear transformation and compute the distance field.

Training Data As described earlier, we applied a linear transformation including scaling and rotation to generate enough training sample flows that we might see in the real-world data. The number of sample flows is 25,000, and the data set for training and testing is split into the ratio of 9:1. Some examples of the training data are illustrated in Figures 37.

Patch Normalization Since the training data set is defined in the 2D unit domain (i.e. $\mathcal{X} \times \mathcal{Y} = [-2, 2]^2$), input patches of numerical data sets are also scaled into this domain. Thus, given velocity patch slices $v_p : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{D}$, defined in the domain $\mathcal{X} \times \mathcal{Y} = [x_{min}, x_{max}] \times [y_{min}, y_{max}]$, we compute $\bar{v}_p$ in the unit domain via:

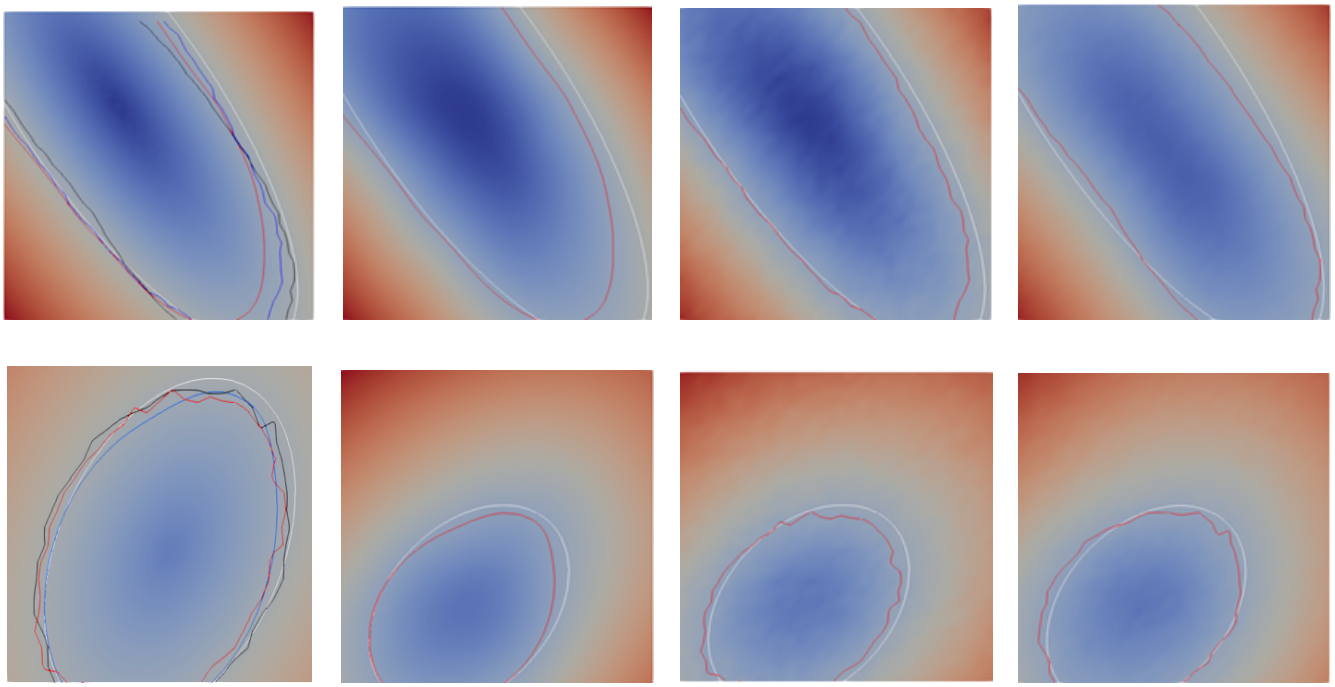
$$\bar{v}_p = \begin{bmatrix} \frac{1}{x_{max} - x_{min}} & 0 \\ 0 & \frac{1}{y_{max} - y_{min}} \end{bmatrix} \tag{14}$$

Furthermore, we globally normalize the magnitude of the training data to $[-2, 2]$ during training. Thus, re-scaled patches from the validation data set are finally normalized by the same factors that have been applied to the training data before it was fed to the network.

Implementation We implemented all of our models using Keras [16] with Tensorflow [1] as backend. The networks are trained on each data set for 100 epochs using an Adam optimizer [47] with a learning rate of 0.001 and the mean squared error (MSE) loss function. For Resnet, the learning rate is variable based of the epoch number. We use a learning rate equal to 0.001 until epoch number 80 and after that it is decreased to 1e-4. We choose a batch size of 256. Finally, the model is selected, which shows the minimum test error during training. All visualizations are created with the visualization toolkit Paraview.

5.6 Results

In the following sections, we first apply our networks to synthetic data generated with our parametric model, before testing it on the extraction of the boundary from vortices in unseen numerical data. Afterwards, the performance is discussed.



(a) Comparison (b) CNN (c) Resnet (d) Unet

Figure 37: Results of vortex extraction on test splits. The L2 errors are 0.12, 0.06 and 0.02 for the three architectures, respectively.

5.6.1 Testing on Synthetic Data

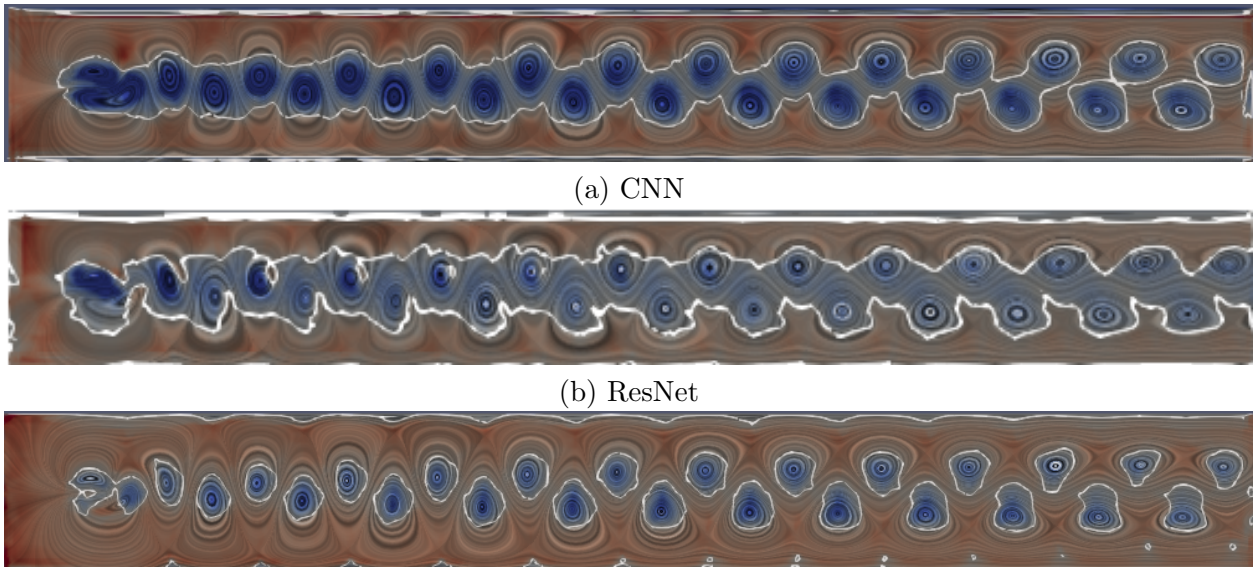
We have applied the generated synthetic sample flows to train a CNN, a Resnet, and a Unet, as illustrated earlier, using the velocity vectors as the input. To evaluate their accuracy, we applied these three trained networks to the extraction of the vortex boundaries from a number of synthetic flows. Figure 37 compares the results obtained from these three networks on two synthetic vortices. From these results, we see that the Unet obtains a boundary closer to the ground truth (i.e., the white ellipses) than the other two networks.

5.6.2 Validation of Network on Numerical Data

Next, we apply our networks to unseen numerical data to evaluate how well the network is generalized, given its synthetic training data. After training, we validate our networks on a few real world data, e.g., the CYLINDER data set. As the input window size of our models is fixed, we

have evaluated this data set by sliding a lookup window over the entire domain. Alternatively, we also set the whole patch as the target and average all overlapping patches. In order to evaluate the CYLINDER flow, which has a resolution of 80×640, we slide a lookup window of size 16×16 with stride 1 and sequentially predict parameters for the center pixel (e.g., 159 steps for 40,625 windows with a batch size of 256). We use stride 1 in order to cover all input data of CYLINDER by shifting one unit at a time. So, we do not miss any particles.

Error Metric and Smoothness We use mean squared error and mean absolute error. Figure 38 shows the result of our models for CYLINDER data using velocity as input.



(a) CNN

(b) ResNet

(c) Unet

Figure 38: Comparison of boundary extraction with different architectures of neural network using the CYLINDER flow. The input to the networks is velocity patches. Our method shows Unet outperforms the other networks.

5.7 Future Work

In our future work, we are going to test our models on other datasets and measure the performance. In addition we plan to add inertial particles to the trajectory input data to see how this extra

information can help the network. In addition, we plan to compare our method with the region-based method discussed in Chapter 2 and the elliptic LCS proposed by Haller et al. [33]. Also, we propose a custom loss function by adding the partial derivative to the loss function in order to prevent jargon in the boundary and get smooth results. Finally we will add transfer learning and evaluate the strength of our models with different inputs.

6 Conclusions and Future Work

In this dissertation, I concentrate on the analysis and visualization of vortices – an important flow feature that is responsible for many important physical events in flows (e.g., transportation of mass and energy). However, due to the lack of a unified definition of vortices, it is challenging to identify and characterize vortex behaviors. My dissertation contributes to addressing this problem by introducing three different approaches that characterize the geometric and physical properties of vortices. First, I introduced the first Morse decomposition computation framework for the topological analysis of 3D piece-wise linear vector fields defined on uniform grids. I applied this computation framework to a number of 3D flows to reveal the recurrent dynamics, including vortical flow behaviors. A GPU based implementation of this framework was introduced to accelerate its performance. However, this approach only characterizes vortices based on their geometry behaviors (i.e. swirling or rotational flow), which may miss other vortices that are hidden by strong translational flow motion (e.g., the flow behind a cylinder). To address this limitation of the topological analysis, in my second approach, I proposed a physics-based strategy to analyze and explore the pairwise relations among attributes in unsteady flow. This approach provides a general and powerful way to explore linear relation, i.e. correlations and non-linear relation, i.e. dependency among attribute-pairs derived from unsteady flow. With this correlation computation framework, I discovered that certain pairs of attributes (e.g., Q and acceleration) have different correlation behaviors within and outside vortex areas, which enables us to identify vortex regions more reliably. Nonetheless, this correlation computation framework cannot identify the exact boundaries of vortices. To extract the vortex boundaries, I proposed a machine learning (especially a neural network) framework. To address the difficulty of getting a large training set with various vortex configurations represented, I introduced a parametric vector field synthesis framework. I applied the generated large training set to train three different neural networks, including a CNN, a Resnet, and a Unet. My initial experiments found that Unet outperforms the other two networks in terms of the accuracy in vortex boundary extraction.

Although our proposed method works across many datasets, there are some limitations. For the topology-based approach, our method cannot sufficiently refine the obtained Morse set for the flow behind a square cylinder due to the limited level of adaptive face sampling (i.e. k). In addition, the proposed CUDA implementation may not be optimal, as it does not utilize the advantage of shared memory mechanism. Furthermore, the memory allocation for particles is performed at the beginning of the algorithm and remains unchanged, which is sub-optimal.

For the physics-based approach, we use a fixed kernel size throughout the computation for one dataset, which may cause the loss of some important features that often have varying scales in space and time. Second, our method is sensitive to the temporal sampling rate provided by the simulation. If the temporal sampling rate is coarse, we see a lot of fluctuation in the pathline computation which interpolation cannot alleviate. We plan to address these limitations in the future and extend our framework to study the attribute relations in ensemble flow data. Finally, we plan to migrate our current implementation onto clusters so that extremely large-scale data can be processed more effectively.

For the deep learning approach, the first limitation is that we need to optimize the parameters for generating training data synthetically. Second, our method is sensitive to the training data and the patterns generated by our synthetic vortex model. There are still some challenges to evaluate a training dataset and how to make sure it could cover all the existing vortex patterns in the real world data. Finally, we would need to use a quantitative metric to compare different architecture results. We plan to address these limitations in future work.

Bibliography

[1] Martın Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Xue Bai, Changbo Wang, and Chenhui Li. “A Streampath-Based RCNN Approach to Ocean Eddy Detection”. In: *IEEE Access* 7 (2019), pp. 106336–106345.

[3] Harsh Bhatia, Valerio Pascucci, and Peer-Timo Bremer. “The Natural Helmholtz-Hodge Decomposition for Open-Boundary Flow Analysis.” In: *IEEE Transactions on Visualization and Computer Graphics* 20.11 (2014), pp. 1566–1578.

[4] Harsh Bhatia et al. “Extracting Features from Time-Dependent Vector Fields Using Internal Reference Frames”. In: *Computer Graphics Forum*. Vol. 33. 3. Wiley Online Library. 2014, pp. 21–30.

[5] Tang Bin and Li Yi. “CNN-based Flow Field Feature Visualization Method”. In: *International Journal of Performability Engineering* 14.3 (2018), p. 434.

[6] Ayan Biswas et al. “An information-aware framework for exploring multivariate data sets”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (2013), pp. 2683–2692.

[7] Simone Camarri et al. “Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers”. In: *XVII Congresso AIMeTA di Meccanica Teorica e Applicata*. Vol. 1. Firenze University Press, Florence, Italy. 2005, pp. 23–34.

[8] Cheng-Kai Chen et al. “Static correlation visualization for large time-varying volume data”. In: *Pacific Visualization Symposium (PacificVis), 2011 IEEE*. IEEE. Hong Kong, 2011, pp. 27–34.

[9] G. Chen et al. “Efficient Morse decompositions of vector fields”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.4 (July 2008), pp. 848–862.

[10] G. Chen et al. “Vector field editing and periodic orbit extraction using Morse decomposition”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.4 (July 2007), pp. 769–785.

[11] Guoning Chen and Shuyu Xu. “An image-space Morse decomposition for 2D vector fields”. In: *Visualization and Data Analysis 2015*. Vol. 9397. International Society for Optics and Photonics. 2015, p. 93970I.

[12] Guoning Chen et al. “Morse set classification and hierarchical refinement using Conley index”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.5 (2012), pp. 767–782.

[13] Jacqueline H Chen et al. “Direct numerical simulation of ignition front propagation in a constant volume with temperature inhomogeneities: I. Fundamental analysis and diagnostics”. In: *Combustion and Flame* 145.1-2 (2006), pp. 128–144.

[14] Min Chen and Heike Jaenicke. “An information-theoretic framework for visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1206–1215.

[15] Min Chen et al. *Information Theory Tools for Visualization*. CRC Press, 2016.

[16] François Chollet et al. “Keras: Deep learning library for theano and tensorflow”. In: *URL: www.keras. io/k* 7.8 (2015). Accessed:2019-03-20, T1.

[17] Alain Crouzil, Ouis Massip-pailhes, and Serge Castan. “A new correlation criterion based on gradient fields similarity”. In: *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. Vol. 1. IEEE. 1996, pp. 632–636.

[18] R Cucitore, M Quadrio, and A Baron. “On the effectiveness and limitations of local criteria for the identification of a vortex”. In: *European Journal of Mechanics-B/Fluids* 18.2 (1999), pp. 261–282.

[19] Liang Deng et al. “A CNN-based vortex identification method”. In: *Journal of Visualization* 22.1 (2019), pp. 65–78.

[20] Soumya Dutta et al. “Pointwise information guided visual analysis of time-varying multi-fields”. In: *SIGGRAPH Asia 2017 Symposium on Visualization*. ACM. 2017, p. 17.

[21] Thomas Ertl, Filip Sadlo, and Markus Uffinger. “A Time-Dependent Vector Field Topology Based on Streak Surfaces”. In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (2013), pp. 379–392. ISSN: 1077-2626. DOI: http://doi.ieeecomputersociety.org/10.1109/TVCG.2012.131.

[22] Martin Falk and Daniel Weiskopf. “Output-sensitive 3D line integral convolution”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.4 (2008), pp. 820–834.

[23] Katharina Franz et al. “Ocean eddy identification and tracking using neural networks”. In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 6887–6890.

[24] Raphael Fuchs et al. “Parallel Vectors Criteria for Unsteady Flow Vortices”. In: *IEEE Transactions on Visualization and Computer Graphics* 14.3 (2008), pp. 615–626. ISSN: 1077-2626. DOI: http://dx.doi.org/10.1109/TVCG.2007.70633.

[25] Raphael Fuchs et al. “Toward a Lagrangian vector field topology”. In: *Computer Graphics Forum* 29.3 (2010), pp. 1163–1172.

[26] Raphael Fuchs et al. “Toward a Lagrangian vector field topology”. In: *Computer Graphics Forum*. Vol. 29. 3. Wiley Online Library. 2010, pp. 1163–1172.

[27] C. Garth et al. “Lagrangian Visualization of Flow-Embedded Surface Structures”. In: *Computer Graphics Forum* 27.3 (2008), pp. 1007–1014.

[28] Yi Gu and Chaoli Wang. “A study of hierarchical correlation clustering for scientific volume data”. In: *Advances in Visual Computing* (2010), pp. 437–446.

[29] Tobias Günther, Maik Schulze, and Holger Theisel. "Rotation invariant vortices for flow visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 817–826.

[30] Tobias Günther and Holger Theisel. "The State of the Art in Vortex Extraction". In: *Computer Graphics Forum*. Wiley Online Library. Zurich, Switzerland, 2018, pp. 149–173.

[31] G. Haller. "Lagrangian coherent structures and the rate of strain in two-dimensional turbulence". In: *Phys. Fluids A* 13 (2001), pp. 3365–3385.

[32] George Haller. "An objective definition of a vortex". In: *Journal of Fluid Mechanics* 525 (2005), pp. 1–26.

[33] George Haller et al. "Defining coherent vortices objectively from the vorticity". In: *Journal of Fluid Mechanics* 795 (2016), pp. 136–173.

[34] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[35] James Helman and Lanbertus Hesselink. "Representation and display of vector field topology in fluid flow data sets". In: *Computer* 8 (1989), pp. 27–36.

[36] Marcel Hlawatsch et al. "Pathline glyphs". In: *Computer Graphics Forum*. Vol. 33. 2. Wiley Online Library. 2014, pp. 497–506.

[37] Julian CR Hunt, AA Wray, and Parviz Moin. "Eddies, streams, and convergence zones in turbulent flows". In: *Studying Turbulence Using Numerical Simulation Databases, 2*. Vol. 1. 1988, pp. 193–208.

[38] Fazle Hussain and Karthik Duraisamy. "Mechanics of viscous vortex reconnection". In: *Physics of Fluids* 23.2 (2011), p. 021701.

[39] J. Sahner, T. Weinkauf, and H.-C. Hege. "Galilean Invariant Extraction and Iconic Representation of Vortex Core Lines". In: *Proc. Eurographics / IEEE VGTC Symposium on*

Visualization (EuroVis '05). Ed. by K. Joy K. Brodlie D. Duke. Leeds, UK, June 2005, pp. 151–160.

[40] Jinhee Jeong and Fazle Hussain. "On the identification of a vortex". In: *Journal of Fluid Mechanics* 285 (1995), pp. 69–94.

[41] Ming Jiang, Raghu Machiraju, and David Thompson. "Detection and visualization of vortices". In: *The Visualization Handbook*. Starkville, Mississippi: Academic Press, 2005, pp. 295–309.

[42] Jens Kasten et al. "Two-dimensional Time-dependent Vortex Regions based on the Acceleration Magnitude". In: *Transactions on Visualization and Computer Graphics (Vis'11)* 17.12 (2011), pp. 2080–2087.

[43] W Kaufmann. "Über die Ausbreitung kreiszylindrischer Wirbel in zähen (viskosen) Flüssigkeiten". In: *Archive of Applied Mechanics* 31.1 (1962), pp. 1–9.

[44] Byungsoo Kim and Tobias Günther. "Robust Reference Frame Extraction from Unsteady 2D Vector Fields with Convolutional Neural Networks". In: *arXiv preprint arXiv:1903.10255* (2019).

[45] Byungsoo Kim and Tobias Günther. "Robust Reference Frame Extraction from Unsteady 2D Vector Fields with Convolutional Neural Networks". In: *Computer Graphics Forum (Proc. EuroVis)* 38.3 (2019).

[46] Byungsoo Kim et al. "Deep fluids: A generative network for parameterized fluid simulations". In: *Computer Graphics Forum*. Vol. 38. 2. Wiley Online Library. 2019, pp. 59–70.

[47] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[48] R. S. Laramee et al. "The state of the art in flow visualization: dense and texture-based techniques". In: *Computer Graphics Forum* 23.2 (June 2004), pp. 203–221. URL: `http://www.VRVis.at/ar3/pr2/star/`.

[49] Teng-Yok Lee and Han-Wei Shen. "Visualization and exploration of temporal trend relationships in multivariate time-varying data". In: *IEEE Transactions on Visualization and Computer Graphics* 15.6 (2009).

[50] Marzieh Berenjkoub Lei Zhang Guoning Chen. "Correlation Study on Attributes of Unsteady Flows". In: *2017 IEEE Visualization Conference (VIS), 1-6 October, Phoenix, AZ, USA*. 2017.

[51] F. Lekien, S.C. Shadden, and J.E. Marsden. "Lagrangian coherent structures in n-dimensional systems". In: *Journal of Mathematical Physics* 48.6 (2007), Art. No. 065404.

[52] Redouane Lguensat et al. "EddyNet: A deep neural network for pixel-wise classification of oceanic eddies". In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE. 2018, pp. 1764–1767.

[53] Yang Liu et al. "A CNN-based Shock Detection Method in Flow Visualization". In: *Computers Fluids* (2019). ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2019.03.022. URL: http://www.sciencedirect.com/science/article/pii/S0045793019300969.

[54] Edward N Lorenz. "Deterministic nonperiodic flow". In: *Journal of the Atmospheric Sciences* 20.2 (1963), pp. 130–141.

[55] Hans J Lugt. "The dilemma of defining a vortex". In: *Recent Developments in Theoretical and Experimental Fluid Mechanics*. Springer, 1979, pp. 309–321.

[56] NASA. *Estimating the Circulation and Climate of the Ocean, Phase II*. www.ecco2.org. Accessed=2018-01-20.

[57] Akira Okubo. "Horizontal dispersion of floatable particles in the vicinity of velocity singularities such as convergences". In: *Deep Sea Research and Oceanographic Abstracts*. Vol. 17. 3. Elsevier. 1970, pp. 445–454.

[58] Paolo Orlandi and Roberto Verzicco. "Vortex rings impinging on walls: axisymmetric and three-dimensional simulations". In: *Journal of Fluid Mechanics* 256 (1993), pp. 615–646.

[59] Armin Pobitzer et al. “A statistics-based dimension reduction of the space of path line attributes for interactive visual flow analysis”. In: *Pacific Vis*. Songdo, Korea, 2012, pp. 113–120.

[60] Armin Pobitzer et al. “The State of the Art in Topology-based Visualization of Unsteady Flow”. In: *Computer Graphics Forum* 30.6 (Sept. 2011), pp. 1789–1811.

[61] F. H. Post et al. “The state of the art in flow visualization: feature extraction and tracking”. In: *Computer Graphics Forum* 22.4 (Dec. 2003), pp. 775–792. URL: `http://cs.swan.ac.uk/~csbob/research/`.

[62] Wieland Reich et al. “Combinatorial vector field topology in three dimensions”. In: *Topological Methods in Data Analysis and Visualization II*. Springer, 2012, pp. 47–59.

[63] Stephen K Robinson. “Coherent motions in the turbulent boundary layer”. In: *Annual Review of Fluid Mechanics* 23.1 (1991), pp. 601–639.

[64] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. Munich, Germany, 2015, pp. 234–241.

[65] I Ari Sadarjoen and Frits H Post. “Detection, quantification, and tracking of vortices using streamline geometry”. In: *Computers & Graphics* 24.3 (2000), pp. 333–341.

[66] I Ari Sadarjoen and Frits H Post. “Geometric methods for vortex extraction”. In: *Data Visualization’99*. Springer. 1999, pp. 53–62.

[67] Filip Sadlo and Daniel Weiskopf. “Time-Dependent 2-D Vector Field Topology: An Approach Inspired by Lagrangian Coherent Structures”. In: *CoRR* abs/1105.5678 (2011).

[68] Natascha Sauber, Holger Theisel, and H-P Seidel. “Multifield-graphs: An approach to visualizing correlations in multifield scalar data”. In: *Visualization and Computer Graphics, IEEE Transactions on* 12.5 (2006), pp. 917–924.

[69] Franz Sauer and Kwan-Liu Ma. "Spatio-Temporal Feature Exploration in Combined Particle/Volume Reference Frames". In: *IEEE Transactions on Visualization and Computer Graphics* 23.6 (2017), pp. 1624–1635.

[70] Nathanaël Schaeffer and Stéphane Le Dizès. "Nonlinear dynamics of the elliptic instability". In: *Journal of Fluid Mechanics* 646 (2010), pp. 471–480.

[71] G. Scheuermann et al. "Visualization of higher order singularities in vector fields". In: *Proceedings of IEEE Visualization '97*. Oct. 1997, pp. 67–74. URL: http://visinfo.zib.de/EVlib/Show?EVL-1997-121.

[72] Cambridge Aerospace Series. *Principles of Helicopter Aerodynamics; 2000; 5 pages; New York, NY US.*

[73] S.C. Shadden, F. Lekien, and J.E. Marsden. "Definition and Properties of Lagrangian Coherent Structures from Finite-Time Lyapunov Exponents in Two-Dimensional Aperiodic Flows". In: *Physica D* 212.3–4 (2005), pp. 271–304.

[74] Shawn C Shadden, Francois Lekien, and Jerrold E Marsden. "Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows". In: *Physica D: Nonlinear Phenomena* 212.3 (2005), pp. 271–304.

[75] Claude Elwood Shannon. "A mathematical theory of communication". In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), pp. 3–55.

[76] Kuangyu Shi et al. "Path Line Attributes - an Information Visualization Approach to Analyzing the Dynamic Behavior of 3D Time-Dependent Flow Fields". In: *Topology-Based Methods in Visualization II*. Ed. by Hans-Christian Hege, Konrad Polthier, and Gerik Scheuermann. Mathematics and Visualization. Grimma, Germany: Springer, 2009, pp. 75–88. ISBN: 978-3-540-88605-1. DOI: 10.1007/978-3-540-88606-8_6.

[77] Kuangyu Shi et al. "Visualizing Transport Structures of Time-Dependent Flow Fields". In: *IEEE Computer Graphics and Applications* 28.5 (2008), pp. 24–36.

[78] Bernard W Silverman. *Density estimation for statistics and data analysis*. London, UK: Routledge, 2018.

[79] Primoz Skraba et al. "2D vector field simplification based on robustness". In: *Pacific Visualization Symposium (PacificVis), 2014 IEEE*. IEEE. Yokohama, Japan, 2014, pp. 49–56.

[80] Ahmad Sohankar, C Norberg, and L Davidson. "Simulation of three-dimensional flow around a square cylinder at moderate Reynolds numbers". In: *Physics of Fluids* 11.2 (1999), pp. 288–306.

[81] Stephen M. Stigler. "Francis Galton's Account of the Invention of Correlation". In: *Statistical Science* 4.2 (May 1989), pp. 73–79. DOI: 10.1214/ss/1177012580. URL: http://dx.doi.org/10.1214/ss/1177012580.

[82] Carlos Michelén Ströfer et al. "Data-driven, physics-based feature extraction from fluid flow fields". In: *arXiv preprint arXiv:1802.00775* (2018).

[83] Andrzej Szymczak and Nicholas Brunhart-Lupo. "Nearly recurrent components in 3D piecewise constant vector fields". In: *Computer Graphics Forum*. Vol. 31. 3pt3. Wiley Online Library. 2012, pp. 1115–1124.

[84] Andrzej Szymczak and Eugene Zhang. "Robust Morse decompositions of piecewise constant vector fields". In: *IEEE Transactions on Visualization and Computer Graphics* 18.6 (2012), pp. 938–951.

[85] T. Weinkauf and H. Theisel. "Streak Lines as Tangent Curves of a Derived Vector Field". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (Nov. 2010), pp. 1225–1234.

[86] T. Weinkauf, H. Theisel, and O. Sorkine. "Cusps of Characteristic Curves and Intersection-Aware Visualization of Path and Streak Lines". In: *Topological Methods in Data Analysis and Visualization II*. Ed. by R. Peikert et al. Mathematics and Visualization. Springer, 2012, pp. 161–176. URL: http://tinoweinkauf.net/publications/absweinkauf12b.html.

[87] Holger Theisel and H-P Seidel. "Feature flow fields". In: *Proceedings of the Symposium on Data Visualisation 2003*. Eurographics Association. 2003, pp. 141–148.

[88] Holger Theisel et al. "Stream line and path line oriented topology for 2D time-dependent vector fields". In: *Proceedings of the Conference on Visualization'04*. IEEE Computer Society. 2004, pp. 321–328.

[89] X. Tricoche, G. Scheuermann, and H. Hagen. "Topology-based visualization of time-dependent 2D vector fields". In: *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym '01)*. May 2001, pp. 117–126.

[90] Xavier Tricoche et al. "Topology tracking for the visualization of time-dependent two-dimensional flows". In: *Computers & Graphics* 26.2 (2002), pp. 249–257.

[91] Georgios H Vatistas, V Kozel, and WC Mih. "A simpler model for concentrated vortices". In: *Experiments in Fluids* 11.1 (1991), pp. 73–76.

[92] Ko-Chih Wang et al. "Statistical visualization and analysis of large data using a value-based spatial distribution". In: *Pacific Visualization Symposium (PacificVis), 2017 IEEE*. IEEE. Seoul, Korea, 2017, pp. 161–170.

[93] Tino Weinkauf and Holger Theisel. "Streak lines as tangent curves of a derived vector field". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1225–1234.

[94] Daniel Weiskopf, Tobias Schafhitzel, and Thomas Ertl. "Texture-based visualization of unsteady 3d flow by real-time advection and volumetric illumination". In: *IEEE Transactions on Visualization & Computer Graphics* 3 (2007), pp. 569–582.

[95] John Weiss. "The dynamics of enstrophy transfer in two-dimensional hydrodynamics". In: *Physica D: Nonlinear Phenomena* 48.2-3 (1991), pp. 273–294.

[96] Alexander Wiebel et al. "Topological flow structures in a mathematical model for rotation-mediated cell aggregation". In: *Topological Methods in Data Analysis and Visualization*. Springer, 2011, pp. 193–204.

[97] Charles HK Williamson. "Vortex dynamics in the cylinder wake". In: *Annual Review of Fluid Mechanics* 28.1 (1996), pp. 477–539.

[98] T. Wischgoll and G. Scheuermann. "Detection and visualization of closed streamlines in planar fields". In: *IEEE Transactions on Visualization and Computer Graphics* 7.2 (2001), pp. 165–172.

[99] T. Wischgoll and G. Scheuermann. "Locating closed streamlines in 3D vector fields". In: *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym 02)*. May 2002, pp. 227–280.

[100] Lijie Xu, Teng-Yok Lee, and Han-Wei Shen. "An information-theoretic framework for flow visualization". In: *IEEE Transactions on Visualization and Computer Graphics* 16.6 (2010), pp. 1216–1224.

[101] Huijie Zhang et al. "Correlation visualization of time-varying patterns for multi-variable data". In: *IEEE Access* 4 (2016), pp. 4669–4677.

[102] Huijie Zhang et al. "Uncertainty visualization for variable associations analysis". In: *The Visual Computer* (2017), pp. 1–19.

[103] Lei Zhang et al. "Enhanced vector field visualization via Lagrangian accumulation". In: *Computers & Graphics* (2017).

[104] Zhiyuan Zhang et al. "Visual correlation analysis of numerical and categorical data on the correlation map". In: *IEEE Transactions on Visualization and Computer Graphics* 21.2 (2015), pp. 289–303.