# Interactive Visualization of Computational Fluid Dynamics Data

Zhenmin Peng

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy

Swansea University
Prifysgol Abertawe

Department of Computer Science
Swansea University

August 2011

# Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

# Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)

Date .....

# Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

# Abstract

This thesis describes a literature study and a practical research in the area of flow visualization, with special emphasis on the interactive visualization of Computational Fluid Dynamics (CFD) datasets.Given the four main categories of flow visualization methodology: direct, geometric, texture-based and feature-based flow visualization, the research focus of our thesis is on the direct, geometric and feature-based techniques. And the feature-based flow visualization is highlighted in this thesis.

After we present an overview of the state-of-the-art of the recent developments in the flow visualization in higher spatial dimensions (2.5D, 3D and 4D), we propose a fast, simple, and interactive glyph placement algorithm for investigating and visualizing boundary flow data based on unstructured, adaptive resolution boundary meshes from CFD dataset. Afterward, we propose a novel, automatic mesh-driven vector field clustering algorithm which couples the properties of the vector field and resolution of underlying mesh into a unified distance measure for producing high-level, intuitive and suggestive visualization of large, unstructured, adaptive resolution boundary CFD meshes based vector fields. Next we present a novel application with multiple-coordinated views for interactive information-assisted visualization of multi-dimensional marine turbine CFD data. Information visualization techniques are combined with user interaction to exploit our cognitive ability for intuitive extraction of flow features from CFD datasets. Later, we discuss the design and implementation of each visualization technique used in our interactive flow visualization framework, such as glyphs, streamlines, parallel coordinate plots, etc.

In this thesis, we focus on the interactive visualization of the real-world CFD datasets, and present a number of new methods or algorithms to address several related challenges in flow visualization. We strongly believe that the user interaction is a crucial part of an effective data analysis and visualization of large and complex datasets such as CFD datasets we use in this thesis. In order to demonstrate the use of the proposed techniques in this thesis, CFD domain experts reviews are also provided.

# Acknowledgements

This thesis wouldn't have come to fruition, without the support of a number of people who directly or indirectly influenced me during my four years PhD study in Swansea. I'm very thankful to all these people. Especially I would like to thank my supervisor Dr. Robert S. Laramee for his endless and tireless guidance, support, care, patience and encouragement during my four years abroad study. I appreciate everything that I learnt from him, including the amazing meeting minutes and the evil coding conventions [Lar07], which made my PhD experience colourful and stimulating. I thank Prof. Min Chen, my ex-second-supervisor, for always being interested in new ideas and for some valuable lessons on how to bridge ideas across different domains. I also thank EPSRC for supporting my PhD financially with grant EP/F002335/1.

I am very thankful to the whole Department of Computer Science. I also want to give special thanks to the Visual and Interactive Computing group; in particular Dr. Mark Jones for his support and encouragement, and my second supervisor Dr. Rita Borgo for her valuable advice on publications. I also would like to thank some research colleagues who have been a pleasure to work with: Tony Mcloughlin, Edward Grundy, Zhao Geng, Hui Fang, Dan Lipsa, Matthew Edmunds, Liam O'Reilly, David Chung, Ben Spencer and Ravi Kammaje. Special thanks go to Tony Mcloughlin, Edward Grundy, and Liam O'Reilly for proof reading part of the thesis.

I also want to thank co-authors of our published research papers: Dr. Guoning Chen, Dr. Nick Croft, Edward Grundy, Dr. Rami Malki, Dr. Ian Masters, and Prof. Chuck Hansen. Without their contributions and collaborations, my research outcomes would not be this fruitful.

My time in Swansea can't be this enjoyable without dear brothers and sisters from Swansea Chinese Christian Church. Thanks God for His protection and letting them be around with selfless love whenever I'm happy or sad.

My final words are dedicated to my family: my father Jianzu Peng and my mother Zhenfeng Luo, for their unconditional love and support although I am so far away from them; my beloved wife, Yajiao Zheng, who has been taking good care of me with her gentleness and patience and without whom my world would be far less wonderful.

# Publications

This thesis is based on the following publications:

- Zhenmin Peng, Edward Grundy, Robert S. Laramee, Guoning Chen, and Nick Croft, **Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach**, *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, to appear, 2011.

- Zhenmin Peng, Zhao Geng, Robert S. Laramee, Nick Croft, Rami Malki, Ian Masters, and Chuck Hansen, **Visualization of Flow Past a Marine Turbine: The Search for Sustainable Energy**, *Technical Report*, Department of Computer Science, University of Wales, Swansea, UK, December 2011. (Under review of IEEE TVCG).

- Zhenmin Peng, Zhao Geng, and Robert S.Laramee, **Design and Implementation of a System for Interactive High-Dimensional Vector Field Visualization**, *Technical Report*, Department of Computer Science, University of Wales, Swansea, UK, December 2011. (Under review of Software: Practice and Experience Journal).

- Zhenmin Peng and Robert S. Laramee, **Higher Dimensional Vector Field Visualization: a Survey**, *Theory and Practice of Computer Graphics (TPCG '09)*, pages 149-163, 17-19 June 2009, Cardiff, UK.

- Zhenmin Peng, Robert S. Laramee, Guoning Chen, and Eugene Zhang, **Glyph and Streamline Placement Algorithms for CFD Simulation Data**, *NAFEMS World Congress (NWC) Conference Proceedings*, the International Association for the Engineering Analysis Community, abstract on page 66 (Full proceedings on CDROM), June 16-19, 2009, Crete, Greece.

- Zhenmin Peng and Robert S. Laramee, **Vector Glyphs for Surfaces: A Fast and Simple Glyph Placement Algorithm for Adaptive Resolution Meshes** in *Proceedings of Vision, Modelling, and Visualization (VMV) 2008*, pages 61-70, 8-10 October 2008, Constance, Germany.

# Table of Contents

# List of Figures

# List of Tables

*Chapter 1*

# Introduction

**Contents**



"In the middle of every difficulty lies opportunity."

- Albert Einstein [1]

## 1.1 Preliminaries: What is Data Visualization?

As defined by the Merriam-Webster dictionary (online at www.merriam-webster.com), *visualization* is (1) the formation of mental visual images, (2) the act or process of interpreting in visual terms or of putting into visible form, (3) the process of making an internal organ visible by the introduction (as by swallowing, by an injection, or by an enema) of a radiopaque substance followed by radiography. In other words, *visualization*, often called data visualization, is a method which transforms the symbolic and abstract data into a visual representation. As the dictionary suggests (1) and (3), it has been widely applied for medical use. This process started as hand-drawn visualizations back in ancient times in order to help people understand and transfer knowledge. An example is illustrated in Figure 1.1, an ancient world map was drawn by Nicolaus Germanus in 1467. By using this people could have a better perception of the world.

[1] Albert Einstein (14 March 1879 - 18 April 1955) was a German-born theoretical physicist who developed the theory of general relativity, effecting a revolution in physics. For this achievement, Einstein is often regarded as the father of modern physics and one of the most prolific intellects in human history.

Figure 1.1: Ptolemy World Map from the 15th century codex of Nicolaus Germanus is an example of visualization applications back in ancient times [Ger67].

Computers brought the ability to collect, create, and store more information. Data can be from a wide variety of scientific sources such as simulation, modelling, or from non-scientific disciplines such as finance, marketing, and business. However, the quantity and complexity of the data exceeds our perceiving ability to derive knowledge from it. Based on this, visualization becomes more and more essential for us to process large amounts of this data. The main benefit of visualization is in mapping the data to an intuitive and effective visual form so that the user can gain a quicker and deeper understanding of the data. Moreover visualization enables us to see geometric structures and patterns that are difficult to infer from a vast collection of raw data. An example of this comparison is visualized in Figure 1.2.

Although the variety and number of data visualization approaches are increasing rapidly, data visualization can be mainly classified into two general categories according to the data properties it is associated with:

- Scientific Visualization: is a subset of data visualization which takes computational scientific data such as engineering simulation or modelling data, and renders the data with realistic volumes, surfaces, and illumination sources in 2D and 3D space. Example visualization techniques include volume visualization, medical visualization, flow visualization, etc [PNB02].
- Information Visualization: this subset assigns a suggestive visual representation to abstract data. Data sources are often from the financial or business domain. Visualization approaches such as histograms, pie charts, scatter plots, and parallel coordinates are frequently used for this subset [War04].

Besides the statement made by Frist Post, one of the most recognised visualization experts in this domain, “it has united the field of scientific and information visualization” in [PNB02],

there are conferences and/or workshops such as the IEEE Visualization conference that divide up papers into the visualization categories shown above. This thesis is concerned with the topic of *flow visualization*, a classic subset of scientific visualization. However, some information visualization techniques are also used for interactive flow information analysis and exploration.

## 1.2 Context of the Work

The focus of this thesis is on the interactive flow visualization of computational fluid dynamics data. It's helpful for the reader to have a look at some context such as what computational fluid dynamics data is and what flow visualization is, before reading the following chapters.

### 1.2.1 Computational Fluid Dynamics Data

Computational fluid dynamics (CFD) is an engineering discipline which applies numerical methods and algorithms to solve and analyse problems that involve fluid flows [And95]. The goals of CFD are to be able to simulate the interaction of fluid with objects and thus accurately predict the behaviour of the fluid flow. Presently, CFD has been increasingly used by many industries in order to both reduce manufacturing design cycles and provide an insight into existing engineering technologies so that they may be tested and improved. Examples of such industries include automotive, power generation, aerospace, chemical engineering and construction. Since computers are becoming even more powerful, the demand for more accu-

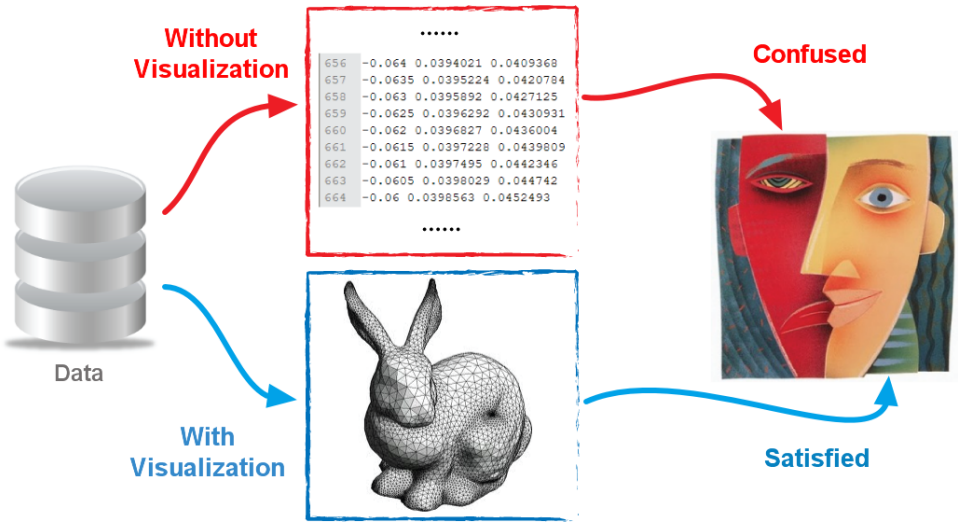


Figure 1.2: A visualization example shows the comparison of the data analysis and exploration of the Standford Bunny data set without visualization and with visualization. By analysing the vertex information from the data set without the visualization, the topology and construction information of the bunny mesh is abstract, difficult and confusing to understand. With the help of visualization, the rendered 3D constructed visual representation of the bunny data set is much easier to perceive and better for cognition.

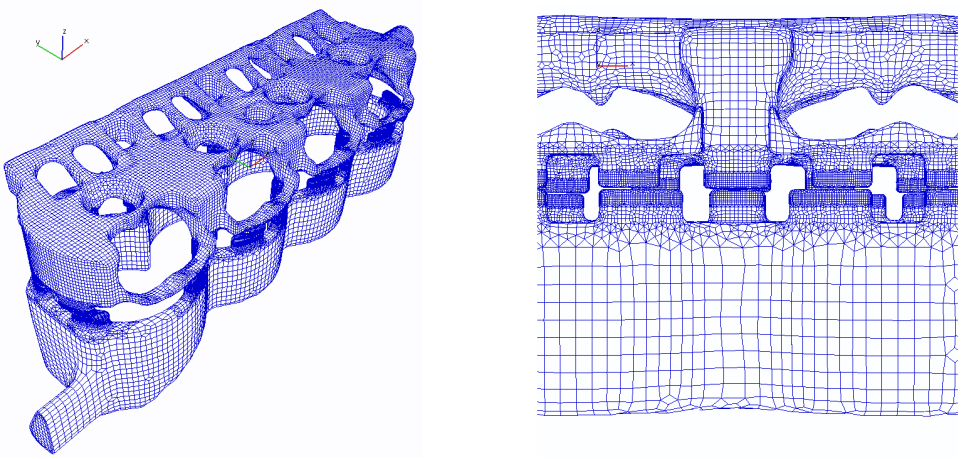

Figure 1.3: Figures demonstrate a cooling jacket CFD simulation dataset with unstructured adaptive resolution boundary grids.

rate CFD simulation results from CFD engineers has been escalated. The size, complexity and dimensionality of the CFD simulation data have dramatically increased in recent years. So the CFD simulation data can be mainly categorised by:

- spacial dimension: 2D planar flow, 2.5D boundary flow, and 3D volumetric flow.
- temporal dimension: static flow or time-dependant flow.
- mesh complexity: structured or unstructured.

An example of 2.5D static unstructured CFD simulation data is illustrated in Figure 1.3. How to visualize this kind of CFD simulation data is a big challenge for visualization researchers mainly due to the computational complexity in handling these unstructured meshes. The flow datasets used in this thesis are all real-world, 2.5D/3D, static, unstructured CFD simulation data.

### 1.2.2 Flow Visualization

Flow visualization visualizes vector data which has a magnitude and direction. The vector data are often obtained from the study of fluid flow or a derivative quantity. In order to illustrate the basis of flow visualization, we look at the mathematical representation of the flow field as the starting point. If the velocity is represented by $\mathbf{v} = d\mathbf{x}/dt$ and we track a massless particle through the vector field, the particle displacement can be described by:

$$d\mathbf{x} = \mathbf{v} \cdot dt \tag{1.1}$$

Since the flow field data we use is from CFD simulation, a numerical integration technique is needed to evaluate equation 1.1. We can solve it with Euler approximation:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{v}_i \cdot \Delta t \tag{1.2}$$

The position of the massless particle at time $i+1$ is a function of the previous position and the previous velocity times an incremental time step $\Delta t$. This is the simplest and perhaps the most common approximation method used to illustrate the flow field. However, if the Euler integration is not accurate enough in a particular case, more accurate approaches such as the second order Runge-Kutta integrator [Cd80] can be used:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \frac{\Delta t}{2} \cdot (\mathbf{v}_i + \mathbf{v}_{i+1}) \tag{1.3}$$

Higher order integrators such as the fourth order Runge-Kutta are also available if even more accurate results are needed.

Flow visualization is one of the classic subsets of visualization, covering a rich variety of industry applications, from the automotive industry, aerodynamics, to weather simulation, meteorology, climate modelling, and medical visualization. These flow visualization techniques can be classified into four general categories: direct, geometric, texture-based and feature-based flow visualization [PVH+03]. These are illustrated in Figure 1.4 and we give brief descriptions of these four categories:

- *Direct Flow Visualization:* Direct visualizations are the simplest solutions to intuitively visualize flow fields. Common approaches include placing arrow glyphs at each sample point to depict the associated vector field and colour mapping. Direct techniques are able to make flow visualization universally and intuitively understandable with less computational cost. So it has been used in many applications to guarantee the interactive analysis of the CFD flow field. However, direct techniques can suffer from a lack of visual coherence. When applied to 3D volumetric flow data, they might suffer from visual complexity and occlusion.

- *Geometric Flow Visualization:* In order to obtain a coherent representation of the vector field, integration-based geometric techniques are applied. A typical example involves defining a set of seeding points, then computing trajectories (e.g. streamlines) from these points through the flow by using various integrator such as equations 1.2 and 1.3, and finally, rendering resulting geometric objects from these trajectories. Due to its comparatively accurate and coherent result, geometric flow visualization has been widely used to visualize almost all types of CFD data. However, visual clutter and occlusion can stem if a poor seeding strategy is employed in 3D. The computational complexity caused by 2.5D and 3D unstructured CFD meshes is also the challenge this visualization needs to address.

- *Texture-based Flow Visualization:* This approach renders with convolved textures to reflect the local properties of the vector field. This category of techniques provides a dense and coherent visualization, even in areas of complicated flow. So it has been widely applied to visualize the 2D and 2.5D flow which is even based on unstructured meshes.

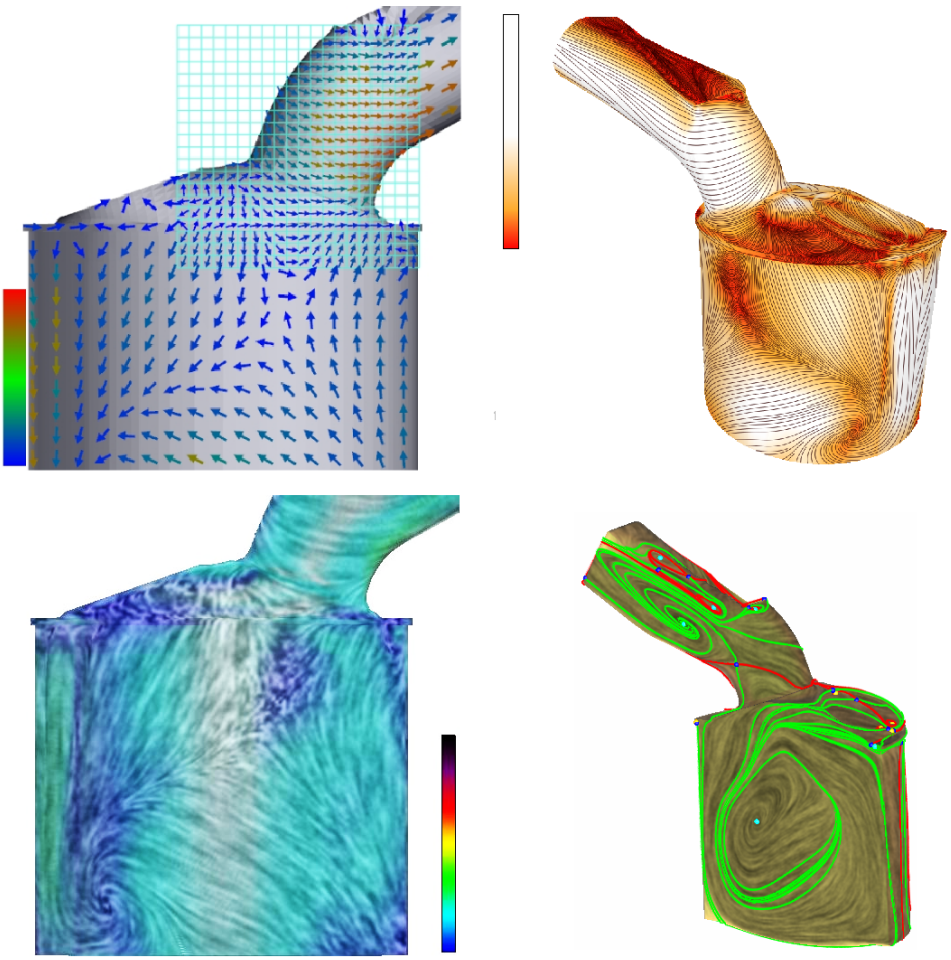

Figure 1.4: The example of the flow at the surface of the gas engine CFD simulation to help illustrate the flow visualization classification: (top-left) direct visualization by the use of arrow glyphs, (top-right) geometric visualization by the use of streamlines [SLCZ09], (bottom-left) texture-based by the use of LIC [LJH03], and (bottom-right) feature-based visualization based on topology extraction [CLZ07].

However, they can also suffer from visual complexity and occlusion when applied to 3D volumetric flow data.

- *Feature-based Flow Visualization:* These higher-level techniques extract subsets of data with features deemed interesting by users before visualization. Hence the visualization is then based on these extracted subsets rather than the whole dataset. This may make the visualization more efficient and suggestive. However, the complexity and computational cost of feature extraction may be high, especially when it is applied to visualize the flow based on 3D unstructured CFD meshes.

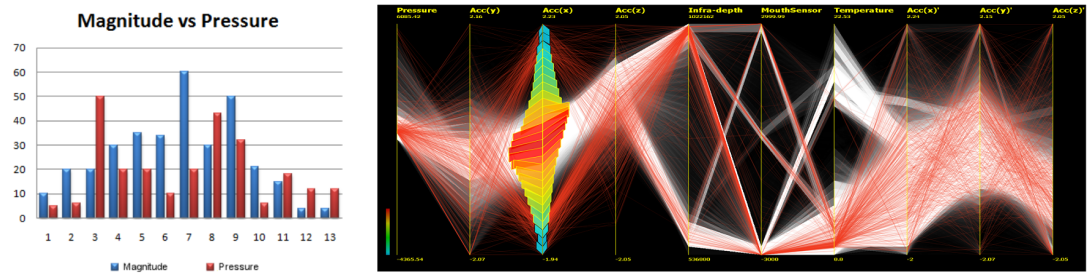


Figure 1.5: (left) A histogram example is used to visualize and analyze the distribution of the velocity magnitude and the pressure information. (right) A PCP visualization result from Geng et al. [GPS+11] demonstrates the ability to identify the correlations and clusters from high-dimensional data.

### 1.2.3 Multi-dimensional Visualization

CFD flow datasets can have many simulation attributes such as mesh resolution, pressure, temperature, density [PGL+11] [PGSC10]. Visualizing these attributes simultaneously is a big challenge for traditional flow visualization approaches. Based on this fact, multi-dimensional visualization, a classic sub-branch of information visualization, is also used to provide an effective visualization and exploration of the high-dimensional CFD flow data. There are numerous multi-dimensional visualization techniques. Here we focus on histogram techniques and parallel coordinate plot (PCP) which have been applied in this thesis. This is because histogram techniques can obtain an overview of the data distribution efficiently without plotting the whole dataset while PCP can identify the correlations between different simulation attributes based on the user selection from histogram table. So this combination of overview and detailed view is applied to analyse the multi-dimensional CFD flow data. Examples are illustrated in Figure 1.5. Since covering all the literature is beyond the scope of this thesis, we recommend the reader consult [War04].

- *Histogram:* This technique provides a graphical representation showing the distribution of data. A histogram consists of adjacent rectangles over discrete intervals. The height of a rectangle is associated with the frequency of the interval. By plotting data using histogram, an overview of the data distribution can be quickly obtained. For multi-dimensional CFD data, multiple histograms can be rendered simultaneously and each histogram presents the distribution of a kind of simulation attributes from the data. Examples of this kind include the histogram table and the polar histogram which are applied in [PGSC10]. However, the correlation of simulation attributes of the CFD simulation dataset is hard to perceive from multiple histograms.

- *Parallel Coordinate Plot (PCP):* PCP is designed for visualizing and analysing high-dimensional data [ID87]. If we imagine an $n$-dimensional space is depicted with $n$ parallel lines and a point in the $n$-dimensional space is represented as a polyline whose vertices are on the parallel axes, then the position of each vertex on the $i$-th axis corresponds to the value on $i$-th coordinate of the point. For multi-dimensional CFD data, the correlation of simulation attributes of the CFD simulation dataset is deemed interesting

and important by CFD engineers. Identification of these correlations and clusters from the multi-dimensional data is the strength of PCP. However, visual clutter and occlusion can stem if the data is very large.

## 1.3 Overview

This section contains summaries for the following main chapters. For continuity, these summaries are arranged to start with corresponding chapter headings.

**Chapter 2: Higher Dimensional Vector Field Visualization: A Survey**

This chapter presents a literature survey with focus on higher-spatial dimensional flow visualization techniques based on the presumption that little work remains for the case of two-dimensional flow whereas many challenges still remain for the cases of 2.5D and 3D domains. The survey provides an up-to-date review of the state-of-the-art of flow visualization in higher dimensions. The reader is provided with a high-level overview of research in the field highlighting both solved and unsolved problems in this rapidly evolving direction of research. This chapter also serves as an introduction step to the main body of this thesis.

**Chapter 3: Vector Glyphs for Surfaces: A Fast and Simple Glyph Placement Algorithm for Adaptive Resolution Meshes**

Chapter 3 presents image-based glyph placement algorithm in order to investigate and visualize flow data based on unstructured, adaptive resolution boundary meshes from CFD. Glyphs can be automatically placed at evenly-spaced intervals. And the user can interactively control the spatial resolution of the glyph placement and their precise location. The implementation supports multiple representations of the flow - some optimised for speed others for accuracy. The result is a tool that provides engineers with a fast and intuitive overview of their CFD simulation results.

**Chapter 4: Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach**

While the previous chapter focuses on the direct flow visualization for CFD simulation data, in Chapter 4 we extend the work to geometric and feature-based visualization. We present a novel, robust, and automatic vector field clustering algorithm that produces intuitive and insightful images of vector fields on large, unstructured, and adaptive resolution boundary meshes from CFD. The most suggestive and important information contained in the meshes and vector fields is preserved and visualized by glyphs or streamlines while less important areas are simplified in the visualization. The motivation behind the approach is the fact that CFD engineers may increase the resolution of model meshes according to importance. A series of synthetic and complex, real-world CFD meshes are tested to demonstrate the clustering algorithm results.

**Chapter 5: Visualization of Flow Past a Marine Turbine: the Information-Assisted Search for Sustainable Energy**

Interest in renewable, green, and sustainable energy has risen sharply in recent years and in Chapter 5 we switch our research focus from the automotive industry to marine turbine design. We develop, explore and present customised visualization techniques in order to help engineers gain a fast overview and intuitive insight into the flow past the marine turbine. The system exploits multiple-coordinated information-assisted views of the CFD simulation data. The application utilises flow visualization and information visualization techniques. To demonstrate the usage of our system, a selection of specialised case scenarios designed to answer the core questions brought out by engineers is described. We also report feedback on our system from CFD experts researching marine turbine simulations.

**Chapter 6: Design and Implementation of a System for Interactive High-Dimensional Vector Field Visualization**

Chapter 6 describes some details about the design and implementation of a generic flow visualization software application framework which incorporates and integrates a selection of scientific and information visualization techniques in order to provide the user an intuitive solution for effective visualization of the multi-dimensional CFD flow simulation data. Our framework provides direct, feature-based and geometric flow visualization techniques and supports information visualization approaches, such as a tabular histogram, velocity histogram, and parallel coordinate plot.

## 1.4 Contributions

The main contributions of this thesis are respectively:

- Chapter 2 provides an up-to-date overview of the state-of-the-art of the most recent developments in the flow visualization in higher spatial dimensions (2.5D, 3D and 4D). It is the first survey to include vector-field clustering approaches for flow visualization. Additionally a table-based classification highlights both mature areas and immature areas in flow visualization.
- In Chapter 3, we describe a fast and efficient image-based glyph visualization technique that (1) automatically places glyphs at evenly-spaced intervals, (2) enables the user to interactively control the spatial resolution of the glyph placement and their precise location, and (3) supports multi-resolution visualization of the flow at surfaces.
- A novel clustering algorithm which couples the properties of the vector field and mesh model from the large, adaptive resolution CFD data into a unified clustering distance measure is presented in Chapter 4. The cluster hierarchy is generated automatically according to the importance of the underlying mesh property and the vector fields. More detailed areas are emphasised while less important or simpler ones are simplified in a

| Techniques \ Chapters | Chapter 2 | Chapter 3 | Chapter 4 | Chapter 5 | Chapter 6 |
|---|---|---|---|---|---|
| Direct FlowVis | ■ | ■ | ■ | ■ | ■ |
| Geometric FlowVis | ■ | ■ | ■ | ■ | ■ |
| Texture-based FlowVis | ■ | ■ | ■ | ■ | ■ |
| Feature-based FlowVis | ■ | ■ | ■ | ■ | ■ |
| Histogram | ■ | ■ | ■ | ■ | ■ |
| Parallel Coordinate Plot | ■ | ■ | ■ | ■ | ■ |
| Image-based Visualization | ■ | ■ | ■ | ■ | ■ |
| Multi-threading | ■ | ■ | ■ | ■ | ■ |
| Selection | ■ | ■ | ■ | ■ | ■ |
| Interaction | ■ | ■ | ■ | ■ | ■ |

Table 1.1: The table illustrates the relevance of each main chapter. The colour is mapped to the relevance: ■ high relevance; ■ adequate relevance; ■ little relevance;

single image. Novel glyph visualizations of cluster attributes including $\theta$-range and $|\mathbf{v}|$-range glyphs are introduced.

- We design a novel application that exploits multiple-coordinated views for interactive visualization of marine turbine simulation data in Chapter 5. Novel information visualization views including a polar histogram and a histogram table are introduced. We experiment with novel knowledge-assisted distortion techniques that provide more visual detail in regions of interest without losing visualization coherency. We describe information-based and knowledge-assisted streamline seeding to investigate the behaviour of the flow past the marine turbine. Novel derived data used to evaluate blade design is computed and visualized based on swirling flow behaviour. We report feedback from CFD experts researching the simulation of flow past the marine turbine.

- Chapter 6 describes the design and implementation of generic flow visualization framework which includes following benefits: (1) The framework handles versatile real-world, unstructured 2.5D, 3D, and $n$D CFD simulation data. (2) Direct, geometric and feature-based flow visualization techniques are integrated in order to support the CFD engineer with intuitive and rich visualizations for effective visual analysis. (3) Information visualization approaches, such as tabular histogram, velocity histogram, and parallel coordinate plot (PCP), are incorporated to enable engineers to gain a deep access to the simulation data and thus focus on parts they deem interesting. (4) Smooth and efficient user interaction is ensured by our multi-threading platform, even when large data sets are processed. (5) Our flow visualization framework is platform independent in terms of both hardware and software.

Table 1.1 is designed to summarise the area of contribution of the main chapters. Colour is mapped to the level of relevance.

*Chapter 2*

# Higher Dimensional Vector Field Visualization: A Survey

**Contents**



> "Learning without thought is labor lost; thought without learning is perilous."
>
> - Confucius [2]

THIS chapter presents a survey of the most up-to-date higher-spatial dimensional (2.5D and 3D) flow visualization techniques based on the presumption that little work remains for the case of 2D flow visualization whereas many challenges still remain for the cases of 2.5D and 3D domains. The high-level overview of research in the field highlighting both solved and unsolved problems inspired our research direction. For example, the ideas for the vector glyph placement algorithm in Chapter 3 and the mesh-driven vector field clustering algorithm in Chapter 4 are based on the fact that no related work has been done for 2.5D flow field visualization of direct or vector filed clustering category while the domain needs effective

[2] Marcus Aurelius (551 BC - 479 BC), was a Chinese thinker and social philosopher of the Spring and Autumn Period.

methods to visualize the flow based on this complex and unstructured 2.5D meshes. This chapter has also been published as a survey paper in TPCG'09 [3].

## 2.1 Introduction

Over the last two decades, *vector field visualization* has developed very rapidly. Its applications range widely from the automotive industry to medicine. As a fascinating sub-branch of scientific visualization, vector field visualization provides solutions which enable users to investigate and analyse some critical features and characteristics of flow phenomena. Although the challenge of two-dimensional flow visualization is deemed virtually solved as a result of the effort invested in this problem, the higher-spatial dimensional flow visualization, (e.g. the visualization on surfaces in 3D (2.5D) and for volumetric flow (3D)), still has many challenges which need to be addressed. We present a survey of higher-spatial dimensional flow visualization techniques in order to provide an up-to-date overview of the most recent developments in flow visualization in higher dimensions highlighting both solved and unsolved problems in this rapidly evolving direction of research.

### 2.1.1 Classification

According to the different needs of the users, there are different approaches to flow visualization. The classification we use here is the four general categories: *direct, geometric, texture-based* and *feature-based* ( [PVH$^+$02][PVH$^+$03][LHD$^+$04]) which are described with details in Section 1.2.2 where Figure 1.4 demonstrates typical examples. So we would not detail them here.

### 2.1.2 Dimensions

A general solution to vector field visualization is not possible for all application due to different interests from users and various characteristics across datasets. According to data characteristics, flow visualization techniques can vary considerably: 2D vector field visualization techniques focus on the flow in planar 2D domains; 2.5D flow visualization techniques are for boundary flows in 3D; and 3D flow visualization techniques are solutions for volumetric 3D flow. Although 3D techniques can offer true 3D flow visualization, the computational cost for rendering is high. Hence, compromises are usually needed to trade off visibility for completeness.

In addition to the spatial dimensions described above, the temporal dimension is of great importance in categorising flow visualization. Steady-state vector field visualization visualizes the instantaneous or static flow representing a single time step while unsteady vector field visualization visualizes the transient or unsteady flow represented by several time steps.

[3] Published as: Zhenmin Peng and Robert S. Laramee, **Higher Dimensional Vector Field Visualization: a Survey**, *Theory and Practice of Computer Graphics (TPCG '09)*, pages 149-163, 17-19 June 2009, Cardiff, UK.

| | | Direct | Vector-field Clustering | Texture-based | Geometric |
|---|---|---|---|---|---|
| 2.5D Flow Data | Steady | | | [LvW95] [vW91]<br>[LPPW95] [LvW95]<br>[BSH97] [SH95] | [vW93a] [SvW91]<br>[MHHI98] [TB96]<br>[SLCZ09] [JL97] |
| | Unsteady | | | [LJH03] [vW02]<br>[vW03] [vW02]<br>[LvWJH04] [LJH03] [vW03]<br>[LSH04] [LJH03]<br>[LGSH06] [GTS+04] [LvWJH04] | |
| 3D Flow Data | Steady | [Dov95] [CM92]<br>[HMK95]<br>[Lar03] | [TvW99]<br>[HWHJ99]<br>[GPR+01] [CH58] | [RSHTE99] [CL93]<br>[GEO02] [JEH01] | [SVL91]<br>[Hul92]<br>[MBC93] [ST90]<br>[vW93b]<br>[BHR+94] [WMW86]<br>[LPSW96] [TK93]<br>[FG98] [JL97] [WG97]<br>[SBH+01] [Hul92]<br>[Gel01] [vW93b] [KM96]<br>[XZC04] [vW93b]<br>[VP04] [LPSW96]<br>[YKP05] [VKP00]<br>[CCK07] [VKP00] [JL97]<br>[LS07] [vW02] [JL97]<br>[UKSE08] [MCG94] |
| | Unsteady | | [GPR+04] [UA01] | [HA04] [CL93]<br>[WE04a] [TvW03]<br>[WSE05] [MB95]<br>[HE06] [Sun03] [WEE03]<br>[WSE07] [WSE05] | [BLM95] [MBC93]<br>[STWE07] [ABCO+01] [CL93]<br>[STW+08] [DGH03]<br>[GKT+08]<br>[vFWTS08] |

Table 2.1: An overview and classification of higher spatial dimensional flow visualization. The survey is grouped based on the classification of flow visualization techniques and the dimensionality of the flow data domain. For feature-based flow visualization, we focus on the vector field clustering techniques in this survey. Each group is then subdivided by steady or unsteady flow solutions. The entries of each sub-group are placed in chronological order. References in sub-script are 2D predecessors

In many cases, further data dimensions represent other data attributes, such as temperature, pressure, or vorticity in addition to above spatial and temporal dimensions. The data of this type is also associated with the terms multivariate and multi-field data. Flow visualization may also take these dimensions into account, e.g., by using color or isosurface extraction.

### 2.1.3 Data Sources

Before outlining some of the most important techniques, a short overview of the data sources is discussed. We focus on visualization of data from computational fluid dynamics (CFD) simulation which deals with data that exhibits temporal dynamics like results from (a) flow simulation (e.g., the simulation of fluid flow through a turbine), (b) flow measurements (possibly acquired through laser-based technology), or (c) analytic models of flows (e.g., dynamical systems, given as set of differential equations). In the most cases, the *velocity* information which is encoded as a set of vectors in a flow dataset represents the focus. To obtain the velocity information from the flow simulations, partial differential equations (PDEs), such as the Navier Stokes, Euler, or Advection Diffusion equations [VM96], are often used. The finite

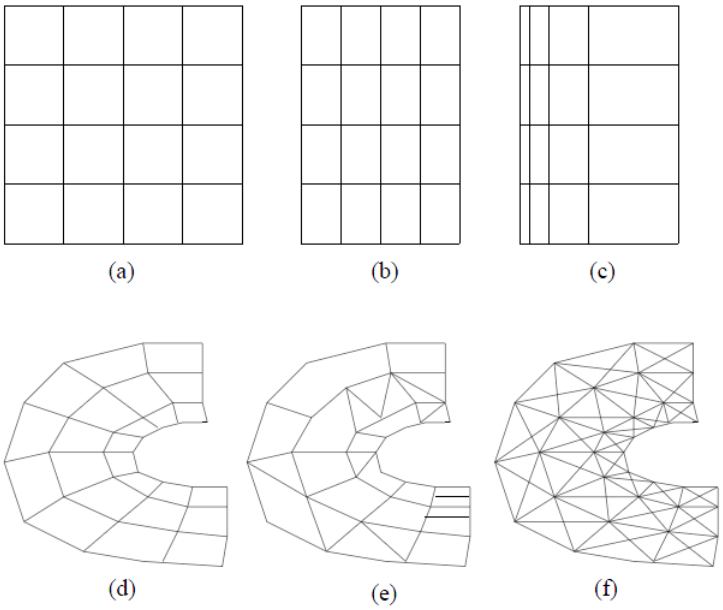


Figure 2.1: Grids used in CFD simulation - (a) Cartesian, (b) regular, (c) general rectilinear, (d) structured or curvilinear, (c) unstructured, and (f) unstructured triangular [Lar03] [WEE03]

volume (FV) and the finite element (FE) analysis, which subdivide the domain into small elements like hexahedral or tetrahedral cells, are used to define the velocity on the computation grid in physical space: unstructured for FE and structured curvilinear for FV solutions. The defined velocity samples $\mathbf{v}_i$ are often stored at vertices of a certain type of grid. Grid types range from simple rectilinear or Cartesian grids to curvilinear grids to complex unstructured grids (see Figure 2.1). In this thesis the CFD data we use is based on 2.5D or 3D unstructured grids such as the mesh shown in Figure 1.3. The challenges stem from these types of grids in higher dimensions are laid on difficulties and complexity in designing algorithms for general point location and flow reconstruction. A lot of research work discussed in this chapter is trying to address these challenges.

### 2.1.4 Overview

We organise the following research literature based on the above classification and dimensions, which can be seen in Table 2.1. The spatial dimensionality of the data domain is applied to classify literature at the first level. Since we focus on higher spatial dimensional domain, 2.5D and 3D are only taken into account. The literature is then subdivided into direct, geometric, texture-based and vector field clustering visualization according to the criteria of four general categories. It is worth pointing out that we concentrate on the vector field clustering visualization rather than the whole feature-based visualization domain. Temporal dimensionality is also used to further classify literature. Most of the literature in each sub-category is presented in chronological order. Related previous work is indicated by sub-scripts in Table 2.1.

The contributions of this chapter are:

- A survey which focuses on the most recent developments in flow visualization in higher spacial dimensions only, 2.5D and 3D.
- A survey to include vector-field clustering approaches as feature-based flow visualization technique for vector field visualization.
- An intuitive table-based overview highlights both mature areas and immature areas in vector field visualization and also indicate previous related work in subscripts.
- Related research is compared and organised according to the classification of flow visualization techniques and the dimensionality of the flow data domain.

The chapter is structured as follows: in Section 2.2, we will discuss some related techniques for direct flow visualisation in higher dimensions. In Section 2.3 geometric flow visualization techniques are discussed based on several different types of geometry primitives. Section 2.4 discusses texture-based flow visualization techniques for higher dimensional flow in three main groups. In Section 2.5 the latest developments in vector field clustering technique as a feature-based flow visualization approach are given. Finally, in Section 2.6 some conclusions and further prospects are presented.

## 2.2 Direct Vector Field Visualization

We start by describing direct vector field visualization methods which are the simplest solutions to intuitively visualize flow fields. This kind of approaches deliver an interactive and quick visual analysis of the flow field with less computational cost. Colour mapping, slicing and glyphs are widely applied in many visualization toolkits. Ward [War02] states that glyph-based visualization has been widely used to convey various information simultaneously by employing intuitive graphs to depict corresponding various variables from abstract data sets. However, direct techniques can suffer from a lack of visual coherence. When applied to 3D flow data, they generally suffer from visual complexity and occlusion.

In order to visualize vector fields on unstructured 3D grids, Dovery [Dov95] extends Crawfis and Max's method [CM92] from regular to curvilinear and unstructured grids by using physical space and parameter space resampling methods. During the physical space resampling, the vector field is linearly interpolated at each sample point, then the physical coordinates of the point are calculated, and lastly related oriented glyphs (plots) are projected from back to front. Although this ensures that sample points are uniformly distributed, physical space resampling is computationally expensive. To address this problem, it can be preferable to resample to parameter space instead. At first, random points are directly generated in parametric space with an area-weighted distribution. Then a relatively accurate and dense resampling can be approximated by mapping the parametrised coordinate to physical coordinate grid points. Vector field visualization on arbitrary 3D surfaces can be efficiently achieved with parameter space

resampling. However, without this pre-processing stage efficient glyph placement on 2.5D unstructured meshes still remains as a challenge for the community.

To address performance issues, Hong et al. [HMK95] use volume rendered vector glyphs which are generated from pre-voxelized icon templates to describe regular, structured vector fields in 3D space. Incremental image updates which re-compute only those pixels on the image plane affected by user input make visualization of the scalar and vector field faster and more interactive. Recently Laramee [Lar03] describes an object-space approach using resampling and vector glyph placement for slices through unstructured, 3D CFD meshes. This method is able to resample any unstructured grid onto any structured grid based on which glyphs are placed. This method is efficient and does not rely on any pre-processing of the data. However, it can't handle 2.5D unstructured meshes.

As we can see direct flow visualization in higher dimensions has received comparatively little attention. This could be partially due to that the challenges for direct vector field visualization on the structured meshes are almost done. However, the same can not said to the flow visualization based on the 2.5D unstructured meshes since there was no direct flow visualization method for surface-based (2.5D) flow field visualization. This is probably due to the difficulties in placing the glyphs on the complex boundary meshes and perceptual problems like visual complexity and occlusion. Attempting to address these challenges, we present an image-based glyph placement technique which conceptually similar to [Lar03] but raises the spatial dimensionality to surfaces (2.5D). Details about this technique can be found in the next chapter.

## 2.3 Geometric Vector Field Visualization

Although the direct flow visualization can quickly depict underlying flow, it suffers from the noncontinuous representation which can not reflect the underlying coherent flow patten. In order to gain a coherent representation of the vector field, integration-based geometric techniques are applied. A typical example involves defining a set of seeding points, then computing trajectories from these points through the flow, and finally, rendering resulting geometric objects from these trajectories. According to the geometric characteristics of these objects, techniques can be mainly categorised into two groups: *line-based* (e.g. streamlines) and *surface-based* (e.g. stream surfaces). The equation (1.2) or (1.3) is normally used to solve the numerical integration for these geometric objects. So as the most traditional and wildly applied flow visualization techniques, lots of effort has been put on the geometric flow visualization. In what follows, we discuss geometric flow visualization techniques in both 2.5D and 3D.

### 2.3.1 2.5D Steady Vector Fields

As we mentioned most vector field domains consist of either 2 or 3 spatial dimensions. The vector filed based on the boundary meshes (2.5D) is a special 3D case. It only focuses on the boundary flow rather than the volumetric flow. To visualize the description of surface-based flow, streamlines, *line-based* trajectories instantaneously tangent to the velocity vector of the

flow, are often used to show the direction of a local fluid element. As the boundary meshes are normally unstructured and it's 3D rather than 2D, the complexity of streamline seeding algorithms increases. The challenges are mainly due to the effort to minimize visual clutter and occlusion caused by streamline placement and, to the extra complexity of another spatial dimension and the unstructured mesh characteristics.

To address the visual clutter and collusion, Van Wijk [SvW91] extends his previous work with surface particles [vW93a]. The basic geometries of streamlines, stream-surfaces and stream-tubes are enhanced by adding massless particles with a small finite size (facets) to the surface and visualizing their behaviour. Several features are shown including: an improved shading model used to decrease jagged edges and strobing artifacts, implementation of Gaussian filters to avoid spatial and temporal artifacts, and a scan-conversion algorithm to improve calculation efficiency. With this improved rendering method, high-quality images and animations are feasible at a reasonable computational cost. The focus of the research is on how to effectively render particles on surfaces.

In order to generate evenly distributed streamlines on 3D curvilinear surfaces, Mao et al. present an image-guided streamline placement technique [MHHI98]. Essentially, this approach extends Turk and Banks' 2D image-guided streamline placement method [TB96] to 3D curvilinear grid surfaces. First, vectors on the 3D curvilinear surface are mapped into computational space. Second, streamlines are generated with desired density distribution by using the extended Turk and Banks' 2D algorithm. And last, streamlines generated in the computational space are mapped back to the 3D surface. Additionally, one of the most interesting parts of this paper is the new energy function which is applied with the position ellipse sampling technique. With help of this energy function, the density of streamlines in the computational space is locally adapted according to the physical space grid density in order to address the mapping distortion caused by non-uniform grid density in the curvilinear grid.

Recently, Spencer et al. [SLCZ09] present a novel, automatic streamline algorithm to visualize the vector field on surfaces in 3D space. This paper describes an image-based algorithm to generate evenly-spaced streamlines fast, simply, and efficiently for any general surface-based vector field. Conceptually, this algorithm can be viewed as an extension of Jobard and Lefer's 2D method [JL97] to 2.5D. In the first stage, the vector field is projected onto the image plane. Based on this operation streamlines can be seeded and integrated. The next step divides the image with a user-defined grid and sequentially attempts to seed a streamline at the centre of each cell. If the z-depth of the seed is non-zero and no streamlines lie closer than $d_{sep}$ to the seed point. A new streamline is traced. A vector field-based seeding method is also used - whenever a seed point along the length of the grid-based streamline curve at regular interval and no streamlines lie closer than $d_{sep}$ to the seed point, a new streamline is traced from it and pushed onto a queue. This process repeats until the queue is empty. The algorithm is enhanced by a proximity testing method which decides when to terminate the particle tracer and an edge detection which ensures that the streamline is seeded in a continuous, bounded area. See Figure 2.2 for more detail. However this technique is image-based which suffers from visual inconsistency when view is changed.

As we can see in this section, most of the research effort is on streamline seeding strategies for visualizing 2.5D steady boundary flow to address the visual clutter and occlusion. Noteworthy is that the effective solution to the challenge of the streamline seeding on surfaces is from Spencer et al. [SLCZ09]. This is the only solution of its kind that handles both general surfaces and unstructured, complex and adaptive resolution meshes. It is also fast enough to support exploration through user-interaction. So this technique is adopted for visualizing boundary flow in our projects in this these. Examples can be found in following chapters. However, other visualization techniques for visualizing 2.5D unsteady flow, such as stream-surfaces, pathline and streakline seeding on surfaces, remains a big challenge since there is no related research work done yet. It might be partially due to the computational and perceptual complexity.

### 2.3.2 Geometric Visualization of 3D Vector Fields

In this section we discuss geometric techniques used to visualize 3D volumetric vector fields. As we mentioned geometric flow visualization techniques can be mainly categorised into two groups: *line-based* and *surface-based*. *Line-based* geometries, such as streamlines we just described in previous section are still wildly applied in this domain for visualization of 3D steady vector field. Streakline, a line is traced by a set of particles that have previously passed through a unique point in the domain [NHM97], is used for visualizing time-dependent 3D flow. *Surface-based* integral objects are also applied to provide greater perceptual information

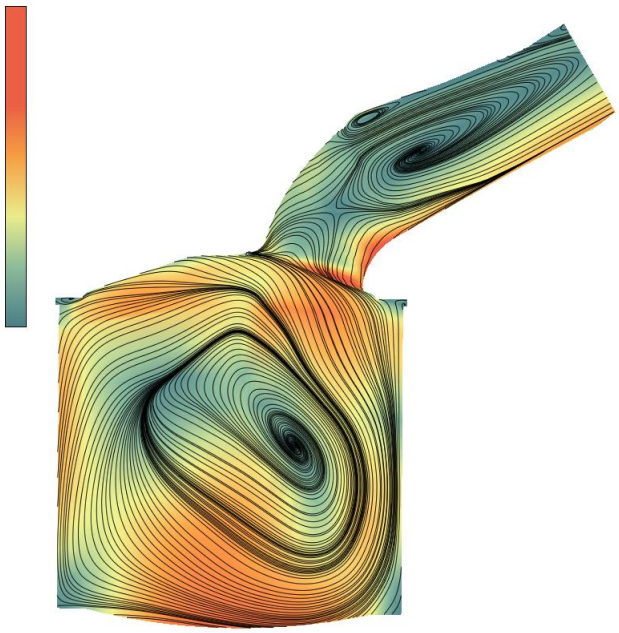

Figure 2.2: Visualization of flow through the gas engine simulation. By setting $d_{test}$ to $0.05 \times d_{sep}$, streamlines overlay highlighting loops within the flow field. [SLCZ09]. Colour is mapping to velocity magnitude.

over line primitives since surfaces suffer less from visual complexity when compared to line-based geometries as many lines can be replaced by a single surface, providing more spatial coherency. The typical surface-based examples are stream-surfaces for steady flow and streak-surfaces for unsteady flow. So the following related literature which has been grouped based on the temporal dimensions is further subdivided to *line-based* and *surface-based* groups.

#### 2.3.2.1 Steady Vector Fields

For *line-based* techniques challenges are perceptual issues. Naively rendering too many or too few lines won't bring an intuitive visualization but results in clutter, occlusions or missing leading result. So how to seed the streamline effectively and efficiently is a hot research topic in the 3D steady flow domain.

Fuhrmann and Gröller [FG98] present a new method to effectively visualize 3D steady flow fields in order to address the problem of the occlusion and perception of distant details in 3D flow fields. Conceptually, this method draws upon the evenly-spaced streamline placement algorithm [JL97] and a texture based technique - FROLIC [WG97]. In order to solve the problem of preserving distance details when visualizing vector fields in a 3D domain, dashtubes - the animated, opacity-mapped streamlines, are automatically placed evenly in 3D space. A texture mapping technique is then applied to maintain the detail along the streamline at a constant level although the velocity of flow changes. Additionally, magic lenses and magic boxes are implemented as interactive tools for users to visualize detailed features of areas deemed interesting by users. However according to the user study users find the magic boxes is hard to use.

Ye et al. [YKP05] present a strategy for streamlines placement in 3D flow fields. With this algorithm, intuitive visualization can be obtained from essential flow patterns captured in sufficient coverage of the field after reducing clutter. This approach is a 3D extension of the 2D flow guided approach [VKP00]. Firstly, critical points need to be extracted in order to distinguish the interesting areas with important flow patterns. Later, different seeding templates are applied around the vicinity of these critical points. Due to the variability of the flow pattern, these seeding templates can transform based on how far one critical point is to another. A unique template-based hybrid map is used to handle this transformation. Then, Poisson seeding is introduced to add streamlines into regions which have sparse coverage. Finally, in order to refine streamlines in 3D space, a filter is applied based on the streamlines' geometric and spatial properties to filter out the streamlines which are too short, with small winding angles, similar to each other, or with very high winding angles.

An adaptive method for streamline placement on steady vector fields in 2D and 3D is presented by Chen et al. [CCK07]. In order to achieve a good balance between feature-based and density-based streamline placement, this new approach draws upon both feature-based streamline placement methods like [VKP00] [YKP05], and density-based ones like [JL97] [MT+03] [LS07]. A similarity metric is applied to measure the similarity among streamlines. This metric consists of Euclidean distance and similarity of shape and direction. Guided by this metric, streamlines are traced to accentuate regions of geometric interest rather than explicitly

enumerating the whole vector field. Additionally, an error metric for streamline representations is introduced to evaluate the generated result when compared to the original vector field. With this metric, comparison and evaluation of various streamline representations can be easily presented in a quantitative way. But only structured meshes are used in this paper.

In order to better display 3D streamlines and reduce visual cluttering in result images, Li et al. [LS07] present an image-based method for streamline seeding and generation. This is the first algorithm of its kind to use an image based approach for the seeding of 3D streamlines. First, 2D depth maps are generated. After, seeds are selected based on the 2D depth maps and then unprojected back to 3D object space before evenly-distributed streamline integrations are generated. By carefully separating streamlines in image space, visual cluttering after the projection from 3D to 2D image can be effectively reduced. Visual clarity can be also enhanced by controlling the density and rendering styles of streamlines according to different user criteria and needs. Additionally, this method provides level of detail rendering, depth peeling, and stylised rendering of streamlines to gain better perception of 3D flow fields. However, since this is image-based approach visual inconsistency is an issue. Also only structured grids are used in this method.

In order to visualize 3D flow in the vicinity of boundary and feature surfaces without losing 3D information due to the inherent projection, Üffinger et al. present a combination of 2D dense texture-based and 3D streamlines visualization [UKSE08]. Conceptually, the idea of this implementation is based on techniques proposed by Max et al. [MCG94]. First, an image-based seeding strategy is applied to obtain an importance-driven distribution of streamline seed points on the surface in object space. Then, a new algorithm for generating streamlines using geometry shaders on the GPU is employed to efficiently trace streamlines based on seed points in object space, thus interactive exploration is enabled. The user can relocate seed points interactively to emphasise regions of interest. But this method is not able to handle large datasets due to the limitation of GPU memory. Additionally only structured grids are supported by this method.

*Surface-based* geometries as the extension to line-based primitives are applied to deliver better coherent representation of 3D flow. Challenges for this type of techniques lies on the perceptual and performance issues. How to deal with converge or diverge flow when surfaces are propagated and how to generate the surface efficiently are questions that the most of the research work in this direction trying to answer.

In order to display local deformation including normal and shear strain and rigid body rotation, Schroeder et al. [SVL91] present a 3D vector visualization technique - the *stream polygon* which is an extension of stream lines. Stream polygons are actually a set of regular, n-sided polygons perpendicular to the local vector. By tracing stream polygons along streamlines , stream tubes are generated to depict properties of the vector field including strain, translation and rotation by the corresponding shapes and radius of polygons. This technique suffers from visual clutter if many stream polygons are generated.

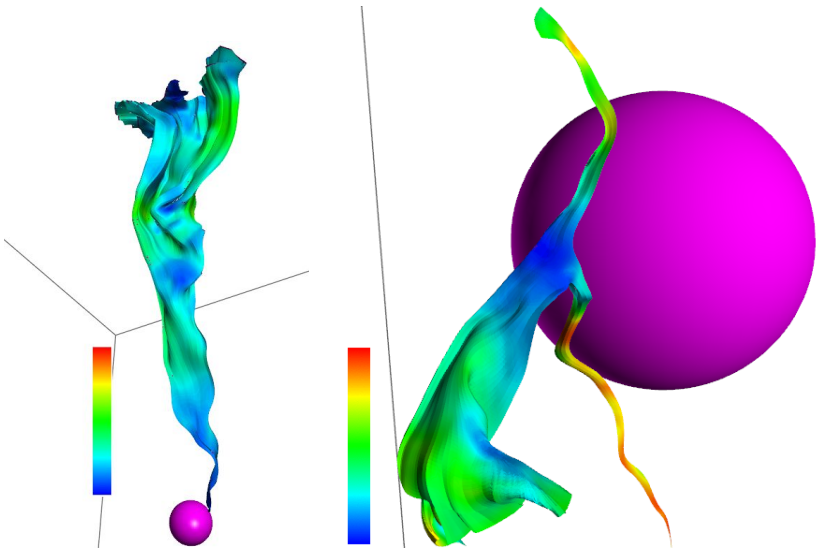

Figure 2.3: Streamsurface visualization of the smoke (left) leaving from a sphere (right) and spliting when it intersects the sphere [MLZ08]. Colour is mapping to velocity magnitude.

Hultquist [Hul92] introduces a novel method to construct *stream surfaces* efficiently. Stream surfaces offer perceptual benefits over seeding many streamlines. Depth perception is enhanced using surface primitives. Visual complexity is also reduced. In this implementation, flow fields can be explored rapidly and effectively from improved sampling densities over the tangent surface, see the similar result in Figure 2.3. During stream surface construction, advancement of the stream surface front is controlled by adaptively adjusting the sampling density, so regions of high divergence can be sampled accurately. Surfaces can be constructed using a greedy triangular tiling between adjacent pairs of streamlines. Ribbon-splitting can be introduced to adaptively refine the polygonal approximation in this process - stream surfaces can be split apart by adding some seeding points into the advancing front of sparsely sampled regions in order to visualize highly divergent flow fields with an acceptable sampling density. On the other hand, ribbons can be merged in areas of convergent flow. However this method has several pitfalls: accuracy of curve, insufficient splitting and overeager merging.

Max et al. [MBC93] introduce flow volumes, the 3D equivalent of streamlines, to visualize more information about the vector field by intuitive volume density which 2D streamlines or ribbons can't offer. In order to render the flow volumes in an efficient way, the method of Shirley and Tuchmann [ST90] is adopted to volume render the flow using a set of semi-transparent tetrahedra. These tetrahedra triangulate in intermediate layers between the layers of vertices in successive time steps. Additionally, a curvature based adaptive subdivision method is applied to handle the sampling fragment if the flow diverges or converges too much. Since only a small volume of the smoke is rendered, flow volumes are fast enough to make user interaction possible. However, the visual quality is not high.

In order to obtain an efficient method for the automatic placement of initial curves, van Wijk presents a new method for construction of stream surfaces [vW93b]. The main concept of his method is based on modelling a stream surface as an implicit surface $f(x) = C$. Streamlines are traced from every sample point in the vector field until they hit a boundary. Then a scalar field is derived such that all of the points common to a streamline are constant. Stream-surfaces are then computed using standard iso-surfacing techniques on the scalar field. However the complexity of generating stream surface in 3D domain is $O(N^4)$.

Brill et al. [BHR+94] introduce a 3D flow visualization technique using objects called streamballs which are based on the metaballs of Wyvill et al. [WMW86]. Spheres split or merge automatically depending on the distance between their center points in order to visualize the divergence or convergence of arbitrary complex flow fields. The premise for visualizing flow data with streamballs is to use the center positions of particles in the flow as skeletons for the construction of implicit surfaces. Spherical surfaces can then be blended with each other to form three-dimensional streamlines and stream surfaces. There are two kinds of streamballs. Using each of the discrete particles as a single skeleton leads to producing discrete streamballs. Grouping several discrete particles together produces a continuous skeleton (i.e. lines, curves etc.). Both forms can be enhanced by radius-mapping and color mapping to visualize local field parameters. However this method only works for regular grids.

Scheuermann et al. [SBH+01] present an algorithm to precisely generate the stream surfaces in tetrahedral grids by exploiting the piecewise linear interpolation. This algorithm is closely related to Hultquist's parametric approach [Hul92]. The stream surface in each cell can be precisely described by using piecewise linear interpolation, since the whole surface is built upon the cellwise analytic description which enables easy access to the local error of stream surfaces and related information of the flow structure in the cells. By using this algorithm, the stream surfaces that approach vortices which can not be properly calculated by previous approaches can now be precisely described. But Only regular grids are supported by this method.

Gelder presents a method to generate stream surfaces in a semi-global fashion for 3D vector fields on the CFD curvilinear grid [Gel01]. This method can solve constraints over a large region of space by integrals instead of locally propagating the solution of a differential equation. The basic idea of this method is similar to the method from van Wijk [vW93b] and the method from Knight and Mallinson [KM96], however, this presented method can address problems of no-slip surfaces and grids with degeneracies which the previous methods can not handle properly.

Two different approaches are presented to better model and render the implicit flow volumes by Xue et al. [XZC04], a slice-based 3D texture mapping and an interval volume segmentation coupled with a tetrahedron projection-based renderer. Conceptually, the implicit flow volumes are the extension of implicit stream surfaces proposed by van Wijk [vW93b]. In the first method, the implicit flow field is rendered directly without the inflow mapping to a scalar field using a dynamic texture operation, thus high interactivity can be obtained. In order to visualize the flow on stream surfaces and time surfaces, the second approach extracts a geometric flow

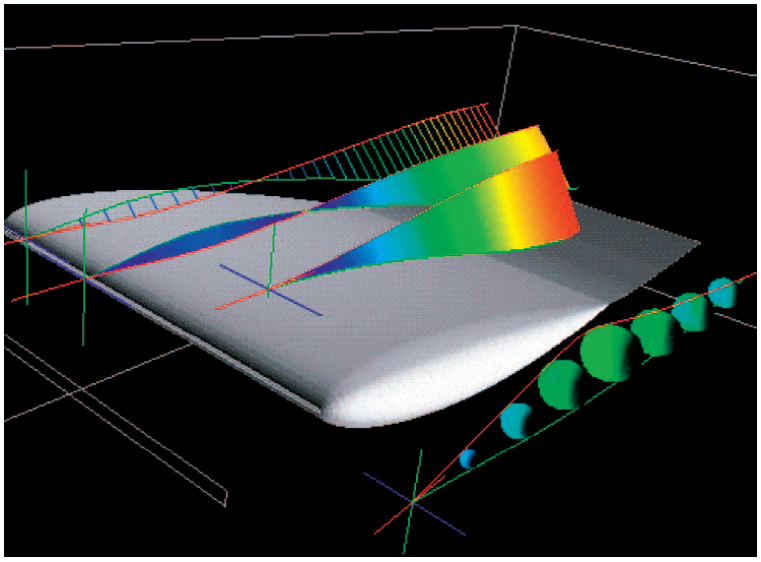

Figure 2.4: Comparing streamlines of two datasets simulated using different turbulence models. The streamlines are compared using line glyphs, strip envelopes, and sphere glyphs to highlight the differences between them. [VP04]

volume from the implicit flow using an iso-contouring or an interval volume routine, hence accurate and detailed result can be gained.

Besides these two main categories, comparing and visualizing the *uncertainty* resulting from the application using different numerical algorithms for numerical integration of geometry primitives is also interesting direction which attracts a lot of attention and research effort.

Lodha et al. [LPSW96] present a system named UFLOW (Uncertainty Flow) to visualize and investigate this uncertainty in fluid flow. This paper actually is a practical step to integrate presentation of data with uncertainty in scientific visualization [BBC91] [TK93]. First, the uncertainty in fluid flow is characterised by the possible sources of uncertainty. In this paper, focus is put on the use of different numerical algorithms to trace streamlines in fluid flow. Then with several visualization techniques including uncertainty glyphs, envelopes animations, priority sequencing, trace viewpoints and rakes applied, this system can intuitively highlight these uncertainties from application of different integration methods for users.

In order to address the need for comparative visualization tools to help analyze the differences on visualizing flow or vector data sets, Verman and Pang [VP04] present a method to compare individual streamlines and streamribbons as well as a dense field of streamlines. Since this method is an extension of UFLOW [LPSW96], it can visualize the difference between various integration methods of streamlines from the same data set and also the difference in streamlines generated from the same seed point location using different integrators in pairwise fashion. Furthermore, it is able to compare the differences between dense fields of streamlines

in order to present a intuitive global view of the differences between vector fields. Additionally, the differences of streamribbons can be visualized by using an overlay or a novel simplified representation according to the user's different needs. See Figure 2.4 for an example.

As we can see geometric flow visualization for 3D steady flow attracts the most attention in flow visualization domain since it's the one of the first flow visualization types and a lot of research work has been done to address corresponding challenges, such as perception and performance issues. For our marine turbine project in Chapter 5, in order to achieve better perception we apply several surface-based geometric visualization techniques (e.g. streamtubes) to reflect the behaviour of the flow past the turbine. It's a growing research domain although a few of the open challenges still remains in this category. For example, interactive stream surfaces seeding strategy for unstructured grids still remains as a big challenge.

#### 2.3.2.2 Unsteady Vector Fields

The following research focuses on geometric visualization in unsteady, 3D vector fields. Pathlines and streaklines are used as the *line-based* techniques for visualizing the unsteady flow.

Becker et al. [BLM95] present a method to generate the 3D analog of streaklines based on time-dependent flow fields. Conceptually, this method is an extension of 3D static flow based flow volumes [MBC93] to visualize unsteady flows in 3D space. Similar to the steady flow volumes which consists of the collection of streamlines that are seeded from on a 2D generating polygon, flow volumes for unsteady flows are based on streaklines which also start on a generating polygon. A series of tetrahedra is constructed along the path of a streakline. Although they are based on static flow volumes, unsteady flow volumes have to take additional considerations into account in order to correctly depict unsteady flow. Firstly, the adaptive step-size integration presented by Lane [Lan94] is used to advect the vertices which are generated by the streakline integration at the given time steps. Additionally, in order to address the difficulty in extending the original subdivision scheme of Max et al. [MBC93] to unsteady flow, adaptive subdivision is applied to insert or delete particles corresponding to the diverging or converging flow. The adaptive subdivision in the downstream direction is also required to keep the distance between layers of particles roughly constant. However The time for constructing and displaying the flow volume at each step is proportional to the number of vertices, so this method is not suitable to handle large datasets.

In order to visually analyze the dynamic behavior of the path lines on 3D time-dependent flow fields, Shi et al. [STW$^+$08] present an information visualization approach which enables the user to intuitively explore intricate 4D flow structures. Although SimVis [DGH03] is the most similar application, this approach puts focus on dynamic flow data rather than scalar data. First, local and global properties of path lines are obtained at selected seeding positions in 4D space. Then, for the purpose of analysing the multivariate data set from previous step, interactive brushing and focus+context visualization are utilized to illustrate how path lines can be used to describe and visualize the underlying flows.

As a natural extension of the streakline, *Streak-surface*, has been applied to visualize the 3D time-dependent flow to achieve a better visual coherence. *Streak-surface* can also be deemed as the dynamic stream-surface which take the temporal dimension into account. We discuss some related work in following literature.

For the purpose of creating and rendering stream surfaces and path surfaces which enable users to interactively manipulate seed curves, even for unsteady flow, Schafhitzel et al. [STWE07] present a point-based algorithm by exploiting GPU-based programming. This algorithm is based on point set surfaces [ABCO$^+$01] and adopts line integral convolution [CL93] to incorporate texture-based flow visualization. First, the streamlines and pathlines are generated by a GPU-based particle tracing algorithm. In order to deal with local flow divergence, a particle-density criterion is applied to obtain an even distribution of those particles. Second, based on the particle traces, the corresponding surfaces are generated and displayed by point set surfaces. Third, in order to compute corresponding flow textures for the texture-based flow visualization, a line integral convolution (LIC) method is applied on the vector field which is stored by the previous particle integration. Since the GPU-based implementation is efficient, user interaction for visualization and exploration on stream surfaces and path surfaces, and even for unsteady vector fields, is possible. However GPU-based approach is sensitive to the datasets of large scale.

A novel algorithm for the computation of integral surfaces is presented by Garth et al. [GKT$^+$08] that generates accurate integral surfaces from unsteady vector fields. As opposed to previous work on surface computation techniques like the method proposed by Hultquist [Hul92], this approach introduces a novel separation of integral surface approximation from the generation of a graphical representation in order to address limitations of previous work. According to this decoupling, the first step approximates a series of timelines using iterative refinement and generates an integral surface skeleton. The second step computes a well-conditioned triangulated representation based on the skeleton. This method can be applied on large time-varying datasets.

Von Funck et al. [vFWTS08] present a new approach to visualize smoke surfaces by utilising semi-transparent streak surfaces. This is the first time that semi-transparent streak surfaces are adopted to interactively visualize time-dependent flow fields. To avoid the problem of expensive adaptive remeshing which prevented streak surfaces from being used in an interactive fashion, the method of coupling the opacity of the triangles to their area, shapes, and curvatures is applied to render smoke using a triangular mesh with a fixed topology and connectivity. Thus, an intuitive and interactive exploration is possible.

Overall, there has been very little work in seeding of integral curves and surfaces in 3D, unsteady flow fields. We think this is mainly due to challenges related to both perception and computational complexity. Although several cutting-edge GPU-based techniques are introduced to accelerate the surface propagation the complexity of parallel computing increases the difficulties in algorithm design. Also the combination of large datasets and complexity of the unstructured meshes still poses big challenges to this domain. Some simplification techniques

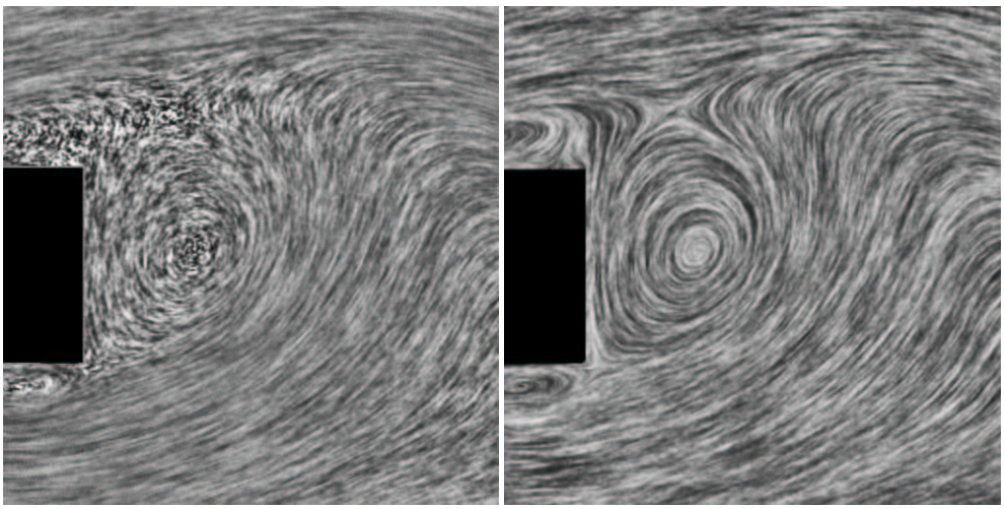

Figure 2.5: Visualization of flow past a box using (left) *Spot Noise* and (right) *LIC*.

such as vector field clustering could be helpful in managing very large datasets and interactive exploration.

## 2.4 Texture-Based Vector Field Visualization

In this section, we survey texture-based visualization methods for vector fields in higher dimensions. The theory behind is that a known texture is distorted according to the local properties of the vector field and then rendered to reflect the pattern of the associated vector field. This category of techniques provides a dense and coherent visualization with lots of details, even in areas of complicated flow, see Figure 2.6. Texture-based flow visualization techniques can be mainly classified into three general categories: *Spot Noise*, *LIC*, and *GPU-based Texture Advection* techniques [LHD+04]. Details of each category will be discussed along with corresponding techniques in following subsections. Since we only focus on 2.5D and 3D texture-based techniques in this chapter, for a complete survey of texture-based methods please refer to [LHD+04].

### 2.4.1 2.5D Vector Fields

The traditional texture-based visualization techniques is based on 2D Cartesian grids. A lot of research has been done for the 2D texture-based visualization, however how to apply these techniques to 2.5D boundary flow on unstructured meshes is nontrivial. Challenges stems from computational complexity of handling boundary curvilinear or unstructured meshes, and conceptual difficulties such as showing the direction of the flow. Following literature is trying to address these challenges.

#### 2.4.1.1 Steady Vector Fields

The first try of applying texture-based visualization from 2D to 2.5D steady flow is from *Spot Noise* techniques. This technique was firstly introduced by Van Wijk [vW91]. In this technique, the so called spot is normally an ellipse or other shape which is warped over a small time step and distributed in a streak fashion to reflect the characteristics of the local flow as shown in Figure 2.5 (left).

In order to apply the spot noise to 2.5D flow visualization, Leeuw and van Wijk [LvW95] present an enhanced spot noise method for surface vector field visualization as an extension of the original spot noise described by van Wijk [vW91]. This enhanced method offers several improvements over the original. To obtain a better visualization of vector fields with high curvature, spot blending is applied to deform the spot into a curved shape according to the local velocity field characteristics. The deformation of the spot is carried out based on a stream surface. Second, with spot filtering implemented, the problem of generating spot noise pictures with a coarse, low-frequency component can be solved, and more homogeneous textures are computed. Third, the spot noise images are rendered fast by graphics hardware, thus the visualization can be rendered at interactive speeds. This method is extended to visualize flow on curvilinear grids.

Later they present a visual simulation of the oil-flow experimental patterns using an enhanced spot noise texture [LPPW95]. This work is based on the enhanced spot noise method [LvW95]. Experimental flow visualization techniques visualize flow field on the surface with oil streaks. The numerical flow simulation can be used to obtain the wall-friction-vector data based upon which a spot noise texture is generated and blended to enhance the visualization of the flow field on the surface. Convergence data is used to scale the intensity distribution function and spot advection is used to simulate oil accumulation in high convergence areas, the final spot noise visualization renders images very comparable to the numerical and experimental visualizations.

Meanwhile *LIC* techniques are also applied to visualize 2.5D steady flow. The *LIC*, namely, Line Integral Convolution, is derived from an algorithm introduced by Cabral and Leedom [CL93]. The original *LIC* approaches convolve and smear the properties of a noise texture using a kernel filter in the direction of the underlying flow. See Figure 2.5 (right). *LIC* was one of the first dense, texture-based techniques able to accurately reflect flow with high local curvature.

Battke et al. [BSH97] introduce an enhanced line integral convolution method for visualizing surface-based (2.5D) vector fields. This method is an extension of the fast LIC approach described in [SH95] and can be applied to 3D arbitrary surfaces whereas previous approaches are restricted to the characteristics of curvilinear surfaces. This method starts with a triangular approximation of the arbitrary surface, then for each triangle a local LIC texture is computed based on local euclidean coordinates, the final step is to generate a smooth textured surface by following stream lines across neighbouring triangles.

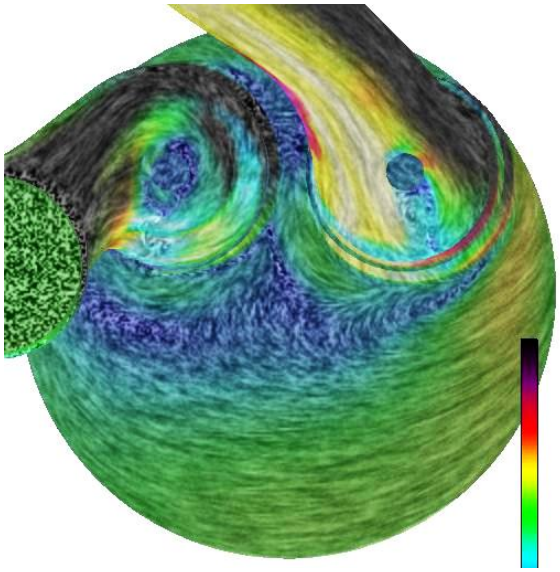

Figure 2.6: Texture-based flow visualization is applied to the surface of an 221K polygonal intake port mesh [LJH03]. Colour is mapping to velocity magnitude.

#### 2.4.1.2 Texture-Based Visualization for 2.5D Unsteady Vector Fields

In order to extend the texture-based visualization from steady to unsteady flow in 2.5D, traditional Spot Noise or LIC techniques may not be suitable for this requirement since performance is the bottleneck for the time-dependent visualization. So quick and efficient algorithms for texture manipulation are needed. *GPU-based Texture Advection* techniques are of this type. This category advect the properties of individual texel or groups of texels in the direction of the associated vector field by utilising the parallel computation from the GPU (Graphics Processing Unit) rather than the CPU in order to realise performance gains. The following research focuses on applying *GPU-based Texture Advection* techniques to visualize surface-based unsteady flow.

Laramee et al. [LJH03] present a method to generate dense representations of unsteady flow on surfaces based on 2D Lagrangian-Eulerian Advection [JEH01] and Image Based Flow Visualization [vW02]. [LJH03] addresses the problem of high computation time required to visualize unsteady flow on surfaces, especially on boundary surfaces of large and complex meshes from CFD data sets. See Figure 2.6. This approach starts with vector field projection which projects the vector field of the surface to the image plane. A color-encoded velocity image which stores the projected vector field is generated. Secondly, the mesh which is applied to advect the textures similar to IBFV [vW02] is distorted according to the discretized Euler approximation of a pathline. Then a noise texture is distorted and attached according to the distorted mesh. The next step is to inject and blend noise in image space. Geometric discontinuities are taken into account using edge detection. Finally, all optional images such as a shaded version of the geometry are rendered.

In the same year another new method to render dense, vector-field guided textures on 3D curved surfaces is presented by Van Wijk [vW03]. As an extension of the original 2D algorithm - IBFV [vW02], this algorithm synthesises textures on curved surfaces directly in image space and also inherits advantages of simplicity, speed, and efficiency in 3D. Each vertex defining the triangular surface contains a position, normal vector and velocity vector. The mesh and its associated vector field is projected to image space. Each vertex position is distorted according to the velocity vector at that position. Advected coordinate positions are computed using Euler approximation of a pathline. Synthesised noise textures are then mapped to the newly advected coordinate positions. After this, noise blending is used to update the visualization as noise texture is advected in the direction of the flow. Finally, a shaded version of the mesh geometry is rendered to enhance depth perception.

A side-by-side comparison between two image space based methods for visualization of vector fields on surfaces: Image Space Advection (ISA) [LJH03] and Image Based Flow Visualization for Curved Surfaces (IBFVS) [vW03] is presented by Laramee et al. [LvWJH04]. Firstly, they identify that both algorithms share several overlapping characteristics such as projection, texture-mapping and noise insertion, whereas the main difference between these two methods is the way the textures advect: ISA adopts the image-based mesh however the IBFVS is driven by the original object-based mesh. Furthermore, the relative advantages and disadvantages of each approach are also presented and suggestions are given regarding when and where they are most suitable.

Later in 2004, Laramee et al. [LSH04] apply a method to extend dense, texture-based visualizations of flow onto iso-surfaces. In terms of the texture-based visualization techniques, the ISA method [LJH03] is adopted as the solution in this case. Combined with the iso-surfacing, a normal mask is applied to address the normal component of the flow to the isosurface. This normal mask, stored in the alpha channel, adjusts the opacity of the image overlay according to the magnitude of cross-flow component to the isosurface. In the texture advection phase ISA [LJH03] is applied. The vector field is projected onto image space. Edge detection is incorporated. The image advects according to the advected texture coordinates. Noise is injected and blent. Finally, an image overlay is employed to enhance the result.

In 2006 Laramee et al. [LGSH06] present a hybrid visualization which combines the advantages from stream surfaces and texture advection techniques. The stream surface generation is based on the method of Garth et al. [GTS+04] while ISA [LvWJH04] is adopted as the texture advection algorithm. This approach conveys properties of the vector field that stream surfaces alone can not. They apply the visualization technique to various patterns of flow from CFD (important to automotive engine simulation including two patterns of in-cylinder flow (swirl and tumble) as well as flow through a cooling jacket). In addition, they explore multiple vector fields defined at the streamsurface such as flow, vorticity, and pressure gradient.

### 2.4.2 Texture-Based Visualization for 3D Vector Fields

When texture-based visualization is applied to visualize 3D volumetric flow, clearly, the biggest challenge is the perceptual issues related to 3D flow visualization such as occlusion, depth perception, and visual complexity. In order to address this challenge, the following research has been dedicated to improve the perception of texture-based flow visualization in 3D.

#### 2.4.2.1 Steady-state Vector Fields

To address the perceptual issues in texture-based visualization for volumetric flow, Rezk-Salama et al. [RSHTE99] present a direct volume rendering approach based on 3D-Texture mapping for visualizing vector fields in an interactive and animated fashion. Conceptually, this approach is a 3D extension to the original LIC [CL93]. With interactive methods including user-controlled transfer functions and volume clipping approaches applied, interactive exploration of the interior structures of the vector field is realised. In order to improve the perception of flow in 3D at a reasonable computational cost, an animated 3D-LIC method is presented via two approaches. The first one is to pre-compute a special 3D-LIC texture for the time-dependent colour table animation. The second one is to clip the 3D-LIC volume interactively according to pre-computed user-defined volumes. However this method does not support unstructured meshes.

Shading is also used to enhance depth perception. Grant et al. [GEO02] present a new method named Lagrangian-Eulerian Time Surfaces (LETS) to visualize ocean flows including vertical motions in 2D and 3D space. This method is built upon the original 2D Lagrangian-Eulerian Advection (LEA) method proposed by Jobard et al. [JEH01]. Initially, LETS distributes a set of particles uniformly on a horizontal plane. Then a time surface is obtained by tracking the horizontal plane in time using a mixture of Eulerian and Lagrangian techniques during each time step. The vertical motion is taken into account in addition to the horizontal motion to displace the time surface according to the flow. Afterwards, a texture computed using texture advection from the dominant horizontal motion is projected onto the evolving surface with shadings to enhance the perception of the final visualization. This method extends LEA to handle 3D flows, however, the method only works for the case of weak vertical velocities rather than the general 3D case.

#### 2.4.2.2 Texture-Based Visualization for 3D Unsteady Vector Fields

Although the perception is still the biggest challenge, to effectively visualize time-dependent 3D flow, texture-based visualization has to take performance issues into account as well. So some enhanced LIC techniques and GPU-based techniques are used to address the speed issues.

Helgeland and Andreassen [HA04] present a new method to more effectively visualize three-dimensional vector fields using LIC. Conceptually, this method is a extension of the original LIC method [CL93]. As the first stage, a seeding-based LIC is employed to fast compute 3D textures using an appropriate input texture in a interactive fashion. The direction of a vector field can be effectively observed from generated images. Secondly, in order to reveal depth

relations among the field lines traced by Seed LIC at a reasonable computational cost, an appropriate transfer function based on a limb darkening shading technique is applied. Finally, the combination of texture-based direct volume rendering and volume LIC is used to depict additional information of scalar quantities when visualizing 3D vector fields.

Weiskopf and Ertl present an interactive texture-based approach for the dense visualization on unsteady 3D flow fields [WE04a]. Although the basic idea of this approach is from 3D IBVF [TvW03], this approach improves and extends 3D IBVF. The first contribution of this paper is that this method can display a full range of velocity values with a fully three-dimensional advection mechanism, thus the problem of velocities being restricted in 3D IBVF is addressed. Second, this GPU-based technique allows a slice of the 3D representation to be updated in a single rendering pass, which makes the interaction possible. Third, an enhanced blending scheme is applied with more flexible noise and dye injection. Moreover, the advection and rendering schemes are extended to transport and display different materials in order to gain more insight of flow fields.

With the goal of high efficiency and good visual perception, Weiskopf et al. [WSE05] present a dense texture-based technique to interactively visualize the unsteady 3D flow. Essentially, this technique is a 3D extension to the texture advection [MB95] by exploiting GPU-based programming. First, a novel 3D GPU-based texture advection mechanism is applied, where the problem of 3D logical memory structure is implemented fast and efficiently by using 2D textures in 2D physical memory. Second, slice-based direct volume rendering is applied to compute volume illumination. Third, some perception-guided volume shading methods are introduced, such as halos with a volumetric importance detector in order to gain good visual perception. Later, they present a technique to interactively visualize the unsteady 3D flow in a dense texture-based fashion [WSE07]. This technique is conceptually built upon their previous texture advection based work [WSE05]. Common to previous work, a 3D GPU-based texture advection mechanism and slice-based volume rendering are applied to achieve efficient 3D texture advection. Additionally, in order to incorporate volumetric illumination, two alternative methods are introduced: first, gradient-based illumination that makes use of GPU-based real-time computation of gradients for the sake of local illumination, and second, line-based illumination which adopts the idea of illuminated streamlines based on vector field directions [ZSH96]. Finally, the problems of clutter and occlusion are addressed by applying perception-guided rendering methods and the volumetric importance function.

In order to address perceptual problems arising from dense presentation like cluttering, Helgeland et al. [HE06] present an interactive texture-based method for visualizing 3D unsteady flow fields based on a sparse representation. At the first stage, a set of particles are evenly distributed throughout the whole domain and then tracked along the time-dependent velocity field by calculating pathlines. Then a novel particle advection strategy inspired by the evenly-spaced streamlines in 2D [JL97] is applied in order to maintain the coherent particle density as time increases. At each time step, directional information is visualized in the output 3D texture. In this way, animation shows the advection of particles, while each frame shows the instantaneous vector field. Conceptually, the idea of this hybrid solution is similar to the ones

used in DLIC [Sun03] and UFAC [WEE03]. Additionally, by decoupling the rendering stage from the rest of the visualization pipeline (based on the method proposed by Li et al. [LBS03]), rendering performance is improved and interactive exploration of multiple fields is also possible.

In general, texture-based visualization is another mature visualization approach able to demonstrate the details of the underlying 2.5D or 3D flow, even the flow pattern is complex. With lots of research effort the problem of 2D or 2.5D, steady or unsteady texture-based flow visualization is close to being solved. However, the same can not be said to 3D volumetric time-dependent flow fields, especially the flow based on unstructured meshes. Perceptual issues stand as the primary bottleneck to solving this challenge. Although texture-based visualization techniques have not been directly applied in my research projects, the idea of efficiently handling flow based on unstructured boundary meshes by projecting it to the image space in texture advection methods [vW02] [LJH03] has been adopted to prepare the vector filed for our image-based glyph visualization in Chapter 3 and the image-based vector field clustering in Chapter 4.

## 2.5 Vector Field Clustering and Visualization

*Feature-based* flow visualization is an approach working at more abstract and intelligent level than *direct*, *geometric*, and *texture-based* categories since this approach extracts subsets of data with features deemed interesting by users. This process is performed before visualization, hence the visualization is then based on these extracted subsets rather than the whole dataset. This may make the visualization more suggestive, efficient and effective. However, the complexity and computational cost of feature extraction may be high. Since this category is abstract and has wildly ranged techniques dedicated to it, such as separation and attachment line extraction [PVH$^+$03], topology extraction [LHZP07], etc. A complete survey based on this category in higher dimension is beyond the scope of this thesis. So we only focus on a particular new feature-based technique, namely, vector field clustering technique for higher dimensional flow visualization. Although vector field clustering techniques are arguably feature-based, we include them here because (1) they have never been included in a vector field survey before and (2) the results resemble both direct and sometimes geometric vector field visualization techniques. Thus they can serve as a bridge between these two categories.

Before we start discussing vector field clustering visualization techniques, a comprehensive and systematic survey focused on scalar clustering algorithms rooted in statistics, computer science, and machine learning is presented by Xu and Wunsch II [XW05]. Firstly, they generalise the clustering analysis procedure using four steps: feature selection or extraction, clustering algorithm design or selection, cluster validation, and result interpretation. Secondly, they detail clustering algorithms in terms of the nature of clusters. Corresponding approaches for clustering different kinds of data sets are discussed and suggested. Lastly, in order to compare different clustering algorithms, applications to some benchmark data sets are presented.

Clustering algorithms can be divided into two groups: either based on constructing a hierarchical structure or based on generating subsets of similar data (known as partitions). Hierarchical clustering algorithms are mainly classified as agglomerative methods and divisive. Since agglomerative methods can generate flexible clustering using binary trees and provide very informative descriptions, they are employed to develop some vector field clustering visualizations like [TvW99] and [HWHJ99]. However, the computational cost for hierarchical clustering can be expensive (e.g. $O(N^2)$) so some of the early hierarchical clustering methods are not capable of handling large-scale data sets. In order to address this problem, some improved hierarchical clustering methods are introduced like BIRCH [ZRL96] which utilises the clustering feature tree to improve the robustness and reduce the computational complexity to $O(N)$. In terms of pre-specified numbered clustering methods, the K-means algorithm is representative and is effective in clustering large-scale data sets. A general solution to data clustering is a major aim of the AI community.

Although the hierarchical clustering is comparatively more expensive to compute, it's easy to implement and thus provide the user direct access to drill down the hierarchy by specifying a certain threshold values. So vector field visualization with this hierarchical clustering feature applied is able to group the underlying flow according to the similarity in terms of the flow characteristics such as direction and magnitude, and thus provide the user both the overview and the detailed view based on the formed hierarchical clusters interactively.

Firstly, Telea et al. [TvW99] present a hierarchical clustering based method that automatically places a limited number of glyphs in a suggestive manner to represent the vector field. It's the first algorithm of its kind and produces both global and detailed information in the same image. In terms of the clustering algorithm, two candidate clusters are selected and merged together in a bottom-up fashion. During this process, a vector field similarity measure is used to evaluate which clusters should be merged. An error metric based on local vector magnitude and direction is introduced to define how a new cluster is merged from two existing ones. Although this algorithm automatically displays the simplified flow, a few clustering parameters are available to refine the result and satisfy different needs of the users, i.e. changing the detail level of the clustering. However this method only works for regular grids.

Rather than the bottom-up hierarchical approach described above, Heckel et al. [HWHJ99] present a method to visualize discrete vector fields in a top-down hierarchical fashion in the same year. This method uses a clustering approach to segment the original vector field into a series of disjoint clusters. Firstly, points from the original vector field data are treated as a single cluster. Then a procedure for splitting clusters is applied using a weighted best-fit plane which partitions the space into convex regions (sub-clusters). Each region has an error measure which calculates the differences between the original discrete vector field and the simplified vector field. With the use of the error measure approach, the recursive procedure of splitting clusters can be terminated when a user-defined threshold value is met. It's also worthy of note that this method does not require a regular grid but the computational cost is heavy. According to the performance result 45 seconds spent on generating only 500 clusters.

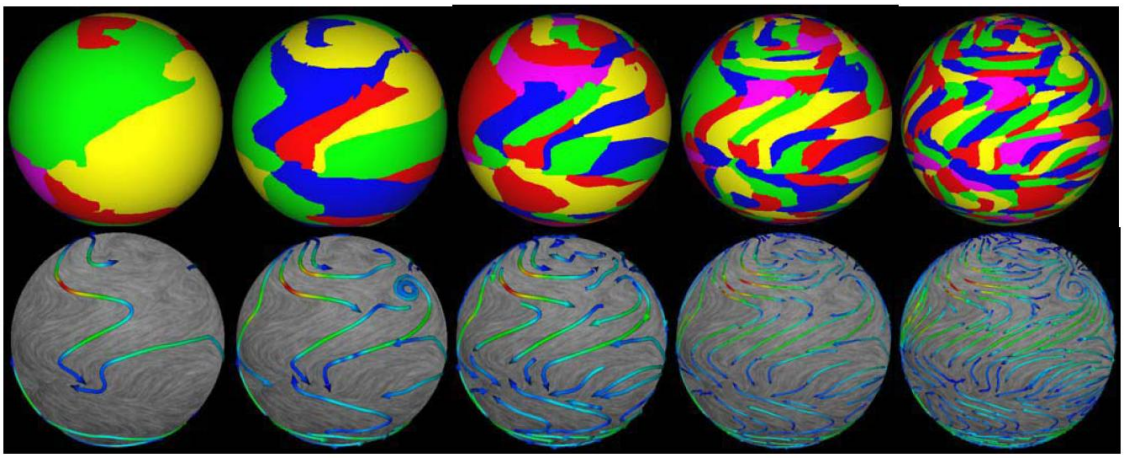

Figure 2.7: Climate dataset decomposition, five coarsest levels (left to right). Cluster images (top row) and flow texture applied with streamline arrow icons (bottom row). [GPR+04]

Instead of constructing a hierarchical structure, generating subsets of similar data in terms of pre-specified numbers of clusters, such as k-means, can be also applied to cluster vector fields in higher dimensions. This approach is able to generate clusters of underlying flow according to the specific cluster numbers quickly enough so that on-the-fly user interactions are enabled. To achieve better vector field clustering results, cross-domain expertise such as numerical analysis for clustering in physics is also used in some related research here.

Garcke et al. [GPR+01] present a new multi-scale method for vector field clustering in 3D space. This continuous clustering method is inspired by the well-known physical phase separation clustering model - the Cahn Hilliard model [CH58]. To classify and enhance the correlation in the cluster sets effectively, this new phase-separation-based continuous clustering method formulates the clustering problem as a diffusion problem rather than a merging or a splitting problem. At the first stage, the set of clusters is implicitly handled by an evolution function with two important energy contributions taken into account: the nucleation of cluster sets and a successive coarsening of the clusters. In accordance with the underlying physical data and based on the evolution function, segments of the flow fields are extracted and classified depending on their location and orientation. Afterwards, a skeletonization approach is applied to highlight the essential features of the refined cluster sets. Finally, various geometric representations are adopted to render the highlighted skeleton in a intuitive way. However the proposed method replies on the pre-generated skeleton shapes or templates. The performance issues also need to be addressed.

In 2004 Griebel et al. [GPR+04] present a vector field clustering method based on an algebraic multigrid [UA01]. Each sample in the flow field is represented by a tensor stiffness matrix that encodes the local properties of the flow field. The algebraic multigrid technique operates on these tensor matrices in order to construct a vector field hierarchy encoding the flow structure. The method is geometry-free and is demonstrated on 2D and 3D vector fields. See Figure 2.7.

However this method only works for regular grids.

As we can see the size of simulation datasets from CFD increases, so does the demand for visualization methods that quickly depict vector fields in a simplified and insightful manner. Vector field clustering algorithms are a desirable solution which offers the advantage of presenting a detailed picture of important or complex areas of the domain while depicting a simplified representation for areas of less importance. However, this category has received relatively little attention. This mainly due to the computational complexity and performance issues. According to the survey there is no related work done for flow visualization of 2.5D boundary flow although effective simplification methods are strongly needed for the complex and large boundary flow datasets. Based on this fact, development of efficient vector field clustering visualization techniques for vector filed based on unstructured boundary meshes is an interesting but challenging direct for future research. Trying to address this challenge, we present a 2.5D vector field clustering method in Chapter 4.

## 2.6 Conclusion and Future Work

This chapter provides the most up-to-date overview of the state-of-the-art in vector field visualization in higher dimensions. The most important literature is included and discussed in this survey. A general solution for vector field visualization is not feasible for all applications. Hence the existence of various vector field visualization techniques provides users different choices to gain an intuitive exploration or a high quality presentation according to their different interests.

As clearly illustrated from Table 2.1, the group of geometric approaches is the most popular since a great amount of effort has been invested in this area. Reasons for the success of this group have to do with its coherent and insightful presentation which reflect the continuity of the flow and help users to gain an intuitive picture of the flow from the data. The major focus has been put on geometric methods for 3D steady flow fields whereas cases of surface-based (2.5D) or 3D unsteady flow fields have received comparatively less attention. Table 2.1 highlights the absence of work that has been done using geometric techniques for unsteady flow on surfaces. The texture-based group is also a mature group which contributes solutions to visualize 2.5D or 3D flow fields densely and coherently. However, it can still suffer from visual complexity when implemented for 3D and 4D flow data.

On the other hand, direct approaches and vector-field clustering approaches have received relatively little attention. As we can see from Table 2.1, there was no direct flow visualization method for surface-based (2.5D) flow field visualization. This is probably due to the difficulties in placing the glyphs on the complex boundary meshes and perceptual problems like visual complexity and occlusion. With respect to vector-field clustering methods as the feature-based flow visualization technique, although some methods have been presented to visualize 3D vector fields, there are still few methods for visualization of 2.5D vector field because of the potential challenges added by clustering vector field on complex, large, and unstructured

boundary surfaces, such as the cooling jacket simulation in Figure 1.3. However, the size of the input data set continues to grow very fast. It would be impractical to visualize this type of data without simplifying. For this reason, simplification approaches are necessary, such as vector-field clustering.

It is also worthy of note that throughout the period of my research, *Feature-based flow visualization* seems to be dominant in flow visualization between 2007 and 2011. This is partially due to the need to address challenges of the increasing size and complexity of CFD datasets, and effectively extracting features from the raw CFD data for flow visualization becomes the most attractive topic to the flow visualization community. For quite a long time, feature-based flow visualization of complex 3D unsteady flow remained elusive due to high complexity and expensive computational cost. However, some significant progress has been made to address this problem. The use of the finite-time Lyapunov exponent (FTLE) and the corresponding Lagrangian Coherent Structures (LCS) [BP02] for finding distinct regions and features such as divergence and convergence from 2D and 3D time-dependent flow are more and more popular based on recent a great amount of related publications [SP07] [GGTH07] [STM08] [GWT+08] [GOPT11] [PTA+11]. FTLE is generic and robust for analyzing fluid behavior. Additionally, interactive selection-based flow visualization [SWC+08] [BMI+07] [JM10] is also attracting attention as another hot branch of feature-based flow visualization. Cross-disciplinary techniques such as image processing and information visualization are applied in multi-linked views to achieve a projection of the raw data. The user is able to select flow feature information intuitively in the projected low-dimensional feature space while the visual output is updated and highlighted in original 3D space. We believe feature-based flow visualization will still follow this trend and continue to be attractive in near future and this is also the research direction that we are interested in.

In future work, based on the survey there are some areas where our research interests are placed at:

- A efficient direct flow visualization approach for unstructured surface-based (2.5D) flow to give a quick overview and detailed view of the data without visual complexity and occlusion.
- A vector-field clustering algorithm as the feature-based approach for complex, large, and unstructured surface-based (2.5D) flow so that a simplified and suggestive image could be automatically obtained for surface-based flow.
- Interactive cross-disciplinary approaches for flow visualization so that the flow features can be intuitively obtained and selected in different statistic space.
- Handling inaccuracy and uncertainty of visualization.

*Chapter 3*

# Vector Glyphs for Surfaces: A Fast and Simple Glyph Placement Algorithm for Adaptive Resolution Meshes

**Contents**



"In these matters the only certainty is that nothing is certain."

- Gaius Plinius Secundus [4]

In the last chapter we learn that there is no direct flow visualization method for surface-based (2.5D) flow field visualization due to the difficulties in glyph placement on the complex boundary meshes and perceptual issues such as occlusion. In this chapter we present a fast and simple glyph placement algorithm as a direct flow visualization method in order to investigate and visualize flow data based on unstructured, adaptive resolution boundary (2.5D) meshes from CFD. The algorithm has several advantages: (1) Glyphs are automatically placed at evenly-spaced intervals. (2) The user can interactively control the spatial resolution of the glyph placement and their precise location. (3) The algorithm is fast and supports

[4] Gaius Plinius Secundus (23 AD - 79 AD), was a Roman author, naturalist, and natural philosopher, as well as naval and army commander of the early Roman Empire.

multi-resolution visualization of the flow at surfaces. The implementation supports multiple representations of the flow–some optimised for speed others for accuracy. Furthermore the approach doesn't rely on any pre-processing of the data or parametrisation of the surface and handles large meshes efficiently. The result is a tool that provides engineers with a fast and intuitive overview of their CFD simulation results. This chapter has been published as a research paper in VMV'08 conference [5].

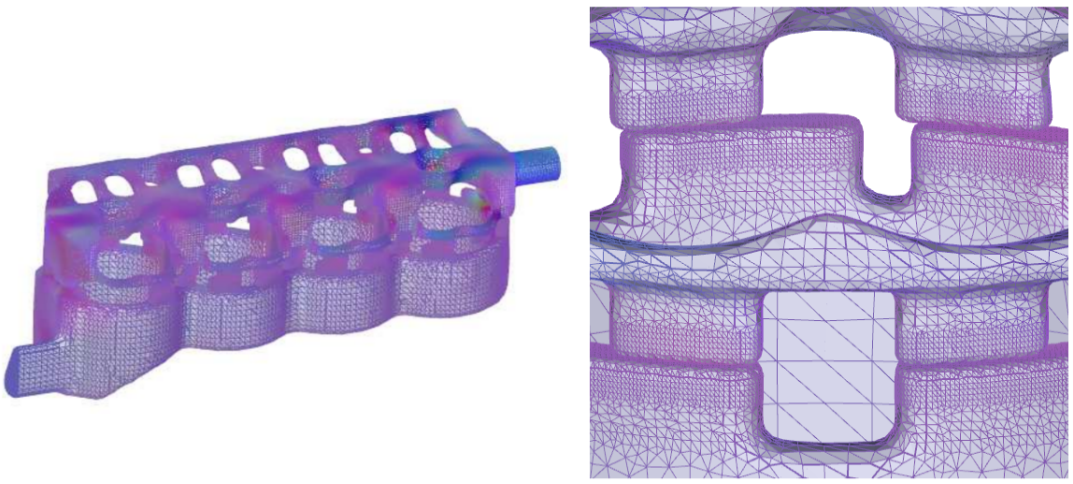

Figure 3.1: The unstructured adaptive resolution boundary grid of a cooling jacket from a CFD simulation. The left image is an overview of the boundary mesh, and the right is a close-up. These images illustrate how complex a typical mesh from CFD can be.

## 3.1 Introduction

Ever increasing attention is invested in order to find reasonable and efficient solutions for analysing and visualizing the flow from computational fluid dynamics in last three decades. As the size of simulation data sets increases, so does the need for effective visualizations that provide insight into the data. A tremendous amount of time and money is spent on simulation in order to speed up the manufacturing process. Constructing objects in software should be faster than building their real hardware counterparts.

Out of all the possible visualization techniques that can be used to investigate the simulation results, vector glyphs and colour-coding are the most popular tools used by engineers. Vector glyphs offer several advantages. They are intuitive – the depiction of the underlying flow is universally understood. Secondly, they do not accumulate error in the same way that geometric techniques do. Integration-based visualizations such as streamlines have in an inherent

[5] Published as: Zhenmin Peng and Robert S. Laramee, **Vector Glyphs for Surfaces: A Fast and Simple Glyph Placement Algorithm for Adaptive Resolution Meshes** in *Proceedings of Vision, Modelling, and Visualization (VMV) 2008*, pages 61-70, 8-10 October 2008, Constance, Germany.

error associated with them stemming from the approximations made in the underlying computation. Thirdly, glyphs are easy to implement. No complicated algorithms or data structures are needed. Thus they are featured in every software application. However, glyphs also have their drawbacks. Optimal vector field glyph placement is a challenge, especially in the context of CFD applications. Figure 3.1 shows a typical, triangulated boundary mesh produced from a CFD model. Its unstructured, adaptive resolution characteristics make the placement of vector glyphs difficult. If we naively place a vector glyph at every sample point on the surface, then the glyphs are either too small to see or so large that they overlap and result in clutter. Another drawback is that the density of glyphs corresponds with the density of mesh polygons. This variation is unrelated to the vector values themselves. Also, the user has no control over the glyph placing. Furthermore, rendering so many glyphs degrades performance time greatly. Most of the glyphs would be occluded.

While glyph-based visualization has been widely applied to tensor field and medical visualization [RP08] [War02], glyphs for vector field visualization have received relatively little attention. This may be due to the difficulties in placing glyphs on unstructured, adaptive resolution boundary meshes from the complex CFD data sets and perceptual problems like visual complexity and occlusion. In order to address these challenges, we present a fast and simple glyph placement algorithm to investigate and visualize flow data based on unstructured, adaptive resolution boundary meshes from CFD yielding the following benefits:

- Glyphs are automatically placed at evenly-spaced intervals, independent of how complex or dense the underlying adaptive resolution mesh is.
- The user can interactively and intuitively control the spatial resolution of the glyph placement as well as their precise location.
- Multi-resolution visualization of the flow at surfaces can be applied to increase detail in areas deemed interesting by the user.
- Glyphs are never generated for occluded or otherwise invisible regions of the surfaces.
- The algorithm is fast, enabling user interaction features such as zooming, translating and rotation.
- Our approach enables various representations of the flow, optimised for either speed or accuracy, in a natural way.

The algorithm relies neither on pre-processing of the data nor on parametrisation of the surface. It also handles large numbers of polygons efficiently. The key to the algorithms speed and simplicity is transferring computation that would normally take place in object space to image space. The approach is especially useful because engineers often start their investigation of simulation results by looking at the surface for an overview.

The rest of the chapter is organised as follows: Section 3.2 provides an overview of related research work. The placement algorithm and user options are described in multiple stages in

Section 3.3. Section 3.4 gives the performance and visualization results. Conclusions and suggestions for future work are presented in Section 3.5.

## 3.2 Related Work

Ward [War02] states that glyph-based visualization has been widely used to convey various information simultaneously by employing intuitive graphs to depict corresponding various variables from abstract data sets. Our work focuses on applying this intuitive depiction in image-space as well as developing an efficient and fast glyph placement algorithm to illustrate the vector field accurately. Previously, related techniques have been proposed in order to improve glyph-based visualization. In this section we describe these related techniques. We emphasise the glyph placement related techniques in two main categories: tensor field and vector field data. Within each category, techniques are discussed with respect to the dimensionality of the given data: 2D, 2.5D (for surfaces in 3D) and 3D.

### 3.2.1 Tensor Field Glyph Placement

The majority of related work has not focused on vector field glyph placement but rather tensor field glyph placement. Laidlaw et al. [LAK+98] apply an elliptical tensor field glyph placement algorithm for the visualization of 2D Diffusion Tensor Image (DTI) data from the spinal cord of a mouse. The regular array of ellipsoids are normalised by size for a visualization that is more easily deciphered. Instead of placing tensor glyphs on a regular-Cartesian array, Kindlmann and Westin [KW06] present a glyph placement algorithm that shapes and positions the glyphs in a smooth and continuous fashion resulting in a visualization free of holes and without overlapping glyphs. The artifacts of the underlying grid structure then disappear. Hlawitschka et al. [HSH07] present an accelerated version of the tensor field glyph packing algorithm. The goal of their algorithm is to support interactive data exploration.

Additionally, a surface-based (2.5D) glyph placement strategy for medical visualization is proposed by Ropinski et al. [RMSS+07]. The algorithm works in object space and is based on isosurfaces. The volume is searched voxel-by-voxel for locations through which the chosen isosurface passes. Afterward, a glyph is placed at every cell that encompass the given isovalue such that it is located on the specified isosurface. Axis aligned rays are cast into the volume in order to detect visible portions of the isosurface. Then only visible portions remain in the final rendering. An approach that requires volume searching, isosurfacing, and ray casting is overly complex and not optimised for speed. Sigfridsson et al. [SEHW02] present a hybrid volume rendering and glyph-based visualization for 3D tensor data based on interactive glyph placement. The glyphs are placed manually with a 3D cursor.

### 3.2.2 Vector Field Glyph Placement

Vector field glyph placement has received comparatively little attention. A vector glyph placement approach is described by Klassen and Harrington [KH91]. Three-dimensional glyphs are placed at regularly-spaced intervals on a 2D plane. Shadows on the plane are added to the

glyphs to highlight their orientation. In order to depict the vector fields on curvilinear and unstructured grids, Dovey [Dov95] presents a vector glyph placement algorithm for slices through 3D curvilinear and unstructured grids. He describes two different object-space approaches for resampling a vector field defined on a 3D unstructured or curvilinear grid onto a regular planar slice. The most computationally expensive part of the procedure for interpolating a simulation result value onto an arbitrary new point is locating the cell that contains the point. This process can be very costly in terms of processing time even when spatial data structures are used to accelerate the search. Hong et al [HMK95] use volume rendered vector glyphs which are generated from pre-voxelized icon templates to describe regular, structured vector fields in 3D space. Incremental image updates which re-compute only those pixels on the image plane affected by user input make visualization of the scalar and vector field faster and more interactive. Laramee [Lar03] describes an object-space approach using resampling and vector glyph placement for slices through unstructured, 3D CFD meshes. The algorithm we describe here is conceptually similar but raises the spatial dimensionality to surfaces (as well as planar slices). Our algorithm is also faster, simpler, and more efficient.

## 3.3 Intuitive Glyphs on Surfaces

This section presents the details of the algorithm starting with a short discussion of why we chose an image-based approach.

### 3.3.1 Object Space vs. Parameter Space vs. Image Space

In order to construct a fast and simple glyph placement algorithm on surfaces, we develop an approach which can deal with large and complex flow data sets from CFD efficiently and interactively.

One possible solution to depict the flow data on the surfaces is to render glyphs directly in object space. Placing a glyph at each data sample point constructs the well-known hedgehog visualization. However, typical drawbacks are obvious. Most of the glyphs may be occluded. This is especially true of the example shown in Figure 3.10 (left). Furthermore, glyphs will either be too large, resulting in visual clutter or too small to perceive.

Secondly, parameter space is another possible approach. If a global parametrisation of the surface can be computed then the challenges posed by glyph placement are simplified. But the process of parameter sizing the surface globally is very complex. CFD data sets contain a large number of polygons. Some involve an especially complex topology. Also parametrisation may result in some distortion when the parametrised surface is mapped back on to physical (3D) space.

Image space is a good alternative to address these challenges. With the use of image space, a 3D vector field can be projected onto the 2D image plane to simplify the problem. That means only visible polygons are sampled and no extra time is spent on generating glyphs for the polygons hidden from the user's view-point. The problem of how to properly place glyphs to represent

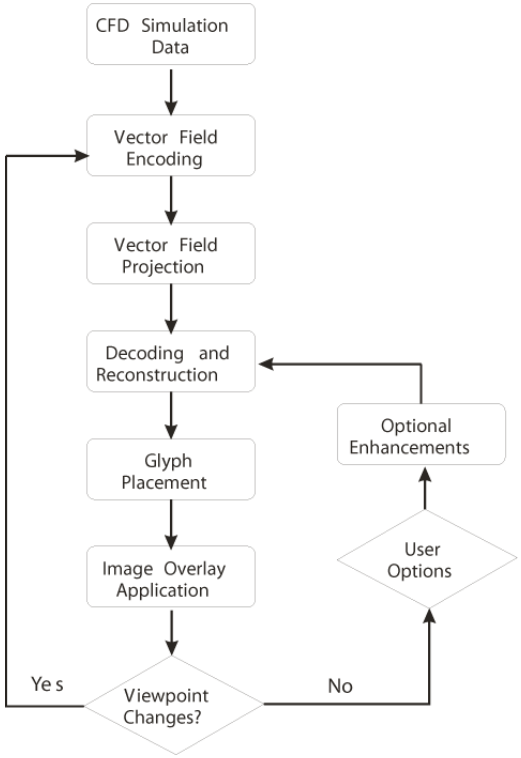


Figure 3.2: An overview chart of our algorithm for the fast generation and simple placement of vector field glyphs for surfaces.

the vector field on the surfaces in 3D space is then greatly simplified to finding the optimal placement in 2D space. By exploiting this approach, user interaction techniques, which would otherwise not be possible, may be applied. However using an image-based approach does bring new challenges, both conceptual and technical. We describe these challenges in detail in the sections that follow.

### 3.3.2 Method Overview

First the vector field is projected from 3D object space to 2D image space, this is done by exploiting graphics hardware. The vector field on the boundary surface from the CFD data set is encoded into the frame buffer. This is followed by both flow reconstruction and glyph placement. The vector field is reconstructed based on the user-defined resolution of an image-based Cartesian mesh. Then the vector glyphs are rendered along with the original surface geometry image overlay. An overview of this process is depicted in Figure 3.2. Several enhancements can be added including various interaction techniques as well as multi-resolution visualizations. Many different user options are available following the reconstruction and glyph placement phases in order to depict the vector field accurately and interactively. If the viewpoint is changed after the final glyph rendering, the next pass will start from the encoding phase therefore only a subset of the algorithm is required, starting with decoding and reconstruction if the

user-defined resampling parameters are changed. More details are given in the sub-sections that follow.

### 3.3.3 Vector Data Encoding and Projection

The vector field values of the boundary are stored at the vertices of the polygonal CFD mesh. A key step into our algorithm is to project the vector field defined at the boundary surface to the image plane. In order to realise this, we use the approach in which the $x$, $y$, and $z$ component values of the vector stored at each vertex of the boundary surface are encoded into $r$, $g$, and $b$ colour values in framebuffer respectively. The formula we use to encode the vector components is the following:

$$c_{r,g,b} = \frac{v_{x,y,z} - v_{min(x),(y),(z)}}{v_{max(x),(y),(z)} - v_{min(x),(y),(z)}} \tag{3.1}$$

Encoding the vector component values in this way yields the following benefits:

- Occluded or otherwise hidden portions of the geometry are automatically culled and are thus eliminated from any further processing.
- Linear interpolation of the vector field is performed automatically by the graphics card hardware.
- No further computation time is spent on polygons whose size is less than one pixel, the occurrence of which is high for CFD meshes (see Figure 3.1).
- The complexity of placing glyphs in object space is reduced to a much simpler problem in image space.

After the component-wise encoding of the vector values, a velocity image is generated. The interpolation of velocity image is necessary for vector field reconstruction. With the aid of hardware-assisted interpolation, we are able to decode the vector field values within the original boundary mesh polygons in addition to the vertices. Transferring the velocity image to the main memory completes the vector field projection process.

### 3.3.4 Decoding

Following the projection of the vector field defined at the surface, reconstruction for the vector field is an essential stage for developing an optimal glyph placement algorithm. Before discussing the details of the reconstruction, we describe the decoding phase for the reconstruction. The process of decoding the velocity vector values $x$ from the velocity image is performed according to the following:

$$v_x = c_r \cdot (v_{max(x)} - v_{min(x)}) + v_{min(x)} \tag{3.2}$$

Vector values of $y$ and $z$ can be achieved in the same way. The decoded vector field values used to reconstruct the flow and render the glyphs are then projected onto the image plane. Technically, the velocity at the boundary surface is defined to be zero (no slip boundary condition). What we see here is an extrapolation of the vector field just under the surface to the boundary.

### 3.3.5 Vector Field Reconstruction

After the vector field has been projected to the image plane we perform flow reconstruction. There are many different options when considering the best approach to representing the vector field including sub-sampling, using first-order or bilinear interpolation, and using box, linear and Gaussian interpolation filter functions. Representations may be optimised for speed or for accuracy. In our implementation, we offer options optimised for both. We describe the reconstruction options in more details in the sub-sections that follow.

#### 3.3.5.1 Sub-sampling

The fastest and simplest way to represent the vector field with glyphs is to use sub-sampling. A rectilinear grid, the resolution of which is defined by the user, is placed in image space. The vector field is then sampled at the centre of each grid cell using the decoding described in the previous section. A vector glyph is rendered at the centre of each grid cell based on the sample as shown in Figure 3.3. In the implementation, some measure must be taken to ensure that no glyphs are rendered at cell centres with the background colour. This can be handled either by an explicit test for background colour or by testing the depth buffer for its maximum value. We chose the option of testing the z-buffer value. No glyphs are rendered for cell centres where $z_{depth} = 1.0$. The advantage of this approach is speed. The user may sample the vector field at several frames per second, rotating the resampling grid, changing its resolution and sliding the grid in image space in order to place the glyphs precisely where the user chooses. We note that these user options of specifying the resampling grid resolution, translating and rotating the grid are not arbitrary. They were specifically requested by fluid engineers we talked to when developing an earlier tool [Lar03]. Fluid engineers want precise control over glyph placement and resampling parameters.

#### 3.3.5.2 Average-based representation

Although the above sub-sampling approach provides the desired speed to investigate the simulation result, it does not construct the most accurate representation of the flow since only a sub-set of samples are taken into account. Therefore we also provide the option of rendering glyphs based on the average vector field value of each user-defined resampling grid cell. Instead of sampling the vector field only at the centre point of each cell, this approach samples the vector field pixel-by-pixel over the whole cell. However, a complication can arise with this approach due to discontinuities on the surface. If we do not take edges in the geometry into

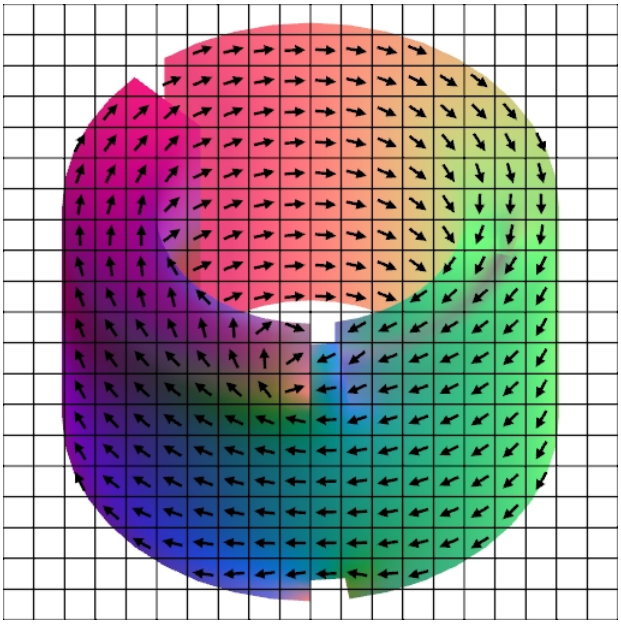

Figure 3.3: Glyphs are rendered at the resampling grid cell centres for visible portions of the boundary geometry and its associated vector field. This example shows a simple ring geometry with a $20^2$ resampling grid.

account then we may end up including undesired velocity values into the final average. To address this problem, we also sample the depth value at each pixel during the average computation. If the depth values of the centre grid cell point, $\mathbf{p}_{center}$, and another sample within the same cell, $\mathbf{p}_{sample}$, differ by more than a threshold value, $\varepsilon_{depth}$, then the pixel with $\mathbf{p}_{sample}$ is not included in the final average. This approach separates the image into distinct regions for accurate flow reconstruction. We emphasise that, in all of our filters, any pixels beyond edge discontinuities are left out of the final result. We do so in order to separate different regions of the geometry that may be at the same depth.

The averaging approach provides high accuracy for representing the flow. The user can gain a precise overview of the vector field based on the boundary surface via the intuitive centre glyph without missing any potentially interesting values. One drawback with this approach however is its cost in computing time.

#### 3.3.5.3 Reconstruction using filters

In order to construct an optimal approach which combines the speed of sub-sampling and the accuracy of an averaging-based representation, we use footprint functions with various interpolation filters. We can use various filters (or filter kernels) to reconstruct the flow at the boundary surface. There are many possible interpolation functions, namely $n^{th}$ order filters where $n$ = 0,1,2,... all with relative advantages and disadvantages (see [KM05] for an overview). We have implemented linear (or first order) and Gaussian filters in our framework. In order to

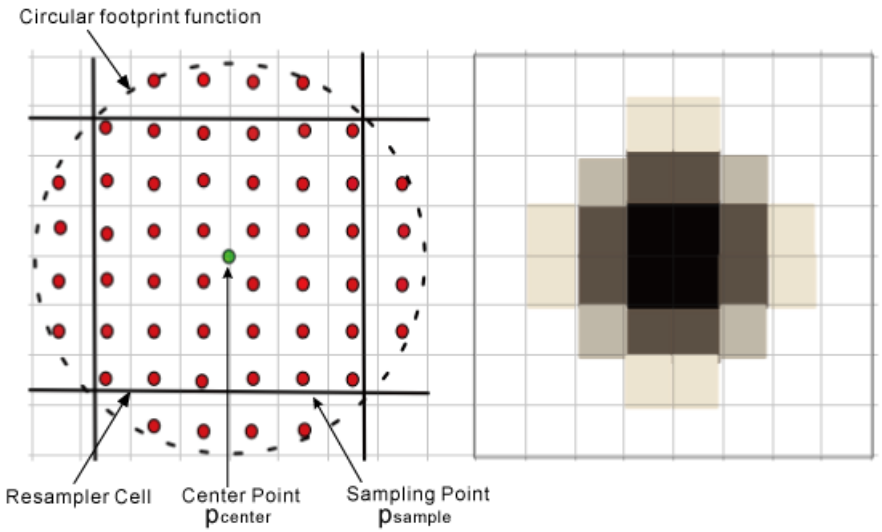


Figure 3.4: Left is the circular footprint function using $8^2$ sized footprint-grid. On the right side, an $8^2$ footprint look-up table with a Gaussian filter kernel.

accelerate the computation, we have implemented these filters as footprint functions similar to Westover [Wes90]. Our image-based resampling approach makes this a natural choice for simple, symmetric 2D footprint functions. Using a 2D footprint function with an elliptical extent, vector field reconstruction is accelerated by pre-computing the contribution of each sample in the footprint's extent and storing it in a look-up table. The extent of each footprint simply encompasses one grid cell in the user-defined resampling grid as shown in Figure 3.4. Furthermore, the footprint represented by each glyph is the same except for a screen space offset. An illustration showing an example footprint and filter kernel is shown in Figure 3.4.

While we believe the use of filter functions implemented using footprint tables represents a balanced trade-off between accuracy and speed, complications arise due to discontinuities in the vector field stemming from mesh boundaries and edges. If we simply contribute each sample's vector values to the filter function we may not get an accurate visualization of the flow at each resampling grid cell centre. This is because the centre points may lie on a different portion of the mesh than surrounding sample points. Figure 3.5 shows a centre-point $\mathbf{p}_{center}$ where a glyph representing the flow is rendered. Also shown is the extent of the footprint function surrounding $\mathbf{p}_{center}$. The extent of the footprint encompasses more samples including undesired ones like $\mathbf{p}_{i,j}$ (shown in red) around $\mathbf{p}_{center}$. Thus we must introduce an additional filtering operation to disregard the contribution of red points $\mathbf{p}_{i,j}$ to $\mathbf{p}_{center}$. We achieve this by taking the depth gradient between $\mathbf{p}_{center}$ and $\mathbf{p}_{i,j}$ into account. If

$$\varepsilon_{depth} > |depth(p_{center}) - depth(p_{i,j})| \tag{3.3}$$

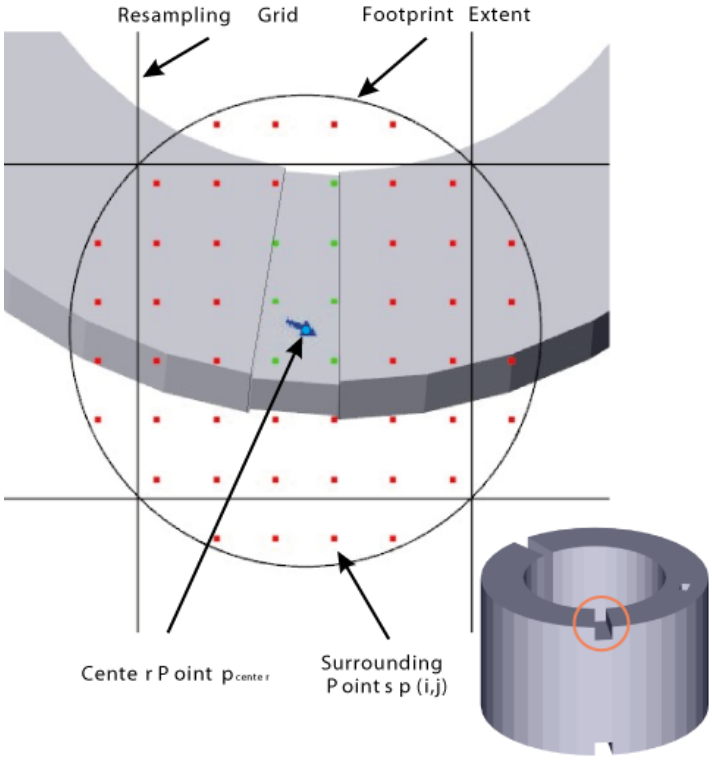


Figure 3.5: The circular footprint function with Gaussian filter kernel is applied at the surface of the ring which has edges as shown. The blue point is the centre point of the resampling cell. Green and red points are sample points defined by the footprint function. Green contribute to the final representation whilst red have been filtered out by equation (3).

then samples $\mathbf{p}_{i,j}$ contribution is not added to the final value at $\mathbf{p}_{center}$. Here $\varepsilon_{depth}$ is a user-defined threshold. In practice, we have found a value of 0.003 to be a good threshold value.

In order to implement the filtering operation, we use a neighbour-based selection method to select sampling points $\mathbf{p}_{i,j}$accurately and efficiently. Equation (3) helps us detect relevant portions of the boundary geometry to sample and which samples to filter out. We test each sample within the extent of the centre point's footprint kernel. We start at $\mathbf{p}_{center}$ and test each adjacent neighbours in a looping fashion. If we find contributing points $\mathbf{p}_{i,j}$ in the first loop of neighbours surrounding $\mathbf{p}_{center}$ then we increase the radius of the loop by one unit and repeat the process. If a discontinuity is found, all neighbours beyond the discontinuity are filtered out. This iterative process stops when either (1) a loop with no contributing samples is found or (2) the extent of the footprint is reached. An illustration is shown in Figure 3.6. $\mathbf{p}_{i,j+1}$ is a neighbour from $\mathbf{p}_{i,j}$. Once $\mathbf{p}_{i,j+1}$ is rejected, the unvisited neighbour points beyond $\mathbf{p}_{i,j+1}$ are not needed.

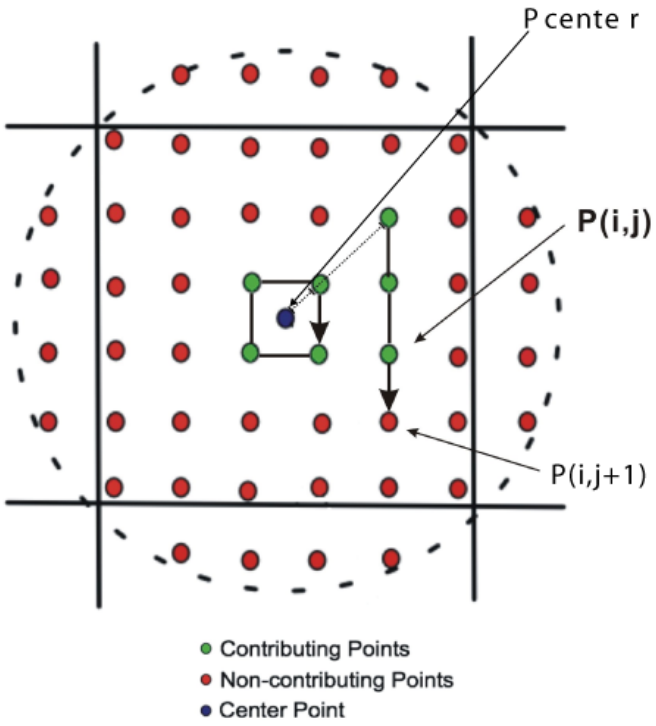


Figure 3.6: Candidate samples are found by searching the kernel extent in a looping fashion starting at $\mathbf{p}_{center}$. Sample points $\mathbf{p}_{i,j}$ do not contribute if equation (3) is true.

### 3.3.6 Image Overlay Application

Along with the glyph placement, an optional image overlay is used for the resulting visualization of the vector field on surfaces by applying colour, shading, or any attribute mapped to colour. In the implementation, we generate the image overlay following the construction of the velocity image once for each static scene. Once the view-point is changed, image overlay needs to be regenerated. By exploiting OpenGL's glDrawPixels() function, rendering an image is much faster than rendering the complex triangulated object each time a user parameter is changed.

### 3.3.7 Glyph Placement, User Options and Enhancements

Many user options can be applied to enhance the usability and flexibility of glyph placement. The user options enable engineers to gain more insight of the flow on boundary surfaces. We describe these user options in more detail individually. Many of the user-interaction techniques we describe would not be possible with an object space approach.

- User-Defined Resampling Grid Resolution: The user may interactively specify the resolution of the resampling grid in image space. The higher the resolution, the more accurate the linear and Gaussian filter kernels become. The lower the resolution, the faster the

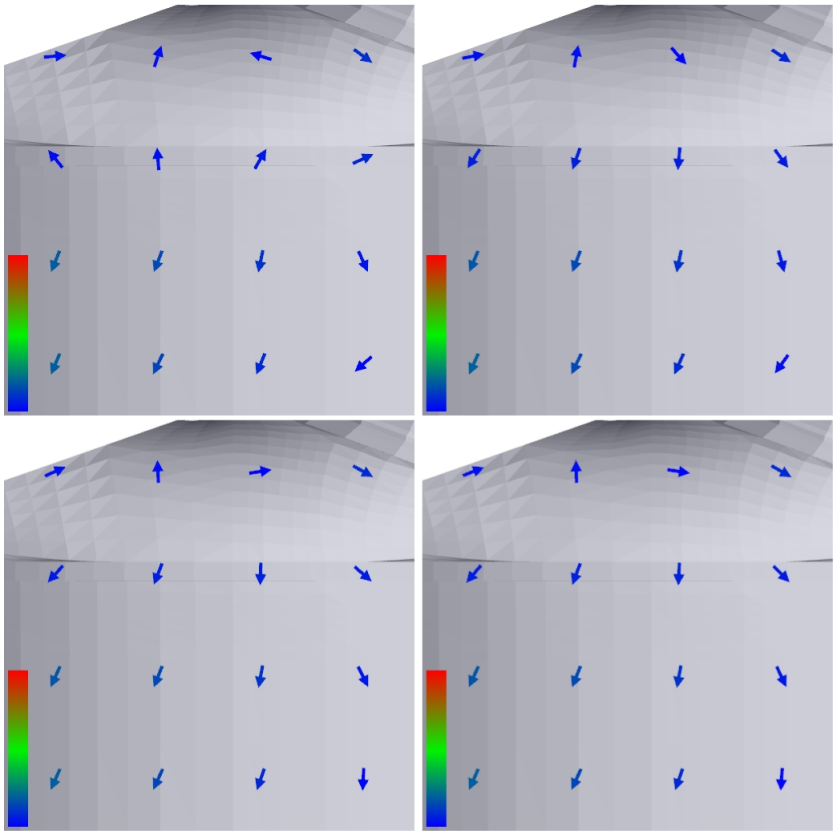

Figure 3.7: A close-up view of the vector field on the surface of an 79K polygonal gas engine simulation mesh. Here we illustrate a comparison of various vector field reconstruction options: (top, left) Sub-sampling, (top, right) Averaging, (bottom, left) Linear filtering and (bottom, right) Gaussian filtering. The accuracy of depicting vector field around the edge between the cap and the cylinder via Linear and Gaussian filter is quite similar to the result from the averaging which is the most accurate, but with much less computational cost than averaging.

interaction becomes. Users desire faster speed for interaction and higher accuracy for analysis and presentation.

- Grid Translation and Rotation: In order to precisely place the vector field glyphs at user defined points, we provide the option of moving the resampling grid around as well as rotating the grid around the centre point. These options were specifically requested by CFD engineers, as well as control of the resampling grid resolution.

- Glyph Scaling: This user option adjusts the size of each glyph in order to avoid overlapping and occlusion of glyphs. With the help of glyph scaling, glyphs can be rendered in proper size to make the vector field easily perceived.

- Multi-Resolution Flow visualization: Our tool also offers a multi-resolution representation of the flow. The user may define a sub-grid, with its own higher (or lower) grid

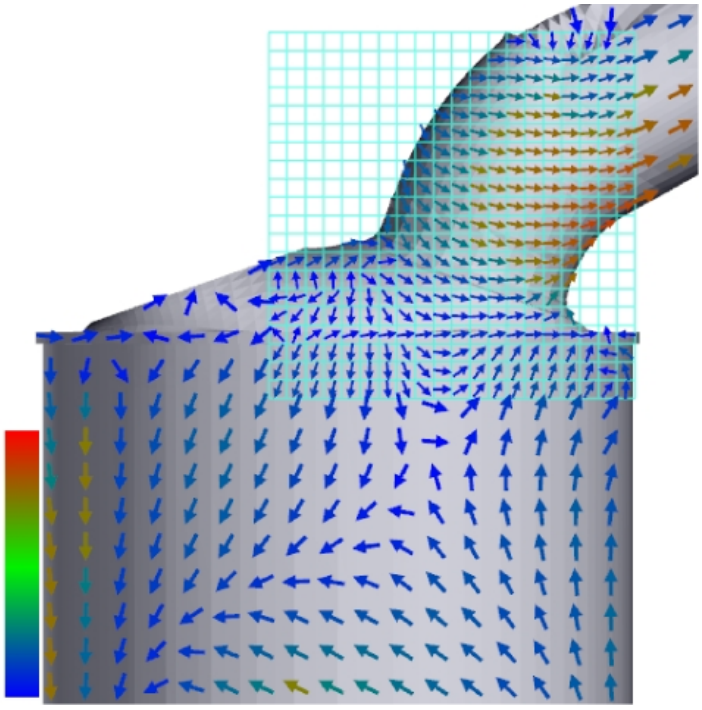

Figure 3.8: A coloured multi-resolution visualization of low resolution and high resolution glyphs applied with Gaussian filter is rendered to visualize the flow at the surface of a gas engine simulation. Colour is mapped to velocity magnitude.

resolution. The user can then position this sub-grid over any area of interest interactively. All of the options built into our framework can be applied to the sub-grid as well: different reconstruction options, as well as grid scaling and rotation (see Figure 3.8).

The accuracy of the vector field representation increases automatically when the user zooms in on a boundary. The higher sampling frequency is a natural benefit of the approach. One of the consequences of using an image-based approach is that the glyphs remain fixed in their positions as the object moves under rotation or translation. This can be handled by unprojecting the vector glyphs back to the object-space surface. However glyphs are not generated for portions of the geometry that are occluded or outside the current view. Finding a perfect solution to this problem is a part of our future work.

## 3.4 Performance and Results

As our glyph-based visualization is focused on unstructured, adaptive resolution boundary meshes from the complex CFD data sets, we evaluate our visualization on simulation data sets with these characteristics. Figure 3.9 shows a comparison of brute-force hedgehog placement and our glyph-based method applied on a surface of an intake port mesh composed of 221K polygons. The intake port has highly adaptive resolution boundary surface and for which no

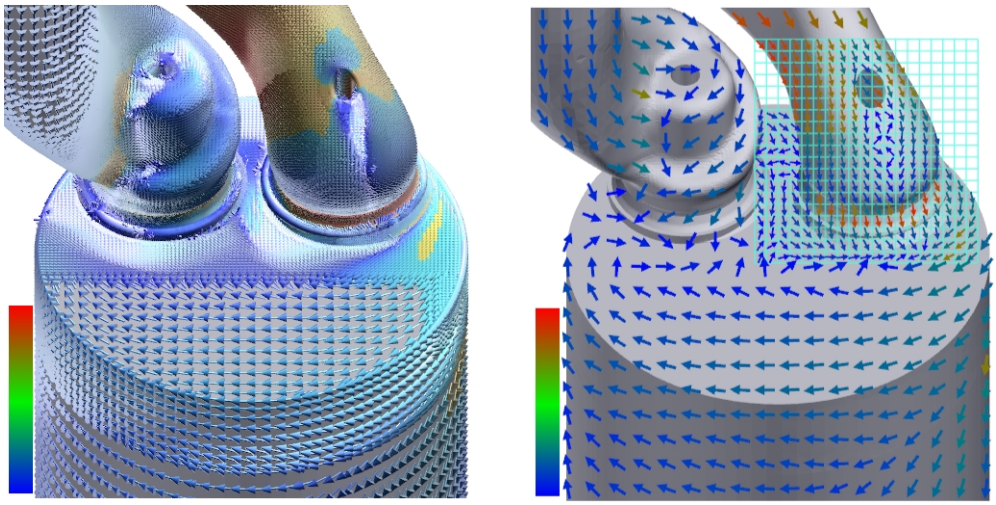

Figure 3.9: The comparison of brute-force hedgehog visualization (left) and our multi-resolution glyph-based visualization which is powered by Gaussian filter (right) applied in order to depict the flow at a surface of an intake port mesh composed of unstructured, adaptive-resolution 221K polygons. Notice how the glyphs are cluttered using the hedgehog approach (left). Also notice that the uneven appearance resulting from the underlying mesh that have nothing to do with the actual flow. Glyphs are colour-coded according to velocity magnitude.

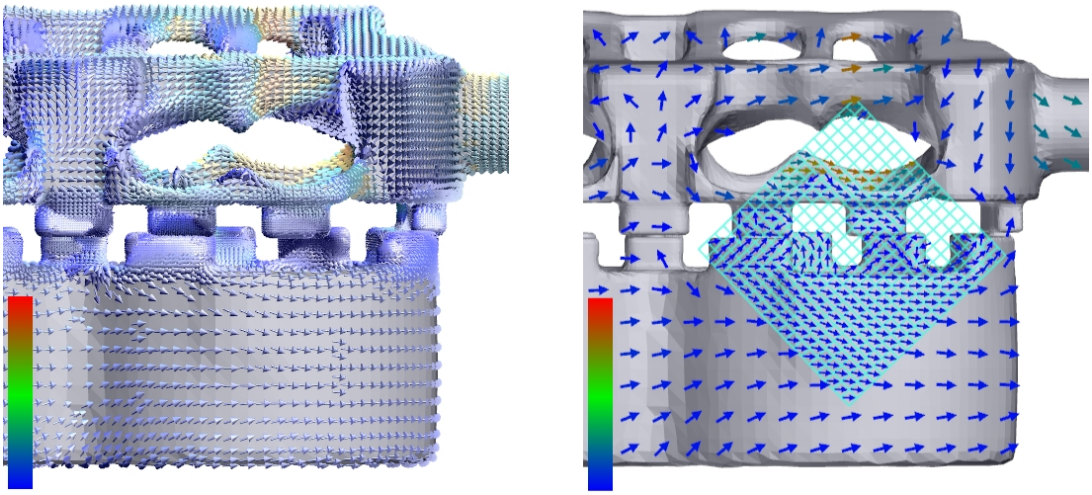

Figure 3.10: Another comparison of brute-force hedgehog flow visualization (left) and glyph-based flow visualization which is powered by Gaussian filter and multi-resolution (right) applied at the surface of a cooling jacket - a composite of 228K unstructured, adaptive-resolution polygons.

global parametrisation is easily computed. As we can see from the left picture, most glyphs overlap or are occluded. Using a hedgehog approach 664k glyphs are rendered. However, our approach renders only about 400 glyphs. Also, the distribution of glyphs is uneven. This

| Data Set | Resampling Rate (FPS) | | | |
|---|---|---|---|---|
| | Sub-sampling | Average | Linear | Gaussian |
| Ring (10K) | 59(29) | 2.5(2.0) | 30(17) | 30(16) |
| Combustion Chamber (79K) | 59(20) | 1.9(1.8) | 29(11) | 29(12) |
| Intake Port (221K) | 59(11) | 2(1.5) | 29(8) | 30(7.5) |
| Cooling Jacket (228K) | 59(9.5) | 1.9(1.7) | 29(8.2) | 29(7.8) |

Table 3.1: Sample frame rates for the visualization algorithm applied with $15^2$ fixed resolution of user-defined resampling grid with about 75% image space area covered. An image of $512^2$ pixels is used.

| Data Set | Resampling Rate (FPS) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | $5^2$ | | $10^2$ | | $20^2$ | | $50^2$ | |
| | SS | Gaussian | SS | Gaussian | SS | Gaussian | SS | Gaussian |
| Ring (10K) | 59(29) | 59(28) | 59(29) | 59(20) | 59(29) | 15(11) | 58(23) | 2.5(2.4) |
| Combustion Chamber (79K) | 59(20) | 59(19) | 59(20) | 58(20) | 59(20) | 13(9.5) | 58(20) | 2.3(2) |
| Intake Port (221K) | 59(11) | 59(10) | 59(11) | 58(8.5) | 59(9.7) | 15(6.2) | 58(10) | 2.4(1.7) |
| Cooling Jacket (228K) | 59(10) | 58(9.4) | 59(9.5) | 58(7.4) | 59(10) | 14(6.5) | 59(9.4) | 2.4(1.9) |

Table 3.2: Sample frame rates for the visualization algorithm applied with sub-sampling (SS), a Gaussian filter function (Gaussian), varying the resolution of user-defined resampling grid and about 75% image space area covered.

appearance is the result of the underlying mesh and have no relation to the flow itself. On the right, our method places glyphs in an intuitive and efficient fashion enabling engineers to get a fast and clear overview of the flow on the surface. At the same time, with the aid of a multi-resolution option, more details on the interesting areas can be obtained. The vector field on the complex cooling jacket boundary meshes (from Figure 3.1) can be also efficiently visualized by our intuitive glyph-based method (Figure 3.10), especially compared to a hedgehog visualization. Because of the fast speed of our method this glyph-based visualization allows users to translate, rotate and zoom in the object interactively to get better insight of the CFD data sets. We encourage the reader to view the supplementary video for more results.

In order to compare the various reconstruction options implemented in our framework, we evaluated sub-sampling, averaging, the linear filter function and the Gaussian filter function on a PC with an Nvidia Geforce 8600GT graphics card, a 2.66 GHz dual-processor and 4 GB of

RAM. The performance times reported in Table 1 were obtained using a fixed $15^2$ resolution resampling grid with about 75% image space coverage. The first times illustrated in the FPS column are for the static case of (no change to the view point) only changes to the user options from section 3.7. The times shown within parenthesis depict the dynamic case of changes to the viewpoint. In terms of the overview chart presented in Figure 3.2, the construction of a velocity image, image overlay, reconstruction of vector field, as well as glyph placement need to be computed in the dynamic case. From Table 1, we can see that sub-sampling is the fastest while averaging is the slowest. Linear and Gaussian filter functions are in the middle as a balance between computation speed and high accuracy.

Table 2 shows performance times in order to compare sub-sampling and a Gaussian filter function with various resolutions of the user-defined resampling grid. The performance time of our algorithm depends on the resolution of the user defined resampling grid used to place relevant glyphs and the number of polygons in the original surface mesh. But in the static case, the algorithm depends mostly on the coverage of the area in image space (75% is covered in our cases) rather than the number of polygons in the original surface mesh. Table 2 indicates that the higher the resolution of the resampling grid, the lower the performance. When the resolution is higher than $20^2$, the performance speed drops off. Hence we use a $20^2$ default resolution like in Figures 3.9 and 3.10 of the resampling grid for good performance and high accuracy. In general, our goal is to provide users for fast performance times for interaction and exploration. For presentation and analysis, users then have the option of increasing the accuracy.

## 3.5 Conclusion and Future Work

In this chapter we propose a fast and simple glyph placement algorithm for investigating and visualizing boundary flow data based on unstructured, adaptive resolution boundary meshes from CFD. We show that the algorithm effectively and automatically places glyphs at evenly-spaced intervals, independent of geometric and topological complexity of the underlying adaptive resolution mesh. We have also demonstrated that the spatial resolution and precise location of the glyph placement can be interactively and intuitively adjusted by the user in order to gain better visualization results. In addition, multi-resolution visualization can be applied to highlight details in areas deemed interesting by the user. Furthermore, the efficiency of our algorithm is reinforced by the fact that no computation time is wasted on occluded polygons or polygons covering less than one pixel. Due to the efficiency and speed of the algorithm user interaction such as zooming, translating and rotation is enabled. The framework supports various representations of the flow optimised for both speed and accuracy. No pre-processing of the data or parametrisation is required.

We would like to extend the work to visualization of unsteady 3D flow. Challenges stem from both the resampling performance time and perceptual issues. Future work also includes using floating-point texture in order to encode and decode the vector field.

*Chapter 4*

# Mesh-Driven Vector Field Clustering and Visualization: an Image-Based Approach

**Contents**



> "Every day you may make progress. Every step may be fruitful. Yet there will stretch out before you an ever-lengthening, ever-ascending, ever-improving path. You know you will never get to the end of the journey. But this, so far from discouraging, only adds to the joy and glory of the climb."
>
> - Sir Winston Churchill [6]

LAST chapter presents an interactive evenly spaced glyph placement algorithm for visualizing the flow based on complex boundary surfaces from CFD datasets. Although a quick and intuitive visualization of the boundary flow field can be obtained, the area with more important flow such as divergent and convergent flow can not be easily highlighted

[6] Sir Winston Churchill (1874 - 1965), a British politician and statesman known for his leadership of the United Kingdom during the Second World War.

by evenly spaced glyphs. So a more suggestive and effective flow visualization technique is needed. Out of the wide variety of existing flow field visualization techniques, vector field clustering algorithms offer the advantage of capturing a detailed picture of important areas of the domain while presenting a simplified view of areas of less importance. This chapter presents a novel, robust, automatic vector field clustering algorithm that produces intuitive and insightful images of vector fields on large, unstructured, adaptive resolution boundary meshes from CFD. Our bottom-up, hierarchical approach is the first to combine the properties of the underlying vector field and mesh into a unified error-driven representation. In order to visualize the variation of velocity magnitude and direction within each cluster, we also introduce novel visualizations of clusters inspired by statistical methods. We apply our method to a series of synthetic and complex, real-world CFD meshes to demonstrate the clustering algorithm results. This chapter has been published as a research paper in IEEE TVCG journal [7].

## 4.1 Introduction

Over the last two decades, vector field visualization has developed very rapidly. Its applications range from the automotive industry to medicine. Vector field visualization provides solutions that enable engineers to investigate and analyse critical features and characteristics of the flow from computational fluid dynamics (CFD). However, visualizing and analysing vector fields on complex boundary surfaces from CFD still remains a challenging task due mainly to the large, unstructured, adaptive resolution characteristics of the meshes used in the modelling and simulation process (see Figure 4.1). Unstructured meshes necessitate either computationally expensive neighbour searching or the explicit storage of mesh topology, which, for large meshes, can cause non-trivial memory overhead. Adaptive resolution meshes cause problems with scale. Some portions of the mesh may be too fine to visualize concurrently with coarse resolution regions. Additionally, as the size of simulation datasets from CFD increases, so does the demand for visualization methods that quickly depict vector fields in a simplified and insightful manner. Finding a good solution that handles both complex boundary surfaces from CFD and provides a suggestive picture of vector fields is a challenge that we address.

Out of all the possible visualization techniques that can be used to simplify and present simulation results, vector field clustering algorithms are a desirable solution which offer the advantage of presenting a detailed picture of important or complex areas of the domain while depicting a simplified representation for areas of less importance. In general, clustering algorithms are based on agglomerative hierarchical grouping techniques which are widely used in the information visualization domain [XW05]. However, vector field clustering algorithms have received relatively little attention. In this chapter, we present a novel, automatic vector field clustering algorithm that handles vector fields on complex boundary surfaces from CFD. It produces simplified but insightful images based on an error-driven distance measure.

[7] Published as: Zhenmin Peng, Edward Grundy, Robert S. Laramee, Guoning Chen, and Nick Croft, **Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach**, *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, to appear, 2011.

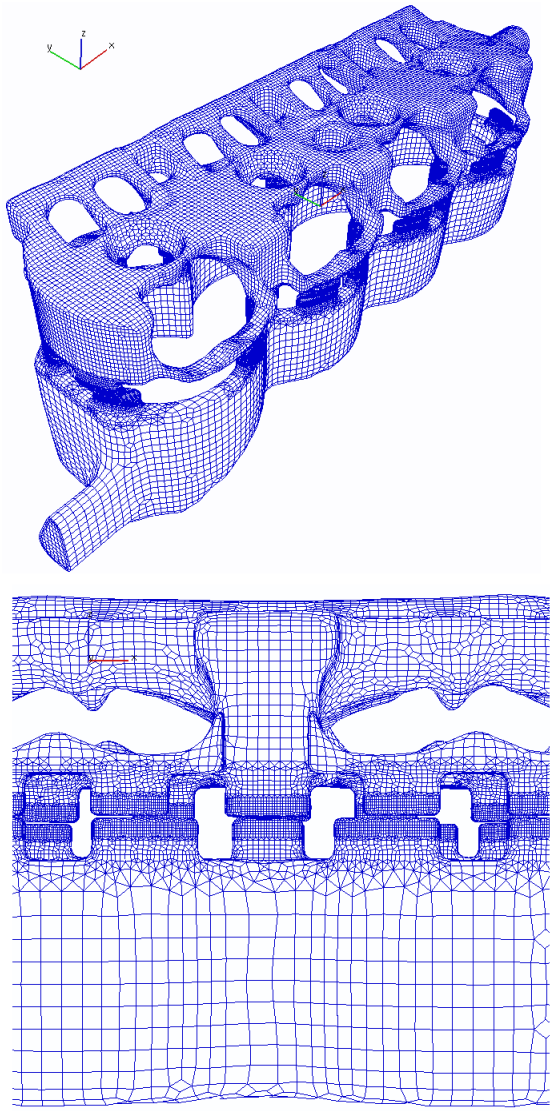

Figure 4.1: The unstructured adaptive resolution boundary grid of a cooling jacket from a CFD simulation. The upper image is an overview of the boundary mesh, and the bottom is a close-up. These images illustrate how complex a typical mesh from CFD can be. The finest mesh resolution is used at the gasket between the cylinder block (bottom) and the cylinder head (upper component half). The gasket is modelled with the finest resolution mesh because the gasket holes are very small with high curvature. Polygons in this mesh differ in size by six orders of magnitude.

One of the primary motivations behind our method stems from the semantics of meshes from CFD. When constructing a model, portions of the geometry which are more important may be modelled with a very fine resolution in order to obtain simulation results with higher accuracy. Constituents of a model deemed less interesting to the engineer may be represented with a coarser resolution mesh in order to speed up the simulation process. The resolution of a region

of the mesh can reflect the importance of that region to the engineer; thus, it is an encoding of the engineers knowledge of the problem. More than 50% of the time spent on an industry grade CFD process (meshing, simulation, and visualization) is devoted to defining and generating the mesh [VM96]. The mesh resolution can also depend on the size of the individual components found within the model itself. Very small constituents of the geometry are treated with detailed, high resolution meshes. Larger features are modelled with a coarser resolution. Our clustering algorithm and resulting visualization takes both the semantics of the underlying mesh and the characteristics of the vector field into account.

The main benefits and contributions of this method are:

- A novel clustering algorithm which couples the properties of the vector field and mesh model into a unified clustering distance measure. The model and implementation are general enough such that any arbitrary attribute of the CFD simulation results may be incorporated into the clustering distance measure.
- A cluster hierarchy is generated automatically according to the importance of the underlying mesh and the properties of the vector fields. More detailed areas are emphasised while less important or simpler ones are simplified in a single image.
- Large, adaptive resolution meshes are handled efficiently because clusters are never generated for occluded or otherwise invisible regions of vector fields on the surfaces.
- We introduce novel visualizations of cluster attributes including $\theta$-range and $|\mathbf{v}|$-range glyphs. Our approach enables various clustering and visualization resolutions, optimised for either speed or accuracy.

The approach enables a range of user-controlled visualization parameters such as automatic glyph and streamlet placement options. Glyphs can be automatically placed at the centre points of clusters. Streamlets can be seeded and traced from cluster centre points to render an intuitive overview of vector fields, see Figure 4.2 (d). The algorithm is robust because it handles meshes with holes, discontinuities, and jagged edges. Furthermore, the algorithm does not rely on a parametrisation of the surface. The key to the technique's robustness and simplicity is its image space approach. However, in order to achieve these benefits several challenges, both technical and perceptual, must be overcome.

The rest of the chapter is organised as follows: Section 4.2 provides an overview of related research work. The algorithm and user options are described in multiple stages in Section 4.3. Section 4.3 also describes two important data structures used to simplify and accelerate the process. Section 4.4 gives the performance and visualization results. The CFD domain expert review on this method is presented in Section 4.5. Conclusions and suggestions for future work are presented in Section 4.6.

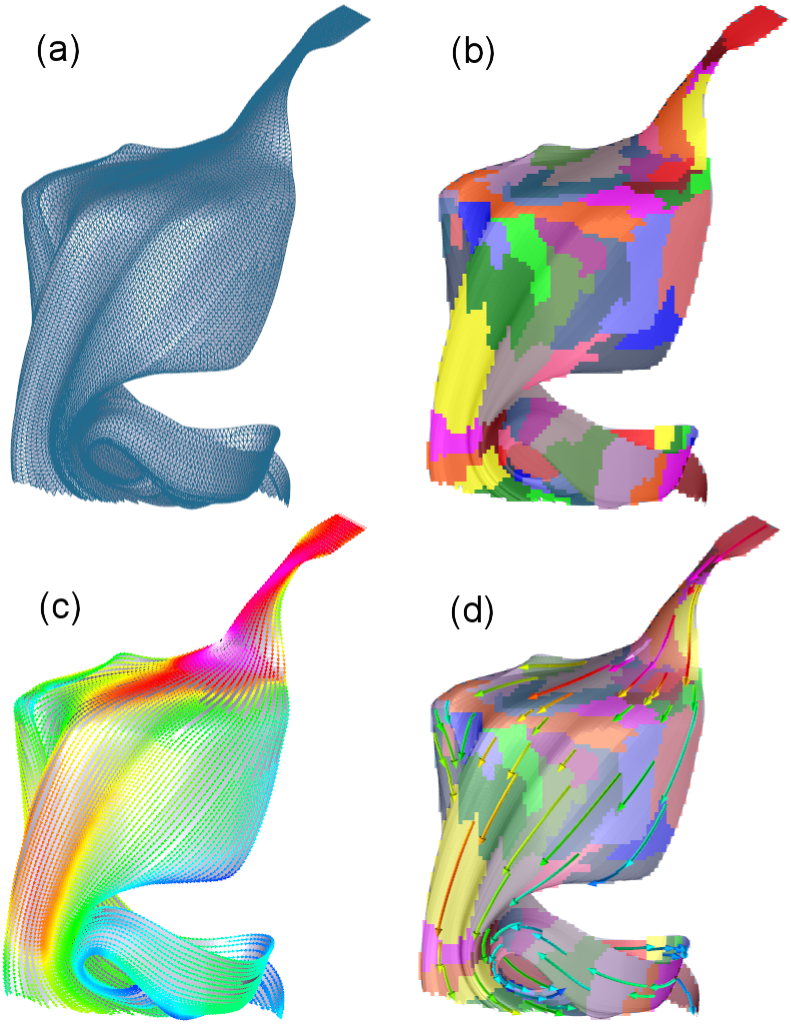


Figure 4.2: The visualization of flow at the stream surface of a gas engine simulation [LGSH06, LWSH04]. (a) The stream surface mesh is composed of unstructured, adaptive-resolution polygons. For stream surface generation, we use the algorithm of Garth et al. [GTS+04] which handles unstructured, adaptive resolution meshes. (b) Clusters rendered with $\boldsymbol{\varepsilon} = 15\%$. (c) The comparison of glyph-based hedgehog visualization and (d) cluster-based streamlet visualization with semi-transparent clusters. Here we can visualize vector field clusters on stream surfaces for the first time. The colour of (c) glyphs and (d) streamlets is mapped to velocity magnitude.

## 4.2 Related Work

Xu and Wunsch II [XW05] present a comprehensive and systematic survey focused on scalar clustering algorithms rooted in statistics, computer science, and machine learning. They generalise the clustering analysis procedure using four steps: feature selection or extraction, clustering algorithm design (or selection), cluster validation, and result interpretation, they detail clustering algorithms in terms of the nature of clusters. To compare different clustering algo-

rithms, applications to some benchmark data sets are presented.

Clustering algorithms can be divided into two groups: either based on constructing a hierarchical structure or based on partitioning into a collection of disjoint sets. Hierarchical clustering algorithms are mainly classified as agglomerative methods and divisive. Since the agglomerative methods can generate flexible clustering using binary trees and provide very informative descriptions, they are employed to develop some vector field clustering visualizations, such as Telea and Van Wijk [TvW99] and Heckel et al. [HWHJ99]. The computational cost for hierarchical clustering can be expensive (e.g. $O(N^2)$ where $N$ is the number of data samples.) so some of the early hierarchical clustering methods are not capable of handling large-scale data sets. To address this problem, some efficient hierarchical clustering methods are introduced like BIRCH [ZRL96] which utilises the clustering feature tree to improve the robustness and reduce the computational complexity to $O(N)$.

There have been very few previous clustering methods targeted at vector fields, especially when compared to the large volume of flow visualization literature in general [LHD$^+$04, MLP$^+$10, PVH$^+$03].

Telea and Van Wijk [TvW99] present a hierarchical clustering method which automatically places a limited number of glyphs in a suggestive manner to represent the vector field. It is the first algorithm of its kind and produces both global and local information in the same image. In terms of the clustering algorithm, two candidate clusters are selected and merged together in a bottom-up fashion. During this process, a similarity measure is used to evaluate which clusters should be merged. An error measure based on local vector magnitude and direction is introduced to define how a new cluster is merged from two existing ones.

Heckel et al. [HWHJ99] present a method to visualize discrete vector fields in a hierarchical fashion. This method uses a clustering approach to segment the original vector field into a series of disjoint clusters. The algorithm is applied in a top-down fashion. Firstly, points from the original vector field data are treated as a single cluster. Then a procedure of splitting clusters is applied using a weighted best-fit plane which partitions the space into convex regions (sub-clusters). Each region has an error measure which calculates the differences between the original discrete vector field and the simplified vector field. With the use of the error measure approach, the recursive procedure of splitting clusters can be terminated when a user-defined threshold value is met. This method is free of discretization introduced by a regular grid or sub-sampling for multi-resolution analysis.

Garcke et al. [GPR$^+$01] present a multiscale method for vector field clustering in 2D and 3D space. This continuous clustering method is inspired from the well-known physical phase separation clustering model - the Cahn-Hilliard model [CH58]. In order to classify and enhance the correlation in the cluster sets effectively, this phase-separation-based continuous clustering method formulates the clustering problem as a diffusion problem rather than a merging or a splitting problem. In accordance with the underlying physical data and based on the evolution function, segments of the flow field are extracted and classified depending on their location

and orientation. Then, a skeletonization approach is applied to highlight the essential features of the refined cluster sets. Finally, various geometric representations are adopted to render the highlighted skeleton in an intuitive way. All the results feature uniform resolution, rectilinear grids.

Griebel et al. [GPR+04] present a vector field clustering method based on algebraic multigrid [UA01]. Each sample in the flow field is represented by a tensor stiffness matrix that encodes the local properties of the flow field. The algebraic multigrid technique operates on these tensor matrices in order to construct a vector field hierarchy which describes the flow structure. The method is geometry-free and is demonstrated on 2D and 3D vector fields.

Du and Wang [DW04] present a Centroidal Voronoi Tessellation based method for the simplification and the visualization of vector fields. In the method, the generators of the tessellation are treated as centres of the surrounding Voronoi regions which are deemed as the clusters. A distance function in both the spatial and vector space is applied to measure the distances between the centre and surround clusters and thus determine the set of generators with the smallest distances as the final centres of the clusters. The proposed method is to minimise the global error function. The algorithm is demonstrated on rectilinear 2D and 3D vector fields.

McKenzie et al. [MLD05] present a global vector field clustering method as an extension to Du and Wang's previous work [DW04]. By taking the direction, gradient, curl and divergence into account to drive the clustering process, this method can provide a meaningful segmentation of the input vector field. It is one of the few methods that is demonstrated on tetrahedral meshes. The semantics of the uniform resolution data are not considered.

There are several recent texture-based algorithms that visualize flow at the boundary surfaces of simulation data [LvWJH04, vW03, WE04b] including streamsurfaces [LGSH06]. However, texture-based methods are different class of visualization techniques from vector field clustering with different characteristics. Firstly, texture-based methods treat the flow *uniformly*. The resolution of the textures is generally constant. The one exception is that of Telea and Strzodka[TS06]. Secondly, texture-based methods fail to depict the downstream direction of the flow in a still image. Users must rely on animations of the flow for visualizing the downstream direction. Thirdly texture-based methods do not reflect the semantics of the underlying mesh in the resulting visualization. For a survey of texture-based methods see Laramee et al. [LHD+04].

Another class of flow visualization techniques focuses on the *topology* of the vector field [LHZP07, HH91, TS01]. Topological methods do not depict the downstream direction of the flow in a still image. They generally rely on another visualization method to do so. Furthermore, topological methods present challenges with respect to interpretation. Only specialists will fully understand the information conveyed by a topological skeleton of the flow. This class of methods also ignores the semantics of the underlying mesh. Visualization of the topological skeleton of the flow is not the goal of the work presented here. If a user would like to visualize the topological skeleton of a vector field, we recommend an algorithm tailored specifically to

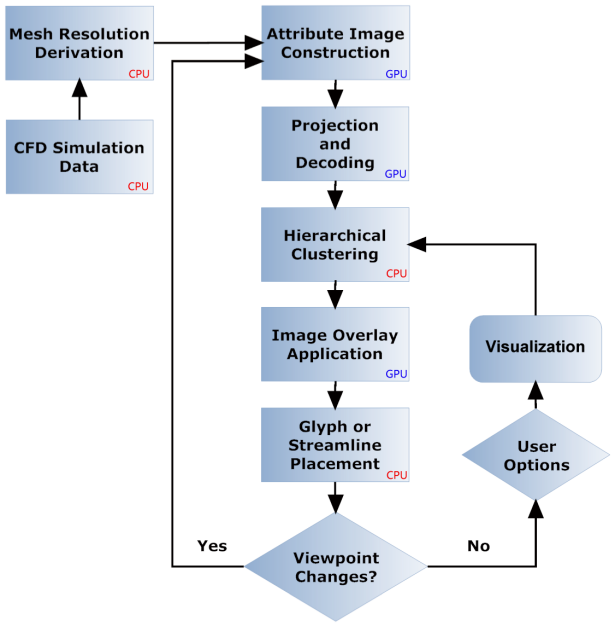


Figure 4.3: An overview chart of our mesh-driven vector field clustering algorithm for surfaces.

do so [CMLZ08, CML+07]. The method here, on the other hand, generates both detailed and summary information in the same visualization. See Laramee et al. for a overview of topology-based approaches [LHZP07].

To our knowledge the algorithm presented here is the first vector field clustering method that takes the properties of the underlying mesh into account in addition to the underlying vector field. This is also the first of its kind to target surfaces from CFD and to handle large, unstructured, adaptive resolution meshes efficiently. In general, previous work deals with data on structured grids.

## 4.3 Mesh-Driven Vector Field Clustering for Surfaces

In this section, we present details of the our algorithm including both the model and its implementation.

### 4.3.1 Method Overview

Figure 4.3 shows an overview of our method. The input includes a generic triangular mesh as a geometric representation for boundary meshes. Each vertex $i$ has a position $\mathbf{r}_i = (r_{ix}, r_{iy}, r_{iz})$, a normal vector $\mathbf{n}_i = (n_{ix}, n_{iy}, n_{iz})$, a velocity vector $\mathbf{v}_i = (v_{ix}, v_{iy}, v_{iz})$ and the neighbouring topology. For simplicity we assume that the velocity is confined to the surface (possibly by projection), i.e. $\mathbf{v}_i \cdot \mathbf{n}_i = 0$. Note that any CFD simulation attribute can be stored at the vertices such as temperature, pressure, kinetic energy etc. This is also true of derived attributes like

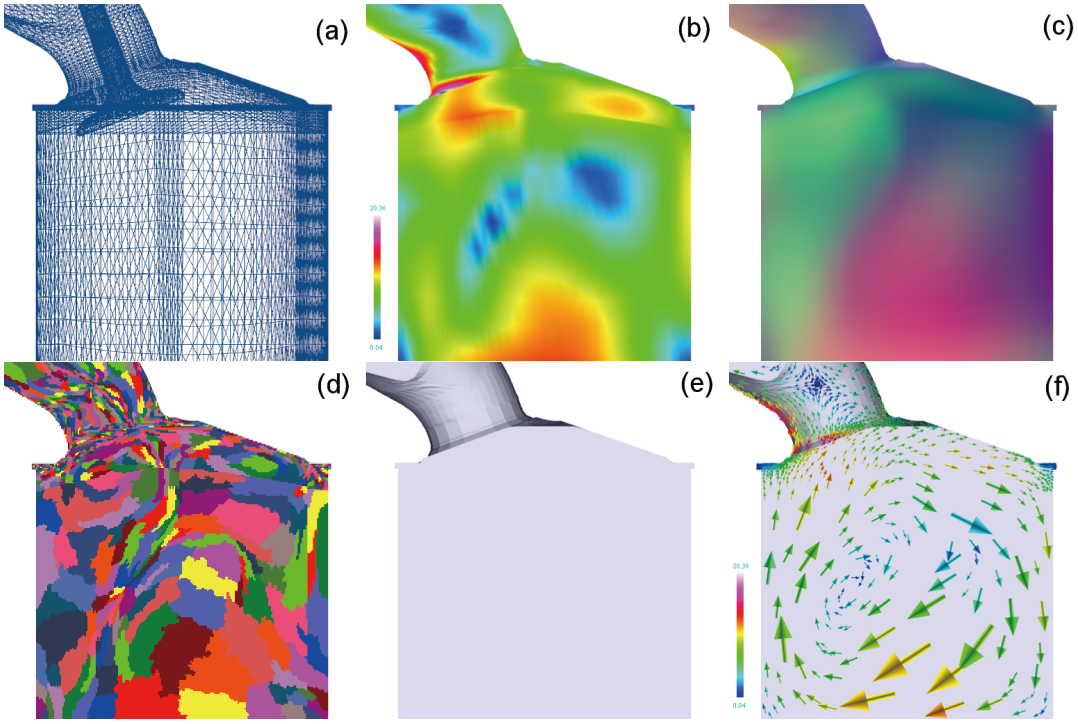


Figure 4.4: Here are 5 constituent images, plus a 6th final image, used for the visualization of surface flow on a gas engine: (a) the initial adaptive resolution mesh, (b) a velocity magnitude colour mapped image, (c) the attribute image corresponding to the vector field and underlying mesh resolution, (d) a colour mapped image which shows the different resulting clusters, (e) an image overlay, (f) the final visualization using glyphs, and the image overlay. Colour in (b) and (f) is mapped to velocity magnitude. The tumble motion depicted in the lower-right image is consistent with previous visualization of the same data [LWSH04].

gradients and mesh resolution as described in Section 4.3.2.

A mesh resolution measure, $m$, is computed at each vertex in a pre-processing phase. Step two is to construct an attribute image. The attribute image is a data structure which holds a simplified view of the model's data attributes including the vector field and the quantified mesh resolution. In order to generate a simplified representation of vector fields for surfaces, a bottom-up hierarchical clustering is applied to the data using a distance measure which includes a local description of the mesh. Then glyphs or streamlets are placed automatically based on the user-defined error measure along with the original surface geometry. Additionally, various enhancements and user options, like distance measure weighting coefficients, can be used to customise the clustering process and thus the final visualization result. An overview of this process is depicted in Figures 4.3 and 4.4. It's also worth mentioning that if the viewpoint is changed after the final visualization rendering, the next pass will start from the attribute image construction. Starting with the hierarchical clustering if the clustering distance measure parameters are changed, only a subset of the algorithm is required. Details are given in the

sub-sections that follow.

One consequence of storing the clusters in a list is the implicit ordering of hierarchy construction. In clustering terminology, the choice of seed clusters affects the result. The fact that the choice of seeds changes the result is a property inherent to many clustering algorithms [XW05], not just the one presented here. In our algorithm, seeds are always processed in the same order, from the top-left to the bottom-right in a rasterized fashion.

### 4.3.2 Mesh Resolution Derivation

Meshes from CFD data sets are adaptive resolution meshes generated according to the importance of constituents in the model. Components of the geometry requiring more accurate simulation results, such as the gaskets of Figure 4.1 and the intake port of Figure 4.4, are modelled with a dense resolution mesh. Some features are considered less important, like the cylinder block (Figure 4.4), and are modelled with a coarse resolution mesh for fast computation. The semantics of the underlying mesh are an important characteristic and can be incorporated into the clustering algorithm along with characteristics of the vector field.

We define the resolution of the mesh around a vertex, $v$, as the density of triangles composed of $v$ and two other vertices. The higher the resolution of the mesh, the smaller the corresponding polygons and thus the shorter the polygonal edge lengths. To derive a resolution measure, we use the length of edges in a 1-Ring neighbourhood. We use a vertex-centred edge mean function as a preprocessing stage of our clustering algorithm. Each vertex, $i$, in the mesh has a given set of edges, $e_n$, in a 1-Ring neighbourhood as depicted in Figure 4.5. The mean length is computed from this set of edges. The resulting mean length $\bar{e}$ represents the resolution of the local neighbourhood around $i$ and is stored as a derived attribute at each vertex in the data set.

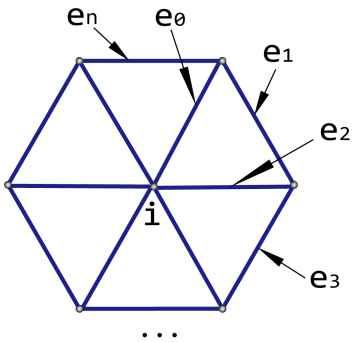


Figure 4.5: A composition of 6 triangles which share a common vertex $i$ represents a 1-Ring neighbourhood of the underlying mesh from which a local resolution measure is derived. $e_i$ here is scalar.

$$\overline{e} = \frac{1}{n+1} \sum_{i=0}^{i=n} e_i \tag{4.1}$$

The resolution in this local region of the mesh is approximated by $m = \frac{1}{\overline{e}}$. Note that this is not the only way to calculate a resolution measure for the underlying mesh and that this is only an approximation. There are other options, such as an area-based mean function or computing the density of vertices per unit area.

### 4.3.3 Attribute Image Construction

The mesh resolution $m_i$ is stored at each vertex, $i$, of the polygonal CFD mesh along with $\mathbf{v}_i$. A key step is to project these attributes defined at the boundary surface to the image plane. We encode the object-space vector field, ($v_{ix}$, $v_{iy}$, $v_{iz}$), and the object-space mesh resolution, ($m$), of each vertex into the $R$, $G$, $B$, and $\alpha$ components of a high precision texture. The formula to encode the components is:

$$\mathbf{C}_i = \frac{\mathbf{v}_i - min(\mathbf{v}_i)}{max(\mathbf{v}_i) - min(\mathbf{v}_i)} \text{ where } \mathbf{C}_i = (C_R, C_G, C_B) \tag{4.2}$$

The formula for encoding the object-space mesh resolution is:

$$m_i = \frac{e_{min}(e_{max} - \overline{e}(i))}{\overline{e}(i)(e_{max} - e_{min})} \tag{4.3}$$

Where $e_{max} > e_{min}$. Encoding mesh attributes in this way yields the following benefits:

- Occluded or otherwise hidden portions of the geometry are automatically filtered out and eliminated from any further processing.
- The input to the clustering algorithm is projected from an unstructured mesh to a uniform, rectilinear grid.
- Interpolation of vector components and the mesh resolution is performed automatically by the graphics hardware.
- No more computation time is spent on polygons whose size is less than one pixel, the occurrence of which is high for CFD meshes (see Figure 4.1).
- The complexity of vector field clustering in object space is reduced to a much simpler problem in image space.

After the encoding of the vector field and mesh resolution, an attribute image is rendered. Note that high-precision (RGBA_32F_ARB) textures can be used to limit quantisation and prevent clamping. The attribute image is used as the input to the clustering procedure (not for visualization purposes). A sample attribute image is shown in Figure 4.4 (c).

### 4.3.4 Projection and Decoding

Following the construction of the attribute image, the projection of the attributes defined at the surface is done by the graphics hardware during the rasterization stage. Since the vector components $\mathbf{v}_i$ and mesh resolution $m_i$ are stored in the framebuffer, projected values can be decoded from the framebuffer with:

$$\mathbf{v}_i = \mathbf{C}_i(max(\mathbf{v}_i) - min(\mathbf{v}_i)) + min(\mathbf{v}_i) \tag{4.4}$$

$$\overline{e}(i) = \frac{e_{min}e_{max}}{e_{max}m_i - e_{min}m_i + e_{min}} \tag{4.5}$$

Where $e_{min} > 0.0$. These decoded values are used to reconstruct the required field attributes as input to our clustering algorithm. Interpolation of the attribute image is necessary for reconstruction. By using hardware-assisted interpolation, we can decode mesh attributes within the original boundary mesh polygons in addition to the information stored at the vertices. See Figure 4.4 for an example.

### 4.3.5 Hierarchical Clustering

For a simplified representation of vector fields at surfaces, a bottom-up hierarchical clustering method is applied to the attribute image using a unified clustering distance measure that includes a local description of the mesh.

#### 4.3.5.1 A Mesh-Driven Vector Field Clustering Measure

To quantify the similarity (or difference) of the characteristics of different clusters, a clustering distance measure is needed. In our clustering algorithm, a clustering error measure is applied to quantify the local characteristics of the vector field and the local description of the mesh. A cluster, $\psi$, is defined as a five-tuple: $\psi(\mathbf{r}, |\overline{\mathbf{v}}|, \alpha, m, \varepsilon)$ where $\mathbf{r}$ is the center-point of the cluster, $|\overline{\mathbf{v}}|$ is the mean velocity magnitude, $\alpha$ is the mean velocity direction, $m$ is the mean mesh resolution, and $\varepsilon$ is the local clustering error. We formulate the distance between clusters with the following measure:

$$\boldsymbol{\varepsilon}(\psi) = c_d\frac{d}{d_{max}} + c_{|\overline{\mathbf{v}}|}\frac{|\overline{\mathbf{v}}|}{|\mathbf{v}|_{max}} + c_\alpha\frac{\alpha}{\alpha_{max}} + c_m\frac{m}{m_{max}} \tag{4.6}$$

where $c_d + c_{|\overline{\mathbf{v}}|} + c_\alpha + c_m = 1.0$. The components of the error measure are:

- Euclidean Distance ($d$): this constituent measures distance between two cluster centres, $\psi_A$, $\psi_B$ using $d = |\psi_A^{\mathbf{r}} - \psi_B^{\mathbf{r}}|$. This component encourages the clustering algorithm to group clusters whose centres are in close proximity. See Figure 4.6 (c). The Euclidean distance between two points in 3D space can be computed using a combination of $(x, y)$

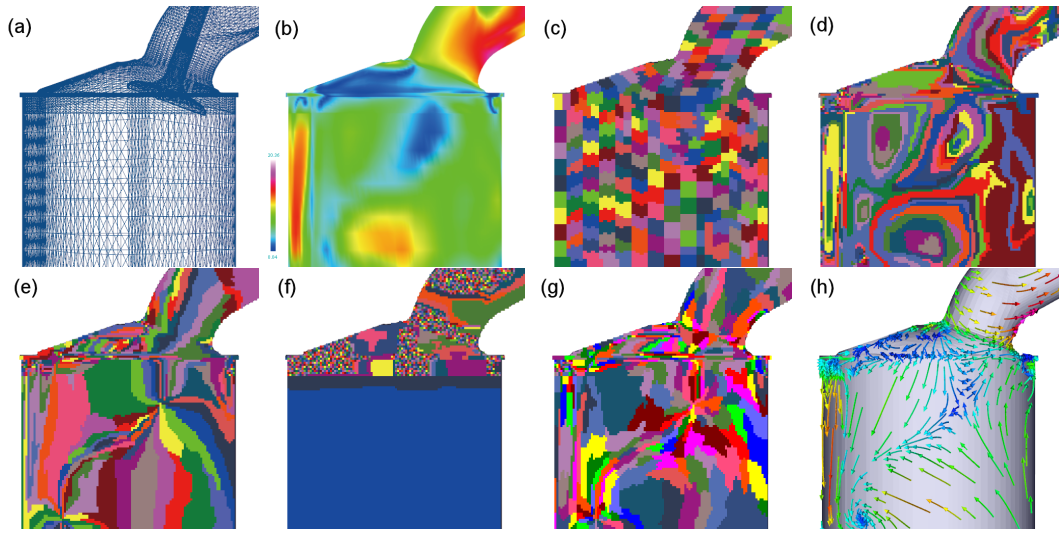


Figure 4.6: Constituent images from the result of clustering process driven by the error measure, used for the visualization of surface flow on a gas engine: (a) the initial adaptive resolution mesh, (b) a velocity magnitude colour mapped image, (c) the result of clustering process by setting $c_d = 100\%$, (d) the result from setting $c_{\overline{\mathbf{v}}} = 100\%$, (e) the result of $c_\alpha = 100\%$, (f) the result of $c_m = 100\%$, (g) the result by setting $c_d = c_{\overline{\mathbf{v}}} = c_\alpha = c_m = 25\%$, (h) streamlets are used to illustrate the vector field based on the attribute field clusters. Colour is mapped to velocity magnitude.

image space coordinates and information from the depth buffer for the ($z$) component (see Section 4.3.5.3). The maximum distance, $d_{max}$, is the length of a diagonal for the geometry's bounding box. It groups clusters visually.

- Velocity Magnitude ($|\overline{\mathbf{v}}|$): this constituent measures the difference between the mean velocity magnitude of two clusters, $\psi_A$, $\psi_B$ using $|\overline{\mathbf{v}}| = \left|\psi_A^{|\overline{\mathbf{v}}|} - \psi_B^{|\overline{\mathbf{v}}|}\right|$. This component encourages the clustering algorithm to group the clusters which have similar magnitude. See Figure 4.6 (d). The maximum velocity, $|\mathbf{v}|_{max}$, is the largest magnitude of the given data set.

- Direction ($\alpha$): this constituent compares the velocity direction of two clusters, $\psi_A$, $\psi_B$ using $\alpha = |\psi_A^\alpha - \psi_B^\alpha|$. It drives the clustering algorithm to group the clusters which have similar direction. See Figure 4.6 (e). The maximum direction value, $\alpha_{max}$, is 180°.

- Mesh Resolution ($m$): this component differs from the other three. Rather than comparing the vector field features, it sums the resolution of the local mesh from two clusters, $\psi_A$, $\psi_B$ using $m = \psi_A^m + \psi_B^m$. The effect of this measure is to accumulate error proportional to the local density and thus present the user with a more detailed representation in regions with a denser mesh. In other words, multiple, shorter edges accumulate more error than a single long straight edge, thus for a given $\varepsilon$, clusters in high resolution mesh regions are generally smaller. See Figure 4.6 (f). The maximum mesh resolution, $m_{max}$, is the largest value of $m$ derived in Section 4.3.2.

- Local Error($\varepsilon$): this is not used as part as the distance clustering measure but is simply the result stored from the previous pairing of two clusters and represents the combined local cluster error.

Each component of the distance measure, including $\varepsilon$, is stored in the normalised range $[0, \cdots, 1]$ in order to unify the various measures. Note $\varepsilon$ is a measure of how well a centroid describes a region of data. Optional user-defined weighting coefficients, $c_d$, $c_{|\overline{\mathbf{v}}|}$, $c_\alpha$, $c_m$ are introduced to adjust the contribution of the first four components to the final clustering distance measure. The fully automatic version of the algorithm essentially ignores the weighting coefficients - thus minimising the need for interaction. These components may be treated independently from one another. By changing the weighting coefficients different distance measures can be tailored to address the user's needs. The effect of varying these coefficients is shown in Figure 4.6. We note that adjusting $c_d$, $c_{|\overline{\mathbf{v}}|}$, $c_\alpha$, and $c_m$ is optional. Their values are set to 25% by default throughout this chapter unless noted otherwise. Modifying coefficients is user-dependent. The distance measure is the key that drives the algorithm and produces a suggestive and simplified depiction of the vector field on surfaces. See the combined clustering result in Figure 4.6 (g) and also the final visualization (h). The larger glyphs indicate areas of more uniform flow. The same saddle point and vortex visualized in the (h) of Figure 4.6 is consistent with previous work [LJH03]. The cluster geometry is stored as a list of pixels and their edges. The centre of a cluster is simply the average position of each pixel centre. If the user would like more control over the shape of the clusters in order to avoid U-shaped clusters for example, then $c_d$ in equation (6) may be increased. The effect of increasing $c_d$ is shown in Figure 4.6. Users interested in regions of similar velocity magnitude increase $c_{|\overline{\mathbf{v}}|}$. Users interested in seeing features such as sources, sinks, and saddle points increase $c_\alpha$. Users who would like more detail in areas of high mesh resolution increase $c_m$.

#### 4.3.5.2 Generating the Cluster Hierarchy

Our algorithm is an agglomerative, average-linking clustering process. As the basic component of the cluster hierarchy each cluster is stored as a node containing $\mathbf{r}$, $|\overline{\mathbf{v}}|$, $\alpha$, $m$, $\varepsilon$. Given a cluster, $\psi_A(\mathbf{r}, |\overline{\mathbf{v}}|, \alpha, m, \varepsilon)$, the bottom-up, hierarchical algorithm is conceptually a search process. $\psi_A$ tests its neighbours looking for a merge candidate. Neighbours share a common cluster edge. The candidate with the minimum clustering distance, $|\psi_A - \psi_B|$, is chosen and a new parent cluster $\psi_{AB}$ is formed. Initially, the smallest clusters are the individual pixels of the attribute image with zero error. The clustering and tree generating process starts in the top-left corner and seeds in rasterized fashion. This process is repeated for each cluster until only one root cluster remains.

Although this method is easy to apply, the computational cost becomes a problem since each cluster $\psi_B$ which can be merged with a given cluster $\psi_A$ needs to be tested using the distance measure. In order to accelerate the hierarchical process, we introduce another structure to speed up the candidate cluster searching process. This structure consists of a lookup table $\psi^{LUT}$ and a neighbour list $\psi^l$ for each cluster. The lookup table accelerates the search process essentially by a trade-off between storage space and processing time. Instead of performing a traversal

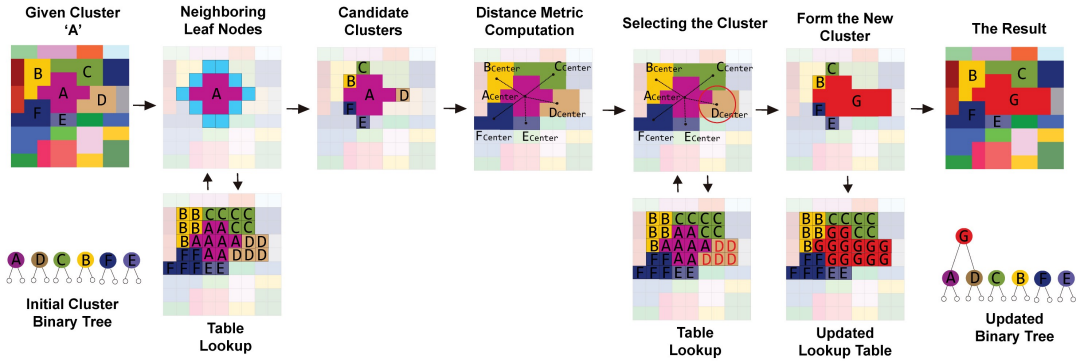


Figure 4.7: An example illustrates the clustering method. It starts with a given cluster, $A$, and a cluster list $L(\psi)$ which stores the initial leaf clusters for the hierarchical clustering process. After the search process is finished, a new cluster, cluster $G$, is formed from two child clusters, $A$ and $D$, which have the shortest distance $\boldsymbol{\varepsilon}(\psi_A - \psi_N)$. $G$ is added to the back of $L(\psi)$. The corresponding binary tree is updated to store the new parent cluster and its children.

of the entire hierarchy for neighbours, the location of child nodes and parent nodes is stored and updated after each search process. $\psi^{LUT}$ is the same resolution as the leaf clusters and stores the address of each leaf cluster's top-most parent cluster. $\psi^{l}$ stores the address of each adjacent leaf cluster. With this additional information, the candidate cluster $\psi_B$ which may be grouped with the current cluster $\psi_A$ to form a new cluster can be located quickly. We illustrate the process using the example in Figure 4.7.

In order to represent initial leaf clusters of the vector field in image space, a rectilinear grid, the resolution of which is defined by the user, is placed in image space. Cluster attributes are then retrieved at each grid cell using the decoding procedure described in Section 4.3.4. Each leaf cluster contains a neighbour list $\psi^{l}$ and is added onto a central cluster list $L(\psi)$ which stores nodes for the bottom-up hierarchical clustering process. The clustering process starts from the head of the list $L(\psi)$.

Our clustering algorithm begins by traversing $\psi^{l}_{A}$ for the given cluster $\psi_A$. For example in Figure 4.7, the given cluster $A$ searches its adjacent clusters $\psi^{l}_{A}$. The $\psi^{LUT}$ stores top-most parent clusters for these adjacent leaf clusters and also indicates the leaf clusters which have the same top-most parent cluster in the neighbour list. A refined neighbour list which contains only the unique cluster candidates can be obtained for cluster $A$.

$\boldsymbol{\varepsilon}$ is computed for each candidate cluster. The cluster resulting in the smallest error is chosen. In Figure 4.7, $D$ is the candidate selected.

Once a candidate in $\psi^{l}_{A}$ is selected, a new cluster, $G$, can be formed by grouping $A$ and $D$. Meanwhile, $\psi^{LUT}_{G}$ is updated. The new cluster is stored in the form of a parent node whose two child nodes are $A$ and $D$. $G$ is added to $L(\psi)$ to be processed. When $L(\psi)$ contains only

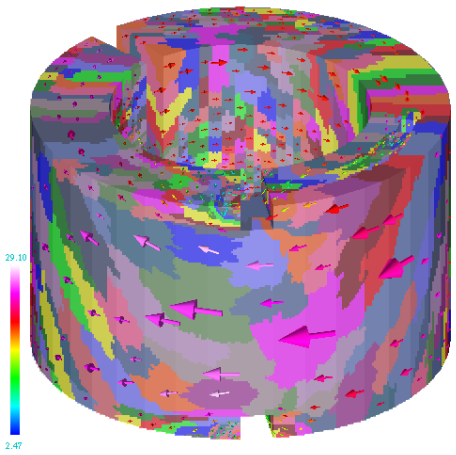


Figure 4.8: The result of the clustering driven by an error measure ($\boldsymbol{\varepsilon} = 12\%$) which uses the depth as a distance measure constituent. Notice how the overall density is much higher towards the rear.

one node, the bottom-up hierarchical clustering is completed.

#### 4.3.5.3 Edge and Discontinuity Detection

During the clustering process, cluster $\psi_B$ chosen to be grouped with $\psi_A$ is found by applying the distance measure to $\psi_A^l$. However, a complication can arise with this approach due to discontinuities and sharp edges on the surface. If we do not take discontinuity in the geometry into account then we may end up grouping clusters that do not belong together. To address this problem, we compare $\psi_A^{depth}$ with $\psi_B^{depth}$. If $\varepsilon_{depth} < |\psi_A^{depth} - \psi_B^{depth}|$ then $\psi_B$ is not grouped with $\psi_A$. This approach separates the image into local regions with boundaries. In practice, we have found a value of $\varepsilon_{depth} \approx 0.3\%$ to be a good threshold.

### 4.3.6 Image Overlay

An image overlay is used for the resulting visualization of the clustering on surfaces by applying colour, shading, or any attribute mapped to color. In the implementation, we generate the image overlay following the construction of the attribute image once for each static scene. Once the view-point is changed, the image overlay is regenerated. By exploiting the glDrawPixels() function from OpengGL, rendering an image is much faster than rendering the complex 3D triangulated object each time a user parameter is changed.

### 4.3.7 General Attribute-based Clustering

Our clustering method can be extended to incorporate *any* general attribute of CFD simulations in the distance measure. Not only can the properties of the vector field and the resolution of the underlying mesh be coupled into a unified distance measure, but also any other CFD simulation attribute can also be used to drive the clustering process such as temperature, pressure, kinetic

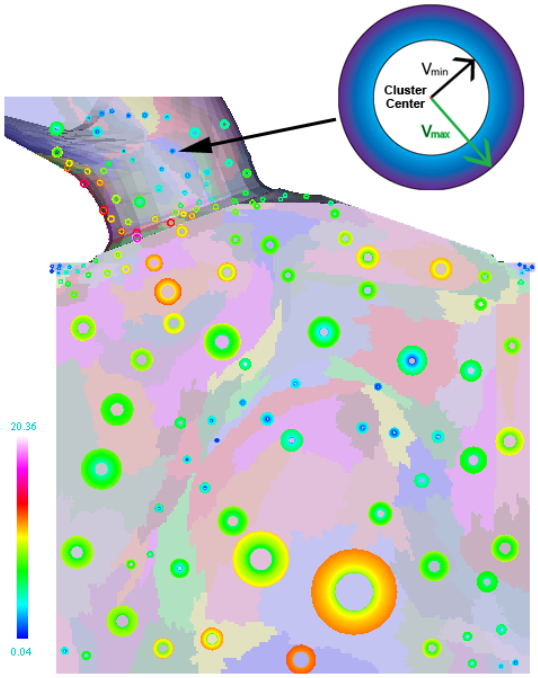


Figure 4.9: $|\mathbf{v}|$-range glyphs: (Top) a close-up look at a $|\mathbf{v}|$-range glyph whose inner radius represents the minimum velocity magnitude while outer is mapped to the maximum. (Bottom) the result image with $|\mathbf{v}|$-range glyphs applied to depict the variation in magnitude within each cluster .The result of the clustering driven by an error measure ($\boldsymbol{\varepsilon} = 15\%$). Glyph colour is mapped to velocity magnitude.

energy and derived data such as gradients. By encoding these values into the attribute image using the method in Section 4.3.3 and building up the hierarchy following the scheme in Section 4.3.5.1, we can obtain arbitrary attribute-based clusters for different user needs. Figure 4.8 demonstrates the effect of using $\psi^{depth}$. For demonstration, we have simply substituted mesh resolution, $m$, with another attribute, $\psi^{depth} = \psi_A^{depth} + \psi_B^{depth}$ in equation (6). In this example, regions further away from this viewer (higher depth value) are denser in the visualization.

### 4.3.8 Visualization Options

Using the binary tree, clusters can be traversed in a depth first search fashion driven by a user-defined error value. In this searching process, the (deepest) clusters where $\psi^{\varepsilon} \leq \boldsymbol{\varepsilon}$ are rendered for visualization. Several user options are available for the visualization of the selected clusters.

#### 4.3.8.1 Colour Coding and Mean Vector Glyphs

A selection of colours can be blended to depict different clusters. This is shown in Figures 4.6, 4.2 (b), 4.8, and 4.15.

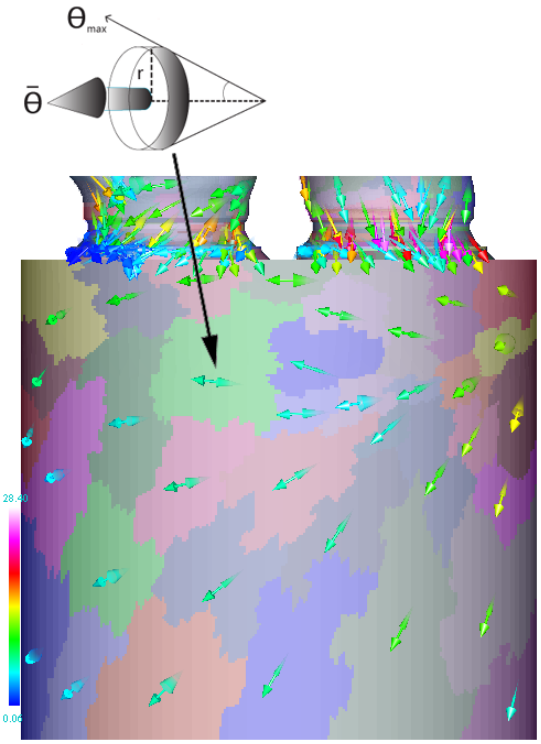


Figure 4.10: (Top) the $\theta$-range glyph whose radius represents the maximum range of vector field direction is illustrated. The result image with $\theta$-range glyphs is shown in (Bottom). The result of the clustering is driven by an error measure ($\boldsymbol{\varepsilon} = 18\%$). Glyph colour is mapped to the velocity magnitude.

Visualization using vector glyphs is one of the most straightforward methods to visualize the flow field. In our algorithm, a mean 3D vector glyph is placed at the centre of each cluster. Glyph size may be mapped to the size of the corresponding cluster while the colour can be mapped to the mean velocity magnitude. See Figure 4.8 and 4.13. The user may zoom in to areas with dense glyphs for more detail. This is one way to resolve occlusion. Other ways include mapping colour to mean velocity magnitude or using streamlets.

#### 4.3.8.2 Streamlets

Cluster-centre-based streamlet seeding is another visualization option. This automatically traces a 3D streamlet tube from each cluster centre until it hits the cluster boundary. The streamlet curves are traced according to the original vector field. In Figures 4.2 (d) and 4.15, a cluster-based streamlet technique (bottom) gives insightful imagery by applying streamlet depiction in the more important areas while yielding a sparse depiction in areas of less importance. Arrow heads can be added to the streamlets to indicate the downstream direction of the flow. Using arrow heads in the visualization can introduce occlusion. To eliminate this the user can disable the arrow heads leaving only streamlet curves. Colour coding cluster according to velocity magnitude is also occlusion free. This is one way to resolve occlusion. Other ways

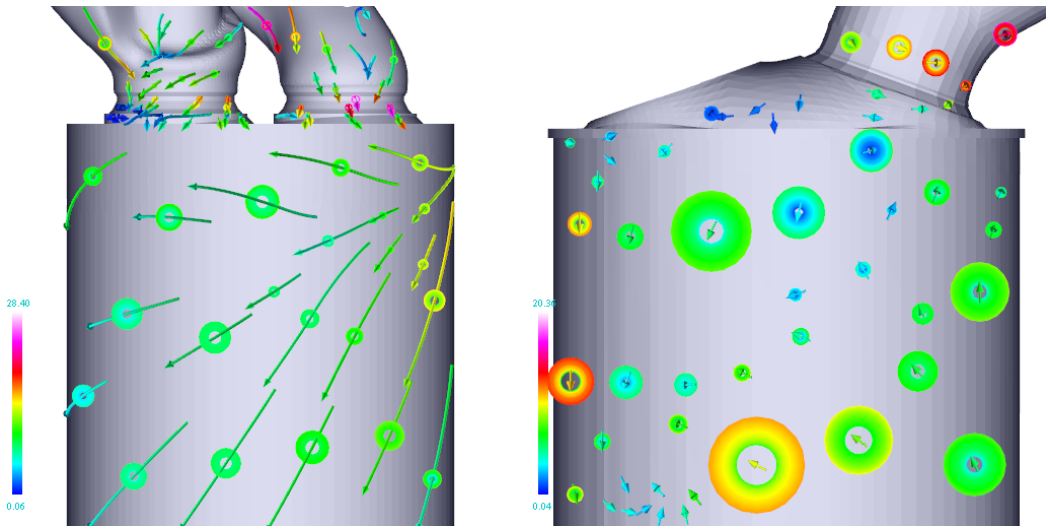


Figure 4.11: (Left) the combination of $|\mathbf{v}|$-range glyphs and streamlet tubes is applied to provide both detailed (the range glyph) and summary (the streamlet) information of the vector field direction. (Right) the detail of the magnitude distribution within each cluster can be obtained by using the $|\mathbf{v}|$-range glyph while the mean glyph is applied to provide the information of the mean magnitude and direction each cluster.

include mapping colour to mean velocity magnitude or using streamlets.

#### 4.3.8.3 $|\mathbf{v}|$-Range Glyphs

Automatically providing more detail in areas of high vector field variance and mesh resolution is deemed helpful to engineers. $|\mathbf{v}|$-range glyphs are used to visualize the variation in vector field magnitude within each cluster. We use a ring-like glyph as in Figure 4.9 (top) where $|\mathbf{v}_{min}|$ is mapped to the inner ring and $|\mathbf{v}_{max}|$ is mapped to the outer ring. The $|\mathbf{v}|$-range glyphs are placed at the centre of each cluster. Regions with relatively large variation in $|\mathbf{v}|$ become obvious. Range glyphs (as opposed to deviation glyphs) have the advantage that they include outliers in the visualization.

#### 4.3.8.4 $\theta$-Range Glyphs:

$\theta$-range glyphs depict the variance in vector field direction within each cluster. A semi-transparent cone-like glyph, as in Figure 4.10, whose radius represents the maximum range of $\theta$ from $\overline{\theta}$ to $\theta_{max}$ is used. Clusters with a large variance in direction are shown with this type of visualization.

#### 4.3.8.5 Hybrid Visualizations

Various visualization options can be combined to provide more details simultaneously, like $\theta$-range glyphs with $|\mathbf{v}|$-range glyphs, glyphs with $\theta$-range glyphs, $\theta$-range glyphs with streamlet tubes (Figure 4.11).

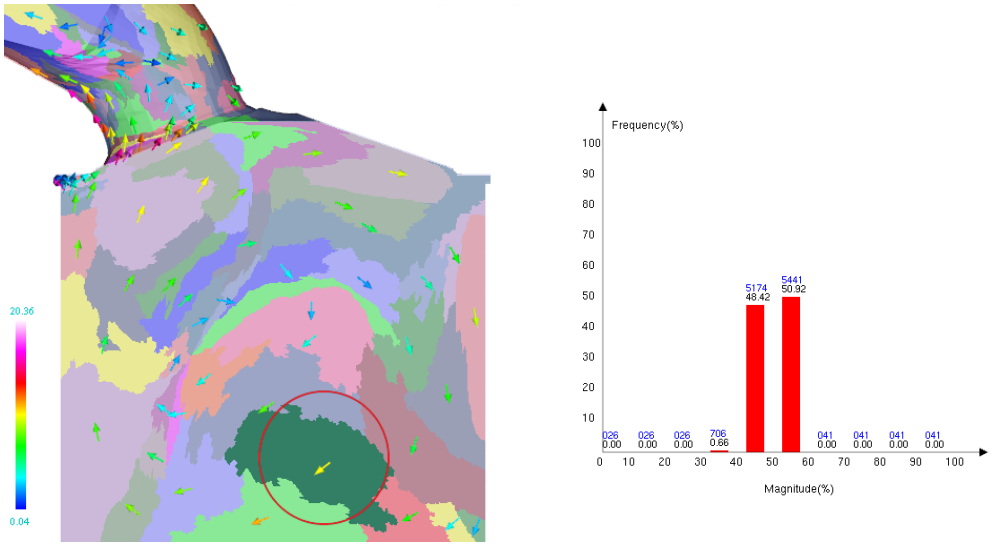


Figure 4.12: The detail of the magnitude distribution within the focus cluster (in dark green) in the scientific visualization window (left) is simultaneously visualized by a histogram in a sub-window (right). For example, the fifth bar in the histogram indicates that there are 5174 leaf clusters, 48.42% of the total leaf clusters of the focus cluster, whose velocity magnitude is ranging from 40% to 50% of the maximum velocity magnitude.

#### 4.3.8.6 Multiple, Coordinated Views

In order to provide more details of the vector samples within each cluster, a histogram visualization is incorporated in an information visualization window to depict the vector magnitude distribution of each cluster. The user can interactively click on a cluster in the scientific visualization window, then the histogram will automatically display the magnitude distribution of its leaf clusters in the information visualization window. An example is shown in Figure 4.12.

### 4.3.9 Image vs. Object Space Clustering and Accuracy

Before discussing accuracy, it is important to note that this is an error-driven visualization method. The user chooses the maximum level of error $\boldsymbol{\varepsilon}$ represented in the individual clusters and the binary-tree data structure is then traversed. The largest clusters, $\psi$, where $\psi^{\varepsilon} \leq \boldsymbol{\varepsilon}$ are rendered. In addition, error in any visualization can stem from factors such as discrete numerical simulation, discrete data sampling, linear interpolation, and numerical integration schemes. For a more detailed discussion of accuracy in vector field visualization, see Chen et al. [CMLZ08].

However, we can compare the accuracy of our approach which uses a hardware accelerated attribute image with that of a pure CPU object-space approach. This comparison is done based on synthetic flow data sets. In some of our synthetic data sets, we have doubled the resolution of the domain in the top-left quadrant in order to vary the mesh resolution. See Figures 4.17

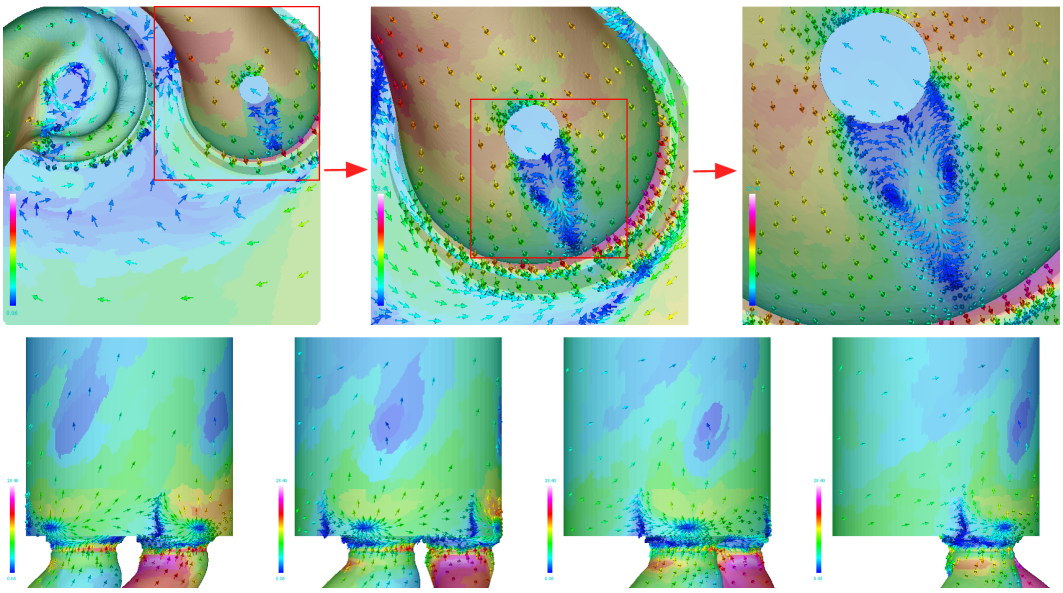

Figure 4.13: Glyph visualization applied to depict vector field clusters on the top surface of the diesel engine during zooming in and rotation. This set of images illustrate that the clustering method produces consistent results when the user zooms in on a portion of the geometry (top) or rotates the geometry (bottom). Colour is mapped to velocity magnitude.

and 4.18. The leaf clusters of object-based clustering algorithm are based on the highest-resolution, object space data samples while the image-based method retrieves each leaf node cluster from the attribute image data structure. Apart from this, they use the same clustering scheme. In Figures 4.17 and 4.18, we can observe the image-based clustering method (Figures 4.17 and 4.18, right) produces the same simplified presentation of vector fields as an object-spaced approach (Figures 4.17 and 4.18, left). One advantage of the object space approach is that the clustering can be performed once for the whole geometry. However, this also produces clusters from hidden surfaces which are occluded or outside the view-point. Also the object space result is in general slower and more memory intensive than the image space approach. Although the object-based clustering algorithm is very accurate, the high computational cost of the distance checking and neighbour finding slows down the clustering process especially if the algorithm is applied to large, unstructured data sets from CFD. Whereas our image-based approach depicts the flow just as well as the object-based method. This makes the image-based clustering algorithm very suitable for visualization purposes. We observe that if an engineer is interested in exact velocity values, they simply click on the mesh at the point of interest to retrieve it as in Figure 4.12 (rather than using clustering).

## 4.4 Performance and Results

As our vector field clustering algorithm targets large, unstructured, adaptive resolution boundary meshes from CFD, we tested our algorithm on a range of simulation datasets with these

characteristics. Colour is mapped to velocity magnitude in our examples.

Figure 4.14 shows a comparison of the hedgehog flow visualization and our mesh-driven clustering method applied to a surface of an intake port mesh composed of 221k unstructured, adaptive resolution polygons. As we can see from the left, most glyphs overlap or are occluded. Using the hedgehog approach 221k glyphs are rendered while our method renders only about one hundred glyphs. Most importantly, the distribution of glyphs in the hedgehog visualization is completely driven by the underlying mesh without considering the flow field itself resulting in severe artifacts. Using texture-based or topological methods to visualize the flow does not depict the downstream direction of the flow. Moreover, texture properties are of uniform resolution. However, our method combines the properties of the underlying vector field and mesh into a unified clustering distance measure which drives the clustering process, and then places glyphs or traces streamlines based on the cluster centres in a simplified and insightful fashion. This enables engineers to get a fast and clear overview of the flow on the surface. The characteristics of the swirl motion depicted here are consistent with previous results [LWSH04]. See Figure 4.14 (right). Despite the high level of geometric complexity (Figure 4.1), our clustering method can also efficiently visualize the vector field on a complex cooling jacket boundary mesh (Figure 4.15). Because of the processing efficiency this clustering method allows users to translate, rotate and zoom in the object interactively to get better insight of the CFD data sets. We encourage the reader to view the supplementary video for more results. Figure 4.13 illustrates the effect of zooming (top) and rotation (bottom). The image-space approach produces consistent results when the user zooms in on a portion of the geometry or rotates the geometry. Zooming and rotation are also demonstrated in the supplementary video.

We can apply clustering to stream surfaces for the first time. In Figure 4.2, the image (a) shows the complexity of the underlying stream surface mesh with jagged edges. The image (b) shows resulting vector field clusters on the stream surface. The remaining two images compare (c) hedgehog visualization with (d) our clustering based streamline visualization.

In order to report the time taken to build up the vector field hierarchy, we test our clustering

| Data Set | Total number of clusters | | | |
|---|---|---|---|---|
| | 131071 | 32767 | 8191 | 2047 |
| Ring (10k) | 10.07s | 1.06s | 0.085s | 10.17ms |
| Combustion Chamber (79k) | 10.06s | 1.07s | 0.087s | 10.06ms |
| Intake Port (221k) | 10.13s | 1.07s | 0.088s | 10.65ms |
| Cooling Jacket (228k) | 10.20s | 1.09s | 0.09s | 10.96ms |

Table 4.1: Cluster hierarchy generation timing figures for total cluster quantities. An image resolution of $512^2$ is used with about 75% image space area covered. The total numbers of clusters include the leaf and parent node clusters.

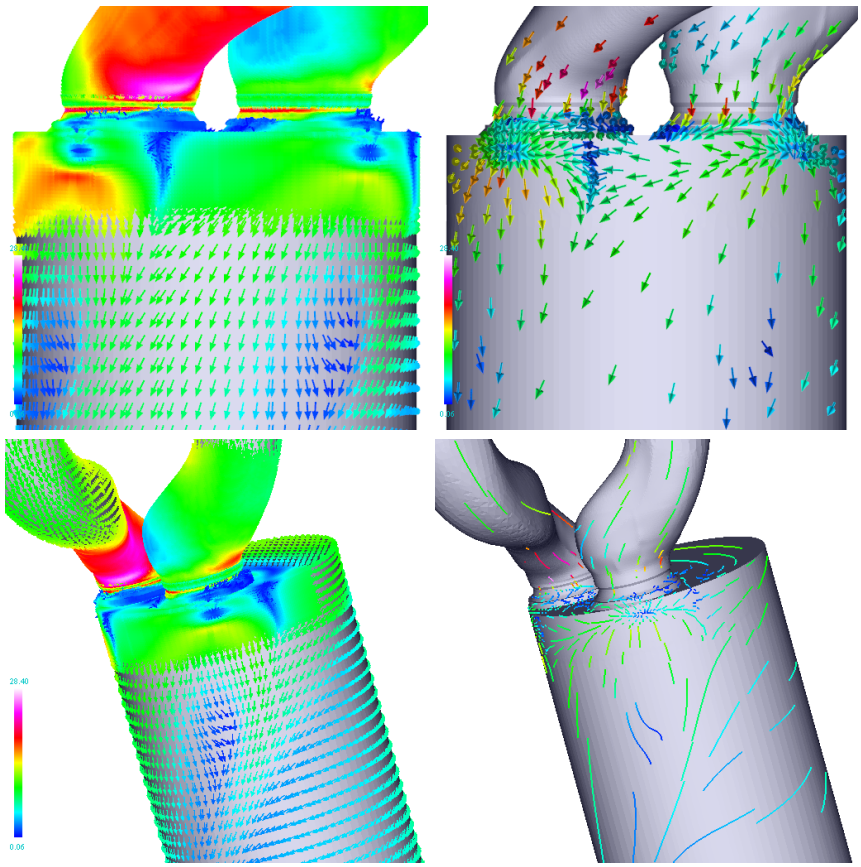

Figure 4.14: A comparison of the hedgehog flow visualization (left) and our vector field clustering visualization with normalised glyph representation with $\boldsymbol{\varepsilon} = 25\%$ (upper-right) and streamlet with $\boldsymbol{\varepsilon} = 35\%$ (bottom-right) applied to flow fields at the surface of a intake port mesh - a composite of 221k unstructured, adaptive-resolution polygons. Colour is mapped to velocity magnitude.

method on a PC with an Nvidia Geforce 8600GT graphics card, a 2.66 GHz dual-processor and 4 GB of RAM. The timings in Table 4.1 were obtained from a $512^2$ resolution image with 75% coverage. From Table 4.1, the performance time reveals that the algorithm depends mostly on the total number of clusters including parent node clusters rather than the number of polygons in the original surface mesh. Furthermore, Table 4.1 indicates that the smaller the size of the initial cluster, the lower the performance, (although the more accurate the result). For interaction, users then can choose fewer initial cluster quantities for faster speed. Our algorithm supports interactive frame rates for up to 25000 leaf clusters. The user can increase the number of clusters for higher accuracy.

We observe that the performance of our algorithm compares closely to the performance times of previous vector field clustering algorithms where reported [TvW99, HWHJ99, GPR$^+$04] that operate on uniform rectilinear grids. Furthermore, the method presented here handles a much larger number of clusters (one-two orders of magnitude) more than previous algorithms.

We have tested our algorithm on data sets with over one-half million clusters without problems.

The clustering could be performed in object space with generally slower performance with a pre-processing step. However as a pre-processing step, object-space methods do not allow interactive changes to visualization parameters. User parameters can be explored and then be saved by experts. Those settings can then be loaded by non-experts. For non-expert users, we have default values for the components of the error measure: $c_d = c_{\bar{\mathbf{v}}} = c_\alpha = c_m = 25\%$, these values are based on our experience of testing real-world engine simulation datasets. Plus, rendering large numbers of occluded glyphs in object space becomes a burden on performance. Object-space flow visualization methods have not demonstrated themselves to be a viable option for this kind of (unstructured, adaptive resolution) mesh data. Stalling [Sta97], Forssell and Cohen [FC95] have all tried to parametrise the surface for visualization. However not all surfaces are easily parametrised, e.g. the cooling jacket (Figure 4.1) or isosurfaces. Secondly, parametrisation introduces a distortion by the mapping between parameter and physical space. Stalling [Sta97] and Carr and Hart [CH02] have tried packing the surface triangles into texture space. However, these algorithms are only developed for uniform resolution meshes composed of isosceles triangles. Plus, much computation time is spent on hidden polygons or polygons smaller than one pixel. Spencer et al. [SLCZ09] have tried flow visualization using evenly-spaced streamlines in both image and object space. However, the object space version runs 3-4 orders of magnitude slower than the image space version of the same algorithm.

We have implemented the vector field clustering method of Telea and Van Wijk [TvW99] and compared it with ours. Figure 4.16 shows the result. Note that both methods produce similar results with the exception of the region with a high-resolution mesh. Our method generates smaller clusters in that region reflecting the characteristics of the underlying mesh and $2-3$ times faster than [TvW99] in this case.

There are some cases where a projection to image space could cause performance to slow down. For example, in a case where the average polygon covers several pixels, very few polygons fall outside the view frustum, and there are no occluded portions of the surface. Our image space approach could be slower than an object space approach. However, in the majority of cases, the image-space approach accelerates performance.

## 4.5 Domain Expert Review

In order to evaluate the usability of our method from the client side - the CFD experts, we invite Dr Nick Croft, a CFD simulation domain expert from the School of Engineering at Swansea University, to help us on the evaluation of the presented algorithm. His review is as below.

“The use of glyphs in the visualization of directional quantities, for example velocity or magnetic fields, would at first seem obvious. The ability to display direction, magnitude and usually other information in a single entity has an apparent advantage over any other approaches. Glyphs require a number of pixels to convey the multitude of information associated with

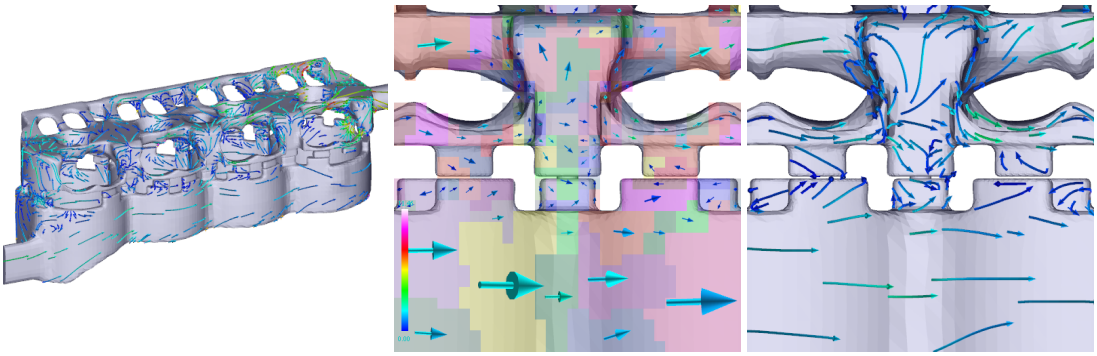

Figure 4.15: Various visualization options applied to depict vector field clusters on the cooling jacket surface - a composite of 228k unstructured, adaptive-resolution polygons: (left) the glyph visualization illustrating an overview of the flow field on the surface using $\boldsymbol{\varepsilon} = 25\%$, (middle) the glyph visualization and (right) the cluster-based streamlet visualization providing a close-up view on the area deemed interesting with $\boldsymbol{\varepsilon} = 38\%$ and $\boldsymbol{\varepsilon} = 35\%$.

them. When a model consists of millions of solution points, which are often concentrated in areas of interest, it is often nearly impossible to understand the information in a plot using glyphs. Overlap of the glyphs, the ability to associate a glyph with its position and the ability to identify features associated with the slower speeds all lead to problems. One solution to these problems is to plot a subset of the glyphs. This approach certainly offers a possibility to display less information and so the detail of the plot can be seen. The question associated with this approach is how does one coarsen the data without compromising the information being conveyed? Any approach will lose information and understanding what has been lost is important in understanding whether the coarsened plot is representative of the data. Any experimental measurement or computational prediction has an error bar which can at best be estimated. Coarsened data has an equivalent error, relative to the original data, but this is known and can be visualized and analysed.

Whilst it is obvious that clustering based on the data will minimise the coarsening error, the more conventional approaches are based on physical separation of coarsened data points or grouping similar number of elements. It is interesting to see the clusters when each of the techniques is used in isolation on a mesh. The concave clusters associated with magnitude clustering and the elongated clusters associated with direction are not ideal but these measures should contribute to increase the representative nature of the coarsened data. Distance based clustering reduces the problems with overlapping glyphs but this needs to be moderated through mesh based clustering to ensure detail in fine mesh regions, usually associated with rapidly changing results and often areas of interest. Whilst it is difficult to see how each of these influences can be combined to produce the “best” plot it is clear that none on their own provide a perfect option.

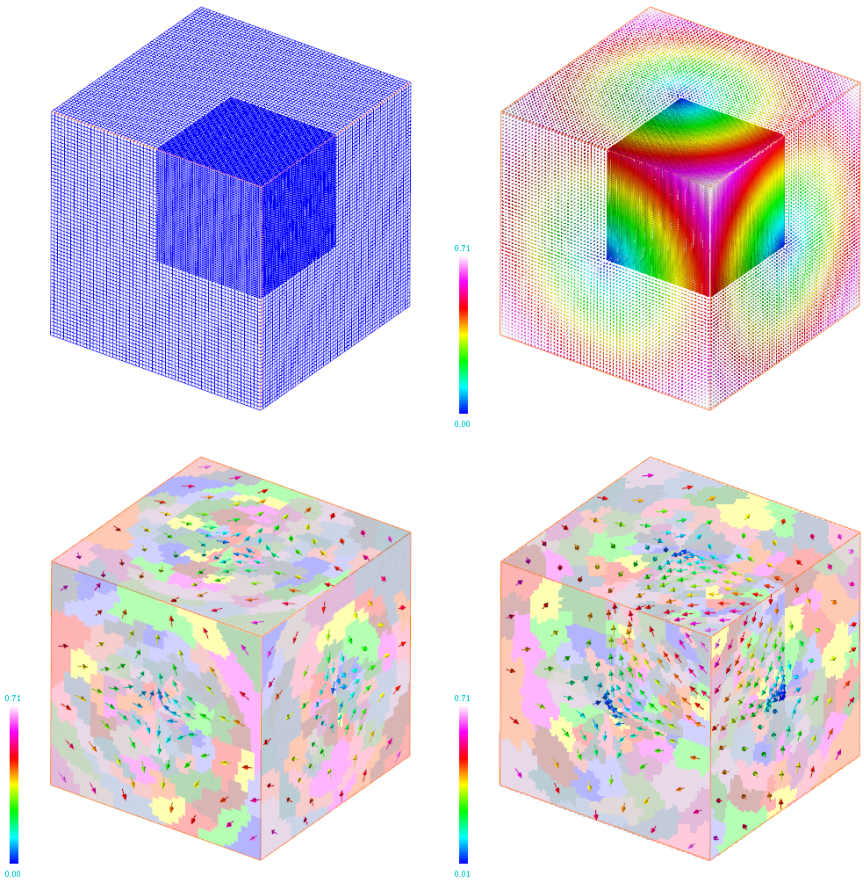


Figure 4.16: The visualization of object vs. image flow from synthetic flow data sets. (top,left) The image illustrates the multi-resolution mesh of synthetic data. (top,right) Hedgehog flow visualization. (bottom,left) The clusters generated using the object-based clustering algorithm of Telea and Van Wijk [TvW99]. (bottom,right) The results from our image-based clustering algorithm.

As has been stated above the clustering of data loses information. As an analyst it is as important to know what information has been lost as what has been retained. The $\theta$ and $|\mathbf{v}|$-range plots presented in Section 4.3.8 display this information. These are not plots that would normally be used to display the results but they are vital to understand how representative the clustered data is of the simulation results. If an analyst is to make a design decision based on clustered data they need to be aware of the uncertainty contained in that data. The $\theta$ and $|\mathbf{v}|$-range plots prove a simple graphical representation of that uncertainty."

## 4.6 Conclusion and Future Work

In this chapter we propose a novel, automatic mesh-driven vector field clustering algorithm which couples the properties of the vector field and resolution of underlying mesh into a uni-

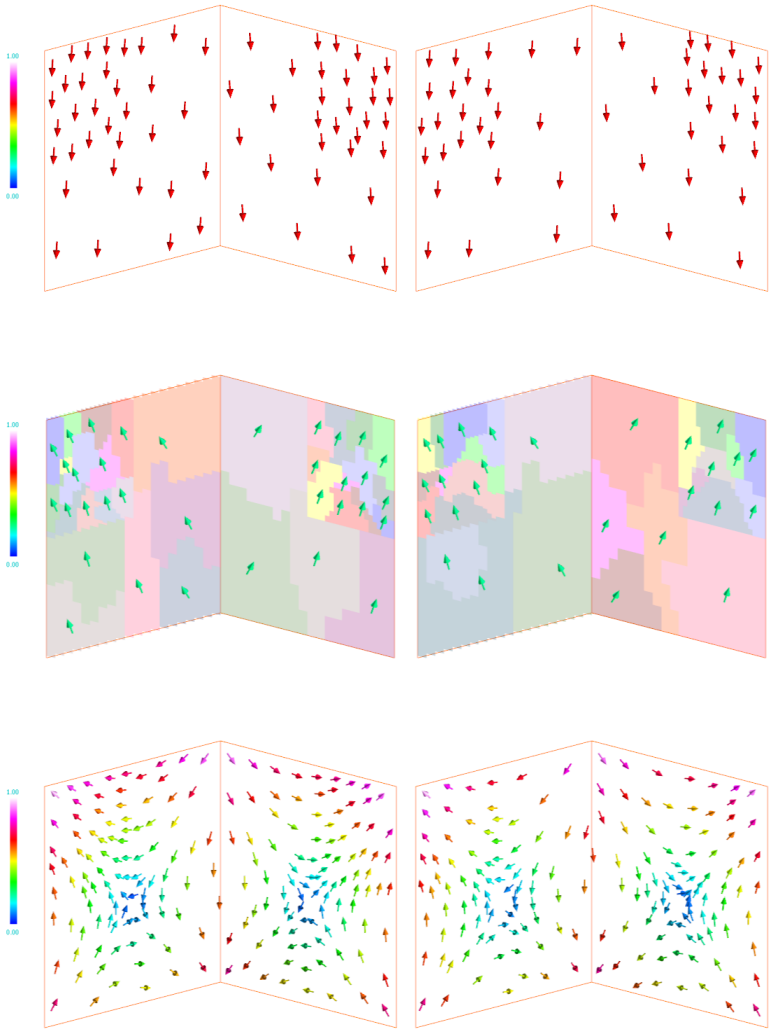

Figure 4.17: The visualization of image vs. object based clustering from synthetic flow data sets. (left) Images shows the clusters generated using a full-precision, object-based clustering algorithm. (right) The results from our novel, image-based clustering algorithm. Planes are at right angles to one another.

fied distance measure for producing intuitive and suggestive images of vector fields on large, unstructured, adaptive resolution boundary meshes from CFD. We have shown that our algorithm clusters vector fields effectively by applying the distance measure with user-defined weighting coefficients, independent of geometric and topological complexity of the underlying adaptive resolution mesh. The cluster hierarchy is generated automatically according to the importance of the underlying mesh and the properties of the vector fields for emphasising vector fields in important regions. No computation time is wasted on occluded polygons or polygons covering less than one pixel. Performance is independent of the data size. So in the

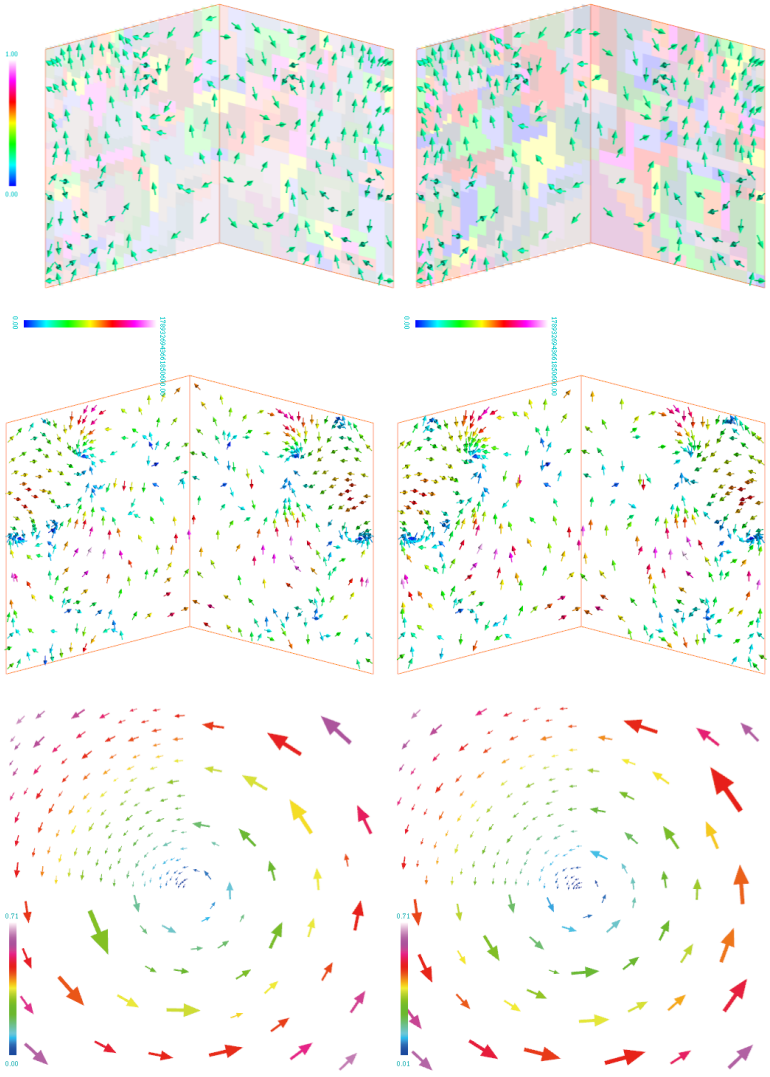

Figure 4.18: The visualization of image vs. object based clustering from synthetic flow data sets modified. (left) Images shows the clusters generated using a full-precision, object-based clustering algorithm. (right) The results from our novel, image-based clustering algorithm. The differences are indistinguishable from a visualization perspective.

most cases our method is much quicker than other methods such as [TvW99], see Table 4.1. We have shown that our framework is general enough to incorporate any data attribute into the clustering distance measure. New visualization inspired by statistics such as the $|\mathbf{v}|$-magnitude and $\theta$-range glyphs have been introduced. Additionally, our algorithm supports user interaction such as zooming, translating and rotation. The accuracy of the visualization is compared to a pure object-based approach. No parametrisation of the surface is required. We have also

demonstrated the robustness of the technique and the ability of the algorithm to handle real-world, complex CFD data sets. Clustering can be applied to stream surfaces for the first time.

As future work we would like to explore the possibility of transferring more of the computation to the GPU. Future work also includes the investigation of different measures for the derivation of mesh resolution. We would also like to extend the work to visualization of unsteady, 3D flow. However, challenges stem from both the resampling performance time and perceptual issues. We would also like to introduce a glyph to represent the standard deviation of each cluster, namely:

$$\sigma_v(|\mathbf{v}_0|\cdots|\mathbf{v}_{n-1}|) = \frac{1}{n-1}\sum_{j=0}^{n-1}(|\mathbf{v}_j| - |\overline{\mathbf{v}}|)^2 \tag{4.7}$$

where $n$ is the number of vector samples in the cluster. Glyph placement for U-shaped clusters is also an area of future work.

*Chapter 5*

# Visualization of Flow Past a Marine Turbine: the Information-Assisted Search for Sustainable Energy

**Contents**



"Energy is eternal delight."

- William Blake [8]

AFTER we finished the mesh-driven vector field clustering project, CFD engineers approached us and wanted us to help them visualize their marine turbine CFD datasets. Due to the high dimensionality of the given datasets no off-shell commercial visualization tools can effectively reveal the inner relationship among dimensions. Tending to address this challenge, in this chapter we develop, explore and present customised visualization techniques in order to help engineers gain a fast overview and intuitive insight into the flow past the marine turbine. The interactive multiple-coordinated view technique has gained its our popularity in exploring the data with high dimensionality as we discussed in Chapter 2.

[8] William Blake (1757 - 1827), was an English poet, painter, and printmaker.

So this presented system exploits multiple-coordinated information-assisted views of the CFD simulation data. Our application consists of a tabular histogram, velocity histogram, parallel coordinate plot, streamline graph and spatial views. Knowledge-assisted distortion is applied to provide more visual detail in regions of interest without losing visual coherency. Information-based streamline seeding is used to investigate the behaviour of the flow deemed interesting to the engineer. Specialised, application-specific information based on swirling flow is derived and visualized in order to evaluate turbine blade design. By incorporating multi-threading, the system provides the smooth and efficient user interaction. To demonstrate the usage of our system, a selection of specialised case scenarios designed to answer the core questions brought out by engineers is described. We also report feedback on our system from CFD experts researching marine turbine simulations.

## 5.1 Introduction and Motivation

As we approach the inevitable depletion of fossil fuel-based sources of energy such as oil and coal, ever increasing attention is paid to renewable energy sources. The United Kingdom has several natural resources from which sustainable energy may be generated. Solar and geothermal resources are difficult or expensive to exploit in the UK. Many believe that wind and ocean tides are rich energy resources. Currently around 1.5 % of electric energy is generated from wind, and this figure is expected to increase to 3.3 % [BWEAB07]. Although much work has been invested in exploiting wind power, effort to extract power from ocean tides is small in comparison. However, oceans contain a large amount of renewable, green, sustainable energy.

A marine turbine makes use of kinetic energy available in moving water, similar in concept to a wind turbine being powered by the wind. This application is gaining in popularity because of relative advantages marine turbines offer. First, the ocean tide is a more predictable resource compared to wind. Second, marine turbines are installed underwater. Hence the visible landscape remains unaffected. Last but not least, due to the slow rotation speed of the turbine, the tidal stream system has lower ecological impact compared to barrages. Based on these premises, the marine turbine has a promising future as a renewable source of energy. However the cost of installation and maintenance for a marine turbine is large. Therefore simulation tools are essential in order to minimise this cost.

Modelling and simulation are carried out to investigate how the flow past the marine turbine is affected and thus develop a better and more efficient marine turbine system. Some of the central questions that CFD (Computational Fluid Dynamics) engineers seek the answers to in their research are:

1. How does the flow past a marine turbine behave?
2. To what spatial extent does a marine turbine have a significant impact on the downstream flow structure?

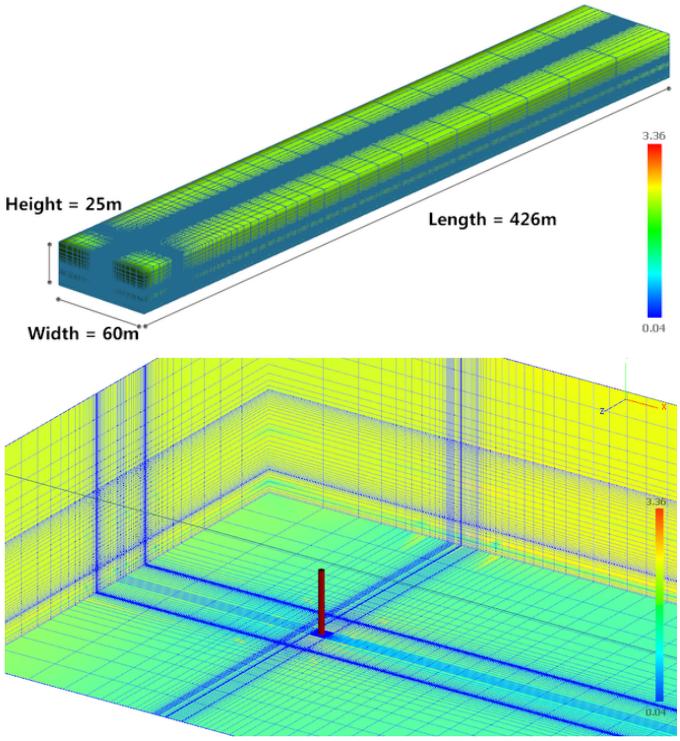


Figure 5.1: Schematic of the flat-bed geometry (top) dimensions of the domain. Lines highlight the adaptive resolution properties of the mesh which is much denser near the rotor area. (bottom) Detail of the mesh close to the pylon.

3. What kind of blade and pylon design can both maximise the energy drawn from the passing current while simultaneously minimising its impact on the momentum of the flow?

4. How closely can turbines be packed in a given region of the ocean floor such that each may effectively draw energy from the ocean tides?

There are multiple challenges, both technical and perceptual, to overcome to answer these questions. Challenges stem from the adaptive resolution, unstructured mesh used for the simulation as well as the high dimensionality of the CFD data. Commercial off-the-shelf tools do not always provide adequate visualization solutions for each user's needs.

In this chapter, we develop, explore and present customised information-assisted interaction and visualization techniques that help engineers answer these questions. We also present approaches for the engineers to gain a fast, intuitive and helpful insight into their simulation results. Our contributions are the following:

- We present a novel application that exploits multiple-coordinated views for interactive visualization of marine turbine simulation data. Two of the information visualization

views are a novel polar histogram of velocity and a histogram table that provide an overview of the high-dimensional data space. Other views are an interactive parallel coordinates and a novel streamline graph visualizations that support analysis and exploration of the CFD data.

- We experiment with novel knowledge-assisted distortion techniques that provide more visual detail in regions of interest without losing visualization coherency.
- We describe information-based and knowledge-assisted streamline seeding to investigate the behaviour of the flow past the marine turbine. Novel derived data used to evaluate blade design is computed and visualized based on swirling flow behaviour.
- We report feedback from CFD experts researching the simulation of flow past the marine turbine.

By utilising customised information and spatial visualization views which incorporate multi-threading, the simulation results can be explored and analysed interactively and more efficiently than standard commercial software. The user can multi-select or brush any attributes deemed interesting from an information visualization view, and the corresponding 3D visualization will be updated in the spatial view simultaneously.

The rest of the chapter is organised as follows: Section 5.2 provides an overview of related research work. The background of the simulation results is described in Section 5.3. Section 5.4 presents the detail of the application framework and its components. Section 5.5 discusses a selection of specialised case scenarios. The domain expert review of the application is provided in Section 5.6. Conclusions and suggestions for future work are presented in Section 5.7.

## 5.2 Related Work

Henze presents a multiple, linked view based system, called Linked Derived Spaces [Hen98], to visualize and analyse time-dependent CFD datasets. The objects from datasets can be interactively examined in various coordinate systems according to data attributes, like velocity, pressure etc., while the spatial connectivity and temporal characteristics are preserved. In this chapter, we implement some customised interactive information-based views: spatial visualization views and information visualization views to help engineers gain better understanding of the marine turbine dataset.

In order to make the design of multiple view based system more systematic and efficient, Baldonado et al. [WBWK00] present a set of guidelines for the design of multiple view systems. The first four guidelines (diversity, complementarity, parsimony, and decomposition) provide the designers with suggestions on selection of multiple views. The last four (space/time resource optimisation, self-evidence, consistency, and attention management) help designers make decisions on view presentation and interaction.

WEAVE [GRW+00] provides transparent linking between custom 3D scientific visualization and multidimensional statistical representations with interactive colour brushing for the user to select interesting regions. WEAVE is used to analyse and visualize simulated data of a human heart to allow scientists to more effectively and interactively study the structure and behaviour of the human heart.

Kosara et al. [KBH04] present an approach called the TimeHistogram as an extension to the traditional histogram which takes time into account for large and complex data sets. The 3D histogram is generated by arranging a number of regular 2D histograms along a third time axis to give users an intuitive overview of the development of a dimension over time. This is partly the inspiration behind the histogram table view feature in this chapter.

Stasko et al. [SGLS08] present a visual analytic system called Jigsaw, which provides an analyst with multiple perspectives (views) on a document collection. The system mainly focuses on exploring connections between entities across the documents and then applying a type of visual index onto the document collection for better investigation.

A technique powered by cross-filtered views is presented by Weaver [Wea09] for visual analysis of multiple dimensional international political event data. Cross-filtering is focusing on fast and flexible interactive visual analysis on fine-grained relationships buried in massive information from multiple data sets. By cross-filtering data values across pairs of views, analysts can quickly and interactively express sequences of multidimensional set queries to obtain the relation information they deem interesting.

Kehrer et al. [KFH10] discuss opportunities for interactively visual analysing the multi-run climate data. This is based on the integration of statistical aggregations with selected data dimensions in coordinated multiple view framework. Traditional robust estimates of mean, variance, skewness, and kurtosis statistical moments are integrated in an iterative visual analysis process as well as measures of outlyingness.

In order to alleviate the modifiable areal unit problem during geospatial analysis, Butkiewicz et al. [BMS+10] present a probe-based interface with coordinated multiple views for the exploration of the results of a geospatial simulation of urban growth. Firstly, the interface alerts the user if any potential unfairness is found during region based comparison. Then problem outliers are provided for user to evaluate. Lastly, semi-automated tools are provided to help the user to correct the detected problems.

Busking et al. [BBP10] present a technique with multiple strongly-linked views for visual shape space exploration and validation. The 3D object view provides local details for a single shape while the high dimensional points in shape space are applied in a 2D scatter plot to offer the global aspects. They introduce a new view called shape evolution view which visualizes the shape variability along a single trajectory in shape space.

Piringer et al. [PBK10] introduce HyperMoVal as interactive visualization of 1D CFD simu-

lations to support multiple tasks related to model validation. HyperMoVal can be linked to other views and further extends the possibilities for comparing known and predicted results, investigating regions with a poor fit, assessing the physical plausibility of models also outside regions covered by validation data, and comparing multiple models.

Wang et al. [WDeC+10] present an interactive visual analytics system based on multi-linked views as an extension to current bridge management systems. This system enables bridge managers to customise the visualization and data model so that it can provide interactive exploration, information correlation and domain oriented analysis to fit different needs.
With the exception of Henze [Hen98] all of the above systems focus on non-CFD applications. The most closely related system to ours is called SimVis. Doleisch et al. [Dol07][DGH03] present the SimVis application for interactive visual exploration and analysis of large, time-dependent, and high-dimensional data sets resulting from CFD simulation. Instead of the traditional automatic feature extraction, SimVis provides more freedom and user options to gain new insight into the data. Information visualizations include simple 2D/3D scatterplots, time-dependent histograms [KBH04], and parallel coordinates. The application also includes a fuzzy classification to make the transition from selected to non-selected region smoother. SimVis is used in several case studies and application examples. Many of them are automotive [LJH03][DMG+04].

In this chapter, we present a framework which is comparable to SimVis, however, our system is distinct from SimVis in the following ways:

- Our system is able to process and visualize the simulation data directly from Tecplot [Tec], the commercial visualization toolkit used by engineers at our university. No data conversion process is required.
- The multiple-linked views are especially customised according to the engineer's needs for interactive exploration of the simulation of flow past the marine turbine, a novel application.
- We introduce a novel streamline graph plot and a new 3D polar velocity histogram to facilitate analysis on swirling flow and its behaviour during evolvement.
- A histogram table provides a complete overview of the high-dimensional data.
- We also introduce a distortion technique as a user option to provide more details in the region of interest within a limited screen space.

SimVis does not feature a histogram table, polar histogram, streamline graph, or distortion. Additionally, since the size of the simulation datasets here is non-trivial, maintaining the smooth user interaction and rendering a large number of selected objects simultaneously is a challenge. In order to address this and deliver a good user experience, we develop a multi-threaded scheme inspired by Piringer et al. [PTMB09] for our system. Users can interact with the visualization result even as it is still being generated in the background.

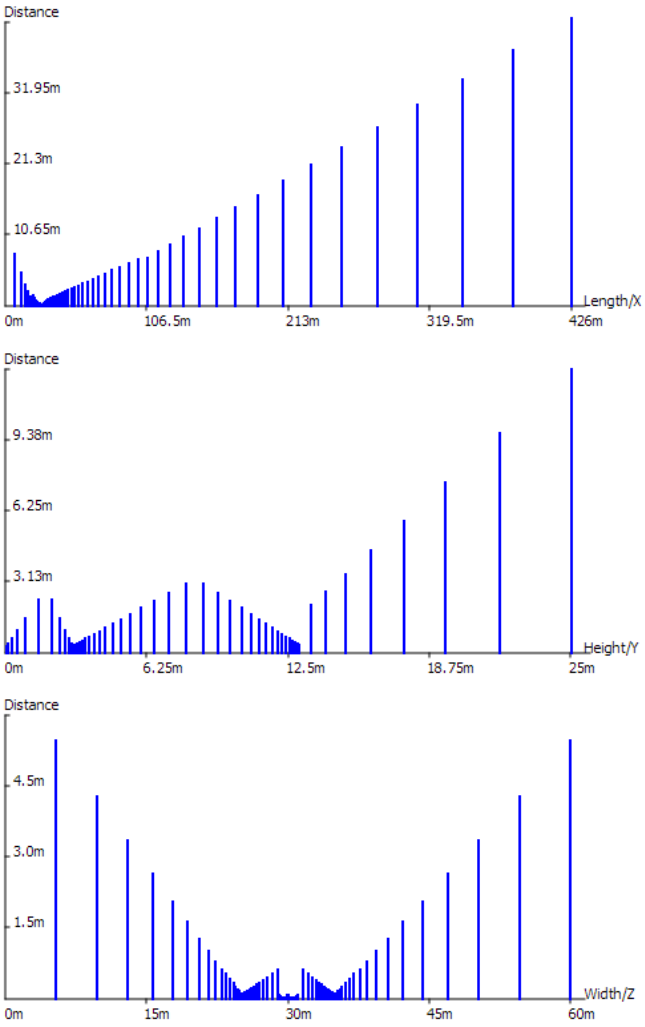


Figure 5.2: The top chart shows the distribution of yz slices along the x axis. The x axis in the chart represents the depth of a slice, and y axis indicates the distance between two adjacent slices. The middle chart shows the distribution of xz slices along the y axis while the bottom shows the distribution of xy slices along the z axis. We can easily see that slices are unevenly distributed.

## 5.3 Background of Simulation Results

Results presented are from a CFD simulation of a tidal stream turbine [WCM+10] [WCMW10] [WCMC09] [WMT+09]. We focus mainly on two representative simulations: a simulation with a single turbine and one with multiple rotors. We briefly describe the purpose of each and the adaptive resolution mesh used to generate each. We discuss some important simulation attributes which are key to answering the engineer's questions.

**Single Turbine Simulation:** To investigate the effect that the supporting structure (Figure 5.1(bottom)) has on the performance of a tidal stream turbine, the CFD simulation is carried out in a (length) 426 m $\times$ (height) 25 m $\times$ (width) 60 m underwater domain with a flat base [WCMC09]. The dimensions of the bounding domain for a rotor with 10 meters diameter are shown in Figure 5.1(top). The width of the domain is six times the diameter of the rotor in order to minimise the effect of the side boundaries on the flow past the turbine. The length of the downstream channel is 400 meters while the height is taken from bathymetry data at the actual turbine site in the Bristol (UK) Channel whose depth is 25 meters below low astronomical tide. The tidal stream turbine model consists of a 12 meter high vertical supporting tower - the pylon as illustrated in Figure 5.1(bottom). The centre of the rotor is 25.5 meters downstream from the in-flow boundary and is positioned on the middle plane in the width dimension. The mesh is constructed of approximately 350k hexahedral elements, the resolution of which is knowledge-based. Engineers are most interested in the behaviour of the flow near the pylon and blades. Thus the resolution of the mesh is highest there. Mesh resolution then drops off with distance from the blade. Figure 5.2 depicts the distribution of data samples along $x$, $y$ and $z$ axis. The height of each line represents the distance from this data sample to its neighbour along a given axis.

The simulation employs the boundary element momentum (BEM) [WCMC09] method. This method considers the time averaged influence of the rotors on the water. This results in the addition of a force term in the momentum equation which is only a function of radial and axial position relative to the rotor axis. The rotor shape is built into the source through its geometry and characteristics rather than the mesh. This method is unsuitable for predicting transient effects of blade motion but does allow time average effects to be resolved.

**Multiple Turbine Simulation:** In order to explore the impact from other turbines and determine how closely multiple turbines can be placed in a region so that each may effectively draw energy from the tidal flow, a simulation with four turbines is carried out in a 448m $\times$ 30m $\times$ 180m domain whose initial conditions are similar to the single turbine simulation. Three turbines are evenly spaced along width-axis at 100.5m downstream from the inlet-flow boundary. The fourth one is placed 132m directly after the front middle turbine. In contrast to the single turbine simulation, this simulation doesn't feature a pylon structure embedded in the mesh since the focus of this simulation is on the influence on the flow from multiple turbines rather than the pylon itself. The mesh consists of over 2.37 million hexahedral elements. The resolution is also knowledge-based.

Both simulations compute several parameters to describe the flow. Some of the important simulation attributes are: (1) Flow Velocity. (2) Relative Pressure: this describes the water pressure value relative to the out-bound flow boundary. Rapid change of the water pressure can seriously affect marine life. (3) Density: the attribute represents the density of water. This attribute may vary due to pollutants such as sediment. (4) Turbulent Kinetic Energy (TKE): the mean kinetic energy (per unit mass) associated with rotational flow. (5) Turbulent Dissipation Rate (TDR): the rate at which turbulent energy is absorbed and converted into heat. (6) Turbulent Viscosity: this describes the diffusive mixing of the flow turbulence. These attributes are deemed as key

for the CFD engineers to effectively understand and describe the flow as it passes by the rotor. In addition, we derive two attributes requested by the domain experts (7) swirl flow and (8) tangential velocity both of which measure types of swirling flow behaviour. These customised derived attributes are described in more detail in Section 5.1.

**Mesh Topology Computation:** In order to accelerate neighbour finding and point location for streamline tracing, the topology of the mesh is computed and stored. In this case the order of mesh cells varies in different regions, which makes the topology information extraction non-trivial. Additionally, the mesh is non-uniform, (See Figures 5.1). To address the problem of extracting the topology from this kind of mesh, we introduce a 3D lookup table in order to efficiently store the topology of each cell for streamline tracing. To illustrate this data structure, we use a 2D example shown in Figure 5.3 (top). In Figure 5.3 (top-left), a 2D mesh is composed of 14 rectangular cells. The sequence of the original mesh cells is indicated by a unique ID. If we want to trace a streamline, we could perform a naive search of the whole mesh to locate the cell that contains the streamline point. However, if the mesh data is very large and complex, this search procedure is expensive. Thus we create a corresponding lookup table $LUT$ shown in Figure 5.3 (top-right). The $LUT$ is made up of the projected cell coordinates along the $x$ and $y$-axis. In this example, we have 7 projected cell $x$-coordinates $(x_0, x_1, \cdots, x_6)$ and 7 along the y-axis $(y_0, y_1, \cdots, y_6)$, such that the cell resolution of the $LUT$ is $6 \times 6$. Each cell in the table stores the original cell ID. With this table structure, we can locate streamline points easily and quickly just by comparing coordinates of the streamline point $\mathbf{p}(x, y)$ with the projected cell

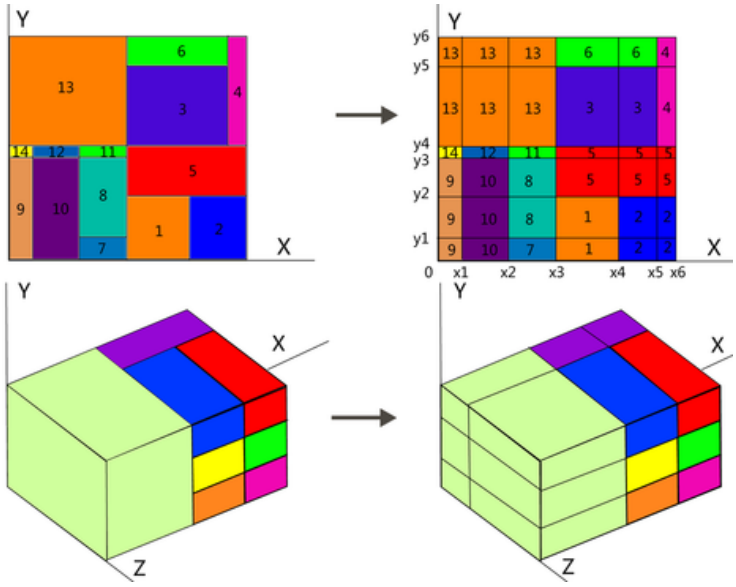


Figure 5.3: An example illustrates the mesh topology lookup table data structure in 2D (top) and 3D (bottom). (top-left) the original 2D mesh composed of cells whose numbers indicate the sequence. (top-right) a corresponding 2D lookup table is generated for the 2D mesh. (bottom) the process of a 3D lookup table being generated.

coordinates along $x$ and $y$-axis. For example, if $x_3 < \mathbf{p}(x) < x_4$ and $y_3 < \mathbf{p}(y) < y_4$, then cell 5 is identified. We extend this 2D $LUT$ to 3D by applying the 2D version to each layer of the original mesh, see Figure 5.3 (bottom).

The first procedure named $SortProjectedLists$ projects the boundary of cells onto the $x$, $y$ and $z$ axis, and stores these unique coordinates into lists $CoordListX$, $CoordListY$, and $CoordListZ$ respectively. Based on these lists, we generate the $LUT$ with the corresponding resolution, $NumCellX \times NumCellY \times NumCellZ$. For example, in Figure 5.3 (bottom) we have 4 unique values in $CoordListX$, 4 in $CoordListY$, and 3 in $CoordListZ$. Then a $LUT$ with $3 \times 3 \times 2$ resolution is constructed. Lastly, we initialise each $LUT$ cell, $c$, in the $LUT$ with the ID of the original cell, $e$, which encloses $LUT$ cell $c$. So the $LUT$ looks like the one in Figure 5.3 (bottom).

When a seeding point $\mathbf{p}(x,y,z)$ is given, $\mathbf{p}(x,y,z)$'s coordinates are compared with values in lists $CoordListX$, $CoordListY$, and $CoordListZ$ respectively. The index of the $LUT$ cell enclosing $p(x,y,z)$ can be calculated by $x\_index \times NumCellY \times NumCellZ + y\_index \times NumCellZ + z\_index$. Once the index is obtained, the original mesh cell ID can be retrieved from the $LUT$ to which the index is pointing.

By using the mesh topology $LUT$, our streamline tracing is much faster than Tecplot's [Tec]. We test both streamline functions on the Single Turbine Simulation which has over 389k hexahedra. It takes approximate $0.15s$ for Tecplot to trace a streamline while ours requires only about $9ms$ which is over 15 times faster. Based on this quick streamline tracing, our multi-threading powered system allows the user to interact or interfere with the streamline visualization even during it is still being generated, which Tecplot does not support.

We note that other cell location schemes are available [GJ10]. A full comparison of state-of-the-art methods is beyond the scope of this chapter. However, one advantage of our method is that it works directly with Tecplot's native data format. Thus no data format conversion process is required. This is very important in this special case. We evaluate against Tecplot because this is what the CFD expert users we collaborate with use in practice.

## 5.4 Application Framework

In this section, we describe the details of the system design and implementation. Firstly, an overview provides the general framework of what the system offers and how the user can interact with it. Then the linked information visualization views are described individually along with their motivation. Finally, spatial visualization options are presented based on information and knowledge-assisted queries.

Figure 5.4 shows an overview of the framework. The input includes the generic hexahedral meshes as the geometric representation for flow domain. Each vertex $i$ has a position $\mathbf{p}_i = (x(i), y(i), z(i))$, a velocity vector $\mathbf{v}_i = (v_x(i), v_y(i), v_z(i))$, and other simulation related

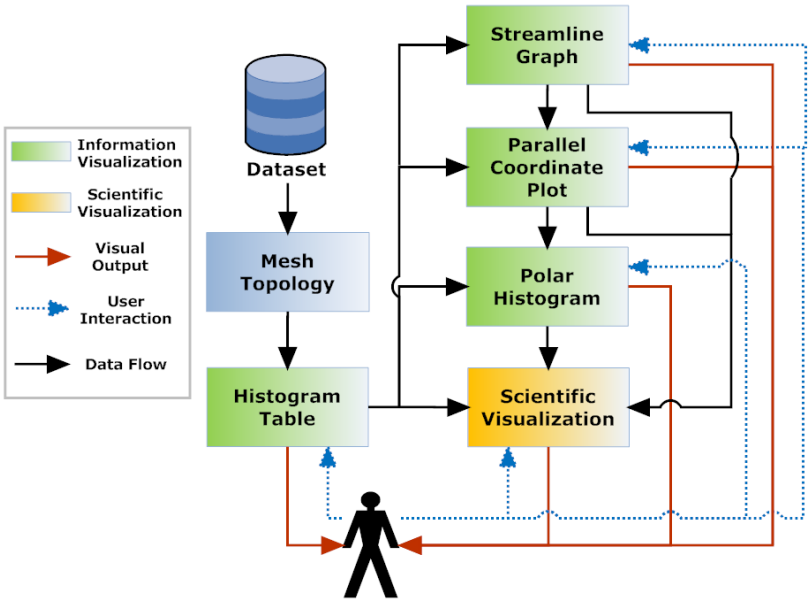


Figure 5.4: An overview chart illustrates the design and work flow between the user and our application framework.

attributes such as pressure, kinetic energy etc. The mesh topology is computed in order to accelerate streamline tracing. After the topology construction, various information visualization approaches are employed to gain insight into the data. The histogram table provides an intuitive overview of the multi-dimensional attributes of the whole simulation. Based on the histogram table, the user can focus on attributes they deem interesting, while the polar histogram and PCP (Parallel Coordinate Plot) simultaneously depict the details of the focus attributes. The polar histogram presents an intuitive description of the flow velocity distribution. The PCP highlights the relationship between CFD attributes to support exploration. The user can interact between different information visualization approaches to obtain the final spatial visualization result. In addition, the streamline graph is used to quantify the streamline curvature so that the specific streamline which has the most swirl can be obtained. In this section, we discuss these information visualization views in more detail.

### 5.4.1 Histogram Table

In order to quickly and efficiently present a large amount of multidimensional data, it is desirable to provide a quick overview of the whole data set. For this, we incorporate a histogram table. The histogram table represents the distribution of multidimensional information across the data set in an interactive visualization. As illustrated in Figure 5.5, the histogram table consists of a stack of histograms. Each individual histogram describes the distribution of a given simulation attribute. The name of the attribute is indicated as well as minimum and maximum values. The number of frequency intervals is defined by the user. The height of each bar is mapped to the volume of the mesh containing the data. Data range is colour-mapped.

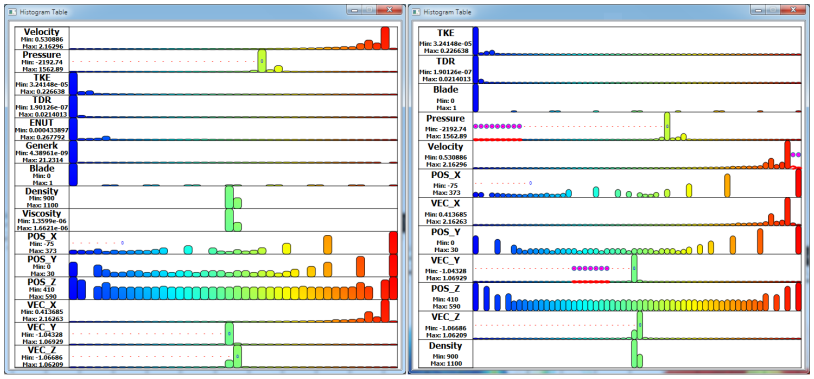


Figure 5.5: A set of histogram tables to provide an overview of the multidimensional information from the turbines simulation. (left) The default histogram table. (right) The sequence of attributes is reordered, and the attributes of ENUT, Generk, and Viscosity are excluded. The number of frequency intervals is increased to 60. The user selected bars are highlighted by the frame with dotted red lines and ellipses in fuchsia.

Red dash labels indicate the negative values, while blue zero labels are applied to highlight the categories ranging from negative to positive.

Based on the histogram table, users can interactively brush or multi-select bars (categories) they deem interesting or important, and thus other visualization views are updated and rendered simultaneously to provide the detail based on the selection. A user option is implemented to render the histogram table in landscape or portrait mode for better layout of linked views.

### 5.4.2 Polar Velocity Histogram

We have a view called the polar histogram for an integrated view of velocity distribution in a spherical coordinate system. See Figure 5.6. The velocity distribution is useful for the user to explore the direction in which the majority or minority of velocity points. Although both

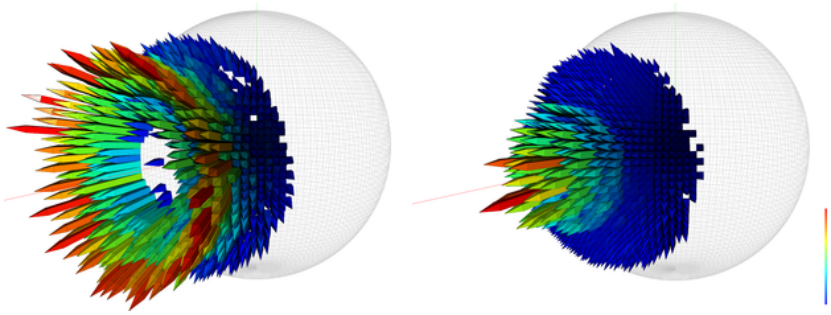

Figure 5.6: The polar histogram illustrates the velocity distribution of the tidal flow around blade elements from the multiple turbines dataset. (left) The velocity distribution of the flow with negative relative pressure while (right) shows the flow with positive pressure.

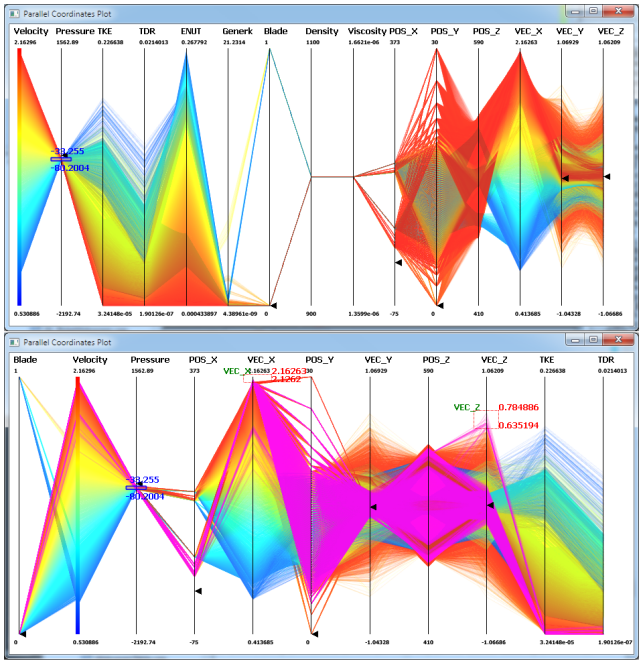


Figure 5.7: PCPs demonstrating the relationship among attributes when the user brushes the pressure ranging from $-80.2004$ to $-33.255$. (upper) The default view. (lower) The sequence of attributes is interactively reordered. The user brushes regions involving the flow with the highest positive velocity along x and z axis, and the selected polylines are highlighted in fuchsia. Colour is mapped to velocity magnitude.

the histogram table and the parallel coordinates plot have $v_x$, $v_y$, and $v_z$ attributes to indicate the velocity distribution, a polar representation is more intuitive because the $x$, $y$ and $z$ vector components are integrated. The polar histogram is based on a sphere whose bin resolution can be customised by the user. Each cell of the sphere wireframe represents a direction range. The height of the stack corresponds to frequency of vectors pointing in this direction.

### 5.4.3 Parallel Coordinate Plot (PCP)

A PCP is also integrated to help the user analyse and explore the multivariate data. The advantage of PCP is that it facilitates identification of correlation and clusters. We can have 15 parallel axes which represent simulation attributes. See Figure 5.7. Each axis reflects the distribution of a specific attribute. The name of the attribute is labelled with minimum and maximum values. When the user brushes or multi-selects categories from the histogram table, the PCP is updated to offer a detailed view of the selection. The user can examine polylines and may find some interesting correlation of variates. For example, all the samples with high velocity magnitude are associated with low pressure. Additionally, the PCP also enables the user to brush polylines of interest. Selected samples are highlighted in the spatial visualization to offer a more precise view of the area of interest.

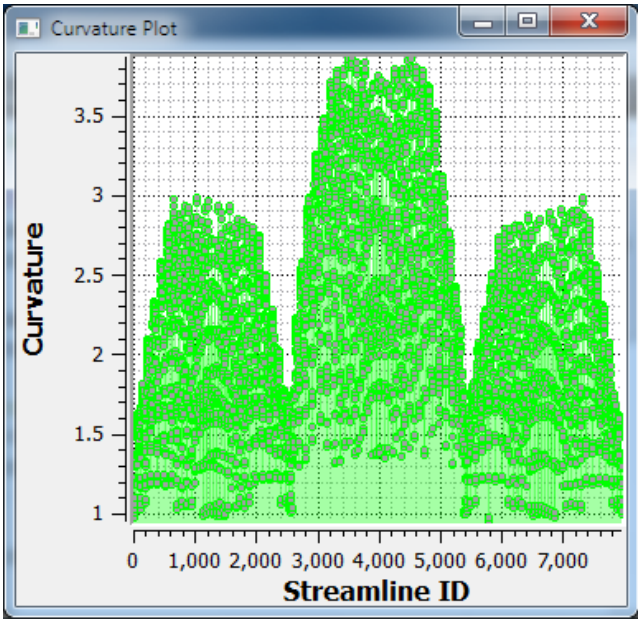


Figure 5.8: The streamline graph details the swirling rate of each streamline. The selected points on the graph are highlighted in red

### 5.4.4 Streamline Graph

Additionally, a novel streamline graph view is added to investigate specific streamlines based on swirl. During the streamline integration, the curvature is measured at each streamline vertex based on the angle formed by the line segments joined there. Once the streamline integration is finished the sum and the average of the curvature are stored at their respective seed positions. The higher the value is, the more swirl the streamline represents. We plot these values onto a line graph whose $x$ axis presents the streamline and $y$ axis maps to the curvature. See Figure 5.8. Each point on the plot presents a streamline. The user can select any point from the plot. The corresponding streamlines are rendered in the spatial view for further exploration.

### 5.4.5 Scientific Visualization

The scientific view renders the 3D result, for example the mesh, colour-mapping vector glyphs and streamlines, according to the user-specified filtering in the information visualization views. To enhance the visualization clarity, knowledge-assisted distortion and information-based streamline seeding can be applied.

#### 5.4.5.1 Distortion of Geometry

The resolution of the simulation mesh is non-uniform. Portions near the rotor and the middle are much denser than others as we can see in Figure 5.1. It is not always possible to visualize the entire domain at full data resolution simultaneously and render insightful imagery around

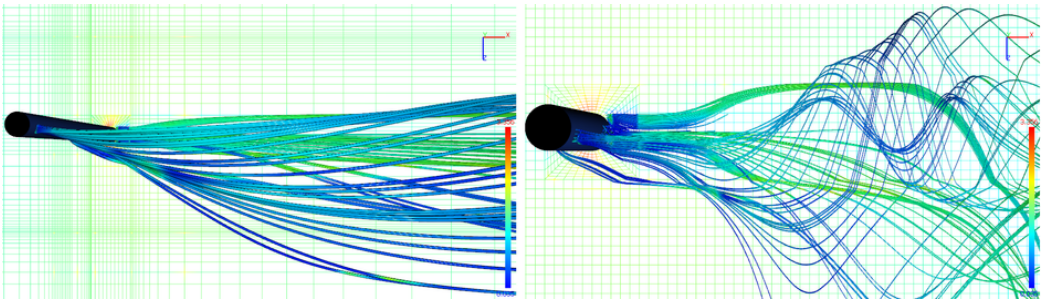

Figure 5.9: Streamlines depict the flow past the turbine. (left) Based on the original mesh without distortion, curves are crowded together and overlap, even zooming can't alleviate the overcrowding. (right) With distortion enabled, the region of the simulation which is deemed interesting by the user is expanded so that streamlines can be easily viewed.

the rotor geometry. Either we obtain a less-detailed view of the whole domain or we can zoom in on an area of interest for greater detail at the expense of an overview of the flow. However when the user zooms in (Figure 5.9 (left)) in the traditional way, they may loose their orientation. Zooming allows the user to focus on the region of interest, for example the area around the rotor, but other parts of object that are out of the focus are cropped. Zooming can't offer a full view of the model simultaneously, which is helpful to the engineers. Hence we have implemented a user option to distort the domain guided by the mesh resolution. This distortion is inspired by the Focus+Context frameworks which enlarge the area of interest without removing other parts [CCCF96, CCF97]. The distortion converts each mesh cell to the overall average sized cell. The average size is computed automatically. The distortion enables users to intuitively observe details of the visualization in the region of interest while keeping the overview of the whole visualization of the past flow in the same view, see Figure 5.9.

#### 5.4.5.2 Information Assisted Streamline Seeding

One of the commercial visualization packages used by engineers at our university is Tecplot. Tecplot offers streamline seeding rakes which can be used to manually seed streamlines throughout the domain. This is time-consuming and error-prone. A manual search does not guarantee that the users discover the features they are looking for. Although fully automatic seeding strategies exist - engineers will always want control over the seeding. This is because complete knowledge of how any automatic algorithm works is required for correct interpretation of the result. Plus, not all features of interest can be known a priori and then extracted. Our system enables the user to seed streamlines based on the knowledge of the domain expert and the information provided by the histogram table, velocity histogram, streamline graph and PCP. See Figure 5.10. We illustrate some case scenarios in the next section.

## 5.5 Application Use Case Scenarios

In this section, we apply our application to specialised turbine-centred scenarios in order to answer questions raised by the engineers, and demonstrate how engineers benefit from its use during the analysis process, as well as being able to obtain new findings which conventional commercial visualization tools, like Tecplot, may overlook. To highlight the difference our application makes, a comparison to the use of Tecplot is discussed.

### 5.5.1 Visual Analysis of Swirling Flow

Swirling flow is important in this application because it indicates torque on the turbine blades. This in turn is a measure of rotor performance. Quantifying, analysing, and visualizing swirl can aid in rotor design. To help engineers locate, quantise and visualize swirling flow in the simulation result and explore its behaviour, a query concerning the swirling flow is input to illustrate where the most swirling occurs and how it behaves after passing the turbine.

To locate where the most swirling flow occurs, the tangential velocity about the rotor axis is recommended by the domain experts for the swirl quantification. Figure 5.12 demonstrates how we calculate the tangential velocity. In order to calculate the tangential velocity (green) $\mathbf{v}_t(\mathbf{p})$ of the given sample point $\mathbf{p}$, the yz-plane which includes $\mathbf{p}$ is drawn. On the yz-plane, $\mathbf{v}(\mathbf{po})$ (black), the vector from point $\mathbf{p}$ to $\mathbf{o}$, the connecting edge of the rotor axis and the perpendicular yz-plane, can be easily calculated. $\mathbf{v}(\mathbf{po})$ is perpendicular to the rotor axis as well as $\mathbf{v}_t(\mathbf{p})$, so that the vector perpendicular to $\mathbf{v}(\mathbf{po})$ on the yz-plane, $\mathbf{v}(\mathbf{po})^{\perp}$, has the same direction as $\mathbf{v}_t(\mathbf{p})$. As the projection on the yz-plane, $\mathbf{v}_{yz}(\mathbf{p})$ (pink) can be obtained from the original velocity $\mathbf{v}(\mathbf{p})$ (light blue) on $\mathbf{p}$. By using $\mathbf{v}_{yz}(\mathbf{p})$ and $\mathbf{v}(\mathbf{po})^{\perp}$, $\theta$ can be obtained via:

$$\theta = \arccos\left(\frac{\mathbf{v}_{yz}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{po})^{\perp}}{\|\mathbf{v}_{yz}(\mathbf{p})\| \, \|\mathbf{v}(\mathbf{po})^{\perp}\|}\right) \tag{5.1}$$

now the magnitude of the tangential velocity $\mathbf{v}_t(\mathbf{p})$ which we use to measure the swirling flow can be calculated by:

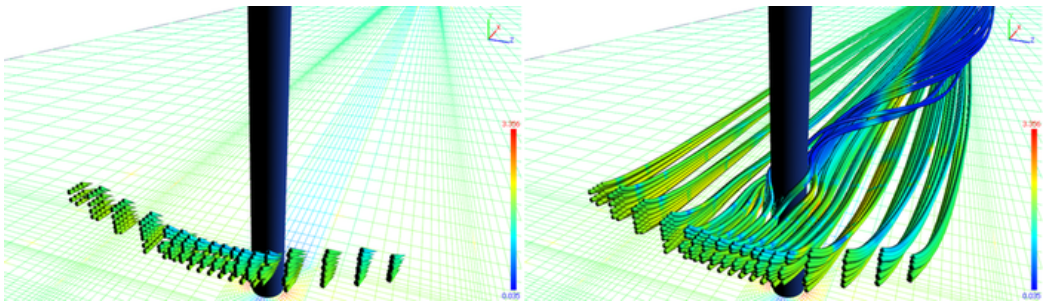

Figure 5.10: Glyphs (left) and knowledge-guided streamlines (right) are seeded to visualize the behaviour of the flow with maximum velocity momentum in negative $y$ direction. The flow starts down toward the bottom and then it turns upward after it passes the pylon.

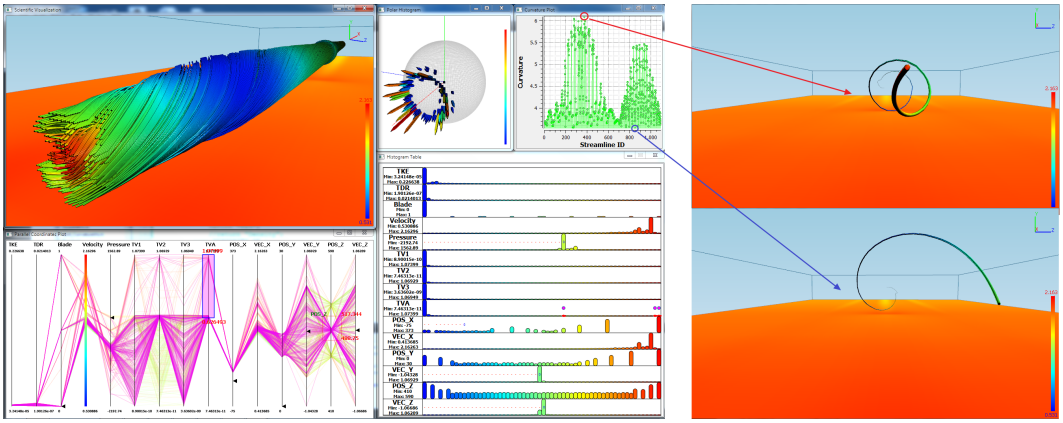

Figure 5.11: The swirling flow about the axis of centre rotors is derived and analysed. TVA indicates the tangential velocity of the nearest axis. TV*n* indicates the tangential velocity about the axis of rotor *n*. For example, TV2 shows the tangential velocity about rotor 2's axis, at the centre in this case. By selecting the centre rotor from the PCP, streamlines are rendered which demonstrate the behaviour of flow about the centre rotor. The streamline graph also provides a diagram showing the curvature of each streamline. By selecting the streamlines with highest and lowest curvature, the linked spatial visualization is updated.

$$\|\mathbf{v}_t(\mathbf{p})\| = \|\mathbf{v}_{yz}(\mathbf{p})\| \cos(\theta) = \frac{\mathbf{v}_{yz}(\mathbf{p}) \cdot \mathbf{v}(\mathbf{po})^{\perp}}{\|\mathbf{v}(\mathbf{po})^{\perp}\|} \tag{5.2}$$

We compute equation (5.2) for every sample of the simulation result. The derived tangential velocity field is stored as an additional attribute of the data. We plot the tangential velocity in

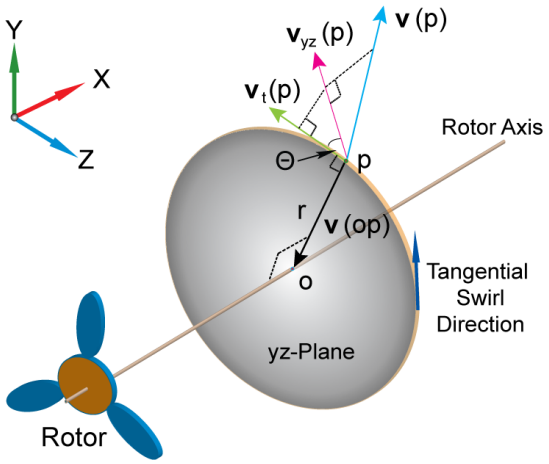


Figure 5.12: The diagram illustrates the derivation of $\mathbf{v}_t(\mathbf{p})$ (green arrow), the tangential velocity of the flow at point $\mathbf{p}$.

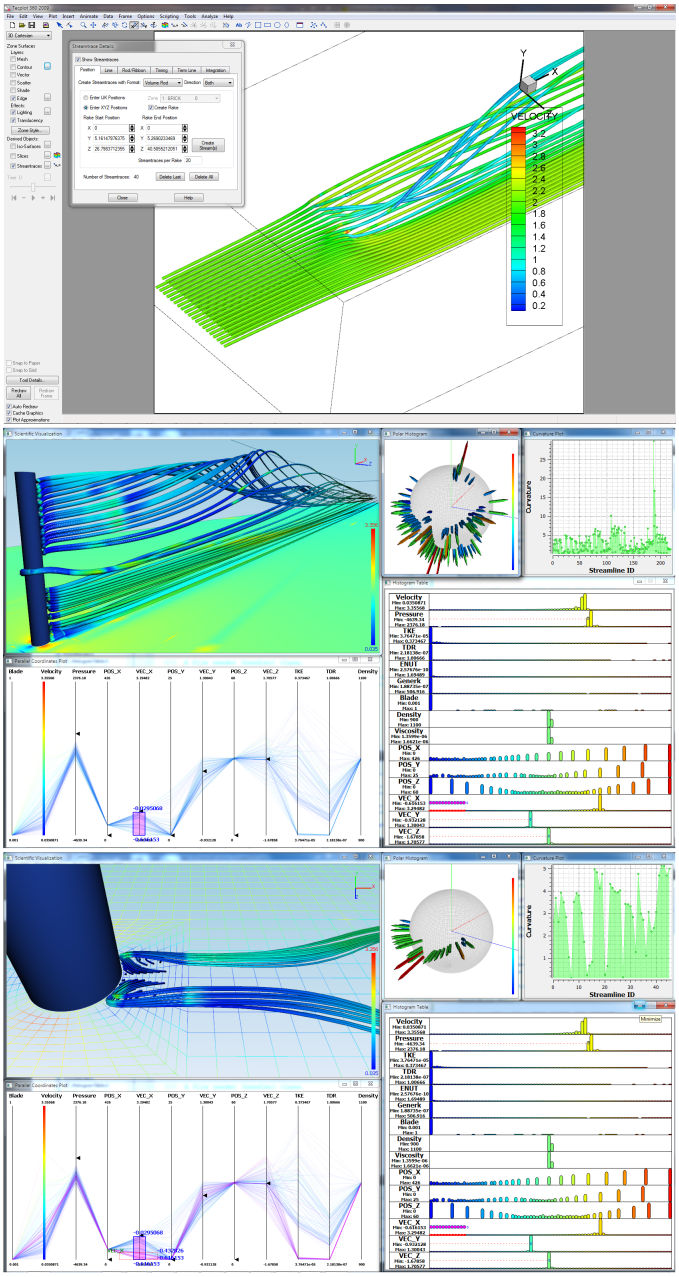


Figure 5.13: This scenario extracts and analyses reverse flow. (top) Streamlines are traced from manually seeded seeding rakes in front of the flow domain in Tecplot [Tec]. (middle) In our framework, the reverse flow can be intuitively obtained from the histogram bars which contain negative velocity values. Streamline seeds are automatically placed based on the selection only in order to visualize the behaviour of the reverse flow past the pylon. (bottom) The distortion and the user interaction on PCP are applied to filter and enhance the final visualization.

the histogram table and PCP so that engineers can analyse the swirling flow. In Figure 5.11, we simply select histogram bars representing high swirling flow about the axis of the nearest rotor from histogram table and a group of median swirling flow for comparison. The PCP is updated to reveal the relationship to the other simulation attributes. We focus on the flow about the centre rotor by brushing the corresponding position from the PCP. Most of the swirling flow has comparatively high velocity magnitude. The streamline visualization depicts the origin of the swirling flow and how it evolves. When we showed the domain experts the visualizations in Figure 5.11, they were immediately surprised that the streamline seeds exhibit asymmetry. They expected complete symmetry. Further investigation is required to identify the source of the asymmetric swirl flow behaviour. The streamline graph provides a detail view of the curvature of each streamline. The periodicity in the streamline graph obviates the periodicity of the swirl behaviour in the flow domain. By selecting the point we deemed interesting, the spatial view is updated to highlight the selected streamline.

### 5.5.2 Areas of reverse flow

To visualize how the flow past marine turbine behaves and how the pylon design impacts on the passing current, a visual query for reverse flow is provided to illustrate the analysis process. Reverse flow is deemed detrimental because it draws useful kinetic energy from the overall flow current. Minimising the reverse flow is one of goals for the optimal blade and pylon design. However, locating and visualizing reverse flow in the simulation result and investigating its behaviour is a challenge for engineers using conventional visualization tools.
Tecplot provides a manual selection as a basic analysis approach. The user can click on the geometry to retrieve the corresponding simulation data attributes. However, it is difficult and time-consuming for the user to manually search for reverse flow. We can use its seeding rake to generate streamlines to visualize the flow. But the placement of streamlines seeds is manual. A manual search of the domain with a seeding rake is time-consuming and error-prone.

In the system a histogram table is developed to provide the user with an intuitive and quick overview of the information contained in the simulation data. The user is able to select negative vectors along x axis (reverse flow), by simply brushing the corresponding bars in the histogram table. See the highlighted bars in Figure 5.13 (middle). The computation is concentrated on the user selection. The PCP conveys the correlation between different attributes of the reverse flow and the polar histogram renders the distribution of reverse flow in the spherical coordinate system. Streamlines in scientific visualization view are seeded in regions which contain reverse flow and traced to depict its behaviour. We call this knowledge-based seeding because it exploits the knowledge of the domain expert directly. See Figure 5.13. Additionally, by interacting with other attributes from the PCP view, the initial selection can be analyzed.

### 5.5.3 Optimal Placement of Turbines

Maximising the energy drawn from the passing tide while minimising the number of turbines is one of the key challenges the engineer faces. The amount of the kinetic energy drawn from the tidal current may depend on where the marine turbine is positioned, especially with respect

to surrounding turbines. Engineers would like to pack turbines as closely as possible to maximise the amount of energy converted from the tidal currents. However, placing turbines too close together makes them ineffective because an upstream turbine reduces flow momentum for those downstream. Thus a trade-off is made. In order to explore the impact of turbines in proximity to one another and determine how closely multiple turbines can be placed in a given region, a query concerning spatial extent of tidal current is input to explore how much energy each turbine can extract and how the flow recovers after passing each blade. In order to answer this question, we use isosurfaces to visualize the spatial extent of tide momentum.

Isosurfaces are useful in order to help CFD engineers study boundaries of the fluid flow around objects. Isosurfacing is a standard visualization tool in commercial visualization toolkits. The flow along x-axis in the positive direction is that which the turbine can draw the most energy from. Simulation experts consider flow that has recovered 90 % (or more) of its momentum with respect to the average current (represented by the inlet) to be a good candidate for converting energy to the turbine. When we choose 90 % of the inlet flow as the isovalue, corresponding isosurfaces in Figure 5.14 depicts that three front turbines can draw most energy out of the passing flow. The flow behind the front three turbines does not return 90 % of its original momentum with the current domain. The turbine downstream can not draw energy out of flow effectively. We reduce the isovalue to 35 % of the inlet flow and the corresponding isosurface, Figure 5.14, illustrates that the isosurface for the front-middle turbine just ends before the one behind. The turbine behind can only draw the energy up to 35 % of the inlet flow. This visualization illustrates the spatial extent that the marine turbines have with respect to the surrounding tidal momentum. If flow momentum has linear behavior the downstream distance between turbines must be 2-3 times greater. More simulation is necessary to confirm this hypothesis.

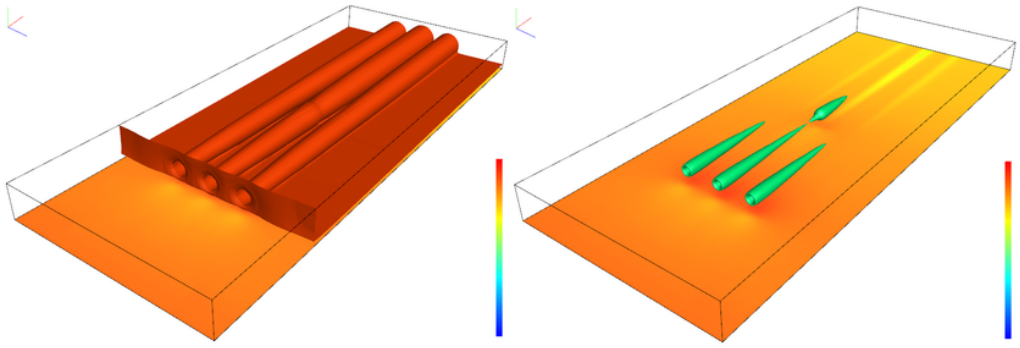

Figure 5.14: Isosurfaces depict the spatial extent of the tidal flow passing around turbines. (left) We choose 90 % of the inlet flow as the isovalue to trace the isosurface. We learn that the first three turbines are more efficient than the one downstream. (right) Then we decrease the isovalue to 35 %. We find the one downstream can only draw up to 35 % the energy from passing flow.

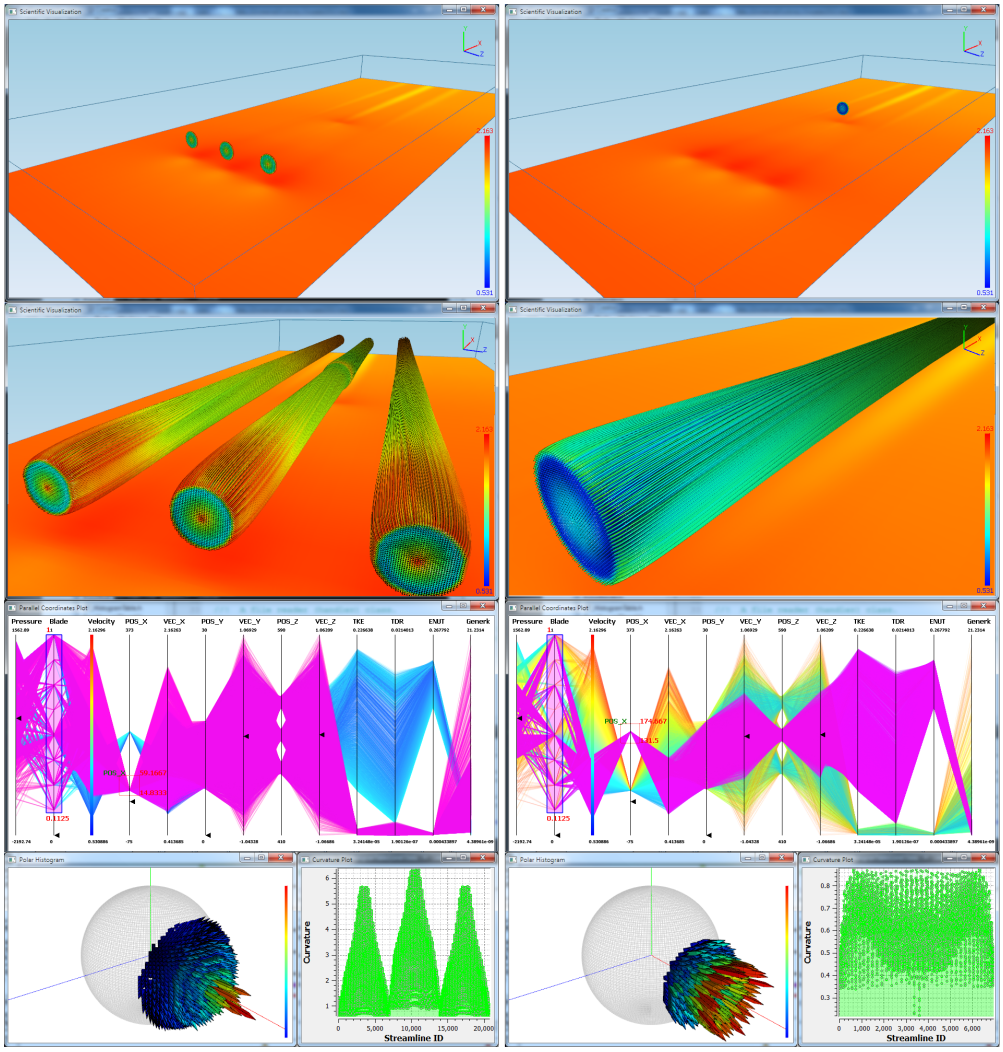


Figure 5.15: This scenario analyses the flow past the time-averaged blades. We select bars representing blades in the histogram table, and blade elements are rendered in the spatial view. (left): We brush front rotors from PCP by selecting corresponding values from *pos_x*. The polar histogram shows most of the flow hitting blades travels in the positive $x$ direction, which has high velocity but low TKE and TDR. (right): The turbine downstream is selected. The majority of the associated flow has comparatively low energy but high TKE and TDR. Streamlines are traced to visualize the behaviour of the flow respectively.

### 5.5.4 Flow Past the Time-Averaged Blades

In order to enhance the user's understanding of the correlation between attributes of each turbine, our customised multiple linked views system provides a more flexible and effective interface to the simulation data. We focus on the flow which directly contacts the turbine. We brush bars representing the blade in the histogram table and the corresponding flow is conveyed. The four blade elements are displayed in the scientific view. The polar histogram shows that the

majority of the flow hitting the blade travels in the positive $x$ direction. The PCP reveals the details of the attribute correlation for the user selection. The colour legend is mapped to flow magnitude. In order to isolate and analyse the flow contacting the front turbines only, we brush front blades from PCP. See Figure 5.15. The highlighted polylines illustrate that the majority of the passing flow contains high velocity (see polyline distribution for $velocity$ and $vec\_x$) while the turbulent kinetic energy (TKE) and turbulent dissipation rate (TDR) are comparatively low. We trace streamlines to visualize how the flow behaves in Figure 5.15. When we select the downstream blade from PCP. We find some interesting features. The majority of flow has comparatively low energy (velocity) and the corresponding TKE and TDR are high. From the polar histogram, we can see that the majority of the flow does not move straight toward the turbine, which is not ideal for energy extraction. Streamline seeding is also applied.

### 5.5.5 Areas of Min/Max Pressure

In order to locate and analyse flow with minimum or maximum pressure a corresponding scenario is provided. We brush the minimum and maximum pressure values from the histogram table. See the highlighted bins in Figure 5.16. The polar histogram reveals that most of the flow travels in the positive $x$ direction. The PCP is also updated to show the correlation between each simulation attribute. The colour applied is mapped to pressure value. We start with brushing the minimum value of the negative pressure it in the PCP. See Figure 5.16. The highlighted polylines reveal that the flow with high negative pressure has comparatively high velocity in the positive $x$ direction. However, for the maximum pressure we find that the corresponding flow occurs around turbines and contains average energy. Streamline visualization is used to depict the path of the flow.

## 5.6 Domain Expert Review

To evaluate our presented method in this chapter, we invite CFD experts Dr Nick Croft, Dr Rami Malki and Dr Ian Masters from the School of Engineering at Swansea University to help us on the evaluation of the presented algorithm. Their feedback is as below.

"Computational simulation often produces result files that contain relatively little data of significance. From the engineers perspective the primary aim of visualization is to cut through the irrelevant data to highlight the important features. This highlighting comes with two main aims, to explain and to understand. These two aims come with slightly different constraints on the visualization software. Most commercial visualization software provides a solution route for the first aim. This solution requires accurate physical representation of the geometric data as well as informative representation of the data either through techniques such as contours, glyphs, streamlines or isosurfaces. For the engineer to explain data there is a requirement to understand it and this is a task that is harder to achieve through the use of commercial packages.

Understanding data often requires the loosening of the dominance of the spatial data that is fundamental to the explanation. One of the basic techniques of simulation involves the placement of many data points in areas of rapid change, which are usually areas of interest. Being able

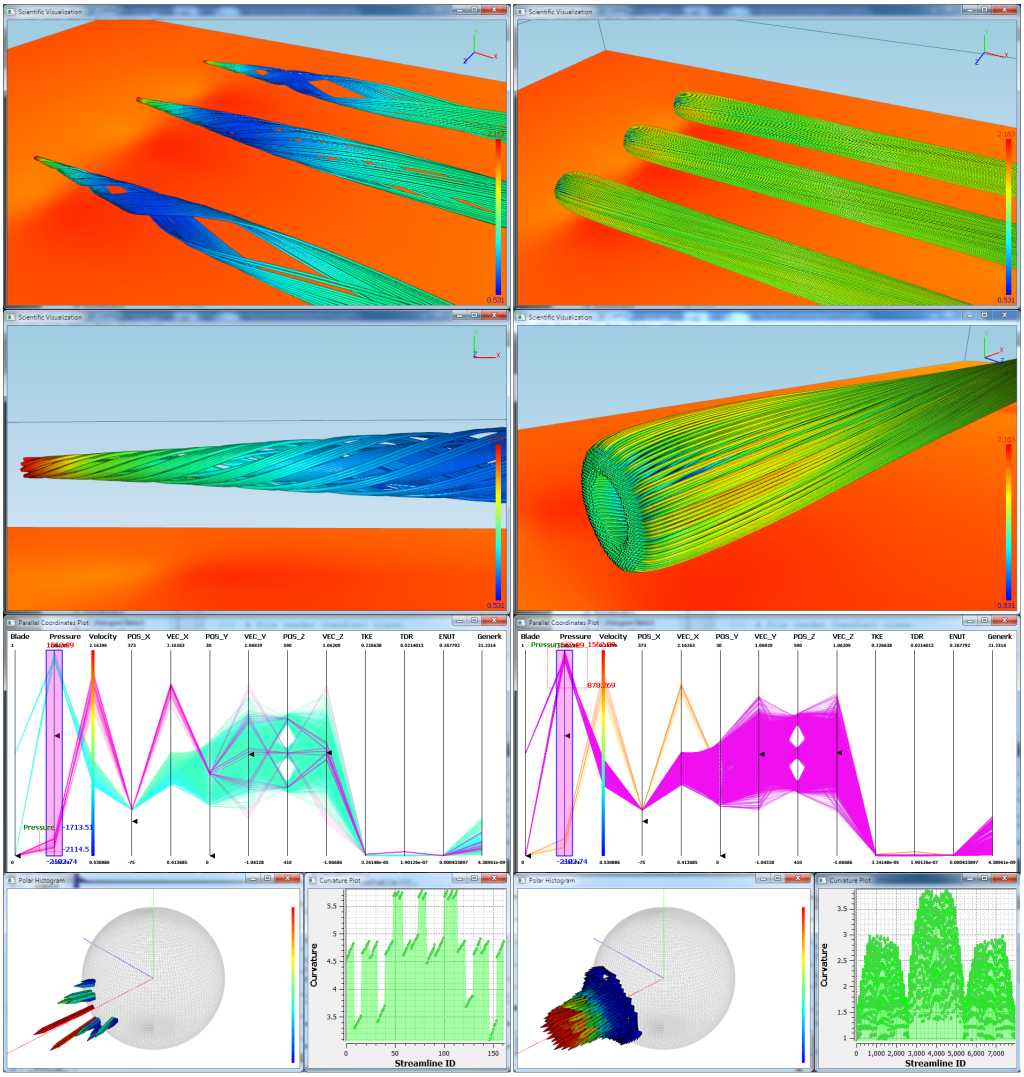


Figure 5.16: The flow with the minimum or maximum pressure is selected and visualized for analysis. (left column) The flow with high negative pressure is depicted, while (right column) the maximum pressure is illustrated. Streamlines are used to reveal the flow behaviour. Zooming views are obtained: flow with the minimum or the maximum pressure.

to 'see' this data often requires significant zooming into the data. This hides the relationship with the rest of the solution domain. The distortion method described in this contribution is a useful technique that performs the equivalence of a zoom in the fine mesh areas whilst retaining surrounding regions. Spatial data may play a very secondary role in some understanding where the fundamental questions concern the relationship between two variables. The PCP offers an excellent route to investigate questions such as what is the effect on other variables, specifically turbulence, of reverse flow. The multiple displays then provide a route back to the spatial relevance which will be part of any explanation.

The modelling of tidal stream rotors has beginnings in the work of Goldstein [Gol29] who first applied this to helicopter rotors. This classical momentum theory describes the interaction of the rotor in terms of horizontal and tangential interference factors. This has been successfully applied to the tidal energy case [MCOW11]. It is of great benefit for the performance design of rotors to be able to describe CFD results in the same terms. However, the nature of the CFD formulation means that results are described on the individual nodes of individual cells. Therefore there is a strong motivation to recover information equivalent to the interference factors. Axial interference can be relatively easily obtained from the velocity information, defining velocity deficit as the difference between local velocity and the upstream conditions. It is obvious to see how this can be described as an additional parallel coordinate. Swirl is less intuitive and the results presented here have two important features: firstly the term is defined in a mathematical way that is consistent with the classical approach of momentum theory, and secondly it recovers the global rotor flow features from the discretised cells. This novel approach is of great benefit to the user and the intelligent information this provides is vital to the engineering design of these systems."

## 5.7 Conclusion and Future Work

In this chapter we propose a novel application which exploits multiple-coordinated views for interactive information-assisted visualization of marine turbine simulation data such that engineers can gain a fast overview and intuitive insight. The system includes an interactive polar histogram of velocity, a histogram table view for an overview of the CFD data, an interactive parallel coordinate visualization, and a streamline graph which provide further analysis and exploration. Knowledge-assisted distortion is applied to provide more visual detail in regions of interest without losing visualization coherency. Information-based streamline seeding quickly and automatically visualizes the behaviour of the flow deemed interesting by the engineer. The information-assisted views of the data have helped domain experts discover new properties of their data which they were previously unaware of. Multi-threading makes the interaction between the system and the user smooth and efficient. The comparison between one of commercial off-the-shelf visualization tools - Tecplot [Tec] and our system is provided in the case scenarios which are designed to answer core questions brought forth by engineers in order to demonstrate that our system is more suitable for this task. We also report feedback from CFD experts researching the simulation of flow past the marine turbine.

As future work we would like to introduce a more generic, though slightly more computationally expensive method to calculate tangential velocity with the direction of the rotational flow. Assume $\mathbf{r}_A$ is a point on the turbine axis and $\hat{\mathbf{z}}$ is a unit vector in the direction of the axis. The position of any $\mathbf{p}$ can be expressed as $\mathbf{r_p} = \mathbf{r}_A + \lambda\hat{\mathbf{z}} + \mu\hat{\mathbf{r}}$. In this equation $\hat{\mathbf{r}}$ is a unit vector in the radial direction associated with the point and the Greek characters represent distances in the respective direction. The velocity at the point can be expressed in its polar form as $\mathbf{v} = \mathbf{v}_r\hat{\mathbf{r}} + \mathbf{v}_z\hat{\mathbf{z}} + \mathbf{v}_\theta\hat{\theta}$. If $\hat{\theta}$ is known then its dot product with the above equation provides the equation for the tangential velocity $\mathbf{v}_\theta$. $\mathbf{v}\cdot\hat{\theta} = \mathbf{v}_\theta$ is used. To obtain the above equation

from the previous one the orthogonality of the unit vectors in the polar coordinate directions has been used. All that remains is to obtain the unit vector in the tangential direction. The first stage in this task is to recall the equality $\hat{\theta} = \hat{\mathbf{z}} \times \hat{\mathbf{r}} = -\hat{\mathbf{r}} \times \hat{\mathbf{z}}$. Taking the cross product of $\mathbf{r_p} = \mathbf{r}_A + \lambda\hat{\mathbf{z}} + \mu\hat{\mathbf{r}}$ with the unit axial vector and after minor rearrangement the following equation is obtained by $(\mathbf{r_p} - \mathbf{r}_A) \times \hat{\mathbf{z}} = \mu(\hat{\mathbf{r}} \times \hat{\mathbf{z}}) = -\mu\hat{\theta}$. As we are looking for an unit vector the above equation can be rearranged to give $\hat{\theta} = \frac{(\mathbf{r_p} - \mathbf{r}_A) \times \hat{\mathbf{z}}}{\|(\mathbf{r_p} - \mathbf{r}_A) \times \hat{\mathbf{z}}\|}$. This equation only contains the three known quantities and so can be immediately calculated and the result used in $\mathbf{v} \cdot \hat{\theta} = \mathbf{v}_\theta$ to get the tangential velocity. This method provides a generic algorithm to obtain a signed tangential velocity value.

We also would like to explore possibilities of transferring more of the computation to the GPU. We would also like to extend the work to visualize the time-dependent marine turbine flow. Future work also includes the investigation of possible applications on other domains, such as wind turbines.

*Chapter 6*

# Design and Implementation of a System for Interactive High-Dimensional Vector Field Visualization

**Contents**



> "In most people's vocabularies, design means veneer. It's interior decorating. It's the fabric of the curtains of the sofa. But to me, nothing could be further from the meaning of design. Design is the fundamental soul of a human-made creation that ends up expressing itself in successive outer layers of the product or service."
>
> - Steve Jobs [9]

In this chapter we describe the design and implementation of a generic framework incorporating a selection of related scientific and information visualization techniques which are

[9] Steve Jobs (1955 - ), is an American computer engineer and industrialist. He is the co-founder and chief executive officer of Apple Inc.

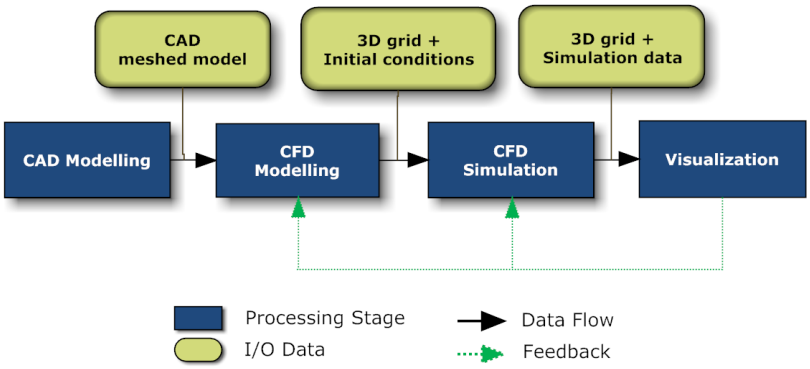


Figure 6.1: The CFD process is composed of modelling, simulation, and visualization stages.

designed and integrated to provide the user solutions for effective visualization of the high-dimensional CFD flow simulation data. In contrast to most research prototypes, the system we present handles real-world, unstructured simulation data. Our framework provides direct, feature-based and geometric flow visualization techniques and supports information visualization approaches, such as a tabular histogram, velocity histogram, and parallel coordinate plot. In order to enable a smooth and efficient user interaction, these visualization options are systematically combined on a multi-threading platform which ensures responsiveness even when processing large high-dimensional data.

## 6.1 Introduction

Over the last three decades, computational fluid dynamics (CFD) has developed very rapidly. Its applications range widely from the automotive industry to medicine [LEG+08]. This is because CFD *modelling* and *simulation* speed up the manufacturing process. Constructing objects and simulating experiments in a software environment is normally faster and cheaper than building and testing physical hardware counterparts in real world. As another important part of the CFD pipeline, the *visualization* process not only provides the engineer the visual result of the simulation, but also verifies or conflicts with the results expected by the engineer so that the original model design can be approved or improved. The CFD process, illustrated in Figure 6.1, is composed of three main stages:

1. Modelling: a 3D structured or unstructured, volumetric or surface mesh is generated to model the physical object. This procedure is based on computer aided design (CAD) modelling.

2. Simulation: a computational simulation of a fluid through the given model in the previous stage is computed in a 3D simulation environment with a set of given initial conditions.

3. Visualization: the simulation result is explored, analysed, and visualized in different ways according to different needs.

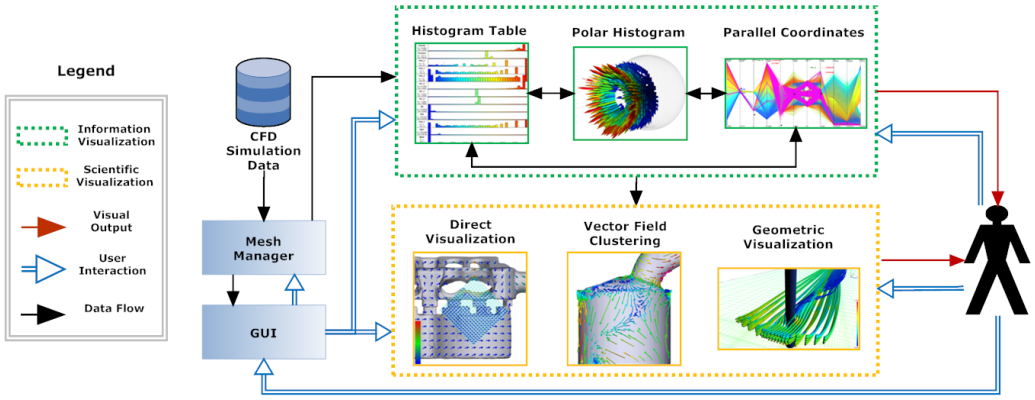


Figure 6.2: An overview the application framework.

Since the size, complexity and dimensionality of the CFD simulation data have dramatically increased in recent years, so has the need for visualization which provides quick and effective insight into the data [PL09]. In order to present a visualization toolkit which is capable of dealing with large, complicated, and high-dimensional CFD simulation data, a comprehensive and versatile visualization framework is needed. In this chapter we focus on the design and implementation of a generic visualization framework which provides the user solutions for effective visualization of the high-dimensional CFD simulation data by combining several scientific and information visualization techniques. This visualization framework yields following benefits:

- The framework handles versatile real-world, unstructured 2.5D, 3D, and $n$D CFD simulation data.
- Direct, geometric, and feature-based flow visualization techniques are integrated in order to support the CFD engineer with intuitive and rich visualizations for effective visual analysis.
- Information visualization approaches, such as tabular histogram, velocity histogram, and parallel coordinate plot (PCP), are incorporated to enable engineers to gain an in-depth analysis of the simulation data and thus focus on parts they deem interesting.
- Smooth and efficient user interaction is ensured by our multi-threading application, even when large data sets are processed.
- Our flow visualization framework is platform independent in terms of both hardware and software.
- The framework is a open source project with simple API provided. The full system document generated by Doxygen [vH] is available online ( `http://cs.swan.ac.uk/~cszp/mt_docs/index.html`). Bob's coding conventions [Lar07] are applied.

We discuss the details of several aspects related to the design and implementation of our flow visualization framework. The corresponding advantages and disadvantages are also discussed.

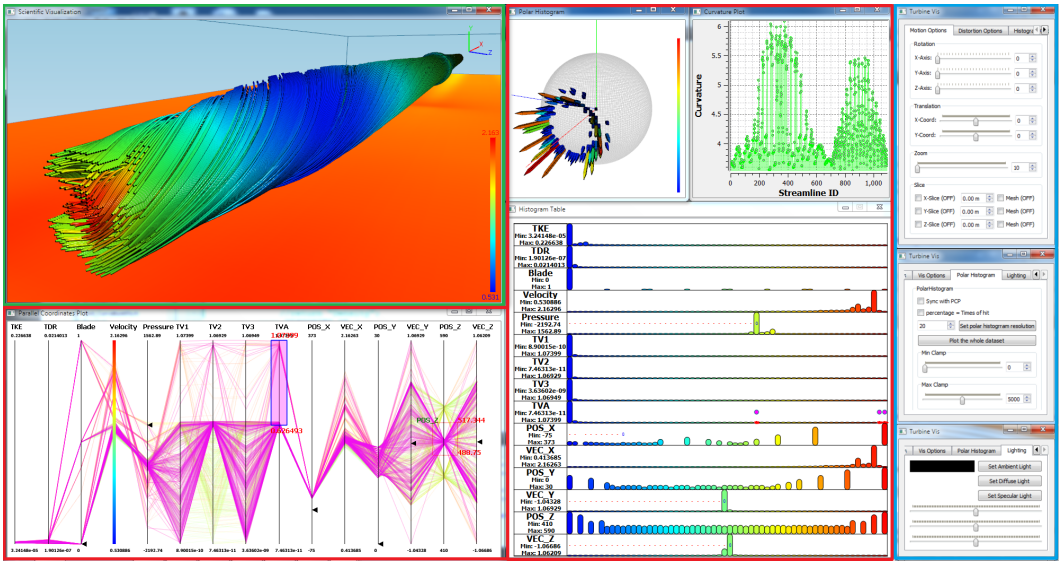


Figure 6.3: A screen shot of our flow visualization framework applied to visualize the flow past marine turbines[PGSC10]. The multi-linked application GUI consists of three distinct parts: (1) Information Visualization Windows (outlined in red) providing various data drilling widgets such as histogram table, parallel coordinate plot, etc. (2) A Scientific Visualization Window (outlined in green) rendering the final visual result in 3D based on the filtered data. (2) The tabbed Console Menu (outlined in sky blue) enables the user to update the parameters intuitively and interactively.

Our presentation here provides much more detail about the design and implementation of our software than a typical visualization research chapter. Also, typical research prototypes are unable to process real-world CFD data (like that we present here).

The rest of the chapter is organised as follows: Section 6.2 provides an overview of related research work. Section 6.3 discusses the overall design of the visualization framework while details of implementation and design of our flow visualization framework are presented in Section 6.4. Section 6.5 evaluates the design and implementation and discusses some advantages and disadvantages of the framework. Conclusions and suggestions for future work are presented in Section 6.6.

## 6.2 Related Work

In this section, we discuss some design and implementation related work. Doleisch et al. [DGH03][Dol07] describe the design and implementation of the SimVis which enables interactive visual exploration and analysis of large, time-dependent, and high-dimensional data sets resulting from CFD simulation. Weaver [Wea04] discusses the design of a multiview visualization system, Improvise, which utilises a shared-object coordination mechanism using visual abstraction language. He also presents a cross-filtered views implementation[Wea09] based on Improvise for multiple dimensional visual analysis. Laramee et al. [LHH05] describe the design and

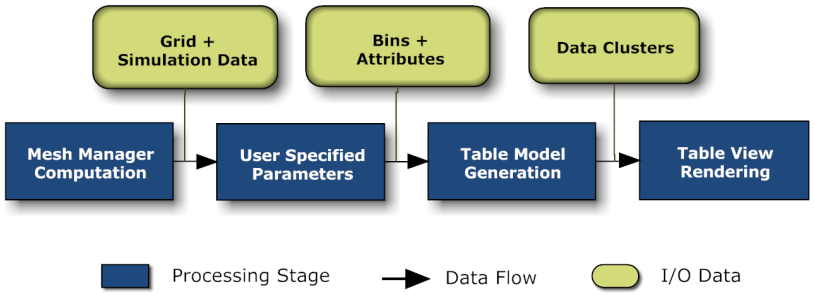


Figure 6.4: The processing pipeline of the histogram table visualization design.

implementation of a flow visualization subsystem which utilises the geometric and texture-based flow visualization techniques. In order to guarantee the quick responsiveness for user interaction even when dealing with large data, Piringer et al. [PTMB09] present a generic multi-threading architecture which allows early cancellation of the visualization thread due to user interaction without common pitfalls of multi-threading. They also present an interactive visualization toolkit, HyperMoVal [PBK10], as an implementation of this architecture in practice. Fisher et al. [FDFR10] present a framework called WebCharts by which existing information visualizations can be plugged into a variety of host applications. WebCharts is like a API of visualization library which encourages greater reuse of existing visualization in host applications, so that users can do visualization locally, yet new visualizations can be updated and obtained via the API from related websites.

## 6.3 Overview of Visualization System Design

Figure 6.2 illustrates an overview of the framework. The input is the CFD flow simulation and associated mesh. It is characterised by the high-dimensional data which includes position, velocity and various simulation attributes such as pressure, viscosity, turbulent kinetic energy, etc. Since the mesh is often unstructured and adaptive resolution, the mesh manager is used to compute and store the mesh topology information as a preprocessing step. After the mesh adjacency construction, various information visualization approaches are employed to gain insight into the data. The histogram table provides an intuitive overview of the multi-dimensional attributes of the whole simulation. After gaining an overview based on the histogram table, the user can focus on attributes they deem interesting, while the polar histogram and PCP simultaneously depict the details of the focus attributes. The polar histogram presents an intuitive visual summary of the flow velocity distribution. The PCP highlights the relationship between CFD attributes to support exploration and analysis. Several flow visualization approaches are applied to provide rich visual analysis. Interactive glyph visualization provides a quick and direct hands-on exploration on the vector field, while a continuous representation can be obtained by a geometric flow visualization such as streamlines. The automatic vector field clustering produces intuitive and insightful images of vector fields. The user can interact between different information visualization approaches to obtain the final scientific visualization result. Figure 6.3 shows our application based on this framework.

## 6.4 System Design and Implementation

In following subsections, we describe two main subsystems in more detail about design and implementation: the **Information Visualization Subsystem** and the **Scientific Visualization Subsystem**. This is where the majority of our research work was done. These two subsystems involve a fairly complex set of classes and associated responsibilities. In order to elaborate these intuitively and show the framework is really working, Standard UML diagrams [Fow03] and Doxygen diagrams [vH] generated based on a project in Chapter 5 are provided. **User Interface Design** is also presented with details at the end of this section.

### 6.4.1 Information Visualization Subsystem

Here we detail how the information visualization subsystem is designed and implemented. Each part of the subsystem is examined with details such as the class relationship, the hierarchy of class components, the collaboration between different classes.

#### 6.4.1.1 Histogram Table Visualization

In order to quickly and efficiently present a large amount of multidimensional data, it's desirable to provide a quick overview of the entire data set. For this, we incorporate a histogram table. See Figure 6.5. The histogram table represents the distribution of multidimensional information across the data set in an interactive visualization. Figure 6.4 illustrates the main processing pipeline of the histogram table visualization subsystem. The input and output data is shown in rectangles with rounded corners and processes are shown in boxes.

The mesh manager preprocesses the input CFD flow simulation data and related mesh in order to obtain the mesh topology information and the range of values for each simulation attribute.

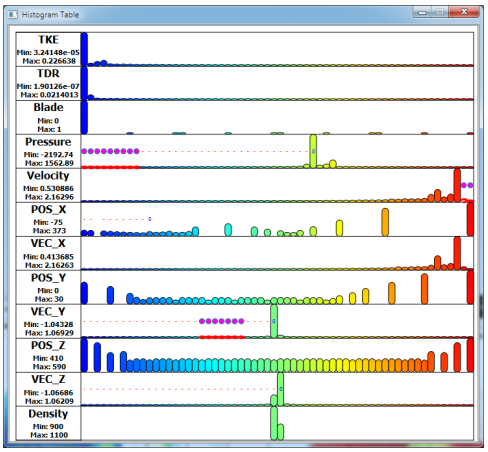


Figure 6.5: A histogram table is used to provide an overview of the multidimensional information from the turbines simulation in Chapter 5.

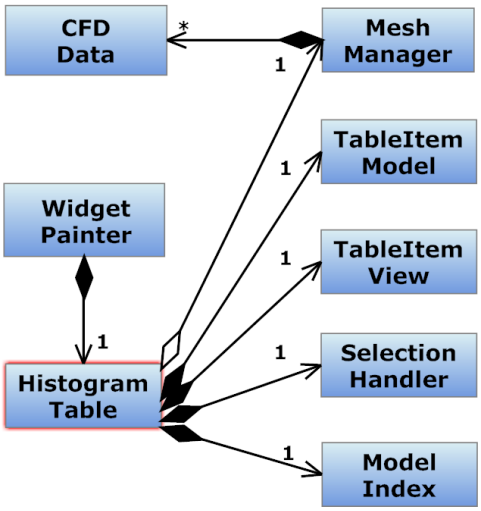


Figure 6.6: A screen shot of relationship among the major components of histogram table visualization implementation. UML notation is used.

After the user specifies their input requirements for the histogram table, such as the bin number of each histogram, the histogram height, the order of attributes and the colour-mapping parameters, a table model is generated to group the preprocessed values into clusters. Based on the table model, the table view renders a histogram table which enables the user interaction on the table model. The user is able to select the items from the histogram table for further exploration.

In order to illustrate the class relationship among the major components of the histogram table implementation, a UML class diagram [Fow03] is drawn in Figure 6.6. The black diamond shape arrow indicates the *composition* which defines a "owns a" relationship while the white

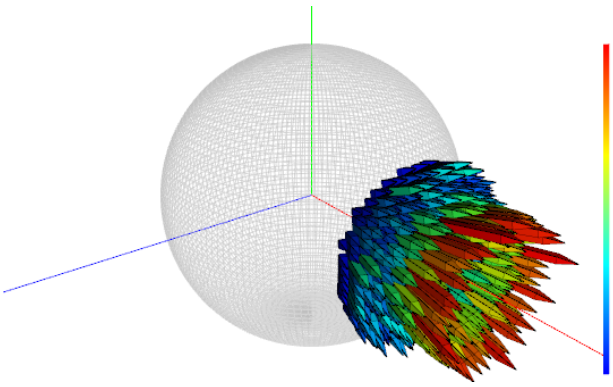

Figure 6.7: A polar histogram is used to illustrate the velocity distribution of the tidal flow around the blade element.

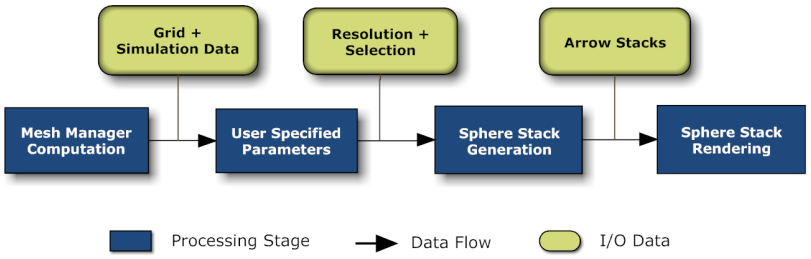


Figure 6.8: The processing pipeline of the polar histogram visualization design.

shows a "has a" *aggregation* relationship. A 1 or $*$ in the figure represents the multiplicity of the relationship. For example, the **TableItem Model** object is owned by the **Histogram Table** object and the relationship is one-to-one. Here we describe the major components shown in Figure 6.6. The **Histogram Table** is the class with the most responsibilities shown in Figure 6.4. The **Mesh Manager** class is responsible for processing and managing the raw **CFD data** from CFD flow simulation. All the information used in this subsystem is filtered and provided by the mesh manager. The **TableItem Model** object is responsible for defining a table model which contains all the grouped information and delegates access to data. The **Model index** is the index object for locating data in the table model. The **TableItem View** object is used to display data from the TableItem Model. The TableItem Model and the TableItem View are constructed based on the model/view architecture. The **Selection Handler** class is responsible for tracking and managing the user selection from the table view. The **Widget Painter** class provides the general 2D rendering mechanism for the histogram table.

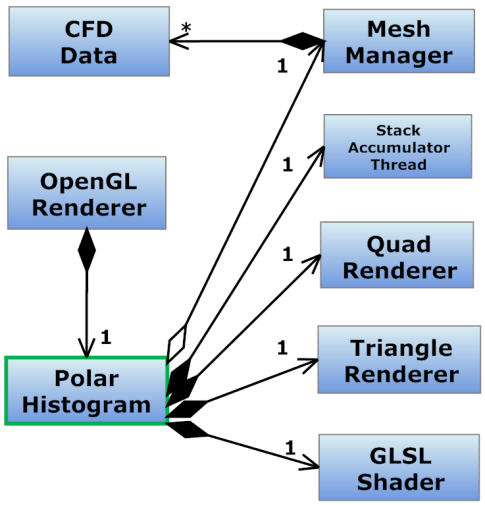


Figure 6.9: A screen shot of relationship among the major components of polar histogram visualization implementation. UML notation is used.

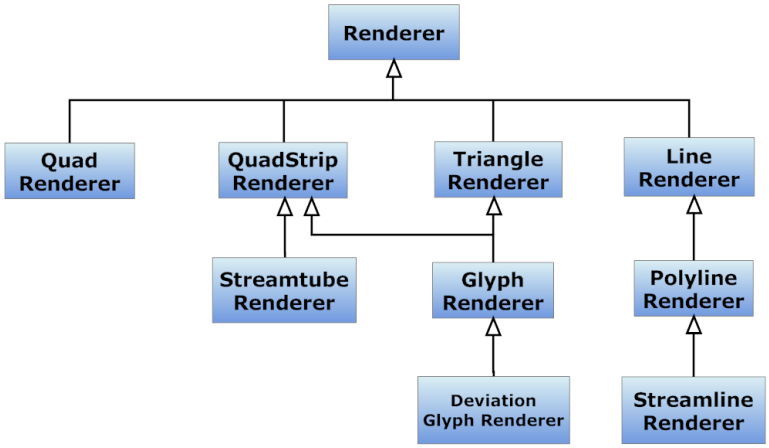


Figure 6.10: A class hierarchy of object Renderer options. UML notation is used.

#### 6.4.1.2 Polar Histogram Visualization

In order to visualize the direction in which the majority or minority of velocity of the whole domain or the user selected region points, we have a view called the polar histogram which is originally inspired from the global animal tracking visualization by Grundy et al. [GJL+09] for an integrated view of velocity distribution in a 3D spherical coordinate system. See Figure 6.7. This visualization delivers an intuitive and direction-oriented visual result of the velocity distribution. Figure 6.8 demonstrates the main design pipeline of this visualization subsystem. Similar to the first step of the histogram table, the preprocessed mesh and simulation data are obtained as the input by the mesh manager. The polar histogram is initialised as a sphere wireframe whose bin resolution is specified by the user. Each cell of the sphere wireframe depicts a range and frequency of velocity direction. Based on the user selection of the data, sphere stacks are generated on top of the corresponding sphere wireframe cells with height associated to the frequency of vectors pointing in the direction. Finally stacks are rendered with arrow tips to represent the direction.

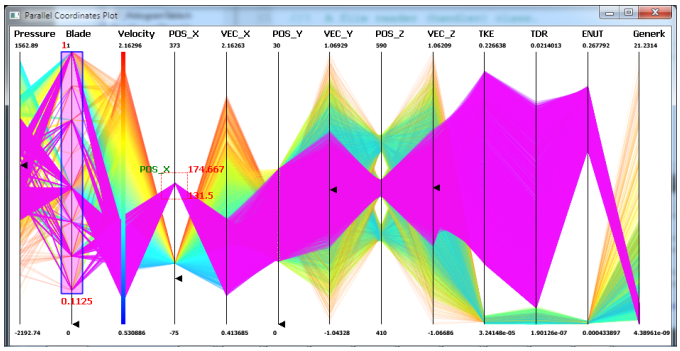


Figure 6.11: A screen shot of the parallel coordinate plot visualization. The user selection is highlighted in magenta.

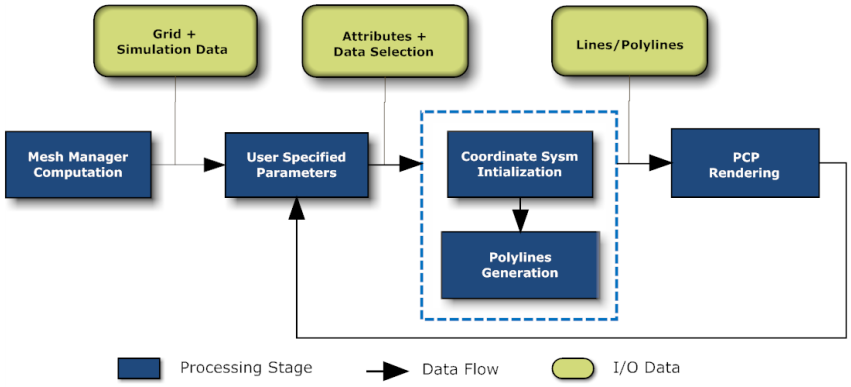


Figure 6.12: The processing pipeline of the parallel coordinate plot visualization design.

#### 6.4.1.3 Parallel Coordinate Plot

Figure 6.9 shows the class relationship between the major components of the polar histogram subsystem implementation, again using UML notation. As the core class **Polar Histogram** coordinates other classes to fulfil all design steps in Figure 6.8. A sphere wireframe with user specified bin resolution is defined to represent different ranges of velocity direction in 3D space by the **Sphere Framer** class. Based on the velocity information provided from the raw CFD data by the **Mesh Manager** class, the **Stack Accumulator Thread** class calculates the frequency of vectors pointing to each velocity direction range (cell) on the sphere wireframe. The multi-threading is applied to make this computation independent from the main working thread in order to prevent the interface from locking up. After the frequency of each cell is obtained, a stack with height associated to the frequency is defined as a cuboid with a pyramid-shaped top by the class **Stack Packer**. The **OpenGL Renderer** is responsible for general rendering of primitives such as points, lines, and polygons. Figure 6.10 shows our **Renderer** options displayed in the class hierarchy in which they were designed and implemented. The hierarchy, following UML notation [Fow03], illustrates the is-kind-of relationship between rendering classes. At the top of the hierarchy the abstract base class, **Renderer**, describes attributes and behaviours that all rendering objects have, such as colour, lighting, anti-aliasing etc. So there is no need to rewrite code for these in other rendering objects but simply inherit these features from this parent class. This type of design makes the project implementation more efficient and controllable. In this case, the sphere wireframe is rendered by the **Line Renderer** class and the pyramid-shaped stack is rendered by join of the **Triangle Renderer** class and the **Quad Renderer** class. In order to make the final rendering result more aesthetically pleasing, a **GLSL Shader** class is used to provide direct control of rendering effects on graphics hardware with a high degree of flexibility.

The correlation of simulation attributes of the CFD simulation dataset is deemed interesting and important by CFD engineers. Identification of these correlations and clusters from the multivariate data is the strength of Parallel Coordinate Plot (PCP) introduced by Al Inselberg [ID87]. A PCP subsystem is integrated to help the user further analyse and explore the

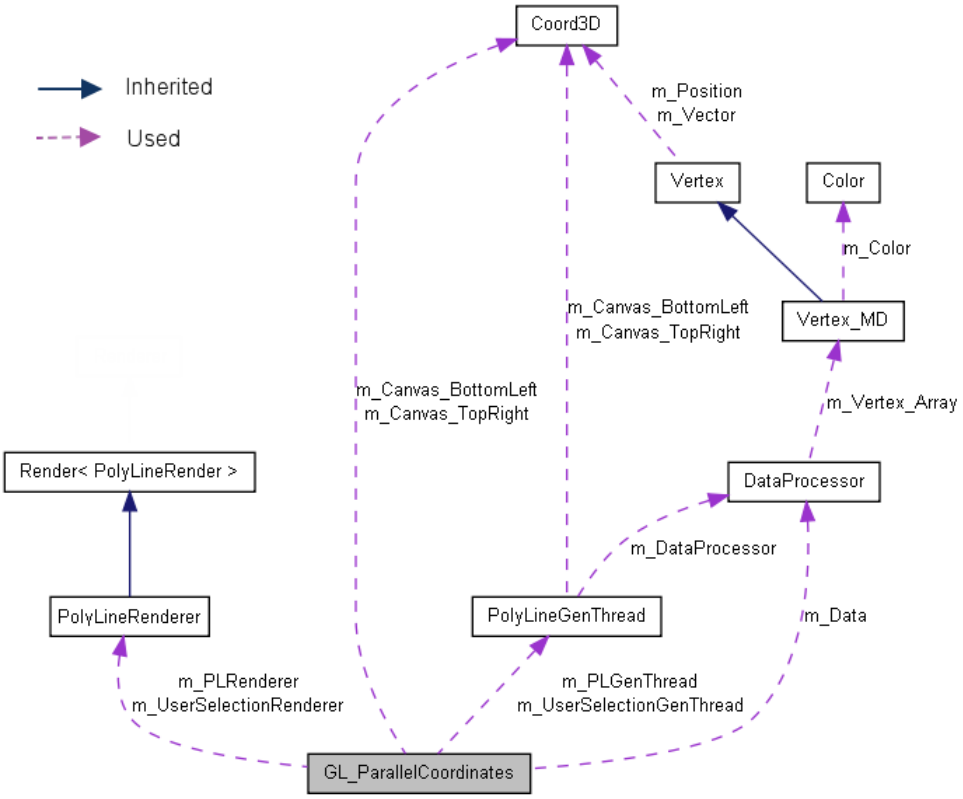


Figure 6.13: A collaboration diagram generated by Doxygen for the class **GL_ParallelCoordinates** as the core class in PCP visualization.

multivariate CFD data. (Figure 6.11) By interacting with the PCP an intuitive pattern of attribute relationship of user selection is highlighted. Figure 6.12 illustrates the design pipeline of PCP subsystem. After the user brushes a portion of the data and specifies which attributes to be investigated for the initialisation of PCP, polylines which reflect the correlation between each attribute are generated. Each axis represents the distribution of a specific attribute. The name of the attribute is labelled with minimum and maximum values.

Instead of using UML structure diagrams such as class diagrams in Figure 6.6 and 6.9, the implementation of PCP is illustrated using the UML behaviour diagram - a collaboration diagram produced from our marine turbine project by Doxygen [vH] in Figure 6.13. Doxygen is a widely used documentation system for various programming languages. It can produce a collaboration diagram to visualize the interactions between classes. In Figure 6.13 the PCP main class is presented as a filled grey box named **GL_ParallelCoordinates**. It directly employs four elements: **Coord3D**, **DataProcessor**, **PolylineGenThread**, and **PolylineRender**. The interactive collaboration between each component is detailed in following:

- **Coord3D** is a 3D coordinate class which provides methods accessing x, y, and z coordinates. Two member objects of this class, **m_Canvas_BottomLeft** and **m_Canvas_TopRight**,

are used as two bounding points, bottom left and top right, to initialise the canvas region for drawing polylines by **GL_ParallelCoordinates**.

- **DataProcessor** is a data handler class that processes the raw CFD data from data files and provides all the information required by each computation and visualization component in the system. This class is the implementation of the mesh manager in Figure 6.12. During the data processing the multidimensional information associated with each vertex can be obtained and stored using the vertex class **Vertex_MD**. **Vertex_MD** is inherited from its parent class **Vertex** which has ability to accessing position and vector information. **Vertex_MD** also maintains other simulation attributes such as pressure, mesh resolution, etc. **Color** is used to present the colour attribute of the vertex according to a given colour mapping during the visualization.

- **PolylineGenThread** is mainly responsible for generating the polylines for PCP visualization. The multi-threading technique is applied to accelerate the computation and make sure the polyline generation process does not freeze the main thread. In the implementation of **GL_ParallelCoordinates** two independent objects of **PolylineGenThread** are created to handle different computation requests. **m_PLGenThread** is the default working thread to generate the polylines based on the input data. **m_UserSelectionGenThread** is more specific. It focuses on the polylines selected by the user from those previously generated by **m_PLGenThread**. This implementation enables the user interactively select polylines deemed interesting for further exploration even the general polyline generation by **m_PLGenThread** is still processing.

- **PolyLineRender** is a renderer class which is inherited from **Render**. See Figure 6.10. This component is associated with **PolylineGenThread**. After polylines are generated and grouped by **PolylineGenThread**, **PolyLineRender** stores them into display lists for the final OpenGL rendering. **m_PLRenderer** and **m_UserSelectionRenderer** are generated for rendering different groups of polylines.

### 6.4.2 Scientific Visualization Subsystem

The scientific visualization subsystem is designed and implemented to provide the 3D spatial result of the filtered flow data from information-assisted views. Scientific techniques for flow visualization can be categorised into four groups: direct, geometric, texture-based and feature-based [PVH+03]. In our framework the direct, geometric and feature-based visualization techniques are utilized for the user to investigate the flow at different levels of abstraction. The design and implementation detail of each of these techniques is provided in following subsections.

#### 6.4.2.1 Direct Flow Visualization

Direct flow visualization is the most basic and intuitive visualization category. It directly maps the visual representation to the data samples without complex transformations or intermediate computation. Colour mapping and arrow glyphs are the most representative examples of this

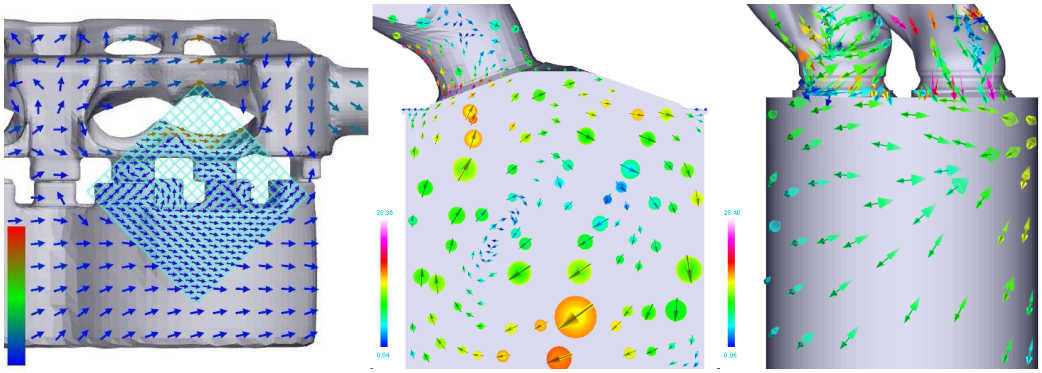

Figure 6.14: Examples show direct visualization results from some of our projects. (left) The glyph placement is directed by the user controlled resampling Cartesian grid on the geometry surface [PL08]. Each glyph is placed at the centre of each cell to directly reveal the direction and magnitude information of the vector at that position. Some range glyphs are also applied along with arrow glyphs to provide statistical information of the vector field variance [PGL+11]. (middle) The ring-shaped magnitude-range glyph is used to visualize the variation in vector field magnitude within each cluster while (right) the cone-like direction-range glyph depicts the range of direction.

category. Note that our design and implementation of direct flow visualization subsystem focuses on glyph-based flow visualization such as arrow glyphs rather than the colour coding.

The pipeline of our direct flow visualization subsystem is shown in Figure 6.15. Based on the data sample the user specifies for visualization, arrow glyphs with corresponding orientation

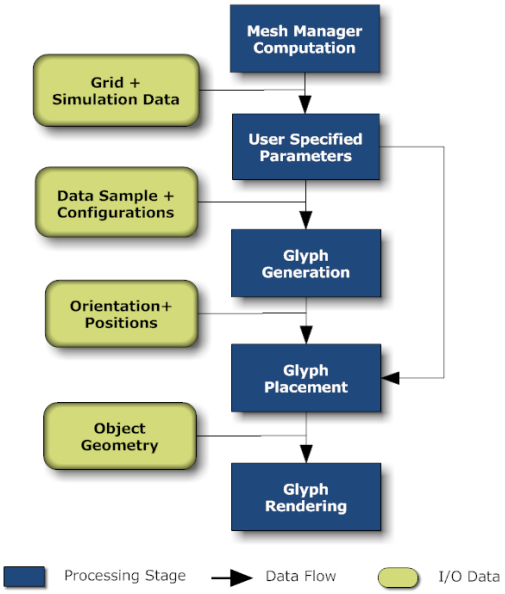


Figure 6.15: The processing pipeline for the direct flow visualization subsystem.

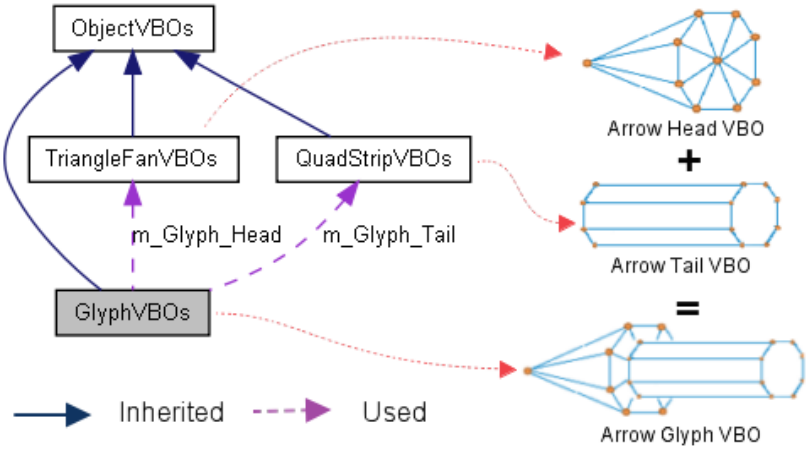


Figure 6.16: (left) A collaboration diagram for the glyph VBO class. (right, top) The triangle fan VBO class is used to generate the cone-like head of the arrow glyph by tiling triangle in the fan fashion. (right, middle) The tube-like tail of the glyph is accomplished by rolling the quad strip using the quad strip VBO class. (right, bottom) by combining the previous two VBOs the final arrow glyph VBO is obtained.

and colour are generated to depict the vector field at the seeding point. Glyphs are rendered along with the original geometry to form the final result.

Direct flow visualization is fully implemented in our projects and some example results are shown in Figure 6.14. It has been used for 2.5D (surface-based) flow visualization [PL08] [PLCZ09] [PGL+11] and 3D flow visualization [PGSC10]. It is worth pointing out that although glyph flow visualization attempts to present intuitive visualization of the vector field of the sample data quickly, the visualization can still suffer from performance issues if glyphs are naively implemented for the large dataset visualization. In order to maintain the high performance of the glyph implementation, the new OpenGL extension - Vertex Buffer Object (VBO) [Ope] is used. VBO allows vertex array data to be stored in high-performance graphics memory rather than the system memory and promotes efficient data transfer, which enables substantial performance gains and more flexibility for the dynamic glyph object implementation. We don't go through the detail of VBO here. For more information the official white paper [Ope] is recommended.

In Figure 6.16 the collaboration diagram shows how each component VBO class is used to form the arrow glyph VBO. **GlyphVBOs**, **TriangleFanVBOs**, and **QuadStripVBOs** are classes inherited from **ObjectVBOs** class which provides the basic VBO operations such as providing attribute arrays for creating VBOs, resetting attribute arrays, etc. Based on the parent class **ObjectVBOs** the **TriangleFanVBOs** class creates VBOs in the fashion of the triangle fan while the **QuadStripVBOs** class uses quad strips. The arrow head of **GlyphVBOs** is described by the **TriangleFanVBOs** class with a cone tiled by a triangle fan which consists of 10 vertices and 16 triangles by default. See Figure 6.16. And the tube shaped tail is represented by

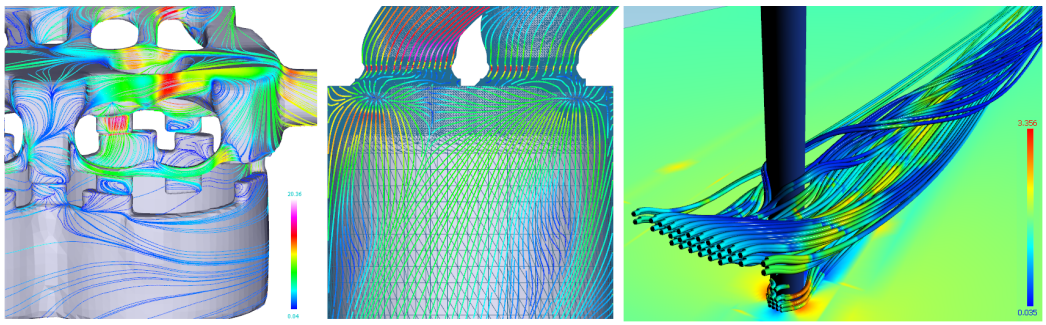

Figure 6.17: Results of Geometric flow visualization implementation in our projects. (left and middle) Colour coded streamlines are used to depict the underlying boundary flow characteristics. (middle) The evenly-spaced streamline technique [SLCZ09] [JL97] can be applied regardless of the associated mesh resolution. (right) Shaded streamlines deliver more depth percept for visualization.

the **QuadStripVBOs** with a default 8 adjacent quads, shown in Figure 6.16. Note that the resolution of **TriangleFanVBOs** and **QuadStripVBOs** can be customised by the user even the default values are provided. VBOs are also used with other visualization subsystems such as the following geometric flow visualization.

#### 6.4.2.2 Geometric Flow Visualization

Visualization of this type generates continuous geometric objects which reflect the underlying vector field. During the computation the missing velocity field between original sparse samples can be reconstructed and applied to the geometric objects by using interpolation and integration. This visualization can provide a coherent visual result of the underlying vector field. Some implementation results are shown in Figure 6.17. In what follows, we discuss the design and implementation of our geometric flow visualization subsystem. Note that it focuses on the geometric objects using integration such as streamlines and tubes rather than other geometric objects such as isosurfaces, since the complete coverage of the geometric techniques is beyond this work. See the survey by McLoughlin et al.[MLP+10] for more research results in this area.

Figure 6.18 illustrates the main processing pipeline for the geometric flow visualization subsystem. The simulation data is the input for the process. After the user specifies the data sample and the configuration attributes such as seeding requirements, the initial seeding positions are generated. Based on these seeds, the streamline is traced by the numerical integration and interpolation. The construction of the streamline is iterative and interactive: an interactive streamline adjustment step is involved at each iteration of the construction to make sure the updated user requirements can be met as soon as possible such as changing the distance between each streamline in evenly spaced streamline visualization [JL97] [SLCZ09]. After the object geometries are ready, the last step is rendering. Feedback from the rendering result helps the user to refine the configurations. The main process of the implementation step is demonstrated by the collaboration diagrams of class **StreamGenerator** and class **StreamTubeRenderThread** in Figure 6.19.

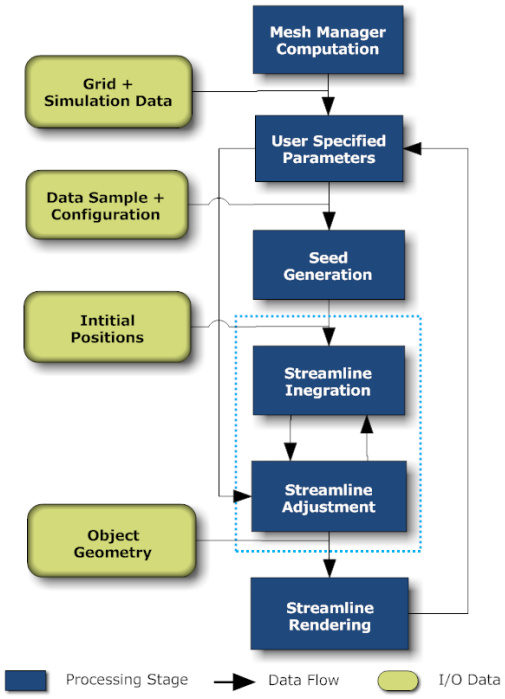


Figure 6.18: The processing pipeline for the geometric flow visualization subsystem.

The **StreamGenerator** class is designed to trace and validate all the stream seeding points and then output each streamline as an array of points. This class provides methods needed by the streamline integration and streamline adjustment stages in the design process. In Figure 6.19 (left) as the component classes **DataProcessor** and **Vertex_MD** have been explained in the PCP section, the functions provided here are very similar, so we focus on the other two classes which are the core components - **Integrator** and **Interpolator**.

- **Integrator**: This class is designed to provide static methods for various numerical integrations such as the first-order Euler integration and the second-order Runge-Kutta integration [PTVF07]. In the subsystem the streamlines are traced using the Euler integration by default with the user specified step sizes. After each streamline is generated by integration, a validation method is provide by **StreamGenerator** to check if the generated point is within the bounding box or not. If it is in the bounding box the iterative streamline tracing proceeds otherwise the integration is ended and the resulting integral paths are stored as arrays and handed over to the streamline renderer.
- **Interpolator**: This class provides various interpolation schemes in a quad or cube grid cell such as the velocity interpolation, color interpolation, mesh resolution interpolation of a point(1D), a line(2D), or a plane(3D). After each valid streamline point is obtained from the integration, the **Interpolator** class is used to interpolate its attribute values like velocity, pressure, etc. based on vertices of the cell which contains this point. By

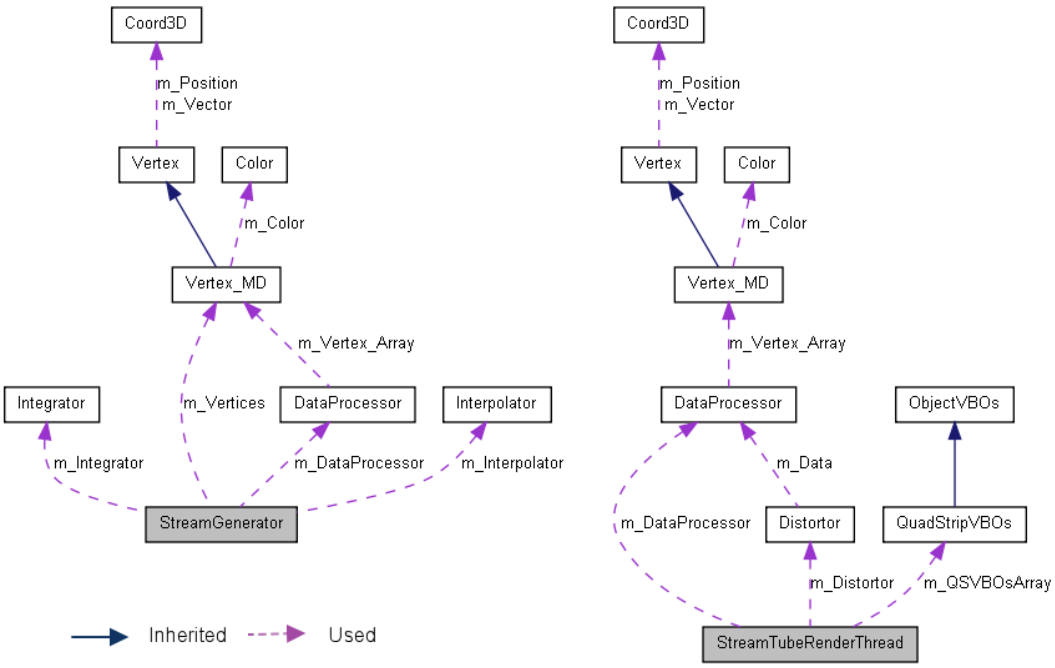


Figure 6.19: Collaboration diagrams for the class **StreamGenerator** and class **StreamTubeRenderThread** in geometric flow visualization subsystem. Diagrams are generated by Doxygen.

using the interpolation, vector field information needed for streamline integration can be reconstructed from the original discrete data samples.

Once streamlines are generated they are ready for rendering. In our subsystem we have two streamline rendering mechanisms: the colour mapped 2D and 3D streamline rendering and the shaded 3D rendering. The former is simpler and faster but tends to suffer from visual clutter while the later is a bit more complex and slower but delivers the better depth perception of streamlines. Here we look at the shaded streamline rendering class **StreamTubeRenderThread**.

The **StreamTubeRenderThread** class is mainly responsible for extending the streamlines generated from **StreamGenerator** to shaded 3D tube-shaped streamlines for rendering. This class employs a multi-threading technique since the tube construction and rendering can be expensive enough to block the main thread. The collaboration diagram is shown in Figure 6.19 (right). To construct the shaded streamlines, **QuadStripVBOs** is used to generate a tube segment with default 8 adjacent quads, like the arrow tail VBO in Figure 6.16 (right, middle), around each streamline segment. See Figure 6.20. The resolution of the quads can be updated by the user. This affects the smoothness of the streamline and the tube construction performance as well. The more quads used the smoother the tube geometry but the slower the tube construction. The **Distortor** class is used to transfer the position of seeding points according to the mesh resolution so that more visual detail in regions of interest can be provided by the

distorted streamlines without losing visualization coherency [PGSC10].

Note that the streamline generation step and the streamline rendering step are separate. The advantage to this type of design and implementation is once streamlines are generated they can be reused again and again for different rendering options if there is no new seeding requirement from the user.

#### 6.4.2.3 Feature-based Flow Visualization

Rather than directly showing the underlying vector field, feature-based flow visualization is employed to provide a 'filtered' visual result by extracting or highlighting the meaningful patterns such as vortices and saddles from the simulation data sets. Those extracted parts deemed interesting by users are defined as the *feature*. For more literature we refer the interested reader to in-depth surveys of feature-based flow visualization by Post et al. [PVH+02] and Laramee et al. [LHZP07]. In our framework we applied an image-based vector field clustering approach (IBVFC) presented by Peng et al. [PGL+11] as a feature extractor and highlighter for the flow visualization. By doing a quick and automatic hierarchical clustering of the vector field according to the user specified errors, this feature-based visualization approach is able to simplify the result with dense visualization on regions with the most suggestive and important information and remain sparse on the less important ones. An example result is shown in Figure 6.21. Note that in this section we only focus on this approach closely rather than other feature-based visualization techniques.

Figure 6.22 demonstrates the design pipeline of our feature-based flow visualization subsystem. In brief, our feature-based flow visualization subsystem simplifies the problem of clus-

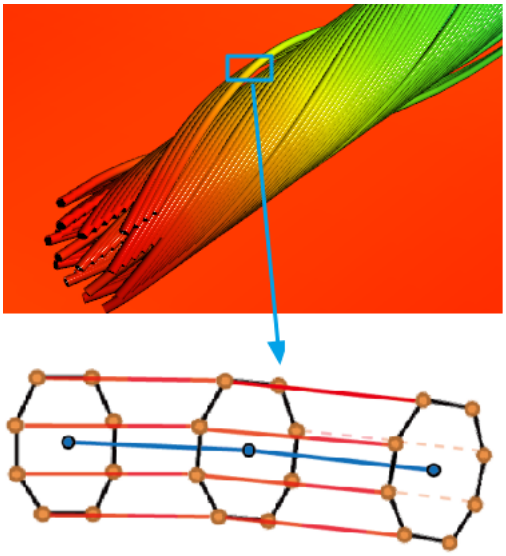

Figure 6.20: This figure demonstrates how the shaded streamline is generated by a collection of quad strips.

Figure 6.21: A screen shot of the visual result comparison between the direct flow visualization (left) and the feature-based visualization (right). (left) The cluttered direct flow visualization can't do a good job of extracting important flow features. (middle) Vector field clusters are generated and coloured in different colours. (right) Based on the generated clusters, shaded streamlets with arrow head are applied to reveal some flow features such as vortices and saddle points which are highlighted by red circles.

tering vector fields on surfaces by confining the clustering process to image space. After the projection to image space, clusters are only generated for visible regions of vector fields on the surfaces. Then various visualization approaches are applied to reflect the vector fields based on those clusters. The main procedures of our IBVFC visualization subsystem are shown on the right part of Figure 6.22: (1) project the vector field to the image plane so that attribute images are obtained, (2) detect geometric edge discontinuities based on the depth image, (3) process a bottom-up hierarchical clustering based on attribute images using a distance-metric, (4) overlay specified visual representations of the original surface geometry such as a semi-transparent representation of the surface with shading, along with (5) the image overlay glyph

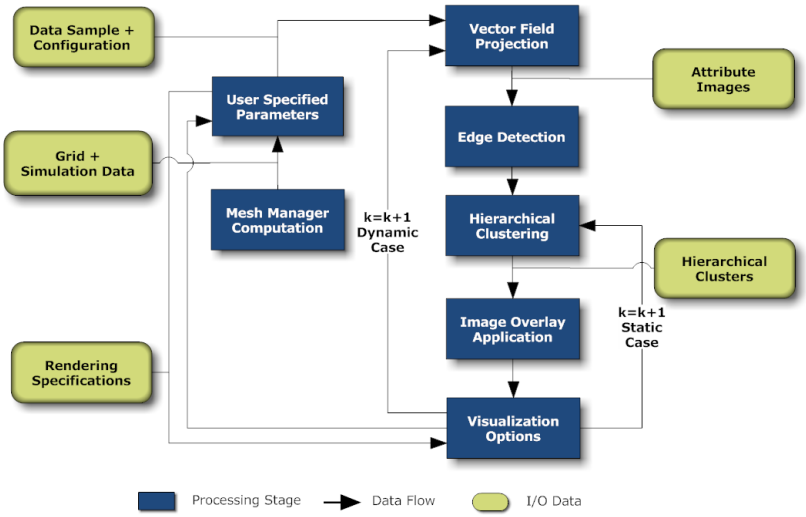


Figure 6.22: The processing pipeline for the feature-based flow visualization subsystem.

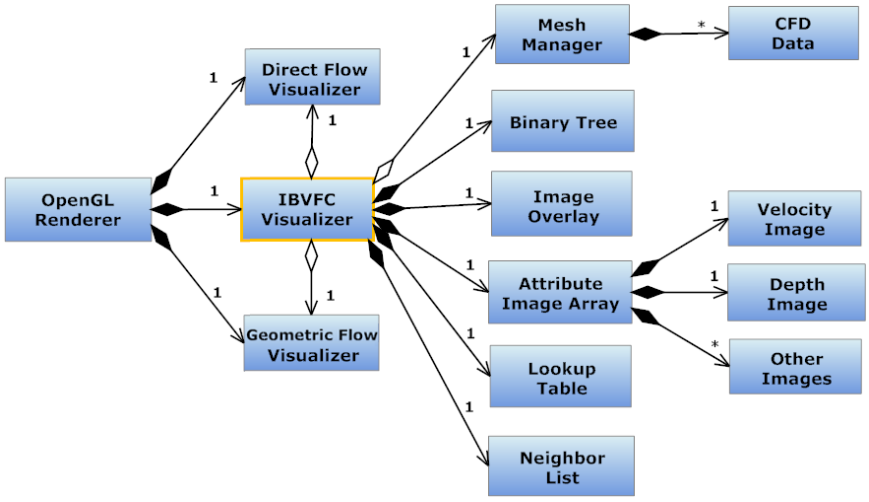


Figure 6.23: The relationship among the major components of the feature based visualization implementation. UML notation is used.

or/and streamline visualization which are applied automatically based on the user-defined error metric and rendering specifications. Additionally, various enhancements and user options, like distance-metric weighting coefficients, can be used to customise the clustering process and thus the final visualization result. It's also worth mentioning that if the viewpoint is changed, such as cases of geometry rotation, translation, and scaling, steps 1-5 of the pipeline are necessary for the next pass, and only a subset (steps 4-7) of the algorithm is required for the static cases if the clustering distance-metric parameters are changed without changing the view-point. Each stage is described in more detail in previous research [PGL+11].

Figure 6.23 outlines the class relationship between the major components of the IBVFC visualization subsystem implementation using UML composition notation. The **IBVFC Visualizer** class is the core class which coordinates each component in Figure 6.22. The **Mesh Manager** class is responsible for handling CFD data sets and providing access to the data information. The **Attribute Image Array** class is used to store and manage the projected attribute images which simplifies the computation from 3D to 2D. This array is the data source for the IBVFC. A **Velocity Image** holds all the projected vector field information which is used to produce the clusters. The **Depth Image** stores the depth map of the projected visible portion. It is used for edge detection. Other image classes such as the **Mesh Resolution Image** in [PGL+11] can be used to add extra parameters associated with the error metric which drives the clustering process. The **Binary Tree** class is a storage class which is used to implement the hierarchical cluster structure. Each node of the binary tree represents a cluster whose error metric reflects its vector field characteristics. The storage classes, the **Lookup Table** and the **Neighbour List**, are used to accelerate the cluster searching and merging process. The **Image Overlay** class stores perceptual information such as the shading and the depth map for the final rendering. It is worthy of note that the **IBVFC Visualizer** has a "is-part-of" relationship with the **Direct Flow Visualizer** and the**Geometric Flow Visualizer** in Figure 6.23. This means

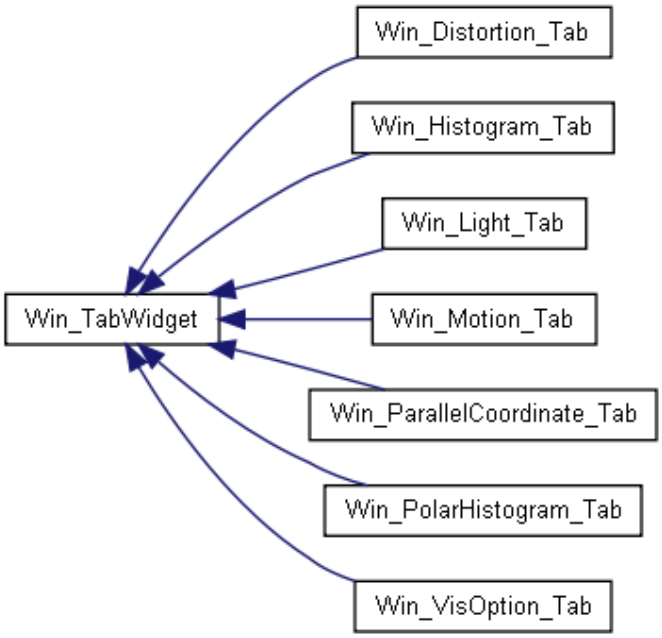


Figure 6.24: A Doxygen inheritance diagram for the console menu tab window.

various visualizations are plug-ins for IBVFC to reveal vector field clusters with different presentations. This design and implementation makes the system easier to extend and maintain. If there are new visualization approaches available for IBVFC we can simply plug them in. At last **OpenGl Renderer** wraps up all the rendering processes.

### 6.4.3 User Interface Design and Implementation

Design and implementation of a friendly and effective user interface (UI) is an essential part of our flow visualization framework. How to design a good UI is a classic topic in the field of human-computer interaction (HCI) community. A literature survey dedicated to the subject is beyond the scope of this work and we refer the interested readers to some influential work by Shneiderman [Shn92], Torres [Tor02], and Cooper and Reimann [CR03]. In the following content we discuss the design and implementation of important components of our framework UI: the graphical user interface (GUI), the event handler, and the multi-threading platform. Note that we use QT library [Nok] to implement each above UI component since the use of QT ensures the platform independence of the user interface. So related QT classes are also mentioned in the following subsections.

**The Outer Body - Graphical User Interface (GUI)** The GUI serves at the frontline of the user interaction. In our multi-linked flow visualization framework shown in Figure 6.3, each subwindow works as a GUI. Firstly, the tabbed console menu window (boxed in sky blue in Figure 6.3) is used. The tab widget not only groups the interactions for the same or similar task but also saves screen space by compacting tabs in the same window. Each tab widget of the console menu is inherited from the class **Win_TabWidget** which provides basic QT graph-

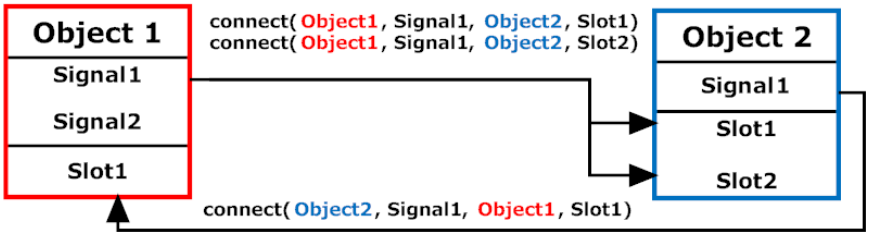


Figure 6.25: A diagram shows how the QT signal-slot callback mechanism works.

ical control elements including check-boxes, sliders, drop-down lists, spinners and buttons. See Figure 6.24. The user can specify parameters to select or update analysis or visualization options by interacting with control elements in the corresponding tab widget. For example if we want to turn on the slice function along $x$ axis and specify the slicing position, we go to the 'Motion Options' tab then tick the check-box 'X-Slice' and update the position value by scrolling the spinner next to the check-box. Additionally, each visualization window is a GUI which supports mouse-event driven user interactions. Information visualizations such as histogram table and PCP allow the user to mouse brush or select the data deemed interesting while the scientific visualization window provides the hands-on interaction for the rendering result such as translation, rotation, and zooming.

**The Brain - Event Handler** After the user input is obtained from the GUI, the class **ControlPanel** is applied as an asynchronous callback centre which handles all the corresponding events from GUI such as key presses, mouse movement, action selections, and timers and then triggers the related functions or computations. In order to implement this functionality to the **ControlPanel** class, the Signal-Slot callback mechanism from QT is used. The concept of Signal-Slot is that objects can send particular signals containing event information which can be received by other objects using special functions known as slots. The Signal-Slot mechanism is contained in all classes which are inherited from the class **QObject**. This means the mechanism can be easily accessed and used in almost every class. One more advantage about the Signal-Slot mechanism is that we can connect as many signals as we want to a single slot, and a signal can be connected to as many slots as we need. See Figure 6.25. This flexibility enables the **ControlPanel** coordinating the communication between multi-linked views more straightforward and efficient.

**The Nervous System - Multi-threading Platform** Since the size of the simulation datasets tends to be non-trivial, maintaining the smooth user interaction among the GUI and rendering a large number of selected objects simultaneously in visualization views is a challenge. In order to address this and deliver a smooth and efficient user interaction, a multi-threading platform inspired by Piringer et al. [PTMB09] is used. **QThread** class provided by QT is inherited by each visualization class so that each visualization view has its own working thread which is independent from the main working thread. Based on this, users can interact with the visualization result even as it is still being generated in the background.

## 6.5 Discussion and Evaluation

After presenting the design and implementation of our flow visualization framework, we now discuss the advantages and disadvantages of implementing each subsystem into the framework and evaluate the framework against the benefits specified in Section 6.1.

**Support for Versatile, Real-world, High-dimensional CFD Data sets** Our framework implements a mesh manager which handles a large, varied collection of real-world CFD data sets: from small geometries to large geometries, from surface-based flow to volume-based flow, from vector field information to multidimensional data, from the automotive simulation to the marine turbine simulation data. The mesh manager is able to convert the data from an ASCII file to a binary file to accelerate the loading process of a large data set. However, the mesh manager has its disadvantages: it has not been separated from the main working thread. So when a large data set is being loaded the user interaction is interrupted until the data loading is finished.

**Support for a Wide Range of Visualization Techniques** A big advantage of implementing different visualization subsystems into the framework is the ability to provide various visualization options. For information visualization we implement the histogram table, the polar histogram and the parallel coordinate plot while the direct, geometric, and feature-based flow visualization techniques are employed for scientific visualization use. It is helpful to provide CFD engineers with a wide range of options since each visualization technique has its own advantages and disadvantages. The usability of this framework is approved by CFD engineers based on their domain expert reviews [PGL+11] [PGSC10]. It's also worthy of note that it is unusual to have this many visualization options integrated to a research prototype. Naturally there are also disadvantages to integrating various visualization subsystem into a framework. Firstly, our all-in-one framework involves a lot of classes, although an object-oriented design pattern is adopted to reduce the complexity and the time spent on learning and understanding the process pipeline. Secondly, large numbers of data transferred simultaneously between subsystems may freeze the user interaction. So the way to combine various visualizations needs to be more efficient, stable and robust.

**Interactivity** Based on our multi-threading platform, each visualization subsystem is able to deliver smooth and efficient user interactions even as the visualization result is still being generated in the background. However, multi-threading adds more complexity to the coding and debugging stages.

**Platform Independence** Platform independence is achieved through the use of the platform independent libraries - QT and OpenGL. QT is an open source cross-platform application framework which not only provides the platform independent GUI library but also supports OpenGL which a widely supported, platform independent graphics library. The downside of QT in this implementation is all OpenGL related. Firstly, the QT OpenGL API is a bit different from the original OpenGL API, which may involve some learning curve. Secondly, the QT OpenGL library could conflict with other OpenGL libraries.

## 6.6 Conclusion

We describe the design and implementation of a generic framework which incorporates information and scientific visualization approaches to provide effective visual analysis of CFD flow simulation data. Direct, geometric, and feature-based flow visualization approaches are integrated with information visualization techniques such as PCP and histogram table to provide an information-assisted visual analysis framework for CFD flow visualization. In contrast to most research prototypes, the system we present handles large, real-world, high-dimensional data. We discuss the design and implementation of visualization subsystems of the frame work and the user interface. We also evaluate the implementation of this framework by discussing the advantages and disadvantages against the goal of our framework. UML and Doxyden diagrams are generated to illustrate the technical content of each component. At last, we believe the outlined principles in this work can be applied in a more general way to other similar projects which handles high dimensional datasets.

*Chapter 7*

# Conclusion and Future Work

**Contents**



> "Never let the future disturb you. You will meet it, if you have to, with the same weapons of reason which today arm you against the present."
>
> - Marcus Aurelius [10]

## 7.1 Conclusions

In this thesis, we focused on the interactive visualization of high dimensional flow CFD datasets, and presented a number of new methods and algorithms to address related challenges in flow visualisation domain. We strongly believe that the interaction is the crucial part of an effective data analysis and visualization on large and complex datasets such as CFD data we used in this thesis. Our interactive flow visualization techniques have been applied on real-world CFD datasets at different abstraction levels: from low abstraction level presentation such as glyph-based flow visualization in Chapter 3 to high abstraction level such as feature-based flow visualization in Chapter 4 and 5. Our projects are also covering different industry areas: from automotive industry (Chapter 3 and 4) to renewable energy industry (Chapter 5).

To conclude this thesis we again reemphasise the main benefits and contributions of this thesis:

[10] Marcus Aurelius (26 April 121 - 17 March 180), was Roman Emperor from 161 to 180. He ruled with Lucius Verus as co-emperor from 161 until Verus' death in 169. He was the last of the "Five Good Emperors", and is also considered one of the most important Stoic philosophers.

- In Chapter 2, we provide a state-of-the-art survey of the most recent developments in steady and unsteady flow visualization in higher spatial dimensions (2.5D and 3D). It is also the first survey which covers vector-field clustering approaches for feature-base flow visualization. This survey serves as a foundation for our flow visualization research as most of the ideas for our projects in this thesis were inspired by it.

- Chapter 3 describes a fast and simple glyph placement algorithm for investigating and visualizing the boundary flow based on unstructured, adaptive resolution meshes from CFD datasets. The algorithm automatically places glyphs at evenly-spaced intervals, independent of geometric and topological complexity of the underlying meshes. Due to the high efficiency of the algorithm real-time user interactions such as zooming, translating and rotation are enabled.

- In Chapter 4, we propose a novel, automatic mesh-driven vector field clustering algorithm which couples the properties of the vector field and resolution of underlying mesh into a unified distance measure for producing intuitive and suggestive visualization of the vector field on large, unstructured, adaptive resolution boundary CFD meshes. The cluster hierarchy is generated automatically according to the importance of the underlying mesh and the properties of the vector fields to emphasis vector fields in the important regions deemed by the user.

- In order to help the CFD engineers to investigate the relationship between each dimension from high dimensional marine turbine CFD datasets, we present a novel application which exploits multiple-coordinated views for interactive information-assisted visualization of marine turbine simulation data in Chapter 5. By incorporating various information visualization techniques a statistic overview of the data can be firstly obtained. Based on the importance found in information visualization views flow visualization techniques can be applied only to regions deemed important by the user. The information-assisted views of the data have helped domain experts discover new properties of their data which they were previously unaware of.

- Chapter 6 describes the design and implementation of generic flow visualization framework which integrates the flow visualization and the information visualization. The framework is platform independent in terms of hardware and software. The API of this framework is also provided.

It's also worthy of note that our research of interactive flow visualization not only focuses on the algorithms and applications but also involves a lot of interactive collaborations. We had close collaboration with CFD experts across our research work from automotive industry to marine turbine design. Their domain expert reviews validated the usefulness of our applications and also inspired some new features for flow visualization applications. Without their valuable feedback and suggestions our search might lose the focus and would not be this successful.

## 7.2 Future Work

Towards the end of my PhD, we built a multi-disciplinary framework for interactive flow visualization at different abstraction levels. The traditional scientific visualization at the lowest level of abstraction was combined with information visualization techniques at higher abstraction levels to deliver effective and interactive exploration of large and high dimensional CFD datasets. This multi-linked exploration and presentation technique is deemed very useful and important because the size, dimensionality and complexity of CFD data have been dramatically increased and hence a mix of visual representations is required. However, finding the right combination of two sides, and adapting techniques from one field to work with others, is one of the challenges we have to confront. Now some possible future research is discussed as follows:

**Effective interaction through linked views** Interaction plays a very important role in visual exploration of large or highly complex CFD data through linked views. Effective interaction not only bonds multi-linked views to form a suggestive representation of complex data at different abstraction levels but also empowers our visual system to focus on the objects of interest within the context of the general structure. In Chapter 5 engineers interactively query the large and complex marine turbine dataset by selecting the regions they deem interesting from the linked information visualization views. This seems more desirable and necessary when the size, complexity and dimensionality of CFD data are getting overwhelming nowadays. So developing effective and rapid interaction between linked views seems to be a way the field is progressing.

In order to achieve more effective interaction, there are several techniques worthy of future work. The first one is the clustering technique. Many recent publications on this technique in flow visualization, such as [WWYM10] [XCML11] [KLG+11], prove its important role in simplifying the complex flow fields. So a potential future work in this domain is that rather than having clustering on the original flow data in 3D space like what we did for our vector field clustering method in Chapter 4, we could apply various clustering techniques to the linked views which has higher abstraction level but lower dimensionality, such as the scatter plot and PCP, so that the computational complexity of clustering process can be reduced to lower dimensional space. This procedure would produce the clusters more quickly and more representatively. Hence efficient interaction can be obtained on those filtered subsets rather than the original large and complex dataset. Another potential direction for future work on linked views is machine learning. Within the exploration process there could be often many repetitive steps for the user to perform, such as looking for a specific pattern or value in a linked view. If the computer can be trained about how the user selects these patterns in a few examples, it can perform an automatic pre-selection of the data for the user so that users can be free from these repetitive jobs but only focus on more essential and unique tasks. In recent work, Ma [Ma07] presented a machine learning assisted volume visualization technique. It's possible to extend this work to our framework in future to enhance the interaction. One more possible extension of our interactive link view technique is to make it web-based available. Since the stable version of WebGL (Web-based Graphics Library) [Gro11], a web-based version of OpenGL, was

released on 3rd March 2011, lots of research effort has been dedicated to explore the power of this new web-based graphics engine [Goo]. Reasons why taking this direction for our future work are: (1) the modern web technology provides much more intuitive, mature and versatile user interaction support. When the WebGL is applied, our linked view flow visualization technique inherits this nice feature to enhance the user interaction; (2) What the client side needs to make the application work is a web browser which supports WebGL. And the fact is that nowadays almost all mainstream web browsers support WebGL, which means headache of deploying the flow visualization application across different operation systems might be gone.

**Close collaboration with domain experts** Another aspect of our future work is to keep collaborating with our 'customers', domain experts, closely. The ultimate goal of CFD flow visualization is to help CFD experts explore the complex and large CFD data more intuitively and effectively. This means the collaboration with CFD experts is really an important aspect of the flow visualization research because they can make sure what we research and develop is what they really want or need. On the other hand they can also inspire some new ideas for the visualization development during the collaboration. However, This important dual-way communication is often overlooked by our flow visualization community. The fact seems that quite lot of research work on CFD flow visualization is aimed at our own flow visualization people rather than the end users, the CFD engineers. Without the validation and evaluation from them how come our research results are proclaimed valuable or useful? In order to avoid this happened to our research, we had very close collaborations with CFD domain experts crossing the automotive industry to the marine turbine design in our research. These collaborations led to successful developments as we can see in previous chapters. This again proves the importance of the collaboration with domain experts. In the future I may work in a different domain as a visualization researcher, but I believe that I will take and apply what I learnt from the collaborations with CFD engineers. Also I hope collaborations between our flow visualization people and the CFD engineers could continue and grow stronger simply because that is what makes the visualization powerful.

# Bibliography


[ABCO+01] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point Set Surfaces. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 21–28, Washington, DC, USA, 2001. IEEE Computer Society.

[And95] J.D. Anderson. *Computational Fluid Dynamics: the Basics with Applications*. McGraw-Hill series in aeronautical and aerospace engineering. McGraw-Hill, 1995.

[BBC91] M. Beard, B. Buttenfield, and S. Clapham. NCGIA Research Initiative 7: Visualization of Spatial Data Quality. Technical paper 91-26, National Center for Geographic Information and Analysis, October 1991.

[BBP10] Stef Busking, Charl P. Botha, and Frits H. Post. Dynamic Multi-View Exploration of Shape Spaces. *Computer Graphics Forum*, 29(3):973–982, 2010.

[BHR+94] M. Brill, H. Hagen, H.-C. Rodrian, W. Djatschin, and S. V. Klimenko. Streamball Techniques for Flow Visualization. In *Proceedings IEEE Visualization '94*, pages 225–231, October 1994.

[BLM95] B. G. Becker, D. A. Lane, and N. L. Max. Unsteady Flow Volumes. In *Proceedings IEEE Visualization '95*, 1995.

[BMI+07] Raphael Bürger, Philipp Muigg, Martin Ilcik, Helmut Doleisch, and Helwig Hauser. Integrating Local Feature Detectors in the Interactive Visual Analysis of Flow Simulation Data. In A. Ynnerman K. Museth, T. Möller, editor, *Proceedings of Eurographics/ IEEE-VGTC Symposium on Visualization 2007*, pages 171–178, 2007.

[BMS+10] Thomas Butkiewicz, Ross K. Meentemeyer, Douglas A. Shoemaker, Remco Chang, Zachary Wartell, and William Ribarsky. Alleviating the Modifiable Areal Unit Problem within Probe-Based Geospatial Analyses. *Computer Graphics Forum*, 29(3), June 2010.

[BP02] L. Barreira and Y.B. Pesin. *Lyapunov Exponents and Smooth Ergodic Theory*. University Lecture Series. American Mathematical Society, 2002.

[BSH97] H. Battke, D. Stalling, and H.C. Hege. Fast Line Integral Convolution for Arbitrary Surfaces in 3D. In *Visualization and Mathematics*, pages 181–195. Springer-Verlag, 1997.

[BWEAB07] British Wind Energy Association (BWEA). Bwea response to john hutton's speech. http://www.bwea.com/media/news/070918.html, September 2007. Last access data: 01/12/2009.

[CCCF96] M. Sheelagh T. Carpendale, T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion Viewing Techniques for 3-Dimensional Data. In *INFOVIS '96: Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96)*, page 46, Washington, DC, USA, 1996. IEEE Computer Society.

[CCF97] M. Sheelagh T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. Extending Distortion Viewing from 2d to 3d. *IEEE Comput. Graph. Appl.*, 17(4):42–51, 1997.

[CCK07] Y. Chen, J. Cohen, and J. Krolik. Similarity-Guided Streamline Placement with Error Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1448–1455, 2007.

[Cd80] S. D. Conte and C. de Boor. *Elementary Numerical Analysis*. McGraw-Hill, New York, 1980.

[CH58] J. Cahn and J. Hilliard. *Free Energy of a Non-Uniform System I.Interfacial Free Energy*, volume 28. J.Chemistry and Physics., 1958.

[CH02] N. A. Carr and J. C. Hart. Meshed Atlases for Real-Time Procedural Solid Texturing. *ACM Transactions on Graphics*, 21(2):106–131, April 2002.

[CL93] B. Cabral and L. C. Leedom. Imaging Vector Fields Using Line Integral Convolution. In *Poceedings of ACM SIGGRAPH 1993*, Annual Conference Series, pages 263–272, 1993.

[CLZ07] G. Chen, R. S. Laramee, and E. Zhang. Advanced Visualization of Engine Simulation Data Using Texture Synthesis and Topological Analysis. In *NAFEMS World Congress Conference Proceedings*. NAFEMS–The International Association for the Engineering Analysis Community, May 22–25 2007. (full proceedings on CDROM).

[CM92] R. Crawfis and N. Max. Direct Volume Visualization of Three-Dimensional Vector Fields. In *VVS '92: Proceedings of the 1992 workshop on Volume visualization*, pages 55–60, New York, NY, USA, 1992. ACM.

[CML+07] G. Chen, K. Mischaikow, R. S. Laramee, P. Pilarczyk, and E. Zhang. Vector Field Editing and Periodic Orbit Extraction Using Morse Decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):769–785, jul–aug 2007.

[CMLZ08] G. Chen, K. Mischaikow, R. S. Laramee, and E. Zhang. Efficient Morse Decompositions of Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):848–862, 2008.

[CR03] A. Cooper and R. M. Reimann. *About Face 2.0: The Essentials of User Interface Design*. John Wiley & Sons, 2003.

[DGH03] H. Doleisch, M. Gasser, and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Data Visualization, Proceedings of the 5th Joint IEEE TCVG–EUROGRAPHICS Symposium on Visualization (VisSym 2003)*, pages 239–248, May 2003.

[DMG+04] H. Doleisch, M. Mayer, M. Gasser, R. Wanker, and H. Hauser. Case Study: Visual Analysis of Complex, Time-Dependent Simulation Results of a Diesel Exhaust System. In *Data Visualization, Proceedings of the 6th Joint IEEE TCVG–EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 91–96, May 2004.

[Dol07] Helmut Doleisch. Simvis: Interactive Visual Analysis of Large and Time-Dependent 3D Simulation Data. In *WSC '07: Proceedings of the 39th Conference on Winter Simulation*, pages 712–720, Piscataway, NJ, USA, 2007. IEEE Press.

[Dov95] D. Dovey. Vector Plots for Irregular Grids. In *IEEE Visualization '95*, pages 248–253, 1995.

[DW04] Qiang Du and Xiaoqiang Wang. Centroidal Voronoi Tessellation Based Algorithms for Vector Fields Visualization and Segmentation. In *Proceedings IEEE Visualization '04*, pages 43–50, Washington, DC, USA, 2004. IEEE Computer Society.

[FC95] L. K. Forssell and S. D. Cohen. Using Line Integral Convolution for Flow Visualization: Curvilinear Grids, Variable-Speed Animation, and Unsteady Flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, June 1995.

[FDFR10] D. Fisher, S.M. Drucker, R. Fernandez, and S. Ruble. Visualizations Everywhere: A Multiplatform Infrastructure for Linked Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1157 –1163, 2010.

[FG98] A. L. Fuhrmann and M. E. Gröller. Real-Time Techniques for 3D Flow Visualization. In *Proceedings IEEE Visualization '98*, pages 305–312. IEEE, 1998.

[Fow03] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Object Technology Series. Addison-Wesley, third edition, September 2003.

[Gel01] A. V. Gelder. Stream Surface Generation for Fluid Flow Solutions on Curvilinear Grids. In *Proceedings of the Joint Eurographics - IEEE TCVG Symposium on Visualization (VisSym '01)*, pages 95–106, May 2001.

[GEO02] J. Grant, G. Erlebacher, and J. O'Brien. Case Study: Visualizing Ocean Flow Vertical Motions Using Lagrangian-Eulerian Time Surfaces. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 529–532, Washington, DC, USA, 2002. IEEE Computer Society.

[Ger67] Nicolaus Germanus. Map of the world, 1467. [Online@`http://http://en.wikipedia.org/wiki/File:Ptolemy_Cosmographia_1467_-_world_map.jpg#filehistory`; accessed 21-July-2011].

[GGTH07] Christoph Garth, Florian Gerhardt, Xavier Tricoche, and Hagen Hans. Efficient Computation and Visualization of Coherent Structures in Fluid Flow Applications. *IEEE Transactions on Visualization and Computer Graphics*, 13:1464–1471, November 2007.

[GJ10] Christoph Garth and Ken Joy. Fast, Memory-Efficient Cell Location in Unstructured Grids for Visualization. *IEEE Transactions on Computer Graphics and Visualization*, 16(6):1541–1550, November 2010.

[GJL+09] Ed Grundy, Mark W. Jones, Robert S. Laramee, Rory P. Wilson, and Emily F. Shepard. Visualization of sensor data from animal movement. *Eurographics/ IEEE-VGTC Symposium on Visualization (Eurovis) 2009, Computer Graphics Forum*, 28(2):815–822, June 2009.

[GKT+08] C. Garth, H. Krishnan, X. Tricoche, T. Bobach, and K. I. Joy. Generation of Accurate Integral Surfaces in Time-Dependent Vector Fields. *Proceedings of IEEE Visualization '08*, October 2008.

[Gol29] Sydney Goldstein. On the Vortex Theory of Screw Propellers. In *Proceedings of the Royal Society*, volume 123, pages 440–465, April 1929.

[Goo] Google. WebGL Experiments. *http://www.chromeexperiments.com/webgl (last accessed 25/11/2011).*

[GOPT11] Tobias Germer, Mathias Otto, Ronald Peikert, and Holger Theisel. Lagrangian Coherent Structures with Guaranteed Material Separation. *Computer Graphics Forum*, 30(3):761–770, 2011.

[GPR+01] H. Garcke, T. Prußer, M. Rumpf, A. C. Telea, U. Weikard, and Jarke J. van Wijk. A Phase Field Model for Continuous Clustering on Vector Fields. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):230–241, July-September 2001.

[GPR+04] M. Griebel, T. Preusser, M. Rumpf, M. A. Schweitzer, and A. Telea. Flow Field Clustering via Algebraic Multigrid. *vis*, 00:35–42, 2004.

[GPS+11] Zhao Geng, Zhenmin Peng, Robert S.Laramee, Rick Walker, and Jonathan C.Roberts. Angular Histograms: Frequency-Based Visualizations for Large, High Dimensional Data. *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG), accepted by IEEE Visualization '11*, page forthcoming, 2011.

[Gro11] The Khronos Group. WebGL - OpenGL ES 2.0 for the Web. *http://www.khronos.org/webgl/ (last accessed 25/11/2011).*, March 2011.

[GRW+00] D. L. Gresh, B. E. Rogowitz, R. L. Winslow, D. F. Scollan, and C. K. Yung. WEAVE: A System For Visually Linking 3-D and Statistical Visualizations, Applied to Cardiac Simulation and Measurement Data. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 489–492, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[GTS+04] C. Garth, X. Tricoche, T. Salzbrunn, T. Bobach, and G. Scheuermann. Surface Techniques for Vortex Visualization. In *Data Visualization, Proceedings of the 6th Joint IEEE TCVG–EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 155–164, May 2004.

[GWT+08] Christoph Garth, Alexander Wiebel, Xavier Tricoche, Ken Joy, and Gerik Scheuermann. Lagrangian visualization of flow-embedded surface structures. *Computer Graphics Forum*, 27(3):1007–1014, May 2008.

[HA04] A. Helgeland and O. Andreassen. Visualization of Vector Fields Using Seed LIC and Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):673–682, 2004.

[HE06] A. Helgeland and T. Elboth. High-quality and interactive animations of 3d time-varying vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1535–1546, 2006. Member-Anders Helgeland.

[Hen98] C. Henze. Feature detection in linked derived spaces. In *Proceedings IEEE Visualization '98*, pages 87–94. IEEE, 1998.

[HH91] J. L. Helman and L. Hesselink. Visualizing Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, May 1991.

[HMK95] L. Hong, X. Mao, and A. E. Kaufman. Interactive Visualization of Mixed Scalar and Vector Fields. In *IEEE Visualization*, pages 240–247, 1995.

[HSH07] M. Hlawitschka, G. Scheuermann, and B. Hamann. Interactive Glyph Placement for Tensor Fields. In *ISVC07*, pages I: 331–340, 2007.

[Hul92] J. P. M. Hultquist. Constructing Stream Surfaces in Steady 3D Vector Fields. In *Proceedings IEEE Visualization '92*, pages 171–178, 1992.

[HWHJ99] B. Heckel, G. H. Weber, B. Hamann, and K. I. Joy. Construction of Vector Field Hierarchies. In *Proceedings IEEE Visualization '99*, pages 19–26, 1999.

[ID87] A. Inselberg and B. Dimsdale. Parallel Coordinates for Visualizing Multi-Dimensional Geometry. In *Proceedings of Computer Graphics International (CGI '87))*, pages 25–44, 1987.

[JEH01] B. Jobard, G. Erlebacher, and M. Y. Hussaini. Lagrangian-Eulerian Advection for Unsteady Flow Visualization. In *Proceedings IEEE Visualization '01*, pages 53–60. IEEE Computer Society, October 2001.

[JL97] B. Jobard and W. Lefer. Creating Evenly–Spaced Streamlines of Arbitrary Density. In *Proceedings of the Eurographics Workshop on Visualization in Scientific Computing '97*, volume 7, pages 45–55, 1997.

[JM10] Chad Jones and Kwan-Liu Ma. Visualizing Flow Trajectories Using Locality-based Rendering and Warped Curve Plots. *IEEE Transactions on Visualization and Computer Graphics*, 16:1587–1594, November 2010.

[KBH04] Robert Kosara, Fabian Bendix, and Helwig Hauser. TimeHistograms for Large, Time-Dependent Data. In *Proc. of the 6th Joint IEEE TCVG - EuroGraphics Symposium on Visualization(VisSym 2004*, Konstanz, Germany, 2004.

[KFH10] Johannes Kehrer, Peter Filzmoser, and Helwig Hauser. Brushing Moments in Interactive Visual Analysis. *Computer Graphics Forum*, 29(3):813–822, June 2010.

[KH91] R.V. Klassen and S.J. Harrington. Shadowed Hedgehogs: A Technique for Visualizing 2D Slices of 3D Vector Fields. *Visualization, 1991. Visualization '91, Proceedings., IEEE Conference on*, pages 148–153, 22-25 Oct 1991.

[KLG+11] Alexander Kuhn, Dirk J. Lehmann, Rocco Gaststeiger, Matthias Neugebauer, Bernhard Preim, and Holger Theisel. A Clustering-based Visualization Technique to Emphasize Meaningful Regions of Vector Fields. In *VMV 2011: Vision, Modeling and Visualization*, pages 191–198, Berlin, Germany, 2011.

[KM96] D. Knight and G. Mallinson. Visualizing Unstructured Flow Data Using Dual Stream Functions. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):355–363, December 1996.

[KM05] A. Kaufman and K. Mueller. *Overview of Volume Rendering*, chapter 1. Visualization Handbook. 2005.

[KW06] G. Kindlmann and C. Westin. Diffusion Tensor Visualization with Glyph Packing. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2006)*, 12(5):1329–1335, September-October 2006.

[LAK+98] D.H. Laidlaw, E.T. Ahrens, D. Kremers, M.J. Avalos, R.E. Jacobs, and C. Readhead. Visualizing Diffusion Tensor Images of the Mouse Spinal Cord. In *IEEE Visualization '98*, pages 127–134, 1998.

[Lan94] D. Lane. UFAT - a Particle Tracer for Time-Dependent Flow Fields. In *Proceedings of Visualization '94*, pages 257–264, Los Alamitos, CA, 1994. IEEE Computer Society Press.

[Lar03] R. S. Laramee. FIRST: A Flexible and Interactive Resampling Tool for CFD Simulation Data. *Computers & Graphics*, 27(6):905–916, 2003.

[Lar07] R. S. Laramee. Bob's Crazy Coding Conventions. Technical report, The Visual and Interactive Computing Group, Computer Science Department, Swansea University, Wales, UK, 2007.

[LBS03] G. S. Li, U. Bordoloi, and H. W. Shen. Chameleon: An Interactive Texture-based Framework for Visualizing Three-dimensional Vector Fields. In *Proceedings IEEE Visualization '03*, pages 241–248. IEEE Computer Society, 2003.

[LEG+08] R. S. Laramee, G. Erlebacher, C. Garth, H. Theisel, X. Tricoche, T. Weinkauf, and D. Weiskopf. Applications of Texture-Based Flow Visualization. *Engineering Applications of Computational Fluid Mechanics (EACFM)*, 2(3):264–274, September 2008.

[LGSH06] R. S. Laramee, C. Garth, J. Schneider, and H. Hauser. Texture-Advection on Stream Surfaces: A Novel Hybrid Visualization Applied to CFD Results. In *Data Visualization, The Joint Eurographics-IEEE VGTC Symposium on Visualization (EuroVis 2006)*, pages 155–162,368. Eurographics Association, 2006.

[LHD+04] R. S. Laramee, H. Hauser, H. Doleisch, F. H. Post, B. Vrolijk, and D. Weiskopf. The State of the Art in Flow Visualization: Dense and Texture-Based Techniques. *Computer Graphics Forum*, 23(2):203–221, June 2004.

[LHH05] R. S. Laramee, M. Hadwiger, and H. Hauser. Design and Implementation of Geometric and Texture-Based Flow Visualization Techniques. In *Proceedings of the 21st Spring Conference on Computer Graphics*, pages 67–74, May 2005.

[LHZP07] R.S. Laramee, H. Hauser, L. Zhao, and F. H. Post. Topology-Based Flow Visualization: The State of the Art. In *Topology-Based Methods in Visualization (Proceedings of Topo-in-Vis 2005)*, Mathematics and Visualization, pages 1–19. Springer, 2007.

[LJH03] R. S. Laramee, B. Jobard, and H. Hauser. Image Space Based Visualization of Unsteady Flow on Surfaces. In *Proceedings IEEE Visualization '03*, pages 131–138. IEEE Computer Society, 2003.

[LPPW95] W. Leeuw, H. Pagendarm, F. Post, and B. Waltzer. Visual Simulation of Experimental Oil-Flow Visualization by Spot Noise from Numerical Flow Simulation. In *Visualization in Scientific Computing '95*, pages 135–148. Springer-Verlag, May 1995.

[LPSW96] S. K. Lodha, A. Pang, R. E. Sheehan, and C. M. Wittenbrink. UFLOW: Visualizing Uncertainty in Fluid Flow. In *Proceedings IEEE Visualization '96*, pages 249–254, October 27–November 1 1996.

[LS07] L. Li and H.-W. Shen. Image-Based Streamline Generation and Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 13:forthcoming, 2007.

[LSH04] R. S. Laramee, J. Schneider, and H. Hauser. Texture-Based Flow Visualization on Isosurfaces from Computational Fluid Dynamics. In *Data Visualization, The Joint Eurographics-IEEE TVCG Symposium on Visualization (VisSym '04)*, pages 85–90,342. Eurographics Association, 2004.

[LvW95] W. Leeuw and J. van Wijk. Enhanced Spot Noise for Vector Field Visualization. In *Proceedings IEEE Visualization '95*, pages 233–239. IEEE Computer Society, October 1995.

[LvWJH04] R. S. Laramee, J. J. van Wijk, B. Jobard, and H. Hauser. ISA and IBFVS: Image Space Based Visualization of Flow on Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):637–648, November 2004.

[LWSH04] R. S. Laramee, D. Weiskopf, J. Schneider, and H. Hauser. Investigating Swirl and Tumble Flow with a Comparison of Visualization Techniques. In *Proceedings IEEE Visualization 2004*, pages 51–58, 2004.

[Ma07] Kwan-Liu Ma. Machine Learning to Boost the Next Generation of Visualization Technology. *IEEE Comput. Graph. Appl.*, 27:6–9, September 2007.

[MB95] N. Max and B. Becker. Flow Visualization Using Moving Textures. In *Proceedings of the ICASW/LaRC Symposium on Visualizing Time-Varying Data*, pages 77–87, September 1995.

[MBC93] N. Max, B. Becker, and R. Crawfis. Flow Volumes for Interactive Vector Field Visualization. In *Proceedings IEEE Visualization '93*, pages 19–24. IEEE Computer Society, October 1993.

[MCG94] N. Max, R. Crawfis, and C. Grant. Visualizing 3D Velocity Fields Near Contour Surfaces. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 248–255, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[MCOW11] Ian Masters, John Chapman, James Orme, and Miles Willis. A Robust Blade Element Momentum Theory Model for Tidal Stream Turbines Including Tip and

Hub Loss Corrections. *Proceedings of the Institute of Marine Engineering, Science and Technology Part A - Journal of Marine Engineering and Technology*, 10(1):25–35, January 2011.

[MHHI98] X. Mao, Y. Hatanaka, H. Higashida, and A. Imamiya. Image-Guided Streamline Placement on Curvilinear Grid Surfaces. In *Proceedings IEEE Visualization '98*, pages 135–142, 1998.

[MLD05] Alexander McKenzie, Santiago V. Lombeyda, and Mathieu Desbrun. Vector Field Analysis and Visualization through Variational Clustering. In *EuroVis*, pages 29–35, 2005.

[MLP+10] T. McLouglin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over Two Decades of Geometric Flow Visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010.

[MLZ08] T. Mcloughlin, R.S. Laramee, and E. Zhang. Easy Stream Surfaces: A Fast and Simple Approach to Stream Surface Generation. Technical report, Computer Science Department, Swansea University, March 2008.

[MT+03] O. Mattausch, T. Theussl, , H. Hauser, and E. Groller. Strategies for Interactive Exploration of 3D Flow Using Evenly-Spaced Illuminated Streamlines. In *Proceedings of the 19th Spring Conference on Computer Graphics*, pages 213–222, 2003.

[NHM97] G. M. Nielson, H. Hagen, and H. Müller. *Scientific Visualization: Overviews, Methodologies, and Techniques*. IEEE Computer Society, 1997.

[Nok] Nokia. QT. *http://qt.nokia.com/products/library (last accessed 25/05/2011).*

[Ope] OpegnGL.org. Vertex buffer object whitepaper. *http://www.opengl.org/registry/specs/ARB/vertex_buffer_object.txt (last accessed 02/06/2011).*

[PBK10] Harald Piringer, Wolfgang Berger, and JÜrgen Krasser. HyperMoVal: Interactive Visual Validation of Regression Models for Real-Time Simulation. *Computer Graphics Forum*, 29(3):983–992, 2010.

[PGL+11] Zhenmin Peng, Edward Grundy, Robert S. Laramee, Guoning Chen, and Nick Croft. Mesh-Driven Vector Field Clustering and Visualization: An Image-Based Approach. *IEEE Transactions on Visualization and Computer Graphics (IEEE TVCG)*, page forthcoming, 2011.

[PGSC10] Zhenmin Peng, Zhao Geng, Robert S.Laramee, and Nick Croft. Visualization of Flow Past a Marine Turbine: The Search for Sustainable Energy. Technical report, Department of Computer Science, Swansea University, UK, Dec 2010.

[PL08] Z. Peng and R.S. Laramee. Vector Glyphs for Surfaces: A Fast and Simple Glyph Placement Algorithm for Adaptive Resolution Meshes. In *Proceedings of Vision, Modeling, and Visualization (VMV) 2008*, pages 61–70, Constance, Germany, 8-10 October 2008.

[PL09] Z. Peng and R.S. Laramee. Higher Dimensional Vector Field Visualization: A Survey. In *Proceedings of Theory and Practice of Computer Graphics (TPCG '09)*, pages 61–70, Cardiff, UK, 17-19 June 2009.

[PLCZ09] Zhenmin Peng, Robert S. Laramee, Guoning Chen, and Eugene Zhang. Glyph and Streamline Placement Algorithms for CFD Simulation Data. In *NAFEMS World Congress Conference Proceedings*, page 66. NAFEMS–The International Association for the Engineering Analysis Community, June 16-19 2009. (full proceedings on CDROM).

[PNB02] Frits H. Post, Gregory M. Nielson, and Georges-Pierre Bonneau, editors. *Data Visualization: The State of the Art*. Springer, 2002.

[PTA+11] Armin Pobitzer, Murat Tutkun, Øyvind Andreassen, Raphael Fuchs, Ronald Peikert, and Helwig Hauser. Energy-scale aware feature extraction for flow visualization. *Computer Graphics Forum*, 30(3):771–780, 2011.

[PTMB09] Harald Piringer, Christian Tominski, Philipp Muigg, and Wolfgang Berger. A Multi-Threading Architecture to Support Interactive Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1113–1120, 2009.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[PVH+02] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. Feature Extraction and Visualization of Flow Fields. In *Eurographics 2002 State-of-the-Art Reports*, pages 69–100, 2–6 September 2002.

[PVH+03] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum*, 22(4):775–792, Dec. 2003.

[RMSS+07] T. Ropinski, J. Meyer-Spradow, M. Specht, K.H. Hinrichs, and B. Preim. Surface Glyphs for Visualizing Multimodal Volume Data. In *Proceedings of the 12th International Fall Workshop on Vision, Modeling, and Visualization (VMV07)*, pages 3–12, nov 2007.

[RP08] T. Ropinski and B. Preim. Taxonomy and Usage Guidelines for Glyph-based Medical Visualization. In *Proceedings of the 19th Conference on Simulation and Visualization (SimVis08)*, 2008.

[RSHTE99] C. Rezk-Salama, P. Hastreiter, C. Teitzel, and T. Ertl. Interactive Exploration of Volume Line Integral Convolution Based on 3D-Texture Mapping. In *Proceedings IEEE Visualization '99*, pages 233–240, 1999.

[SBH+01] G. Scheuermann, T. Bobach, H. Hagen, K. Mahrous, B. Hamann, K. I. Joy, and W. Kollmann. A Tetrahedral-based Stream Surface Algorithm. In *Proceedings IEEE Visualization 2001*, pages 151–157, October 2001.

[SEHW02] A. Sigfridsson, T. Ebbers, E. Heiberg, and L. Wigstrom. Tensor Field Visualisation Using Adaptive Filtering of Noise Fields Combined with Glyph Rendering. *Visualization, 2002. VIS 2002. IEEE*, pages 371–378, 1-1 Nov. 2002.

[SGLS08] John Stasko, Carsten Görg, Zhicheng Liu, and Kanupriya Singhal. Jigsaw: Supporting Investigative Analysis through Interactive Visualization. *Information Visualization*, 7(2):118–132, 2008.

[SH95] D. Stalling and H. Hege. Fast and Resolution Independent Line Integral Convolution. In *Proceedings of ACM SIGGRAPH 95*, Annual Conference Series, pages 249–256. ACM SIGGRAPH, ACM Press / ACM SIGGRAPH, 1995.

[Shn92] B. Shneiderman. *Designing the User Interface: Strategies for Effective Computer Interaction*. Addison-Wesley, 2nd edition, 1992.

[SLCZ09] B. Spencer, R.S. Laramee, G. Chen, and E. Zhang. Evenly-Spaced Streamlines for Surfaces. *Computer Graphics Forum*, 2009. forthcoming.

[SP07] Filip Sadlo and Ronald Peikert. Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction. *IEEE Transactions on Visualization and Computer Graphics*, 13:1456–1463, 2007.

[ST90] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. In *Computer Graphics (San Diego Workshop on Volume Visualization)*, volume 24, pages 63–70, November 1990.

[Sta97] D. Stalling. LIC on Surfaces. In *Texture Synthesis with Line Integral Convolution*, pages 51–64. ACM SIGGRAPH 97, International Conference on Computer Graphics and Interactive Techniques, 1997.

[STM08] Bela Soni, David Thompson, and Raghu Machiraju. Visualizing Particle/Flow Structure Interactions in the Small Bronchial Tubes. *IEEE Transactions on Visualization and Computer Graphics*, 14:1412–1427, November 2008.

[STW+08] K. Shi, H. Theisel, T. Weinkauf, H. Hauser, K. Matkovic, H.-C. Hege, and H.-P. Seidel. Path Line Attributes - an Information Visualization Approach to Analyzing the Dynamic Behavior of 3d Time-Dependent Flow Fields. In *Topo-In-Vis 2007*, Springer series of Mathematics and Visualization, pages 60–74, Grimma, Germany, 2008. Springer.

[STWE07] T. Schafhitzel, E. Tejada, D. Weiskopf, and T. Ertl. Point-based Stream Surfaces and Path Surfaces. In *GI '07: Proceedings of Graphics Interface 2007*, pages 289–296, New York, NY, USA, 2007. ACM.

[Sun03] A. Sundquist. Dynamic Line Iintegral Convolution for Visualizing Streamline Evolution. In *IEEE Transactions on Visualization and Computer Graphics*, volume 9(3), pages 273–282, 2003.

[SVL91] W. Schroeder, C. R. Volpe, and W. E. Lorensen. The Stream Polygon: A Technique for 3D Vector Field Visualization. In *Proceedings IEEE Visualization '91*, pages 126–132, 1991.

[SvW91] J. Stolk and J.J. van Wijk. Surface Particles for 3d Flow Visualization,. In *Proceedings of the Second Eurographics Workshop on Visualization in Scientific Computing*, pages 22–24. To be published by Springer Verlag, April, 1991.

[SWC+08] Dominic Schneider, Alexander Wiebel, Hamish Carr, Mario Hlawitschka, and Gerik Scheuermann. Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14:1475–1482, November 2008.

[TB96] G. Turk and D. Banks. Image-Guided Streamline Placement. In *ACM SIGGRAPH 96 Conference Proceedings*, pages 453–460, August 1996.

[Tec] Tecplot. Tecplot360. *http://www.tecplot.com/ (accessed 25/11/2010).*

[TK93] B. Taylor and C. Kuyatt. Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results. Technical report, National Institute of Standards and Technology Technical Note 1297, Gaithersburg, MD, January 1993.

[Tor02] R. J. Torres. *Practitioners Handbook for User Interface Design and Development*. Prentice Hall, 2002.

[TS01] X. Tricoche and G. Scheuermann. Continuous Topology Simplification of Planar Vector Fields. In *Proceedings IEEE Visualization 2001*, pages 159–166, 2001.

[TS06] Alexandru Telea and Robert Strzodka. Multiscale Image Based Flow Visualization. In *Proc. of SPIE-IS&T Electronic Imaging, Visualization and Data Analysis (VDA) 2006*, volume 6060, pages 1–11, Jan 2006.

[TvW99] A. Telea and J.J. van Wijk. Simplified Representation of Vector Fields. In *Proceedings IEEE Visualization '99*, pages 35–42, 1999.

[TvW03] A. Telea and J. J. van Wijk. 3D IBFV: Hardware-Accelerated 3D Flow Visualization. In *Proceedings IEEE Visualization '03*, pages 233–240. IEEE Computer Society, 2003.

[UA01] U.Trottenberg and A.Schuller. *Multigrid*. Academic Press, Inc., Orlando, FL, USA, 2001.

[UKSE08] M. Üffinger, T. Klein, M. Strengert, and T. Ertl. Gpu-based Streamlines for Surface-Guided 3D Flow Visualization. In *Proceedings of Vision, Modeling, and Visualization (VMV) 2008*, pages 100–106, Constance, Germany, 8-10 October 2008.

[vFWTS08] W. von Funck, T. Weinkauf, H. Theisel, and H.P. Seidel. Smoke Surfaces: An Interactive Flow Visualization Technique Inspired by Real-World Flow Experiments. *Proceedings of IEEE Visualization '08*, October 2008.

[vH] Dimitri van Heesch. Doxygen. *http://www.stack.nl/ dimitri/doxygen/index.html (last accessed 25/05/2011).*

[VKP00] V. Verma, D. Kao, and A. Pang. A Flow-guided Streamline Seeding Strategy. In *Proceedings IEEE Visualization 2000*, pages 163–170, 2000.

[VM96] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Addison-Wesley, February 1996.

[VP04] V. Verma and A. Pang. Comparative Flow Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):609–624, 2004. Senior Member-Alex Pang.

[vW91] J. J. van Wijk. Spot noise-Texture Synthesis for Data Visualization. In Thomas W. Sederberg, editor, *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, volume 25, pages 309–318, 1991.

[vW93a] J.J. van Wijk. Flow Visualization with Surface Particles. *IEEE Computer Graphics and Applications*, 13(4):18–24, July 1993.

[vW93b] J.J. van Wijk. Implicit Stream Surfaces. In *Proceedings of the Visualization '93 Conference*, pages 245–252. IEEE Computer Society, October 1993.

[vW02] J. J. van Wijk. Image Based Flow Visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.

[vW03] J. J. van Wijk. Image Based Flow Visualization for Curved Surfaces. In *Proceedings IEEE Visualization '03*, pages 123–130. IEEE Computer Society, 2003.

[War02] M.O. Ward. A Taxonomy of Glyph Placement Strategies for Multidimentional Data Visualization. *Information Visualization*, 1(3/4):(194-210), 2002.

[War04] Colin Ware. *Information Visualization: Perception for Design, Second Edition*. Morgan Kaufmann, 2004.

[WBWK00] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 110–119, New York, NY, USA, 2000. ACM.

[WCM+10] A. J. Williams, T. N. Croft, I. Masters, C. R. Bennett, S. G. Patterson, and M. R. Willis. A Combined BEM-CFD Model for Tidal Stream Turbines. In *3rd International Conference on Ocean Energy*, Bilbao, Spain, 6 October 2010.

[WCMC09] Alison J. Williams, T. Nick Croft, Medzid Muhasilovic, and Mark Cross. Significance of Site Data When Modelling Tidal Stream Turbines. Technical report, School of Engineering, Swansea University, UK, 2009.

[WCMW10] Alison J. Williams, T. Nick Croft, Ian Masters, and Miles R. Willis. Investigating Environmental Issues related to Operation of Tidal Stream Turbines using a BEM-CFD Model. In *World Renewable Energy Congress XI*, Abu Dhabi, UAE, 25-30 September 2010.

[WDeC+10] Xiaoyu Wang, Wenwen Dou, Shen en Chen, William Ribarsky, and Remco Chang. An Interactive Visual Analytics System for Bridge Management. *Computer Graphics Forum*, 29(3):1033–1042, 2010.

[WE04a] D. Weiskopf and T. Ertl. GPU-Based 3D Texture Advection for the Visualization of Unsteady Flow Fields. In *WSCG 2004 Conference Proceedings, Short Papers*, pages 259–266, February 2004.

[WE04b] D. Weiskopf and T. Ertl. A Hybrid Physical/Device-Space Approach for Spatio-Temporally Coherent Interactive Texture Advection on Curved Surfaces. In *Proceedings of Graphics Interface*, pages 263–270, 2004.

[Wea04] Chris Weaver. Building Highly-Coordinated Visualizations in Improvise. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 159–166, Washington, DC, USA, 2004. IEEE Computer Society.

[Wea09] Chris Weaver. Cross-Filtered Views for Multidimensional Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):192–204, 2009.

[WEE03] D. Weiskopf, G. Erlebacher, and T. Ertl. A Texture-Based Framework for Spacetime-Coherent Visualization of Time-Dependent Vector Fields. In *Proceedings IEEE Visualization '03*, pages 107–114, 2003.

[Wes90] L. Westover. Footprint Evaluation for Volume Rendering. In *SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques*, pages 367–376, New York, NY, USA, 1990. ACM.

[WG97] R. Wegenkittl and M. E. Gröller. Fast Oriented Line Integral Convolution for Vector Field Visualization via the Internet. In *Proceedings IEEE Visualization '97*, pages 309–316, October 19–24 1997.

[WMT+09] Miles Willis, Ian Masters, Sara Thomas, Rebecca Gallie, Jo Loman, Andy Cook, Reza Ahmadian, Roger Falconer, Binliang Lin, Guanghai Gao, Mark Cross, Nick Croft, Alison Williams, Medzid Muhasilovic, Ian Horsfall, Rob Fidler, Chris Wooldridge, Ian Fryett, Paul Evans, Tim ODoherty, Daphne ODoherty, and Allan Mason-Jones. Tidal Turbine Deployment in the Bristol Channel CA Case Study. *Proceedings of the Institute of Civil Engineers - Energy*, 163:93 C105, 2009.

[WMW86] G. Wyvill, C. McPheeters, and B. Wyvill. Data Structure for Soft Objects. *The Visual Computer*, 2(4):227–234, February 1986.

[WSE05] D. Weiskopf, T. Schafhitzel, and T. Ertl. Real-Time Advection and Volumetric Illumination for the Visualization of 3D Unsteady Flow. In *Data Visualization, Proceedings of the 7th Joint EUROGRAPHICS–IEEE VGTG Symposium on Visualization (EuroVis 2005)*, pages 13–20, May 2005.

[WSE07] D. Weiskopf, T. Schafhitzel, and T. Ertl. Texture-Based Visualization of Unsteady 3d Flow by Real-Time Advection and Volumetric Illumination. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):569–582, 2007.

[WWYM10] Jishang Wei, Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. A sketch-based interface for classifying and visualizing vector fields. In *PacificVis'10*, pages 129–136, 2010.

[XCML11] Huaxun Xu, Zhi-Quan Cheng, Ralph R. Martin, and Sikun Li. 3d flow features visualization via fuzzy clustering. *Vis. Comput.*, 27:441–449, June 2011.

[XW05] R. Xu and D. Wunsch. Survey of Clustering Algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678, 2005.

[XZC04] D. Xue, C. Zhang, and R. Crawfis. Rendering Implicit Flow Volumes. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 99–106, Washington, DC, USA, 2004. IEEE Computer Society.

[YKP05] X. Ye, D. Kao, and A. Pang. Strategy for Seeding 3D Streamlines. In *Proceedings IEEE Visualization 2005*, pages 471–476, 2005.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. *SIGMOD Rec.*, 25(2):103–114, 1996.

[ZSH96] M. Zöckler, D. Stalling, and H.-C. Hege. Interactive Visualization of 3D-vector Fields Using Illuminated Stream Lines. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 107–ff., Los Alamitos, CA, USA, 1996. IEEE Computer Society Press.