

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CHRISTIAN AZAMBUJA PAGOT

**Feature Extraction and Visualization from
Higher-Order CFD Data**

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. João Luiz Dihl Comba
Advisor

Porto Alegre, July 2011

CIP – CATALOGING-IN-PUBLICATION

Pagot, Christian Azambuja

Feature Extraction and Visualization from Higher-Order CFD Data / Christian Azambuja Pagot. – Porto Alegre: PPGC da UFRGS, 2011.

108 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Advisor: João Luiz Dihl Comba.

1. Higher-order CFD data. 2. Feature extraction. 3. Parallel vectors operator. 4. Isocontouring. 5. Interval arithmetic. I. Comba, João Luiz Dihl. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Carlos Alexandre Netto

Vice-Reitor: Prof. Rui Vicente Oppermann

Pró-Reitor de Pós-Graduação: Prof. Aldo Bolten Lucion

Diretor do Instituto de Informática: Prof. Flávio Rech Wagner

Coordenador do PPGC: Prof. Álvaro Freitas Moreira

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

To my mother, Lilian da Silva Azambuja, my sister, Marcele Azambuja Pagot, my grandfather, Walter Machado de Azambuja and my grandmother, Celony da Silva Azambuja.

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, João Luiz Dihl Comba, for the pleasant and fruitful collaboration during all the years.

I wish to express my appreciation to my dissertation committee: Professors Luis Gustavo Nonato, Luiz Henrique de Figueiredo and Marcelo Walter. All these gentlemen have contributed valuable suggestions to this work.

I would also to warmly thank Professor Thomas Ertl, who accepted me for an one-year exchange period at the Universität Stuttgart, in Germany. Thanks also to all people at Universität Stuttgart, especially to Professor Daniel Weiskopf, Eduardo Tejada, Filip Sadlo, Joachim Vollrath, Markus Üffinger, and Ralf Botchen, for the direct support with this work.

At the Universidade Federal do Rio Grande do Sul, I thank André S. Spritzer, Barbara Bellaver, Carlos A. Dietrich, Daniel K. Osmari, Denison Linus da Motta Tavares, Francisco M. Pinto, Isabel Cristina Siqueira da Silva, Jonatas Medeiros, Juan M. Ibiapina, Juliano M. Franz, Leandro Augusto Frata Fernandes, Leonardo Fischer, Leonardo Schmitz, Luiz F. Scheidegger, Marcos Slomp, Marilena Maule, Rafael P. Torchelsen, Renato Silveira, Rodrigo Barni, Vinicius Da Costa De Azevedo, Vitor A.M. Jorge, and Vitor F. Pamplona for the scientific discussions and the fellowship along all these years. Also, I would like to thank Professors Anderson Maciel, Carla M. D. S. Freitas, Luciana P. Nedel, Manuel Menezes de Oliveira Neto and Roberto da Silva.

The PPGC-UFRGS has provided the support and equipment that I needed to produce and complete my thesis. My work has been supported by the CNPq (140238/2007-7). Specifically, funding has come from CAPES (Probral 3192/08-3) for the one-year exchange program at the Universität Stuttgart.

My family deserves special mention. My mother, Lilian da Silva Azambuja, my sister, Marcele Azambuja Pagot, my grandfather, Walter Machado de Azambuja, and my grandmother, Celony da Silva Azambuja, have always raised me with their caring and gentle love. I can not find words to express all my gratitude. I love you. My profound gratitude to my girlfriend, Zenires Figueiredo de Azevedo, for all the patience and love, even during a transition moment in her life. I love you.

I would like to thank all the people that generously gave me their support and advice, making the realization of this thesis possible. My apology that I could not mention personally one by one.

Last, but definitely not least, I thank all the taxpayers in this country for supporting the public education.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS	11
LIST OF FIGURES	13
LIST OF TABLES	15
ABSTRACT	17
RESUMO	19
1 INTRODUCTION	21
1.1 Isocontouring	22
1.2 Extraction of Parallel Vectors Line-type Features	22
1.3 Thesis Statements	23
1.4 Results	24
1.5 Organization of this Document	24
2 VISUALIZATION OF CFD DATA	27
2.1 Visualization	27
2.2 Traditional CFD Data	28
2.2.1 Mesh Types	28
2.2.2 Field Description and Interpolation	29
2.3 Visualization of Volumetric CFD Scalar Fields	30
2.3.1 Direct Volume Rendering	30
2.3.2 Isocontouring	31
2.4 Feature Extraction	32
2.5 Higher-Order CFD Data	33
2.5.1 Discontinuous Galerkin Datasets	34
3 ISOCONTOURING HIGHER-ORDER SURFACES	37
3.1 Isocontouring Higher-Order Data	37
3.2 The Proposed Isocontouring Method	38
3.3 Object Space Sampling	39
3.4 Image Space Refinement	40
3.5 Sampling Strategies	41
3.5.1 Sampling Issues	42
3.6 Estimating Quad Sizes using FTLE	42
3.7 Implementation	44
3.7.1 Data Storage	44

3.7.2	GLSL Shaders	45
3.7.3	Computation Flow	45
3.8	Results	46
3.8.1	Performance	46
3.8.2	Isocontouring Quality	47
4	THE PARALLEL VECTORS OPERATOR	53
4.1	Definition	53
4.2	Feature Classification and Filtering	54
4.3	Application Examples	54
4.3.1	Zero Curvature	54
4.3.2	Extremum Lines	56
4.3.3	Ridges and Valleys	57
4.3.4	Other Uses of the Parallel Vectors Operator	59
4.4	Extracting Features	60
4.4.1	Isosurface Based Approach	60
4.4.2	Iteration on Cell Faces	60
4.4.3	Analytic Solution at Triangular Faces	61
4.4.4	Curve-following Methods	61
4.5	State of the Art in Parallel Vectors	62
4.6	Summary of Parallel Vectors Features	62
5	EXTRACTING FEATURES FROM HIGHER-ORDER CFD DATA	65
5.1	Background	65
5.1.1	Feature Flow Fields	65
5.1.2	Interval Arithmetic	67
5.2	PV feature extraction from HO data using RAA	71
5.2.1	Seed Extraction	72
5.2.2	Feature Tracing and Filtering	75
5.3	CUDA Implementation	77
5.3.1	Ordering of Computation	77
5.3.2	Polynomial Evaluation and Storage	77
5.3.3	Seed Extraction	78
5.3.4	Feature Tracing and Classification	79
5.4	Results	80
6	CONCLUSIONS AND FUTURE WORK	87
6.1	Isocontouring of Higher-Order Data	87
6.2	Line-type Feature extraction from Higher-Order Data	88
6.3	Future Work	88
APPENDIX A	MATHEMATICAL DEFINITIONS	91
A.1	Differential Geometry Equations	91
A.2	Directional Newton Method	91
A.3	Convective Derivative	92
A.4	Strawn, Kenright and Ahmad Vortex Core Definition in Terms of the PV Operator	92
APPENDIX B	ISOCONTOURING CODE	93
B.1	Unprojecting Fragments	93

APPENDIX C	IEEE VISUALIZATION 2008 POSTER	95
APPENDIX D	RESUMO EXPANDIDO	
	(EXTENDED ABSTRACT IN PORTUGUESE)	97
D.1	Extração de Isosuperfícies	98
D.2	Extração de Estruturas Lineares Descritas pelo Operador de Vetores Paralelos	99
D.3	Idéia Central	100
D.4	Resultados	101
D.5	Conclusões e Trabalhos Futuros	101
D.5.1	Trabalhos Futuros	102
REFERENCES		103

LIST OF ABBREVIATIONS AND ACRONYMS

AA	Affine arithmetic
AABB	Axis aligned bounding box
AF1	First affine form
AMR	Adaptive mesh refinement
BUBO	Bindable uniform buffer object
CFD	Computational fluid dynamics
DFC	Dinâmica de fluidos computacional
DG	Discontinuous Galerkin
DVR	Direct volume rendering
FFF	Feature flow field
FE	Finite element
FTLE	Finite-time Lyapunov exponent
GLSL	OpenGL shading language
GPU	Graphics processing unit
HO	Higher-order
IA	Interval arithmetic
MC	Marching cubes
MIP	Maximum intensity point
NDC	Normalized device coordinates
NR	Newton-Raphson
PV	Parallel vectors
RAA	Reduced affine arithmetic
SIMD	Single-instruction multiple-data
SPH	Smoothed-particle hydrodynamics
VBO	Vertex buffer object

LIST OF FIGURES

Figure 2.1:	Common element shapes used for 3D domain tessellation in traditional CFD.	28
Figure 2.2:	Mesh types.	29
Figure 2.3:	Element-wise mapping functions transform points from the element’s reference space to the world space.	34
Figure 2.4:	<i>Shock channel</i> dataset structure.	35
Figure 2.5:	<i>Sphere</i> dataset structure.	35
Figure 3.1:	The proposed higher-order isocontouring method pipeline.	39
Figure 3.2:	Isocontouring approximation with point sampling and quad generation.	40
Figure 3.3:	Isosurface refinement through quad rasterization and fragment projection.	41
Figure 3.4:	Neighboring points located in the same and disjoint surfaces.	44
Figure 3.5:	Isocontouring of dataset <i>shock channel</i> with varying quad sizes.	48
Figure 3.6:	Isocontouring of dataset <i>sphere</i> with varying quad size.	49
Figure 3.7:	Continuous isocontouring of a continuous field.	50
Figure 3.8:	Isocontouring performance according to the number of processed elements.	50
Figure 3.9:	Comparison between the proposed method and Knoll’s method.	51
Figure 3.10:	Comparison between our method and a ray traced image.	51
Figure 3.11:	Isocontouring artifacts reduction through the use of a proposed FTLE-based heuristic.	52
Figure 4.1:	Height field: contour map together with the ridges and valley lines.	58
Figure 5.1:	Selection of the current FFF based on the angle formed between the gradient vectors.	68
Figure 5.2:	Feature extraction pipeline.	72
Figure 5.3:	Alignment between quadtree cells and octree cell faces reduces domain dimension.	74
Figure 5.4:	Selection of the best correcting plane based on the angle formed with the predictor move.	76
Figure 5.5:	Pruning extracted feature lines against the actual element boundary.	76
Figure 5.6:	Alignment-based grouping of quadtree cells increases performance during quadtree subdivision.	79
Figure 5.7:	Performance results obtained with the proposed parallel feature extraction method.	82

Figure 5.8:	Performance results obtained with the octree subdivision strategy proposed by Theisel.	83
Figure 5.9:	Several octree refinement levels for one cell of the <i>shock channel</i> dataset.	84
Figure 5.10:	Feature extraction from the <i>shock channel</i> dataset.	84
Figure 5.11:	Comparison between our higher-order feature extraction method and the original method, based on linear data.	85
Figure 5.12:	Extracted line-type features from sphere dataset under increasingly restricting filtering criteria.	86

LIST OF TABLES

Table 4.1:	Examples of parallel vectors expressions that represent line-type features.	64
------------	---	----

ABSTRACT

Computational fluid dynamics (CFD) methods have been employed in the studies of subjects such as aeroacoustics, gas dynamics, turbo machinery, viscoelastic fluids, among others. However, the need for accuracy and high performance resulted in methods whose solutions are becoming increasingly more complex. In this context, feature extraction and visualization methods play a key role, making it easier and more intuitive to explore and analyze the simulation data.

Feature extraction methods detect and isolate relevant structures in the context of data analysis. In the case of flow analysis, these structures could be pressure isocontours, vortex cores, detachment lines, etc. By assigning visual attributes to these structures, visualization methods allow for a more intuitive analysis through visual inspection.

Traditionally, CFD methods represent the solution as piecewise linear basis functions defined over domain elements. However, the evolution of CFD methods has led to solutions represented analytically by higher-order functions. Despite their accuracy and efficiency, data generated by these methods are not compatible with feature extraction and visualization methods targeted to linearly interpolated data. An alternative approach is resampling, which allows the use of existing low order feature extraction and visualization methods. However, resampling is not desirable since it may introduce error due to subsampling and increase memory consumption associated to samples storage. To overcome these limitations, attention has recently been given to methods that handle higher-order data directly.

The main contributions of this thesis are two methods developed to operate directly over higher-order data. The first method consists of an isocontouring method. It relies on a hybrid technique that, by splitting the isocontouring workload over image and object space computations, allows for interactive data exploration by dynamically changing isovalues. The second method is a line-type feature extraction method. The search for features is accomplished using adaptive subdivision methods driven by the evaluation of the inclusion form of the parallel vectors operator. Both methods were designed to take advantage of the parallelism of current graphics hardware. The obtained results are presented for synthetic and real simulation higher-order data generated with the discontinuous Galerkin method.

Keywords: Higher-order CFD data, feature extraction, parallel vectors operator, isocontouring, interval arithmetic.

RESUMO

Métodos de simulação baseados em dinâmica de fluidos computacional (DFC) têm sido empregado em diversas áreas de estudo, tais como aeroacústica, dinâmica dos gases, fluidos viscoelásticos, entre outros. Entretanto, a necessidade de maior acurácia e desempenho destes métodos têm dado origem a soluções representadas por conjuntos de dados cada vez mais complexos. Neste contexto, técnicas voltadas à extração de estruturas relevantes (*features*), e sua posterior visualização, têm um papel muito importante, tornando mais fácil e intuitiva a análise dos dados gerados por simulações.

Os métodos de extração de estruturas detectam e isolam elementos significativos no contexto da análise dos dados. No caso da análise de fluidos, estas estruturas podem ser isosuperfícies de pressão, vórtices, linhas de separação, etc. A visualização, por outro lado, confere atributos visuais a estas estruturas, permitindo uma análise mais intuitiva através de sua inspeção visual.

Tradicionalmente, métodos de DFC representam suas soluções como funções lineares definidas sobre elementos do domínio. Entretanto, a evolução desses métodos tem dado origem a soluções representadas analiticamente através de funções de alta ordem. Apesar destes métodos apresentarem características desejáveis do ponto de vista de eficiência e acurácia, os dados gerados não são compatíveis com os métodos de extração de estruturas ou de visualização desenvolvidos originalmente para dados interpolados linearmente. Uma alternativa para este problema consiste na redução da ordem dos dados através de reamostragem e posterior aplicação de métodos tradicionais para extração de estruturas e visualização. Porém, o processo de amostragem pode introduzir erros nos dados ou resultar em excessivo consumo de memória, necessária ao armazenamento das amostras. Desta forma, torna-se necessário o desenvolvimento de métodos de extração e visualização que possam operar diretamente sobre os dados de alta ordem.

As principais contribuições deste trabalho consistem em dois métodos que operam diretamente sobre dados de alta ordem. O primeiro consiste em um método para extração e visualização de isosuperfícies. O método baseia-se em uma abordagem híbrida que, ao distribuir o esforço computacional envolvido na extração e visualização das isosuperfícies em operações executadas nos espaços do objeto e da imagem, permite a exploração interativa de isosuperfícies através da troca de isovalores. O segundo método consiste em uma técnica para extração de estruturas lineares, onde a avaliação da forma intervalar do operador *parallel vectors*, em conjunto com métodos de subdivisão adaptativa, é utilizada como critério de pesquisa destas estruturas. Ambos os métodos foram projetados para tirarem proveito do paralelismo do *hardware* gráfico. Os resultados obtidos são apresentados tanto para dados sintéticos quanto para dados de simulações gerados através do método de Galerkin descontínuo.

Palavras-chave: Dados de DFC de alta ordem, extração de estruturas (*features*), operador

de vetores paralelos (*parallel vectors*), extração de isosuperfícies, aritmética intervalar.

1 INTRODUCTION

The foundations for research on computational fluid dynamics (CFD) were established in the 1960's, in the paper by Hess and Smith (HESS; SMITH, 1967). Since then, the increase in processing power and the need for accuracy led CFD methods to evolve into more sophisticated methods that can be applied to a broad range of problems, including aeroacoustics (REYMEN et al., 2005; RICHTER; STILLER; GRUNDMANN, 2009), gas dynamics (VAN DEN BERG, 2009; GALANIN; SAVENKOV; TOKAREVA, 2009), viscoelastic fluids (GUÉNETTE et al., 2008), turbo machinery (SUN et al., 2010), transport in porous media (AL-HAMAMRE; AL-ZOUBI; TRIMIS, 2010), among others. This evolution led to increasingly complex solutions, whose analysis has become significantly harder. In this context, feature extraction and visualization techniques started to play a key role.

Feature extraction methods concentrate on the automated analysis, detection and selection of relevant data portions, capturing meaningful structures out of large and intricate data. In the context of flow visualization, examples of such features are creases (ridges and valleys) in scalar fields, as well as separation, attachment, and vortex core lines in vector fields. These methods can significantly reduce the amount of data to be manipulated, thus allowing to focus attention on relevant data. Visualization techniques, on the other hand, allow for a more intuitive and natural way of inspecting data by assigning a visual representation to the selected structures.

The evolution of CFD methods led to solutions represented analytically by higher-order functions. Despite their greater accuracy and efficiency, data generated by these methods are not compatible with feature extraction and visualization methods targeted to linearly interpolated data. The pragmatic approach is resampling, which allows the use of existing low order feature extraction and visualization methods. However, resampling is not a desirable approach since it may introduce error due to subsampling and increase memory consumption, needed for samples storage. To overcome these limitations, more attention has recently been paid to methods that handle higher-order data directly.

In this thesis we propose two methods that operate directly over analytical higher-order CFD data. The first method consists of a previewing system for interactive exploration of isosurfaces defined by higher-order data. Interactivity is achieved by splitting the isocontouring workload over object and image spaces. The second method is an efficient parallel vectors line-type feature extraction method that relies on the use of reduced affine arithmetic and parallel processing to improve performance and allow for guaranteed bounds regarding accuracy with respect to existence, position, and topology of the features obtained.

Both methods were designed to take advantage of parallel graphics hardware. Quantitative and qualitative results, for both methods, are presented for synthetic and real higher-

order simulation data generated with discontinuous Galerkin method. The next sections summarize both methods, for which more detailed descriptions are given in the following chapters.

1.1 Isocontouring

Approximate representation of higher-order data offers a viable visualization approach by allowing a trade-off between rendering speed and accuracy. Low-order representations using an isocontouring algorithm such as marching cubes (LORENSEN; CLINE, 1987) are among the simplest solutions to this problem. Adaptive sampling variations of marching cubes can further reduce the error and capture more complex structures (REMACLE et al., 2006; SCHROEDER et al., 2006). However, resampling approaches introduce error and in some cases lead to memory increase due to the large number of lower-order elements needed to represent the original data.

Usually, methods for directly contouring higher-order data are formulated as a root finding or gradient descent problem. Due to the higher-order nature of the data, there is no closed-form solution for these formulations, and numerical methods must be used instead. Since these numerical methods are typically computationally expensive, they are often computed in a pre-processing step, which was applied in several mesh-extraction (REMACLE et al., 2006; SCHROEDER et al., 2005, 2006) and point-based algorithms (FIGUEIREDO et al., 1992; WITKIN; HECKBERT, 1994; VAN KOOTEN; VAN DEN BERGEN; TELEA, 2007; MEYER et al., 2007). Although interactive rendering is possible since the isosurface is computed during pre-processing time, changing isovalues requires the recomputation of the isosurface and hence the overall visualization might be no longer interactive. The ability to dynamically change isovalues is present in some ray casting or ray tracing isocontouring algorithms (WILEY et al., 2004; NELSON; KIRBY, 2006; KNOLL et al., 2009). However, the evaluation during the rendering step of such numerical methods leads to lower frame rates. It was observed that a hybrid approach, that quickly computes an object space approximation of the desired isosurface, together with a refinement step in image space, could result in a interactive isocontouring method, without the need for resampling.

We propose an algorithm for the interactive approximate contouring of multi cell higher-order data based on two phases. In the first phase coarsely seeded particles are guided by the gradient field for obtaining an initial sampling of the isosurface in object space. The second phase performs ray casting in the image space neighborhood of the initial samples. Since the neighborhood is small, the initial guesses for ray casting tend to be close to the isosurface, leading to fast root finding and thus efficient rendering. Since the object space phase affects the density of the samples, some artifacts can occur in the final rendering. Thus, we also propose a heuristic, based on dynamical systems theory, that adapts the neighborhood of the seeds in order to obtain a better coverage of the surface.

1.2 Extraction of Parallel Vectors Line-type Features

Several traditional features, such as isosurfaces and streamlines, can be described by algebraic or differential equations. This allows for the separation between a feature's description and its extraction procedure. However, for several other non-traditional features, this separation between description and extraction methods is not easy to find. Originally introduced by Roth and Peikert in (PEIKERT; ROTH, 1999), the parallel vectors operator

consists in a mathematical framework to identify line-type features in vector and scalar fields. Through the proposed formulation, several features types can be described analytically by the set of points where two distinct vector fields become parallel or anti-parallel.

In the original parallel vectors method, features are extracted from trilinearly interpolated data by finding intersection points with the faces of grid cells that are later connected by straight line segments. This method is local (solutions are found per cell), robust, and comparably fast. However, it might not be accurate enough since it approximates features by straight segments and suffers from topological ambiguity problems when connecting more than two intersections per cell. In contrast, the original feature flow field method (THEISEL; SEIDEL, 2003) provides a more accurate and smooth feature extraction. The feature flow field has been applied to parallel vectors feature extraction (THEISEL et al., 2005) using a subdivision method for finding seeds per cell in trilinearly interpolated data.

Our approach can be seen as an extension of (THEISEL et al., 2005) to higher-order data. The method looks for linear features in the flow, such as vortex cores, attachment-detachment lines, ridges, valleys, etc., described analytically by the parallel vectors operator. The parallel vectors formulation allow for the analytical description of several line-type features previously described only procedurally. Besides separating the feature description from its extraction procedure, the analytical form of the operator allows for its representation in inclusion form. Thus, the proposed method looks for interesting features using an adaptive space subdivision guided by the evaluation of the inclusion form of the parallel vectors operator. After the subdivision stage, root finding is used to precisely locate *seeds* (points resting on the feature lines). The placed seeds are then used as starting points in a feature flow field-based tracing stage that finally reconstructs the features.

1.3 Thesis Statements

I propose two different methods meant to allow efficient exploration of multi cell higher-order data containing affine mappings between reference and world space. The first method is a hybrid isocontouring technique that is based on the following statement:

Interactive approximate isocontouring of higher-order data is possible by splitting the computation workload between object and image spaces. In object space, a view independent approximation of the surface is computed. The surface approximation will reduce the cost associated to the surface refinement step computed in image space.

Two issues must be addressed to allow for the construction of a hybrid algorithm capable of efficiently extracting isosurfaces in this context. The first one is the development of the method that approximates the desired isosurface in object space. It must be efficient at the same time that its output serves as a good starting point for the subsequent refinement step. The second issue is related to the method used in image space, which must efficiently refine the isosurface based on the previously computed approximation. In Chapter 3 we present a hybrid algorithm based on two stages which accounts for the above mentioned requirements. It also discusses sampling issues related to the proposed approach, presenting a heuristic inspired by dynamic systems theory that reduces sampling artifacts.

The second method presented in this thesis is a technique for efficient parallel vectors line-type feature extraction from higher-order data that is based on the following statement:

The representation of the parallel vectors operator in its inclusion form allows for accurate and efficient approximation of the actual features inside higher-order data. Dataset processing in a per-element basis, instead of per-stage basis, reduces data read back overhead and lends itself well for parallel processing.

The issues raised by the above statement include: an analytic representation for the parallel vectors expression that is manageable, the choice of an inclusion form that can be efficiently implemented and evaluated, and an efficient strategy for data subdivision based on the evaluation of the inclusion form of the parallel vectors operator. Chapter 5 presents strategies that handle each one of the above mentioned requirements. The parallel vectors inclusion form is represented by an 5-term affine form derived from an affine arithmetic extension. The inclusion form of the parallel vectors is used to guide octree and quadtree-based subdivision that efficiently and robustly place seed points used for feature tracing. It is shown how the parallel vectors expression can be efficiently evaluated through the storage of the coefficients of its primitive components.

1.4 Results

The main contributions introduced in this thesis include:

- A hybrid method for the efficient isocontouring of higher-order data;
- A splatting resizing scheme based on dynamical systems theory;
- A method for line-type feature extraction from higher-order CFD data based on the evaluation of the inclusion form of the parallel vectors operator;
- An experimental analysis of an efficient subdivision strategy for parallel vectors line-type feature refinement in parallel architectures;
- A framework for efficient multi cell higher-order data evaluation in the GPU;
- A predictor corrector-based tracing scheme that efficiently reparameterize the field on the correcting plane;
- An alignment-based face grouping strategy that improves performance on SIMD architectures by reducing execution divergence among threads.

1.5 Organization of this Document

This document is organized as follows: chapter 2 presents a brief introduction to the scientific visualization field. Although not focused on the specific subject of this thesis, it gives context and presents related techniques in the field. At the end of the chapter the higher-order data format, and the description of the higher-order datasets used along this work, are introduced.

Chapter 3 presents the proposed isocontouring for higher-order data method. Initially, the state of the art in isocontouring of implicit and multi cell higher-order data is presented. Afterwards, the method is discussed in detail and results presented.

In order to facilitate the understanding of the feature extraction method presented in chapter 5, chapter 4 introduces the parallel vectors operator. The mathematical formulation is presented and it is shown how several existing line-type features can be rewritten in terms of the operator.

Chapter 5 presents the proposed method for line-type feature extraction method for higher-order data. A brief introduction to the concepts of inclusion arithmetic and feature flow fields is first presented. Afterwards, the feature extraction method together with implementation decisions are explained. At the end of the chapter results are presented together with discussions.

Finally, chapter 6 reviews the contributions of this thesis, pointing to directions for future work.

Appendices present complementary material related to the development of this thesis. Appendix A introduces the definition of some mathematical tools used along this thesis. Code and algorithms snippets related to the isocontouring method are presented in Appendix B. Finally, Appendix C contains the poster published at IEEE Visualization 2008.

2 VISUALIZATION OF CFD DATA

In this chapter we provide an overview of the field of scientific visualization and feature extraction with focus on CFD data analysis. We start by providing a brief description of the space discretization approaches used by traditional CFD methods and how the solution is represented and stored. The following sections present visualization and feature extraction techniques targeted to traditional CFD data. We introduce the higher-order data generated by recent CFD methods, pointing out the key features of this type of data. The chapter ends by presenting a detailed description of some higher-order datasets obtained from discontinuous Galerkin fluid simulations which were used to generate results for both methods presented along this thesis.

2.1 Visualization

Visualization is related to making visible those things that normally can not be seen. This is the case, for example, with fluid flow. Fluid is usually transparent (air, water, gas, etc.) and under normal circumstances it is not possible to visually inspect its behavior. Techniques that can be employed to allow the visualization of the fluid flow include, among others, the injection of smoke in wind tunnels or the placement of colored oil over friction surfaces. Visualization of other phenomena, such as the difference of temperature and shock waves, can be accomplished with techniques such as shadow graphs (SETTLES, 2010), an optical technique that captures optical inhomogeneities in transparent media.

In the context of computer science, visualization can be seen as a branch of computer graphics responsible for creating images or graphical representations about underlying data and processes. A description for the visualization field was given in 1987, in the National Science Foundation’s Visualization in Scientific Computing Workshop report (reprinted in (MCCORMICK; DEFANTI; BROWN, 1987)):

“Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science (...)”

The visualization field is usually broken into two almost disjoint branches: scientific visualization and information visualization. Scientific visualization usually deals with the graphical representation of *continuous* data, such as simulation data, e.g. fluid flow simulation, and data obtained from measurement devices, e.g. MRI or CT scans. Information

visualization, on the other hand, deals with discrete data usually of very high dimensionality such as stock market indexes, employee records, social networks, among others. Despite the above classification, sometimes the boundaries between these two branches can not be clearly defined.

2.2 Traditional CFD Data

Traditional CFD methods usually involve a double discretization. First, the physical domain is tessellated into several elements that together form a mesh. Second, the continuous function spaces (infinite dimensional) are replaced by finite expansions. After the discretizations, the solution is approximated for each dataset element. The following sections describe the types of tessellation normally used and how solutions are stored and retrieved.

2.2.1 Mesh Types

Initially, the physical domain Ω is tessellated into a collection of n elements, generating the mesh M_Ω according to Equation 2.1.

$$M_\Omega = \bigcup_{i=1}^n e_i, \quad (2.1)$$

where e_i is the i -th element.

Usually M_Ω is a compatible mesh, with elements intersecting only at the boundary faces, edges or vertices. The four basic element shapes commonly used for 3D domain tessellation are tetrahedra, pyramids, hexahedra, and prisms, as shown in Figure 2.1.

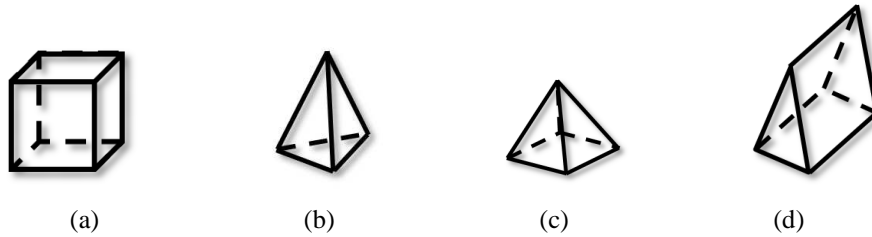


Figure 2.1: The four basic element shapes commonly used for 3D domain tessellation in traditional CFD: a) hexahedron, b) tetrahedron, c) pyramid and d) prism.

According to its overall structure, the assembled mesh can be classified as *structured* or *unstructured*. Structured meshes can be viewed as 2D or 3D arrays of elements represented by quadrilaterals or hexahedra, respectively. Since inter-element connectivity information is implicit, it can be compactly stored as a 2D or 3D array. Curved meshes are also considered as structured. Under the structured class, we find the regular meshes, which consists of regular samplings along each axis. The axis-alignment of the elements in structured meshes allows easier derivative computations (e.g. finite differences), and its regularity permits efficient handling (e.g. isocontouring and visualization algorithms such as (LORENSEN; CLINE, 1987; LACROUTE; LEVOY, 1994)).

Unstructured meshes can be composed by mixed shape elements and need to keep additional element connectivity information, which may affect memory management and

performance. However, these meshes better adapt to arbitrary domain boundaries, being used in cases where fluid flows around obstacles.

There exist also other classes of meshes, such as the *hybrid* and *hierarchical meshes*. While the former is composed by structured and unstructured mesh portions, the latter is organized in hierarchical structures, such as the adaptive mesh refinement (AMR) meshes. Figure 2.2 illustrates some of the above mentioned mesh types.

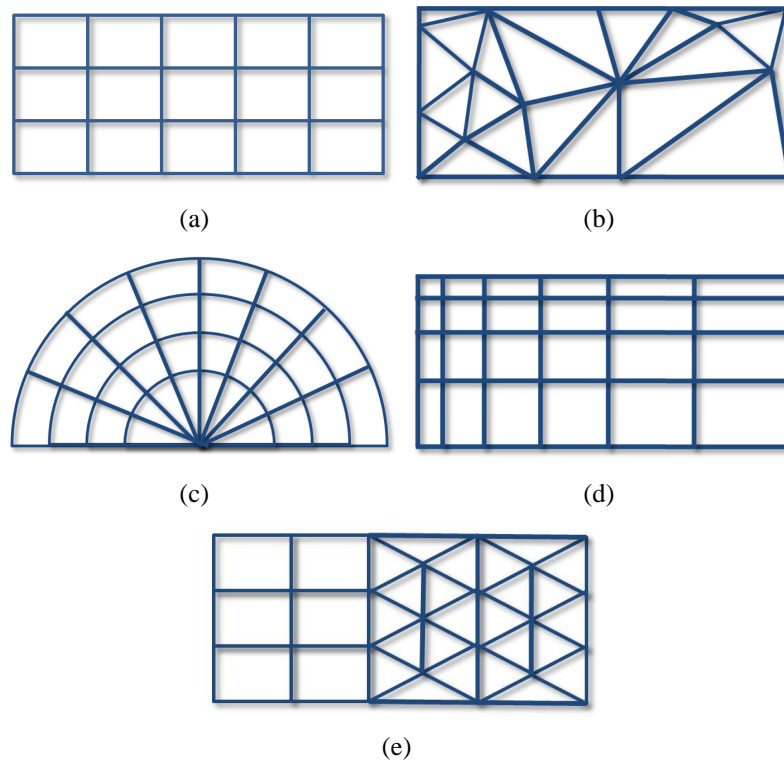


Figure 2.2: Mesh types: a) structured, b) unstructured, c) curvilinear, d) rectilinear and e) hybrid.

In the realm of computer-based flow simulation methods, there exist also *meshless* techniques such as the Smoothed-particle hydrodynamics (SPH)(LUCY, 1977; GINGOLD; MONAGHAN, 1977). According to this method, the fluid is represented by a set of particles. The properties for a given point in the simulation domain can be obtained by summing the *smoothed* contributions of all particles that lie within a certain radial distance.

2.2.2 Field Description and Interpolation

Solutions to CFD methods are represented by values associated to physical quantities and, depending on the type of quantity to be represented, different data types must be used. Physical quantities such as pressure, density or temperature associate a unique value to each point in space and can be represented by scalar fields. Other physical quantities, such as velocity or curl, associate a direction and a magnitude to each point in space. In these cases, such quantities are represented by vector fields, where vector length and coordinates indicate magnitude and direction, respectively.

Traditional CFD methods represent solutions through discrete samples stored at the element vertices. While, for scalar fields, these samples are represented by single scalar values, for vector fields the samples are represented by n -tuples of scalar values.

Real world quantities are continuous and it becomes necessary to reconstruct continuous functions from the discrete data generated by traditional CFD methods. The reconstruction of the continuous function can be obtained with approximation or interpolation techniques. While approximation techniques just "approximates" the discrete function, interpolation techniques generate continuous functions that match the exact function values at the sample points.

The simplest and more efficient interpolation technique is the nearest-neighbor interpolation. As the name suggests, unknown values are filled with values picked from the closest sample available. As a side effect of this approach, the reconstructed function will never be continuous regardless the sampling frequency. One possibility to force continuity between two samples is to require the function to vary linearly between them. This technique is known as linear interpolation, and guarantees C^0 continuity between the samples. When it is required some level of smoothness in the derivatives of the reconstructed function (C^i continuity, where i represents the corresponding i -th order derivative), higher-order interpolation schemes can be used at the cost of more expensive computations.

2.3 Visualization of Volumetric CFD Scalar Fields

CFD data is not always related to vector fields. Scalar fields representing pressures, density, heat, as well as scalar fields derived from vector fields, such as velocity and vorticity magnitude, can be used in flow analysis. A scalar field is a function f that maps every point in the n -dimensional domain to a scalar value (Equation 2.2).

$$f : \mathbb{R}^n \rightarrow \mathbb{R}. \quad (2.2)$$

When associated to units of measurement, scalar values represent quantities such as density, temperature, etc.

Several techniques have been developed for the visualization of scalar fields. These methods can be divided in two classes: direct and indirect methods. Direct methods extract the visualization directly from the scalar data. A classical example of a direct method is ray casting. Indirect methods rely on an intermediary representation for the scalar data, and this is usually the case with isocontouring methods, where the most notable example is the marching cubes algorithm. The following sections present a brief description of each visualization method.

2.3.1 Direct Volume Rendering

Direct volume rendering assumes that the volume data represents a participant medium composed by semi-transparent material. Final images are generated in a three stage process:

- Sampling
- Classification
- Compositing

Initially, rays emanating from the eye viewpoint are cast through each screen pixel, traveling along the volume and sampling it at intervals. This is the sampling stage. Afterwards, samples are classified. Classification usually consists of the assignment of color and opacity values to each sample. Color and opacity values are obtained from transfer

functions, which map every possible scalar value to pre-defined color and opacity values. The last stage computes the final pixel color by compositing the samples along the ray. Several approaches can be used for compositing, depending on the optical model used. Among the most common models, one can find maximum intensity point (MIP) (WALLIS et al., 1989), absorption-only and absorption-emission (BLINN, 1982; KAJIYA; VON HERZEN, 1984).

Direct volume rendering (DVR) is a good visualization approach for measured real-world data with noise and amorphous soft objects. A great advantage of the DVR approach is its intrinsic ability to render contextual information, not limited to a unique scalar value. The main drawback regarding DVR is the high computational cost demanded by the integration during the sampling stage. To mitigate this cost, several approaches have been developed to approximate the integral (ENGEL; KRAUS; ERTL, 2001; ROETTGER et al., 2003). Additionally, several acceleration techniques have been developed taking advantage of the regularity found in structured datasets. These techniques are usually separated in three classes, according to the space where they operate. Image space techniques, such as ray casting (DREBIN; CARPENTER; HANRAHAN, 1988; UPSON; KEELER, 1988), compute the final volume rendering entirely in image space. Object space techniques, such as *splatting* (WESTOVER, 1990; CRAWFIS; MAX, 1993), or *texture slicing* (CULLIP; NEUMANN, 1994; CABRAL; CAM; FORAN, 1994; GUAN; LIPES, 1994), take advantage of structures described in object space to accelerate volume rendering. Hybrid methods combine characteristics of both classes and an example of such method is the *shear-warp* (CAMERON; UNDERILL, 1992; YAGEL; KAUFMAN, 1992; SCHRÖDER; STOLL, 1992), which distributes the computation effort to compute the volume rendering over the object and image spaces.

Unstructured meshes present distinct properties that can not be handled by DVR methods targeted to structured data. Examples of these specificities are rays that can leave and re-enter the volume during tracing and sample locations that are not implicit. Thus, for unstructured meshes, additional techniques have been developed, such as *projected tetrahedra* (SHIRLEY; TUCHMAN, 1990) and the *HAVS* (CALLAHAN et al., 2005), among others.

2.3.2 Isocontouring

Given the 3-dimensional scalar field $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, an isosurface is a surface S that represents a set of points of constant value v (Equation 2.3).

$$S = \{\mathbf{x} | f(\mathbf{x}) = v\}. \quad (2.3)$$

Isocontouring is an alternate approach for scalar field visualization. Opposite to DVR, isocontouring methods usually rely on a intermediary representation of the desired structures, which is then used for the final visualization.

Originally developed for regular meshes (composed by hexahedral elements), the marching cubes (MC) algorithm (LORENSEN; CLINE, 1987) proceeds over the data, taking eight samples at a time and verifying whether they are crossed by the desired isosurface. This verification is based on the values of the eight samples. If two samples present different signs, it indicates that the element is crossed by the isosurface, and inverse linear interpolation is used to find the intersection points over the element edges. These intersection points are later connected and originate polygons inside each element. This computation is executed independently for each dataset element and the union of

generated polygons compose a triangle mesh that represents an approximation of the desired isosurface. MC must be executed every time the isovalue is changed. The method is robust and simple to implement. Since the surface extraction is executed independently for each element, it lends itself well for parallelization.

However, MC suffers from problems related to ambiguity and quality of the generated mesh. Ambiguity emanates from cases where the configuration of the intersection points found for a given element allow for the construction of topologically non-equivalent polygons. Several publications have presented alternatives that can reduce problems related to the topological ambiguity intrinsic to MC. Regarding mesh quality, it depends directly on the quality of each triangle of the mesh, and MC can generate very bad quality triangles under certain circumstances. Several publications have also attacked the mesh quality problem of MC (DIETRICH et al., 2009; SCHREINER; SCHEIDEGGER; SILVA, 2006; GARLAND; HECKBERT, 1997; CROSSNO; ANGEL, 1997; GAVRILIU et al., 2001).

MC was originally developed for the isocontouring of regularly sampled scalar fields, and could not be applied directly over unstructured meshes. Marching tetrahedra (PAYNE; TOGA, 1990) was a marching strategy developed for the isocontouring of unstructured tetrahedral meshes. Although not susceptible to the ambiguity problems encountered in original MC, the method can generate meshes topologically inconsistent with the underlying data. Regarding the contouring of implicit data, marching methods can be used given that a sampled version of the original data is available. However, disadvantages regarding this approach are the inclusion of error in data and increase in memory consumption needed to store the samples.

2.4 Feature Extraction

Increase in computational processing power has allowed for the development of sophisticated CFD simulation methods capable of generating more accurate results. Together with the adoption of time-dependent simulations, these new methods have led to substantial increase in the size of the resulting datasets. Increasingly larger amounts of data have posed new challenges to the scientific visualization community, which has seen feature-based visualization as an option to make the exploration of this data feasible.

By extracting only the relevant structures out of intricate data, feature-based techniques reduce the current data to manageable sizes, allowing for easier inspection and sometimes allowing for interactive data exploration.

There is not a formal definition regarding features. Usually, features represent patterns or structures which are relevant in a certain visualization context. It is important to note that features to be extracted must be "localizable" in the domain represented by the data.

The two main tasks regarding feature-based visualization are the appropriate definition of the feature and the design of an algorithm that is able to isolate features described according to that definition. The description of the features is related to their corresponding dimensionalities. Some features are 0-dimensional (point features), such as critical points in scalar or vector fields. Other features can be represented by 1-dimensional structures, such as vortex cores or detachment and attachment lines. Examples of 2-dimensional features include surfaces such as shock fronts or isosurfaces. In the case of closed surfaces, such as isosurfaces, they additionally delimit a 3-dimensional region in space. These regions can also be seen as 3-dimensional features, e.g., regions where some value is under or above a prescribed threshold.

Another point regarding features is the scope of their definitions. Some features are

only described in a global scope. This is the case for example with *watersheds*. Other features can be described completely in local terms, as for example critical points in scalar fields, which are represented by points where the first-order derivatives are zero. A higher dimensional feature described in local terms is the Eberly height ridge definition (EBERLY, 1996). For the feature extraction method presented in Chapter 5 of this thesis, we focus on line-type features described locally.

2.5 Higher-Order CFD Data

As already discussed, traditional CFD methods are based on point-based discrete data from which element-wise continuous data can be obtained using an arbitrarily chosen interpolation scheme. A commonly used approach is the linear interpolation since it is simple to compute (i.e. local to the element), does not introduce spurious oscillations in the reconstructed function, and guarantees at least C^0 continuity at the element boundaries.

An alternative to avoid the assumption of an arbitrary interpolation would be to provide the original interpolation scheme (possibly higher-order) along with the discrete data, thus allowing for the reconstruction of the exact cell-wise continuous function from the samples. This is the case, for example, with data generated with the discontinuous Galerkin method, for which the solution is given in terms of per-element multivariate higher-order polynomials.

The discontinuous Galerkin framework can be considered as a combination of the finite volume and finite element schemes. Similar to traditional CFD schemes, discontinuous Galerkin assumes a tessellation of the simulation domain into elements. As in finite volume schemes, discontinuous Galerkin allows for solution discontinuities at element boundaries, at the same time that, as in the finite element schemes, it allows the representation of the solution in terms of polynomial expansions. In this case, the coefficients of the polynomials indicate the possible degrees of freedom determined by the formulation of the governing equations.

The degree of the polynomials determine the accuracy order of the solution in space. To increase accuracy, it is sufficient to increase the order of the polynomials. This can be done adaptively, in regions where higher resolution is demanded. Alternatively, the combination of changes in the polynomial degrees can be coupled with local grid refinement to change the accuracy order in distinct regions of the dataset. The solution, represented by the set of element-wise polynomials, is discontinuous (C^0 continuous) at element boundaries. In the interior of each element the solution is C^p ($p \geq 1$) continuous, where p is the degree of the polynomial.

Most finite-element methods define local element operations, which is usually the case regarding discontinuous Galerkin approaches. Element-wise analytic functions describing the solution are defined in the corresponding element reference space. Thus, in order to query the solution value for a point \mathbf{p}_W in world space, one must first locate the element e_i containing the point, transform \mathbf{p}_W to the corresponding element reference space position \mathbf{p}_e using the inverse of the element's mapping function T_i and finally evaluate the solution polynomial F_i at position \mathbf{p}_e in the reference space (Figure 2.3).

The mapping function T , which can be non-linear, maps points from element reference space to the world space. Usually, this function can not be analytically inverted, and numerical approaches must be used instead. In this work we focus on multi cell higher-order data whose mapping function T represents only affine transformations that can be

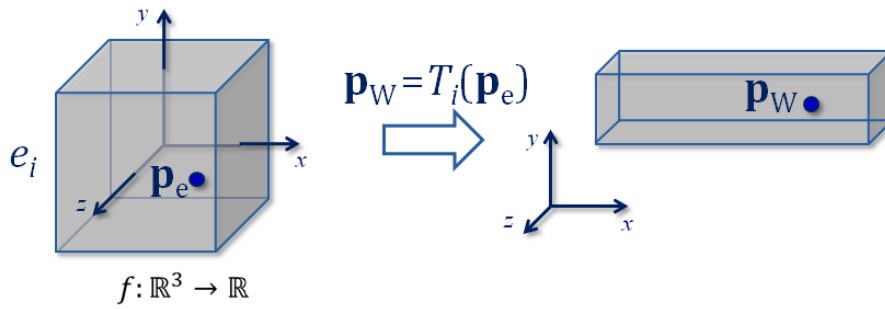


Figure 2.3: Element-wise mapping functions T_i transform a point \mathbf{p}_e in the reference space of the element e_i to a point \mathbf{p}_W in world space. In the general case, T_i may be non-linear, and can not be analytically inverted. In these cases, numerical approaches must be used to compute T_i^{-1} .

inverted analytically.

Higher-order data represented analytically presents several advantages with respect to visualization and feature extraction. The exact solution value for any point can be evaluated and prescribed accuracy checked. Symbolic manipulation of the analytical data representation allows for the exact evaluation of arbitrary higher-order derivatives. This is essential, for instance, in feature extraction approaches that rely on feature descriptions based on solution derivatives (e.g. vortex cores, streamlines, among others). An analytical solution description allows for its evaluation in inclusion form. Together with spatial subdivision methods, this allows to safely discard data portions that do not contain interesting data, focusing computational effort towards relevant data.

2.5.1 Discontinuous Galerkin Datasets

The higher-order data used in this work were generated by spacetime expansion discontinuous Galerkin simulations presented in (GASSNER; LÖRCHER; MUNZ, 2008). Datasets are represented by unstructured meshes whose elements assume polyhedral shapes. The solution is represented by element-wise polynomials of arbitrary degree, defined in each corresponding element's reference space. Each polynomial P is described in a monomial basis of the form

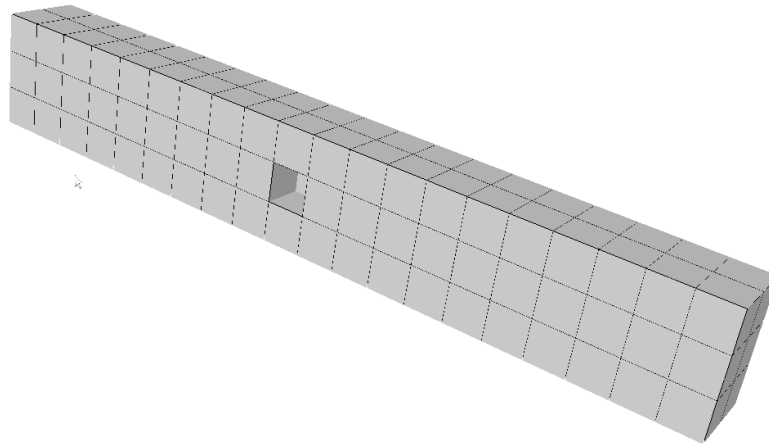
$$P(\mathbf{x}) = \sum_{i+j+k < n} c_{i,j,k} x^i y^j z^k, \quad (2.4)$$

where c is the coefficient of each monomial; x, y , and z are the independent variables in the element reference space; i, j , and k are the powers of each independent variable; and n is the degree of the polynomial.

Mapping functions, that map the element reference space to the world space, are defined separately for each element and consist entirely of translations.

The *shock channel* dataset was generated by a numerical simulation where a propagating shock with Mach number $Ma = 3$ hits a cubic obstacle positioned in the middle of the channel. As the shock advances, a lifted ballistic wave is formed, along with the shock's reflections on the channel walls. Low order numerical schemes face challenges when dealing with shocks and the resolution of their corresponding effects. In this simulation, the high degree of the polynomials (5 and 6, in this case), compensates for the very coarse resolution of the grid ($20 \times 2 \times 3$ hexahedral (cubic) cells). The obstacle is the size

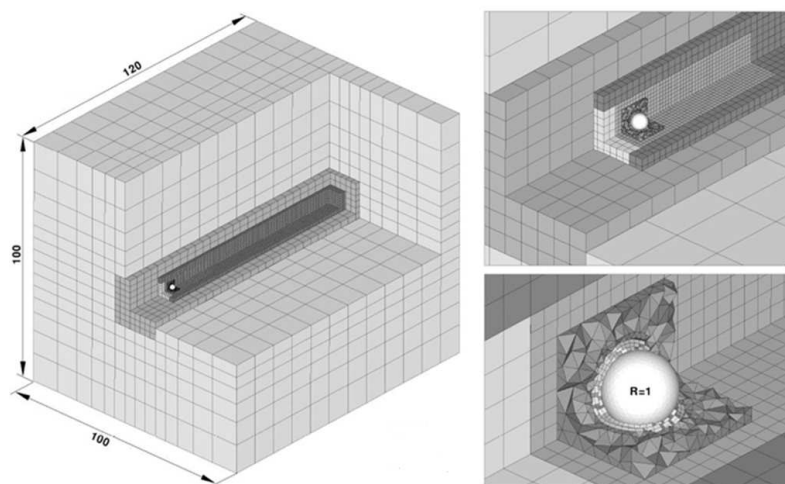
of one cell, resulting in a mesh with 119 cells in total. Figure 2.4 shows the structure of the *sphere* dataset.



(a)

Figure 2.4: *Shock channel* dataset structure. The mesh is composed of 119 hexahedral (cubic) elements. The solution is represented by degree 5 and 6 polynomials described in monomial basis. The mapping functions between reference and world spaces contain only translations.

The second dataset was generated by a hydrodynamical simulation that solves the compressible Navier-Stokes equations. With a Reynolds number of $Re = 300$ and a uniform flow with $Ma = 0.3$ initially set up, the simulation results show a von Karman vortex street roll-up. The unstructured mesh for the *sphere* dataset is composed by 34,535 polyhedral elements that assume the following shapes: hexahedron, tetrahedron, pyramid and prism. The degree of the solution polynomials is 3. Figure 2.5 shows the structure of the *sphere* dataset.



(a)

Figure 2.5: *Sphere* dataset structure. The mesh is composed of 34,535 polyhedral elements. Solution is represented by degree 3 polynomials described in monomial basis. The mapping functions between reference and world spaces contain only translations.

3 ISOCONTOURING HIGHER-ORDER SURFACES

In this chapter we describe the details of a method for interactive approximate isocontouring of higher-order data. The method is based on a two-phase hybrid rendering algorithm. In the first phase, coarsely seeded particles are guided by the gradient of the field for obtaining an initial sampling of the isosurface in object space. The second phase performs ray casting in the image space neighborhood of the initial samples. Since the neighborhood is small, the initial guesses tend to be close to the isosurface, leading to accelerated root finding and thus efficient rendering. The object space phase affects the density of the coarse samples on the isosurface, which can lead to holes in the final rendering and overdraw. Thus, we also propose a heuristic, based on dynamical systems theory, that adapts the neighborhood of the seeds in order to obtain a better coverage of the surface. Results for datasets from computational fluid dynamics are shown and performance measurements for our GPU implementation are given. The contents of this chapter were published as "Interactive Isocontouring of High-Order Surfaces" (PAGOT et al., 2010).

3.1 Isocontouring Higher-Order Data

There is a vast literature on isocontouring. We constrain the discussion below to techniques targeted to higher-order data.

Figueiredo *et al.* (FIGUEIREDO et al., 1992) proposed physically-based approaches for extracting triangle meshes from implicit surfaces. One method is particle-based while the second is based on a mass-spring system, and represents the first attempt on using particles to sample higher-order data. This work inspired Witkin and Heckbert (WITKIN; HECKBERT, 1994) to develop a point-based tool for the modeling and visualization of implicit surfaces. When used as a modeling tool, points represent handles for changing shapes. As a visualization tool, points are projected onto the surface and rendered as discs, with size and distribution adaptively computed according to the curvature of the surface. Meyer *et al.* (MEYER et al., 2007) proposed a technique for isosurfaces generated from higher-order finite element simulations that builds upon the approach of Witkin and Heckbert. Potential functions are used for particles repulsion, giving more control over particle distribution. This method handles cells with curved surfaces and allows for interactive visualization of a given isosurface. Interactive exploration of different isovalues is not viable since every isovalue change typically takes several minutes due to resampling.

Kooten *et al.* (VAN KOOTEN; VAN DEN BERGEN; TELEA, 2007) presented the first interactive particle-based method for implicit surface visualization that runs entirely on the GPU. The projection method is an adaptation of that by Witkin and Heckbert. Particle repulsion relies on a spatial-hash data structure that requires costly and frequent updates, thus preventing its use for rendering large datasets. Although not directly related

to isocontouring, Zhou and Garland (ZHOU; GARLAND, 2006) present a point-based approach for the direct volume rendering of higher-order tetrahedral data. Despite their effectiveness, point-based methods typically only provide a coarse representation for the isosurface, and accurate representations require a huge number of points. Haasdonk *et al.* (HAASDONK *et al.*, 2003) presented a multi-resolution rendering method for *hp*-adaptive data based on polynomial textures. Although effective, this method was designed only for 2D rendering. Schroeder *et al.* (SCHROEDER *et al.*, 2005) presented a mesh extraction method that explores critical points of basis functions to provide topological guarantees on the extracted mesh. Similar to other mesh extraction methods, it employs computationally expensive pre-processing to allow interactive exploration of arbitrary isovalues. Remacle *et al.* (REMACLE *et al.*, 2006) proposed a method for resampling higher-order data inspired by adaptive mesh refinement (AMR) methods. The original higher-order data is down-sampled to a lower-order representation which is suitable for lower-order visualization algorithms like Marching Cubes (LORENSEN; CLINE, 1987). The resampling error threshold can be adjusted, but low thresholds can lead to memory consumption explosion.

A competing strategy is to avoid pre-computation and compute isocontours during rendering. Nelson and Kirby (NELSON; KIRBY, 2006) presented a ray tracing-based method for the isocontouring of spectral/*hp*-adaptive data. The method works by projecting the higher-order functions onto each traced ray and computing the intersection with the isosurface from the resulting univariate function. Visualization error is carefully quantified and reduced. However, it typically takes several seconds to generate the final image, prohibiting interactive exploration of the data.

Knoll *et al.* (KNOLL *et al.*, 2009) presented an interactive interval arithmetic-based method for ray tracing of implicit surfaces on the GPU. It is currently one of the fastest ray tracers for implicits, presenting also robustness with respect to the root-finding process. However, the equations of the implicits must be converted to an inclusion-computable form, which increases the number of arithmetic operations needed to evaluate the equations. This is not a problem in the case of polynomials with only a few coefficients, as can be verified by their results. However, in the case of simulation data we usually have polynomials with hundreds of coefficients and thousands of polynomials to be evaluated for each frame, and the use of interval arithmetic may impact performance.

3.2 The Proposed Isocontouring Method

Interactive rendering rates in the proposed method are obtained by dividing the isocontouring workload between object and image space computations, each defined in a separate phase. The first phase, in object-space, generates an initial sampling of the isosurface by projecting particles (here called *seeds*) along the gradient field onto the surface. Only seeds successfully projected onto the surface are considered for further processing. Each projected seed generates a surface-tangent, seed-centered quadrilateral (quad) that covers the neighborhood of the seed. The second step uses the fragments generated by the rasterization of those quads as the initial points for a ray casting that refines the isosurface representation in image space. Figure 3.1 gives an overview of the method's pipeline.

To compute the initial isosurface approximation, we start by placing a set of seeds uniformly distributed inside each element and projecting them onto the surface. The projection affects the density of the seeds and thus the sampling quality. There are methods described in the literature that handle this problem by using repulsion/attraction or

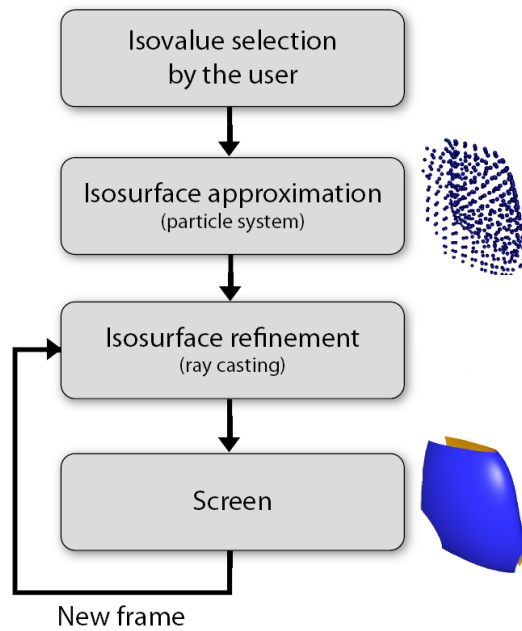


Figure 3.1: The proposed higher-order isocontouring method pipeline: An initial sampling of the isosurface is generated in object space by projecting coarsely seeded particles over the isosurface along the gradient field in object space. Each projected point (seed) generates a surface-tangent, seed-centered, quadrilateral that covers the neighborhood of the seeds. In the second stage the quadrilaterals are rasterized in image space and fragments used as the starting point for a ray casting that refines the final isosurface representation.

birth/kill of seeds in a pre-processing stage (WITKIN; HECKBERT, 1994; MEYER et al., 2007; VAN KOOTEN; VAN DEN BERGEN; TELEA, 2007). Despite their effectiveness, these procedures are computationally expensive, requiring the update of complex data structures that are hard to efficiently map to parallel architectures. Although our experimental results show that satisfactory quality images are obtained even without handling the sampling problem, we additionally propose a heuristic that helps in reducing the artifacts of the final image. This heuristic is implemented as a pre-processing stage that analyzes, under the dynamical systems theory perspective, the seeds behavior along the gradient field during projection. Differently from the previous methods that must compute the pre-processing for each isovalue, this pre-computation is executed only once for the entire dataset and does not depend on a specific isovalue. The next sections give an in depth explanation of each step of our algorithm.

3.3 Object Space Sampling

The task of finding an isosurface for a given polynomial data can be formulated as a root-finding problem. For higher-order polynomials, the lack of closed form solutions leads to the use of iterative numerical methods. Newton-Raphson (NR) is one of the best known methods for numerical root finding. Additionally, it presents good convergence rates when the starting point is already close to the desired solution. However, uniformly distributed seeds may not be close to the solution and NR may fail, leading to poor sam-

pling. One possible solution is to guide the seeds through the gradient field until they get closer to the isosurface and then apply NR to refine the projection. Since we just want to bring the seeds closer to the isosurface, we decided for guiding the points by means of integration. After the integration step, it is assumed that the seeds are closer to the desired isosurface and we start with NR iterations. Directional NR (LEVIN; BEN-ISRAEL, 2002) along the gradient is used, since it has the additional advantage of quadratic convergence. Equation 3.1 shows the formulation used for the NR iterations.

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{f(\mathbf{x}^k)}{\nabla f(\mathbf{x}^k) \cdot \nabla f(\mathbf{x}^k)} \nabla f(\mathbf{x}^k), \quad (3.1)$$

where k is the time step ($k = 0, 1, \dots$), \mathbf{x} is the point position in reference space, and f is the polynomial for the current element.

All computations are performed separately for each seed point since no neighborhood information is needed. Figure 3.2 gives an overview of the first phase.

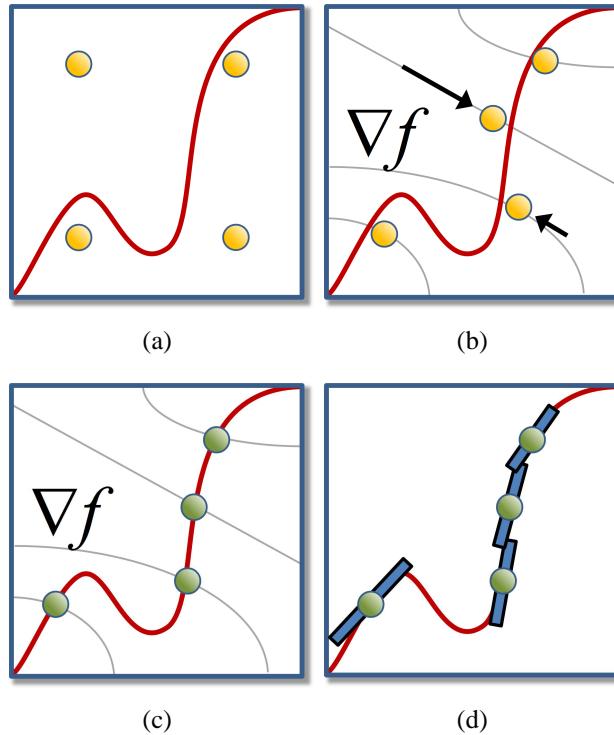


Figure 3.2: Isosurface approximation: (a) Seeds are initially distributed uniformly into each element. (b) Seeds are guided through integration steps along the gradient field towards the desired isosurface. (c) Seeds are projected onto the isosurface using directional NR. (d) Surface-tangent, seed-centered, quads are generated for each seed successfully projected.

3.4 Image Space Refinement

The input for the second phase of our method is a set of surface-tangent, seed-centered quads that together cover the isosurface. Points lying on these quads can be used as starting points for the ray casting that will refine the representation of the isosurface. These points are efficiently obtained through the rasterization of the quads in image space.

The fact that these points are already close to the isosurface implies that we can use the directional NR method along the ray direction, as show in Figure 3.3. Equation 3.2 shows the formulation for the directional NR used for the fragment projection:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{f(\mathbf{x}^k)}{\nabla f(\mathbf{x}^k) \cdot \mathbf{r}} \mathbf{r}, \quad (3.2)$$

where k is the time step ($k = 0, 1, \dots$), \mathbf{x} is the point position in reference space, f is the polynomial for the current element, and \mathbf{r} is the ray vector in reference space.

The generated ray and fragment coordinates are transformed into the reference coordinate system, where the ray-isosurface intersection computations are performed. After the maximum number of iterations is reached, if the fragment is not projected onto the isosurface (within an error tolerance bound) or it has been moved outside of the element, it is discarded. If the fragment is successfully projected, shading is computed and the corresponding pixel color is updated.

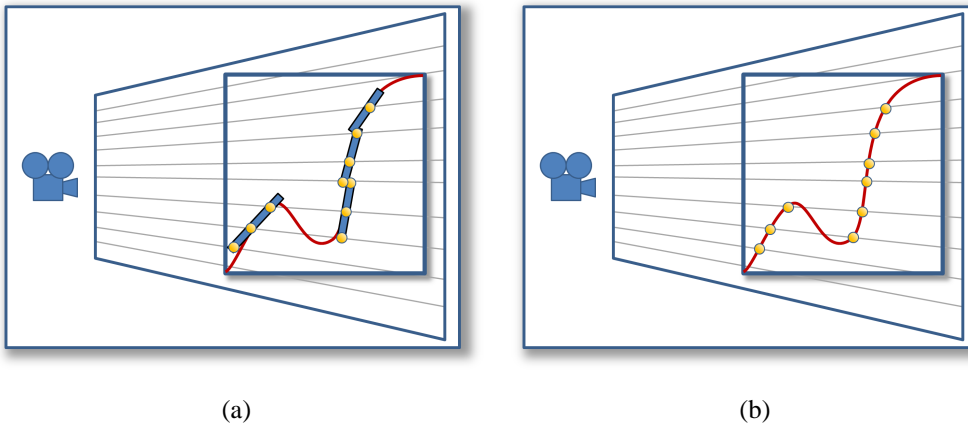


Figure 3.3: Isosurface refinement: (a) Quads are rasterized in image space. (b) Each fragment is projected against the isosurface through directional Newton-Raphson along the viewing rays.

3.5 Sampling Strategies

In both the object space sampling (Section 3.3) and the image-based refinement (Section 3.4) phases, an appropriate sampling for obtaining a complete isosurface representation is required. In some cases holes can result due to improper sampling. As a limit case of the object space sampling, infinitely dense seeding would guarantee a complete isosurface representation, but in practice, no guarantees about the resulting isosurface sampling can be given.

The reason is that in general no assumptions about the behavior of the gradient field can be made, and hence of mapping of seeds to the isosurface. For example, it is often not possible to estimate the maximum distance between two seeds when they are mapped to an isosurface of arbitrary isolevel. A limit case of the image space refinement is a single quad that spans the complete screen, which would be equivalent to a ray casting approach. This would lead to the typically low performance of these methods due to computationally expensive root finding along the viewing rays. Our method operates between the two mentioned limit cases. To motivate the parametrization and a sampling technique, we will first discuss the relevant issues and implications that arise in the context of sampling.

3.5.1 Sampling Issues

The proposed approach processes fragments independently. On one hand, it allows for efficient parallel processing of fragments. On the other hand, the lack of neighborhood information makes it impossible to detect, and to directly fix holes in the isosurface. Holes can be caused by different reasons:

1. root finding may fail, e.g. due to insufficient iteration count,
2. root finding may deliver a point on a nearby isosurface if the quad is not well aligned with the isosurface, i.e. the isosurface exhibits too high curvature with respect to the initial sampling,
3. quad sizes are insufficient to cover the whole surface.

The first case is hard to address explicitly because the efficiency of our method relies on synchronized root finding and hence constant iteration count (see Section 3.7.2). However, this is typically negligible. Since we use quads aligned with the tangent plane of the isosurface at each sampling seed, the second case depends only on the quality of the initial sampling and the last one on the quad sizes. The initial sampling has to be performed each time the isolevel is changed and therefore the exploration of isocontours requires high performance for this operation. This makes the use of expensive approaches not feasible for this step. Unfortunately, it is generally not possible to find a seed distribution for the object space sampling that leads to uniformly sampled isosurfaces for all possible isovalues. This makes non-uniform quad sizes for the image space refinement necessary.

Therefore, we propose a method for estimating sufficient quad sizes in Section 3.6 and place the seeds for the object space sampling regularly inside each element. One problem is that even comparably large quads cannot guarantee hole-free coverage of the isosurface because the sampling points can be arbitrarily distant due to the possible intricacy of the gradient field. Additionally, large quads typically lead to high overdraw and hence to lowered performance. All in all, a balance has to be found between the number of initial seeds for the object space sampling and the sizes of the quads.

For approximate results at high performance that can serve as a preview to more expensive but robust methods such as that by Knoll *et al.* (KNOLL et al., 2009), the sampling in object space can be parametrized and the size for the quads can be derived from that sampling as shown in Section 3.8. As an alternative, we now propose a method to estimate conservative quad sizes for individual quads.

3.6 Estimating Quad Sizes using FTLE

A method to determine quad sizes that guarantee a complete covering of the isosurfaces under the assumption of an appropriate object-space sampling can be derived from dynamical systems theory. The finite-time (or local) Lyapunov exponent (FTLE) was originally defined to measure the predictability of dynamical systems (LORENZ, 1965; HALLER, 2001). More precisely, it measures the exponential growth that a perturbation undergoes when it is advected for a finite time interval in a vector field. The flow map $\phi^T(\mathbf{x})$, which maps a sample point \mathbf{x} to its advected position $\phi^T(\mathbf{x})$ after advection time T , is the basis for the Cauchy-Green deformation tensor $\nabla\phi^T(\mathbf{x})$. Using

$$\Delta^T(\mathbf{x}) = (\nabla\phi^T(\mathbf{x}))^\top \nabla\phi^T(\mathbf{x}) \quad (3.3)$$

and λ_{\max} being the largest eigenvalue, leads to the finite-time Lyapunov exponent (HALLER, 2001):

$$\sigma^T(\mathbf{x}) = \frac{1}{|T|} \ln \sqrt{\lambda_{\max}(\Delta^T(\mathbf{x}))}. \quad (3.4)$$

For our application, we are only interested in the growth of the distance between two seeds as they are guided by the gradient field and therefore we omit the normalization by advection time and the logarithm in Equation 3.4. Additionally, since we look for points on the isosurfaces, i.e. the intersections of isosurfaces with gradient field lines starting at the seeds, the integration along the field lines shall not be limited by integration time (or length). Instead, it has to be limited by the prescribed isolevel l . We define $\phi^l(\mathbf{x})$ to be mapping the position \mathbf{x} to the isosurface of isolevel l along the corresponding gradient field line (the field line is stopped if a critical point of the gradient field is reached). Using Equation 3.4, this would lead to the separation factor

$$s^l(\mathbf{x}) = \sqrt{\lambda_{\max}(\Delta^l(\mathbf{x}))}. \quad (3.5)$$

We follow a direct approach for evaluating $s^l(\mathbf{x})$. To avoid numerical issues and to be able to exclude certain seeding neighbors $\mathbf{n} \in \mathcal{N}(\mathbf{x})$ of \mathbf{x} , as described below, the computation of $s^l(\mathbf{x})$ is not based on the gradient of $\phi^l(\mathbf{x})$, but it is calculated directly:

$$s^l(\mathbf{x}) = \max_{\mathbf{n} \in \mathcal{N}(\mathbf{x})} \frac{\|\phi^l(\mathbf{n}) - \phi^l(\mathbf{x})\|}{\|\mathbf{n} - \mathbf{x}\|}. \quad (3.6)$$

The quad size $d(\mathbf{x})$ at $\phi^l(\mathbf{x})$ is $s^l(\mathbf{x})$ times the corresponding seeding distance $\|\mathbf{n} - \mathbf{x}\|$. If the resolution of the object space sampling is appropriate, using these quad sizes guarantees a complete covering of the isosurface at level l . For obtaining quad sizes that are appropriate for any isolevel, the maximum separation $s(\mathbf{x})$ over all isolevels, computed in a preprocessing step, is used to determine the quad sizes:

$$s(\mathbf{x}) = \max_{l_{\min} \leq l \leq l_{\max}} s^l(\mathbf{x}). \quad (3.7)$$

This quantity relates to the FTLE maximum by Sadlo and Peikert (SADLO; PEIKERT, 2007). Unfortunately, as shown in Figure 3.4, neighboring seeds can get mapped to different isosurface parts, making the resulting quad sizes too conservative. This leads to unnecessary overdraw during rendering and hence lowers the overall performance. As a remedy, we propose a heuristic that reduces the overdraw caused by these cases. The idea is to exclude neighbors of \mathbf{x} from the computation of $s(\mathbf{x})$, that are not located in its vicinity on the same isosurface. There are several possible approaches for detecting if two points are adjacent on the same isosurface. The straightforward approach would be to compute the geodesic distance between the points along the isosurface. However, this is considered impractical and too expensive.

A criterion for testing if a mapped point $\phi^l(\mathbf{x})$ and its mapped neighbors $\phi^l(\mathbf{n}), \mathbf{n} \in \mathcal{N}(\mathbf{x})$, are located close to each other on the same isosurface can be motivated by the fact that the gradient is always aligned with the isosurface normal.

To test if a point and its neighbor are located on the same isosurface, the behavior of the gradient field is analyzed on the straight line connecting them. If the isosurface is planar between the two points, the line follows the isosurface and intersects the gradient field everywhere perpendicularly. If, on the other hand, the isosurface is non-planar between

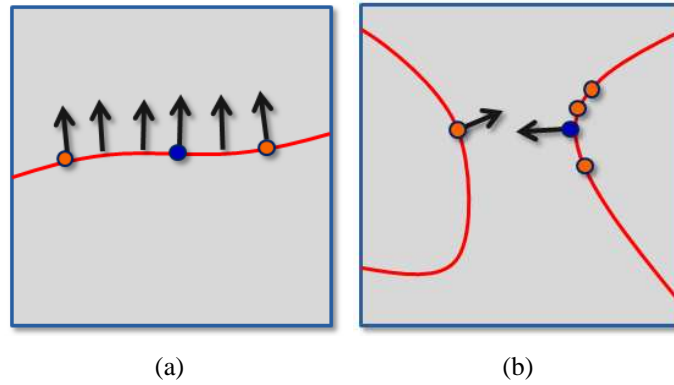


Figure 3.4: (a) Neighboring points projected over the same surface lead to more adequate (i.e. less conservative) estimates for quad sizes. (b) Neighboring points projected over disjoint surfaces may lead to very conservative quad estimates and reduced performance. A heuristic, based on the alignment of the normal vectors at the sampling points, reduces the influence of points laying on disjoint surfaces during quad size computation.

the points or if the points even lie on different isosurface regions, there are positions on the segment where the gradient is not perpendicular. We introduce the following measure a_{\min} for the minimum angle between the gradient and the segment \mathbf{s}_n going from \mathbf{x} to \mathbf{n} :

$$a_{\min} = \max_{t \in [0,1]} \frac{|\mathbf{s}_n \cdot \nabla s(\mathbf{x} + t\mathbf{s}_n)|}{\|\mathbf{s}_n\| \|\nabla s(\mathbf{x} + t\mathbf{s}_n)\|}. \quad (3.8)$$

The neighbor \mathbf{n} is excluded from the computation if a_{\min} exceeds a user-defined threshold. In practice, this threshold can be typically set to 0.5 for suppressing most of the erroneous neighbors in order to prevent too conservative quad sizes and reduce the performance loss due to overdraw.

3.7 Implementation

The proposed method was designed to take full advantage of parallel architectures. Although our current implementation runs on a single GPU, it could be easily distributed over a GPU cluster. We start by first describing the data structures used. Afterwards we go into the details of the implementation.

3.7.1 Data Storage

For the early element culling we use an interval-tree (CIGNONI et al., 1997) that stores the extrema of the scalar field for each element. This tree is stored and processed on the CPU side. The remaining data structures are stored on the GPU through the use of different buffers. Each time the user selects a different isovalue, we need to load the initial set of uniformly distributed seeds and compute the isosurface approximation. Thus we allocate two seed arrays on the GPU: one with the initial unprojected seeds and another to store the projected seeds. For each unprojected seed, its corresponding FTLE value is stored as its w coordinate. Actually, those seed arrays are implemented as vertex buffer objects (VBOs). We refer to VBO_{unproj} for the VBO with unprojected seeds and VBO_{quads} for the VBO that stores the quads. Polynomial, gradient, and element boundary data are read-only and are made available for GLSL through the use of bindable uniform

buffer objects (BUBOs). Since BUBOs are limited in size (usually 64KB) we chose to create a separate BUBO for each element and to bind them on demand.

3.7.2 GLSL Shaders

Both phases of the algorithm are implemented through a set of shader programs. The first step of the algorithm is implemented through a GLSL program composed of a vertex shader and a geometry shader. The vertex shader reads the VBO_{unproj} and projects the seeds onto the isosurface (Section 3.3). These seeds are streamed up to the geometry shader and used to generate surface tangent quads. The quads are resized accordingly to the FTLE value of the seed (Section 3.7.1). In order to record the generated quads into the VBO_{quads} we interrupt the pipeline just after the geometry shader through the use of an OpenGL extension called *Transform Feedback* (or *Stream-Out* in DirectX terminology). The GLSL program responsible for the second step is composed of a vertex and a fragment shader. The vertex shader just redirects the vertices of the quads to the rasterization stage. The fragment shader implements the core of the second step. It performs a ray casting using the input fragment as the starting point for the root-finding iterations (Section 3.4). As said before, fragments that are not successfully projected are discarded. However, it must be observed that fragments successfully projected onto the isosurface are only shaded according to the new position, without having their depth coordinate updated. This procedure greatly increases method's performance by allowing the use of the early-z test. A side effect of this approach is that some noise can appear on some regions of the isosurface. This happens because a fragment closer to the camera can be projected on a far surface, being thus wrongly shaded. This effect is reduced through the comparison between the angles of the polynomial gradient vectors at the fragment positions before and after projection. If the two angles differ above a certain value, it is assumed that the fragment jumped to another surface. In this case one has just to discard the fragment. In both steps, the shaders must compute several evaluations of the polynomials and their gradients. A shader capable of evaluating a general polynomial with an arbitrary degree would contain a loop that could not be unrolled by the compiler. To increase performance, we decided to keep several versions of these shaders, each one targeted to a specific polynomial degree. For the polynomial evaluation itself we use static expressions generated by a multivariate Horner scheme approach (CEBERIO; KREINOVICH, 2004). We decided to keep the number of integration and NR iterations static in order to reinforce thread synchronization, increasing performance.

3.7.3 Computation Flow

When the user chooses a desired isovalue, the elements containing the isovalue are selected. These elements are inserted in a separate list and arranged in front-to-back order through fast sorting, that does not have to be exact. This ordering is not necessary, but it helps the second step of the algorithm to avoid processing occluded fragments, increasing performance. The shaders corresponding to the polynomial degree of the current element are activated. The corresponding BUBO (containing the polynomial, gradient, and boundary data) for the current element is bound and the seeds are projected. After all seeds are projected, we start with the second step. Again, the element list is traversed, and now the ray casting shaders and corresponding BUBOs are bound. Fragments successfully projected onto the isosurface are recorded in the frame buffer.

3.8 Results

Results and performance measurements were obtained on a computer with an Intel Core 2 Quad 2.4 GHz processor, with 4 GB of RAM, Linux operating system, and NVidia GeForce 8800 GTX video card. To demonstrate the method we used synthetic data and two density datasets generated by Discontinuous Galerkin flow simulations. The synthetic dataset is composed by a regular grid containing 4^3 cubic elements. For each element, the scalar field is described by polynomials of degree 4. Data is C^4 continuous inside each element and at element boundaries. The two real higher-order discontinuous Galerkin datasets used for the method demonstration are described in Section 2.5.1.

3.8.1 Performance

For each phase of the algorithm a set of parameters can be adjusted. For the first phase these parameters are the number of initial seeds (n_s), number of integration steps (n_i), number of NR steps (n_{N1}) for seed projection, and the error for the NR projections (ϵ_1). For the second phase the parameters that can be adjusted are the number of NR steps (n_{N2}) for fragment projection, the scale factor for the quad size (s_q), and the error for the NR projections (ϵ_2). Although the large number of parameters may be confusing at first, they offer great flexibility to the user, allowing the tuning towards accuracy or interactivity. From our experience we observed also that in general one can explore both accuracy and performance by adjusting just a few of them, keeping the others at fixed values. For all presented measurements we used screen resolutions of 1024^2 pixels.

All renderings were made with early element culling active and with front-to-back ordering for the elements. Figure 3.5 shows performance measurements for the *shock channel* dataset. The fixed parameters, and their respective values, are: $n_s = 10^3$, $n_i = 50$, $n_{N1} = 10$, $\epsilon_1 = 10^{-3}$, $n_{N2} = 3$ and $\epsilon_2 = 10^{-4}$. These values were chosen through trial-and-error. The only variable parameter is s_q . Due to early element culling only 16 elements ($\approx 7.43\%$ of the total) were processed. The results of Figure 3.6 show performance measurements for the *sphere* dataset. The fixed parameters are: $n_s = 8^3$, $n_i = 50$, $n_{N1} = 10$, $\epsilon_1 = 10^{-3}$, $n_{N2} = 3$ and $\epsilon_2 = 10^{-4}$. These values were also chosen through trial-and-error. Again, the only variable parameter is s_q . Due to early element culling, only 3,781 elements ($\approx 9.13\%$ of the total) were processed.

Figure 3.8 shows how the method scales with respect to the number of elements. These measurements were made with degree 3 polynomials extracted from the *sphere* dataset. It can be seen that the rendering cost does not grow linearly. This can be explained by the use of the early-z test, which avoids the processing of fragments or even elements that are totally occluded. This behavior is reinforced since we use front-to-back ordering for the elements. We may still have some overdraw since we do not order the quads inside the element. However, from our experiments, this has not affected significantly the performance in the general case.

The method proposed by Knoll *et al.* (KNOLL *et al.*, 2009) may look, at first, as a competing approach. Despite the fact that their method is meant for the robust rendering of implicit, it is fast and could be applied separately to each element of our datasets. Figure 3.9 shows a comparison between our method and Knoll's for the rendering of a single element of the *shock channel* dataset. Visual inspection shows that our method is able to achieve higher framerates at better image quality. One reason for the difference in performance is probably related to the extra cost involved with the use of inclusion arithmetic in Knoll's method. Although the use of inclusion arithmetic guarantees error

bounds for the results, it tends to produce inferior results if larger error bounds are used. Since we are not primarily interested in the control of accuracy, we rely on the direct evaluation of the original polynomial representation, which is computationally cheaper and thus can be used in a previewing system.

Regarding the time spent to recompute the object space approximation at each iso-value change, for all our tests, they took less than 2 seconds. In the case of the *shock channel* dataset, it takes less than 1 second to recompute the seed projection. All measurements were made with $n_i = 50$ integration steps and $n_{N1} = 10$ Newton iterations.

3.8.2 Isocontouring Quality

As a previewing system the proposed method does not focus on high accuracy. Nevertheless, it is capable of delivering results of reasonable quality. Figure 3.10 shows a comparison between images generated by our method and the same images generated by POV Ray, a well known, tested, and freely available ray tracer (POVRAY, 2009). The discontinuities observed in these renderings are due the discontinuous Galerkin method used to compute the simulations. Our method handles continuous data properly, as can be seen in Figure 3.7.

Despite the reasonable quality of the images, one may want to have better visual quality for the isocontouring. In such cases the user can switch to a robust isocontouring system, as the one proposed by Knoll *et al.* (KNOLL *et al.*, 2009). However, we have also developed some heuristics in order to allow our method to generate better quality images. The quality of the final sampling depends on the relation between the number of seeds and the size of the quads. The exact relation between these two parameters is difficult to assess since the projection changes seed density. The FTLE-based heuristic tries to estimate “optimal” quad sizes that “work” for all isovalues considering an initial seeding given by the user. Since the FTLE does not take into account the topology of the isosurface, it can be too conservative, by considering distances between seeds in disjoint surfaces. For this case we adopt a test that estimates whether points are in disjoint surfaces (Section 3.6). Figure 3.11 illustrates the sampling problems related to bad seeding/quad size relation and how our heuristic resizes the splats according to that seeding, reducing effectively the artifacts in the final image. It also illustrates the effect of our disjoint-surfaces test on reducing the FTLE conservativeness. Despite the fact that it improves the quality of the final rendering, the FTLE makes the method less attractive for previewing since it reduces its performance.

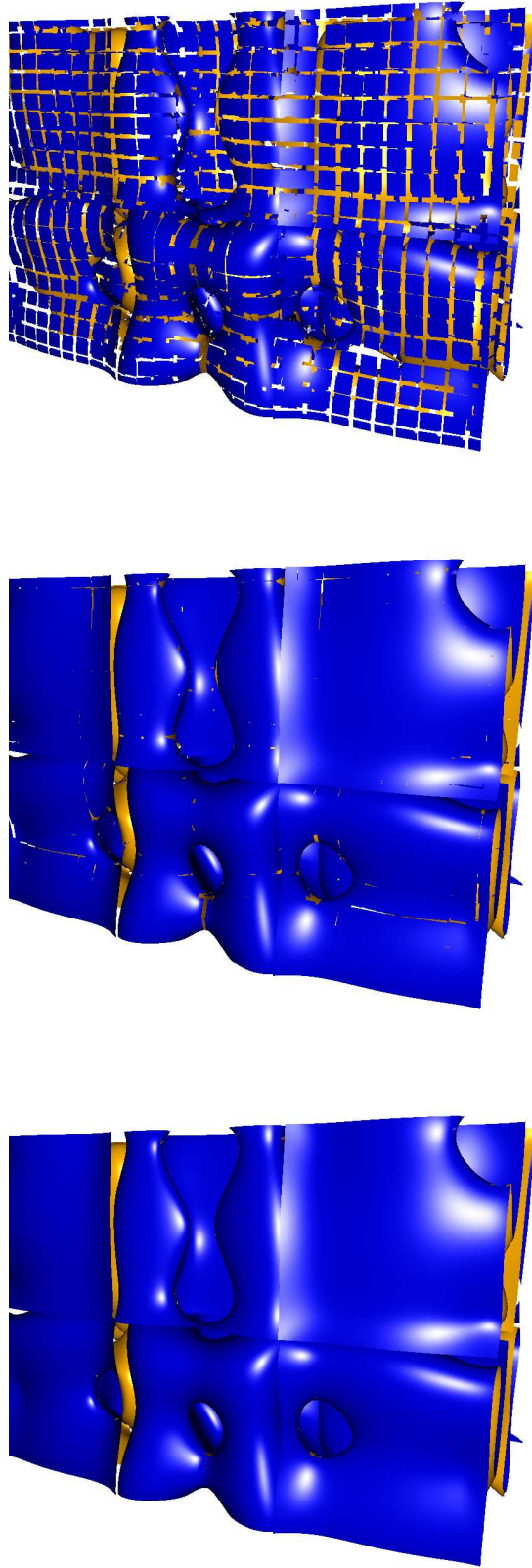


Figure 3.5: Renderings for the isovalue ≈ 2 of the *shock channel* dataset with varying s_q . From top to bottom, the s_q value and corresponding frames per second (fps) are: $s_q = 0.75$ and 45 fps, $s_q = 1.0$ and 39 fps, $s_q = 1.25$ and 29 fps.

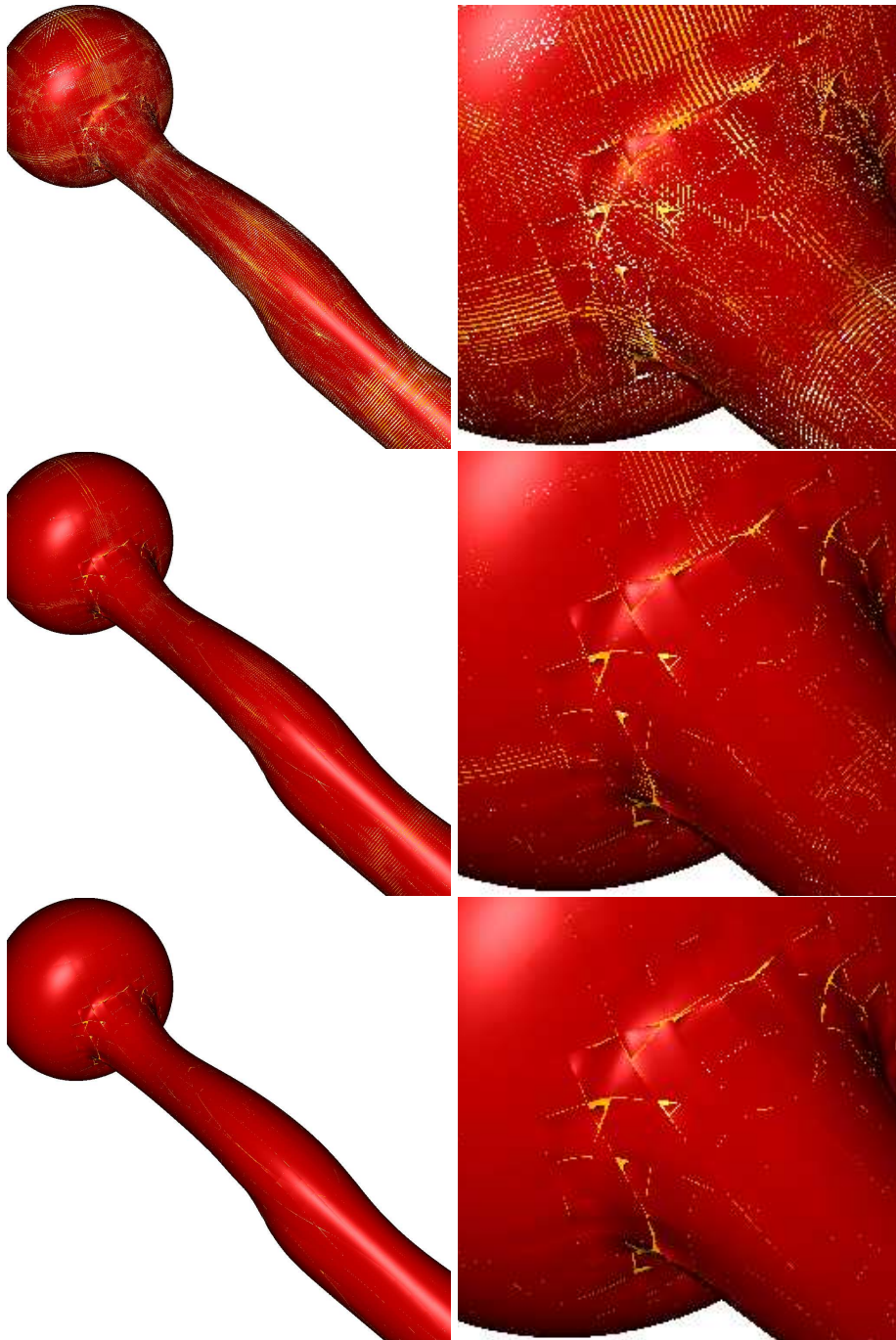


Figure 3.6: Renderings for the isovalue ≈ 0.9983 of the *sphere* dataset with varying s_q . From top to bottom, rendering and zoomed images with the following s_q value and corresponding frames per second (fps): $s_q = 0.5$ and 21 fps, $s_q = 0.75$ and 14 fps, $s_q = 1.0$ and 12.5 fps.

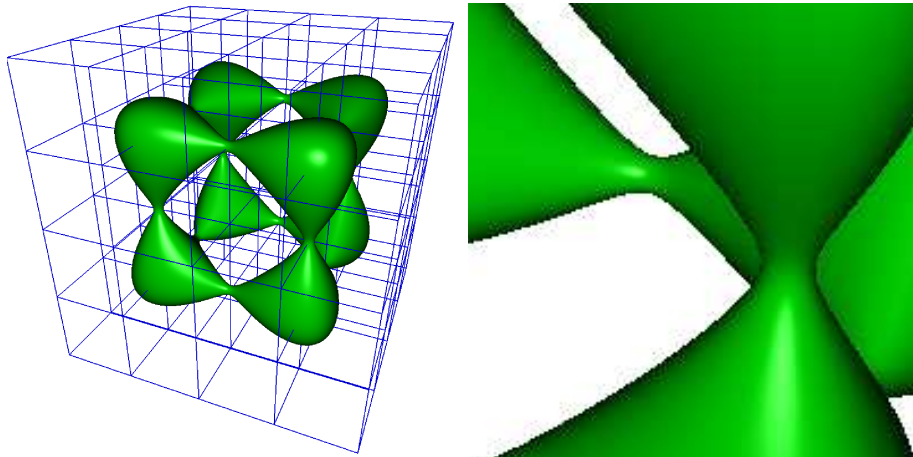


Figure 3.7: Rendering of Tangle dataset sampled with a grid of $4^3 = 64$ elements (left). The dataset is continuous inside and across elements. Zoomed image (right) shows how our method handles the continuity at the borders correctly. Settings for this rendering are: $n_s = 10^3$, $n_i = 50$, $n_{N1} = 10$, $\varepsilon_1 = 10^{-3}$, $n_{N2} = 3$, and $\varepsilon_2 = 10^{-4}$. The isovalue is ≈ -1.98 . Image was rendered at 1024^2 with ≈ 9.6 fps.

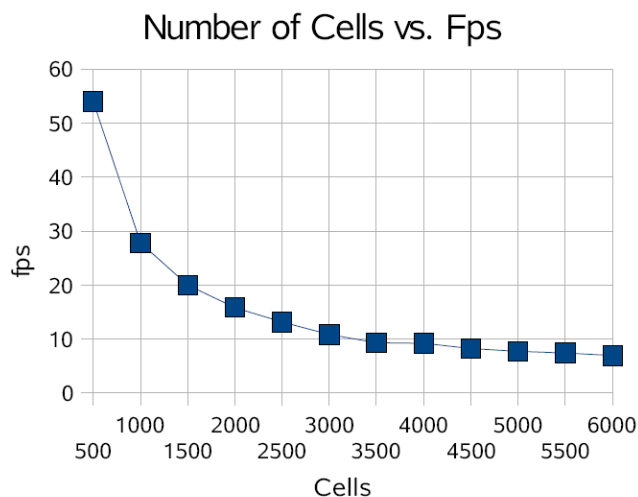


Figure 3.8: Performance (in frames per second, fps) vs. number of processed elements. The non-linearity can be explained by the use of the early-z test, which avoids the processing of occluded fragments.

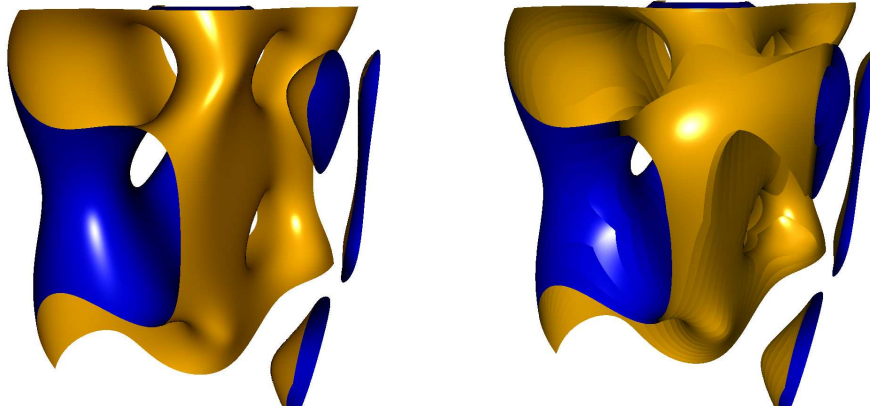


Figure 3.9: Comparison between our method and Knoll's scheme. (left) Polynomial of degree 5 rendered by our method at ≈ 30 fps. With the standard settings (depth = 10 and error = 0), Knoll's method rendered this image at ≈ 8 fps without visible artifacts. (right) After optimization of their parameters (depth = 4 and error = 0) to reach the best possible image quality at ≈ 30 fps. Both images rendered in a 1024^2 pixel screen.

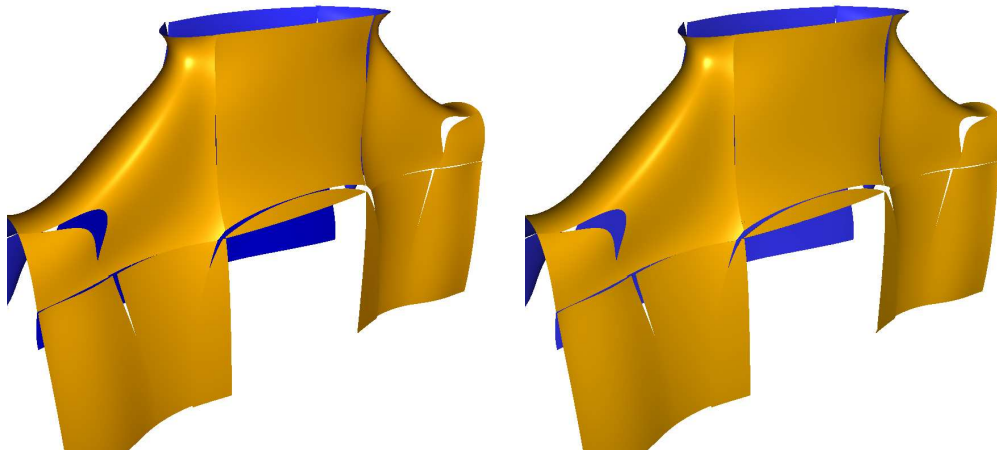


Figure 3.10: Comparison against POV Ray. (left) Image generated with our method, with the following settings: $n_s = 10^3$, $n_i = 50$, $n_{N1} = 10$, $\epsilon_1 = 10^{-3}$, $n_{N2} = 3$, $\epsilon_2 = 10^{-4}$, $s_q = 1.5$ and isovalue ≈ 5.8437 . This scene was rendered at 8.7fps. (right) Reference image generated from the same data and same isovalue with POV Ray. Both images rendered at 1024^2 pixels.

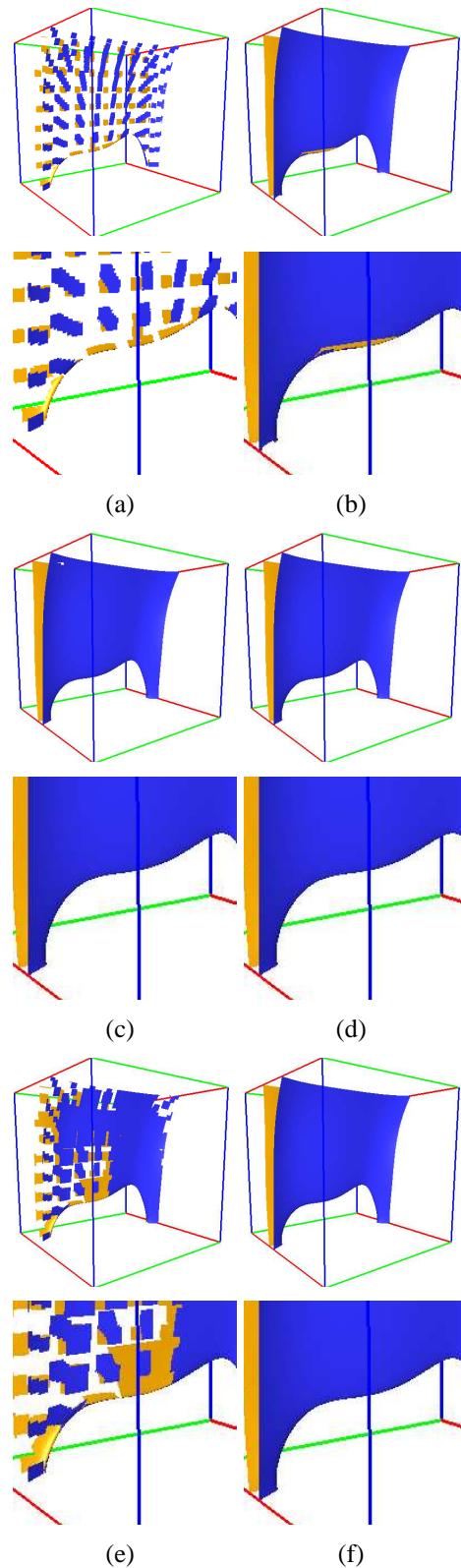


Figure 3.11: Sampling issues related to quad size: (a) Standard quads reduced to 30% of original size. (b) Sampling using standard quads: some artifacts can be seen due to poor sampling. (c) Quads scaled by the FTLE and reduced to 30% of their size. (d) Sampling obtained with quads resized by the FTLE approach. (e) Quads resized by our FTLE approach in conjunction with a heuristic that reduces its conservativeness, reduce in 30%. (f) Sampling obtained with our FTLE approach in conjunction with the heuristic.

4 THE PARALLEL VECTORS OPERATOR

In order to facilitate the understanding of the feature extraction method presented in Chapter 5, this chapter, based on the thesis by Martin Roth (ROTH, 2000) which originally introduces the parallel vectors operator, presents a brief description of the operator definition and properties. The following sections present the formalization, in terms of the parallel vectors operators, of several line-type features typically found in the context of flow analysis. A couple of techniques that can be used to extract line-type features described by linear vector fields are also presented and briefly discussed. A section is devoted to give an overview about current research on parallel vectors. The chapter finishes with a summary of several line-type parallel vectors features and properties.

4.1 Definition

Let \mathbf{u} and \mathbf{z} be two arbitrary n -dimensional vector fields. The set of points for which \mathbf{u} and \mathbf{z} are parallel or one of them is zero is defined as

$$S = \{\mathbf{x} : \mathbf{u}(\mathbf{x}) = 0\} \cup \{\mathbf{x} : \exists \lambda, \mathbf{z}(\mathbf{x}) = \lambda \mathbf{u}(\mathbf{x})\} \quad (4.1)$$

The first term in the equation above avoids $\lambda = \pm\infty$. The second term represents a system of n scalar equations on $n + 1$ unknowns $(x_1, \dots, x_n, \lambda)$. Solution points require that $\mathbf{z} \parallel \mathbf{u}$. If one assumes a local coordinate system with one axis parallel to $\mathbf{u}(\mathbf{x})$, $\mathbf{z}(\mathbf{x})$ will be parallel to $\mathbf{u}(\mathbf{x})$ only if its $n - 1$ components orthogonal to $\mathbf{u}(\mathbf{x})$ are zero. This restricts $n - 1$ degrees of freedom for the solution, which explains the fact that, for independent equations (non-degenerated), the solution is 1-dimensional.

This is the general parallel vectors definition which describes 1-dimensional features embedded in an n -dimensional domain. In this thesis we will restrict the discussion about parallel vectors feature extraction to one-dimensional features in three-dimensional domains.

To extract the parallel vectors feature lines in \mathbb{R}^3 of two continuous vector fields \mathbf{u} and \mathbf{z} defined on $[x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$, we first derive a third vector field \mathbf{w} such that:

$$\mathbf{w}(x, y, z) = \mathbf{u}(x, y, z) \times \mathbf{z}(x, y, z) = \begin{pmatrix} k(x, y, z) \\ l(x, y, z) \\ m(x, y, z) \end{pmatrix} \quad (4.2)$$

Parallel vectors feature lines are defined as the set of points where $\mathbf{w} = (0, 0, 0)^\top$. Equation 4.2 presents a system of three equations on three unknowns. However, these equations are not independent, and the solutions are represented by 1-dimensional closed

manifolds. In the seminal work by Peikert and Roth (PEIKERT; ROTH, 1999) they present some possibilities for the choice of \mathbf{u} and \mathbf{z} whose parallelism correspond to different feature types.

The next section discusses parallel vectors features filtering and classification. The following sections present the parallel vectors definitions for some prominent line-type features in the context of flow analysis. Additionally, a brief description of some techniques used to extract feature lines described by parallel vectors criterion is introduced. The last section presents a table which summarizes the parallel vectors features presented in this chapter.

4.2 Feature Classification and Filtering

As will be seen in the following sections, some feature types can be fully described by parallel vectors expressions. However, this is not the case in general. As shown before, solutions for the parallel vectors expressions are represented by 1-dimensional algebraic curves, which are closed, although most of the features are not representable by closed lines. In fact, usually only some of the requirements needed to define a feature can be expressed in terms of parallel vectors, whose solution represents a superset of points containing the desired structures (*raw features*). In order to fully describe and isolate the desired structures, additional criteria must be applied over the resulting raw features.

Some of these additional criteria classify points as being part of the feature or not. These are called *boolean filtering criteria* or *conditions*, and restrict, select or classify the set of points in the raw features.

Additionally, feature extraction based on local criteria, such as those amenable to a definition by the parallel vectors operator, are often susceptible to numerical noise and false positives. Therefore, it is usually inevitable to perform a subsequent *quality filtering* step. There are several possibilities for the criteria used for quality filtering in the context of line-type features according to (PEIKERT; ROTH, 1999).

Classification and filtering criteria may vary a lot depending on the feature type. Thus, we introduce some filtering and classifications criteria as we present the feature types and the corresponding parallel vectors expressions in the following sections.

4.3 Application Examples

This section presents examples of features that can be described in terms of the parallel vectors operator. First, *primitive* structures of scalar and vector fields, such as curvature and minimum/maximum lines, are described in terms of the introduced operator. Some existing line-type feature definitions, such as vortex cores and separation lines, are then described in terms of these primitive structures, giving rise to parallel vectors expressions which represent equivalent descriptions for these features.

4.3.1 Zero Curvature

Assuming that \mathbf{v} is a steady vector field, the acceleration \mathbf{a} of a particle moving along this field can be determined according to

$$\mathbf{a} = \frac{D\mathbf{v}}{Dt} = (\nabla\mathbf{v})\mathbf{v}, \quad (4.3)$$

as shown in Appendix A.3.

If the acceleration of the particle is aligned with the direction of \mathbf{v} , there are no perpendicular forces bending the current particle trajectory and that the particle is following a straight line. Thus, it is possible to define a parallel vectors expression which identifies locations where streamlines present zero curvature. The loci of zero curvature of a vector field is represented by the set of points where

$$\mathbf{v} \parallel (\nabla\mathbf{v})\mathbf{v}. \quad (4.4)$$

According to differential geometry, curvature vanishes when acceleration is aligned with velocity as shown in

$$\kappa = \frac{\mathbf{a} \times \mathbf{v}}{\|\mathbf{v}\|^3}, \quad (4.5)$$

where κ is the curvature of the streamline (Appendix A.1).

This formulation is important for the description of several line-type features such as vortex core by Sujudi and Haines (SUJUDI; HAIMES, 1995) and separation and attachment lines by Kenwright (KENWRIGHT, 1998; KENWRIGHT; HENZE; LEVIT, 1999) as will be shown next.

4.3.1.1 *Sujudi/Haines Criterion for Vortex Core Definition*

Vortices are prominent structures in the context of flow analysis, and their presence may be desired or not. They play an important role in combustion chambers, where their presence may increase the efficiency of the mixing process. On the other hand, they may cause a drag in aerodynamic profiles. There is no consensus in the definition of a vortex and several different models have been developed.

One of these definitions is the one by Sujudi and Haines, which proposes a rather intricate algorithm for vortex core definition and extraction. According to their definition, vortex core is the set of points where the *reduced velocity* is zero.

For a given point, the reduced velocity is computed by first computing the eigenvalues of the Jacobian of the velocity $(\nabla\mathbf{v})$. If the Jacobian presents only one real eigenvalue, the eigenvector \mathbf{e}_0 , corresponding to the unique real eigenvalue, is calculated. Finally, the reduced velocity is computed by subtracting, from the original velocity, its component aligned with the eigenvector \mathbf{e}_0 .

According to this formulation, the reduced velocity is zero only at locations where \mathbf{e}_0 is parallel to \mathbf{v} . Since \mathbf{e}_0 is an eigenvector of $\nabla\mathbf{v}$, then

$$(\nabla\mathbf{v})\mathbf{e}_0 = \lambda\mathbf{e}_0. \quad (4.6)$$

As explained before, for points on the vortex core we have $\mathbf{e}_0 = \mathbf{v}$. Thus, Equation 4.6 can be rewritten as

$$(\nabla\mathbf{v})\mathbf{v} = \lambda\mathbf{v}, \quad (4.7)$$

that is equivalently written as

$$\mathbf{v} \parallel (\nabla\mathbf{v})\mathbf{v}, \quad (4.8)$$

which is the same formulation for the zero curvature criterion presented in Equation 4.4

As is usual for parallel vectors formulations, the above expression extracts raw features, and post-processing becomes necessary to isolate only the desired structures. Additionally to the parallelism between velocity and the eigenvectors of the Jacobian (guaranteed in Equation 4.8), it is necessary that, at the feature points locations, the Jacobian presents only one real eigenvalue. Martin Roth (ROTH, 2000) suggests the computation of the discriminant D of the Jacobian at each extracted point in order to verify if the Jacobian presents only one real eigenvalue. If, for a given point $D > 0$, the Jacobian presents only one real eigenvalue and the point is part of the vortex core. Otherwise, the point is not part of the feature and must be discarded.

4.3.1.2 Attachment and Detachment Lines

Attachment and detachment lines are structures that indicate, respectively, where the flow abruptly moves away or returns to the surface. These are important structures in aerodynamics since they can increase drag and reduce lift, and therefore their occurrence should be reduced or completely eliminated.

Kenwright *et al.* (KENWRIGHT, 1998; KENWRIGHT; HENZE; LEVIT, 1999) proposed a formulation for attachment and detachment lines that is very close to the vortex core formulation of Sujudi and Haimes presented in the previous section.

As with the Sujudi and Haimes approach, this formulation is based on the alignment of the velocity with the eigenvectors of the Jacobian. Thus, a requirement for a point to be part of an attachment or detachment line is

$$\mathbf{v} \parallel (\nabla \mathbf{v}) \mathbf{v}, \quad (4.9)$$

which is exactly the same criterion as in Sujudi and Haime's approach (Equation 4.8).

Attachment and detachment lines are features defined on a 2-dimensional parameterization (*skin friction field*) of the original 3-dimensional vector field, leading to a 2×2 Jacobian. Feature lines extracted with the above formulation comprise a superset containing the desired features. In order to detect and classify the desired structures (as attachment and detachment lines), as well as to discard points that do not represent features, further analysis of the eigenvalues of the Jacobian must be performed.

4.3.2 Extremum Lines

Extremum lines are lines for which a scalar field s assumes a extremum value (minimum, maximum or saddle) on a plane perpendicular to a vector obtained from a vector field \mathbf{w} defined on the same domain:

$$\mathbf{w} \parallel \nabla s. \quad (4.10)$$

The above parallel vectors criterion defines a new class of linear structures, called *sectional extrema*, which can be used for the description of line-type features. This is the case, for instance, with the Banks and Singer (BANKS; SINGER, 1994) and Strawn, Kenwright and Ahmad (STRAWN; KENWRIGHT; AHMAD, 1998) vortex core definitions.

Banks and Singer (BANKS; SINGER, 1994) proposed a predictor-corrector scheme for the extraction of vortex core lines along the vorticity and pressure fields. According to their scheme, advancing moves are generated along the vorticity while the correction moves minimize the pressure on the plane perpendicular to the vorticity field. Thus, vortex cores should be approximations of vorticity lines.

For the algorithm to start, a set of seeds must be first placed. According to Roth (ROTH, 2000), the vortex core definition by Banks and Singer suggests that, at the vortex core, vorticity is approximately parallel to the pressure gradient

$$\nabla \times \mathbf{v} \parallel \nabla p, \quad (4.11)$$

where $\nabla \times \mathbf{v}$ is the vorticity and p the pressure field.

The above criterion includes points that are located on pressure maximum, minimum or saddle points. Thus, an additional filtering criterion must be employed to reject points that do not satisfy the minimum pressure criterion. This can be accomplished through the computation of the two eigenvalues of the Hessian of the pressure field parameterized on the perpendicular plane. If both eigenvalues are positive it means that the point is located at a pressure minimum and is part of the vortex core. If both eigenvalues are negative (pressure maximum) or present different signs (saddle points), the point is not on a vortex core and must be discarded.

Strawn, Kenwright and Ahmad (STRAWN; KENWRIGHT; AHMAD, 1998) also proposed a vortex core definition that can be expressed in terms of extremum lines. According to their definition, a vortex core is the set of points where the magnitude of the vorticity field \mathbf{w} reaches its maximum on the plane perpendicular to vorticity. Since the magnitude of the vorticity field $|\mathbf{w}|$ can be substituted by \mathbf{w}^2 , the equivalent description according to the parallel vectors is

$$\mathbf{w} \parallel \nabla(\mathbf{w}^2), \quad (4.12)$$

which can be equivalently written as

$$\mathbf{w} \parallel (\nabla \mathbf{w})^T \mathbf{w}, \quad (4.13)$$

according to the derivation in Appendix A.4.

4.3.3 Ridges and Valleys

Regions of minimum and maximum values are usually related to relevant structures, and a proper definition (and detection) of such structures play a key role in data behavior analysis. Among such structures, there are ridges and valleys, which have been used for the extraction of several different types of interesting structures, ranging from topographic features to multidimensional structures in fluid flow.

Ridges and valleys are important structures in topography and their mathematical formalization as curves were already pursued early in the nineteenth century by De Saint-Venant (SAINT-VENANT, 1852) and Breton de Champ (CHAMP, 1854). Figure 4.1 illustrates geomorphological ridges and valleys extracted from a 2-dimensional height field.

Two main approaches have been used for the definition of ridges and valleys. One is based on the concept of watershed (and watercourse) and the other based on the concept of height ridge.

According to the watershed based definition, ridges are extracted as slope lines that depart from saddle points on the height fields, and are traced in positive and negative gradient direction. Despite the usefulness of this type of description, watershed based definition is global in nature, involving the processing of the whole dataset in order to locate the desired features.

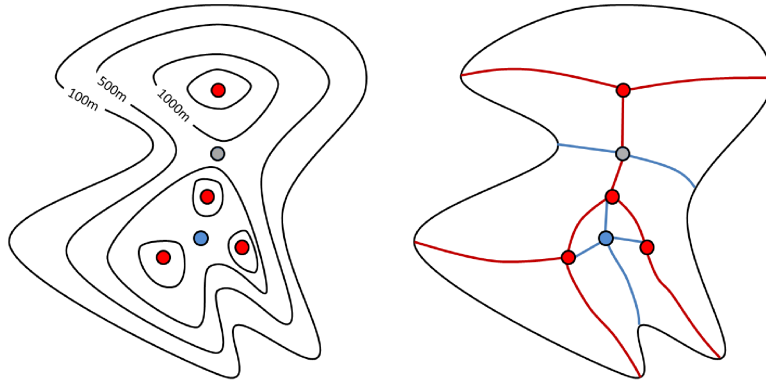


Figure 4.1: Red points: maximum. Blue points: minimum. Gray points: saddle. Left: height field contour map. Right: corresponding height maps ridge (red) and valley lines (blue).

The height ridge approach, on the other hand, represent features not as slope lines, but as solution manifolds of algebraic equations, which are computed only from local properties obtained from the height field and its corresponding derivatives. The most prominent definition of this type is the one by Eberly (EBERLY, 1996).

Due the local nature of the parallel vectors operator, we restrict the discussion of ridges and valleys extraction to techniques that operate locally. More specifically, we concentrate on the one proposed by Eberly, for which an equivalent parallel vectors expression is presented.

According to Eberly (EBERLY, 1996):

“The height ridge definition uses the heuristic that a ridge point should be a point for which the function has a local maximum in the direction for which the graph has the largest concavity (assuming the normal direction is in the positive z direction).”

More formally, a k -dimensional height ridge embedded in \mathbb{R}^n can be defined as the set of points of the scalar field $f : \mathbb{R}^n \rightarrow \mathbb{R}$ where

$$\frac{\partial f}{\partial y_1} = \dots = \frac{\partial f}{\partial y_{n-k}} = 0 \quad (4.14)$$

$$\frac{\partial^2 f}{\partial y_1^2}, \dots, \frac{\partial^2 f}{\partial y_{n-k}^2} \leq 0 \quad (4.15)$$

in the local coordinate frame y_1, \dots, y_n defined by the eigenvectors of the Hessian matrix $\mathbf{H} = \frac{\partial^2 f}{\partial x_i \partial x_j}$. The ordering of the eigenvalues λ_i is defined as follows:

$$\lambda_1 \leq \dots \leq \lambda_n. \quad (4.16)$$

The height ridge counterpart, *height valley*, can be extracted as the height ridges of the inverted function $-f$.

According to (ROTH, 2000), ridge (or valley) lines are the set of points where the slope is locally minimal compared to points located on a line perpendicular to the elevation gradient. Thus, the definition requires a minimum of the magnitude of the gradient

\mathbf{g} of the scalar field f in the plane perpendicular to \mathbf{g} . Ridge and valleys are therefore the minimum lines of \mathbf{g} and can be expressed according to the extremum line formulation presented in Section 4.3.2:

$$\mathbf{g} \parallel (\nabla \mathbf{g})^T \mathbf{g}. \quad (4.17)$$

The vector field \mathbf{g} used in the formulation above is the gradient of a scalar field and $\nabla \mathbf{g}$ represents its Hessian (\mathbf{H}), which is always symmetric. Since the $\mathbf{H} = \mathbf{H}^T$, Equation 4.17 can be simplified and rewritten as

$$\mathbf{g} \parallel \mathbf{H}\mathbf{g}. \quad (4.18)$$

The above formulation would be equivalent to the first part of Eberly's definition (Equation 4.14) except for the fact that, for points laying on features lines, it does not restrict the alignment of \mathbf{g} with the maximal (or minimal) eigenvectors of the \mathbf{H} . In fact, Equation 4.18 extracts all points for which \mathbf{g} is aligned to *any* of the current eigenvectors of \mathbf{H} (PEIKERT; SADLO, 2008). Isolation of desired features can be accomplished by testing the alignment of \mathbf{g} with the eigenvectors of \mathbf{H} at each extracted point. Points where \mathbf{g} is aligned to the maximal/minimal eigenvectors of \mathbf{H} are kept. Otherwise, they are discarded.

At this stage we have a set of points that represents the union of all extracted ridges and valleys, but we are not yet able to determine to which feature type each point pertains. As a final step, points are classified as being part of a ridge or a valley line according to Equation 4.15.

4.3.4 Other Uses of the Parallel Vectors Operator

Levy *et al.* (LEVY; DEGANI; SEGINER, 1990) defined vortex core as the set of points where normalized helicity approaches -1 or $+1$. Normalized helicity is the cosine of the angle between velocity \mathbf{v} and vorticity $\nabla \times \mathbf{v}$. If the vectors are parallel and point in the same direction (despite their magnitude), the cosine approaches $+1$. If the vectors are anti-parallel (point in opposite directions) the cosine of the angle between them approaches -1 . Thus, a vortex core is defined as the set of points where velocity is parallel to vorticity

$$\mathbf{v} \parallel \nabla \times \mathbf{v}, \quad (4.19)$$

which represents the equivalent criterion for extracting vortex cores according to the parallel vectors formulation.

Another example of the use of parallel vectors in vortex core definition is related to curved vortex cores. Linear vector fields can model only vortices whose core is a straight line, and most of the existing vortex core extraction methods (usually based on first derivatives) assume that the vector field is locally linear. However, curved and bended vortices are found in practical data very often, mainly in the field of turbo-machinery, where the fluid is usually constrained to curved channels. Using the method by Sujudi and Haimés (SUJUDI; HAIMES, 1995) as the starting point, Roth and Peikert (ROTH; PEIKERT, 1998) developed a method for the extraction of idealized curved vortex core, which presented satisfactory results when applied over real simulation data. The method makes use of second derivatives of the velocity field, and one of the requirements for a point to be part of the bended vortex core is the parallelism shown below

$$\mathbf{v} \parallel (\nabla \mathbf{a}) \mathbf{v}, \quad (4.20)$$

where \mathbf{v} expresses velocity and \mathbf{a} the acceleration.

This formulation identifies a superset of points that contains the desired features, and further processing is necessary to isolate only the desired structures. For each point satisfying Equation 4.20, it must be assured that the velocity gradient presents only one real eigenvalue. This can be accomplished by the computation of the discriminant D of the velocity gradient at each extracted point, where $D > 0$ for the points lying on the feature.

4.4 Extracting Features

According to the original proposal (PEIKERT; ROTH, 1999), for 1-dimensional features in \mathbb{R}^2 , the extraction of the zero-isolines of the cross product between the two vector fields can be used. However, the extraction in higher dimensional spaces (\mathbb{R}^n , where $n \geq 3$), the choice of the method is not obvious, and several possibilities can be considered. Thus, four possible methods are suggested for the extraction of parallel vectors line-type features. While the first three methods are concerned with finding intersections of the features with faces of the data grid, the fourth is a continuation-based line tracing approach that, used in combination with any of the three previously suggested methods, enables the extraction of topologically correct line features. Following is a brief description of each one of these methods.

4.4.1 Isosurface Based Approach

The parallel vectors feature lines can be seen as the intersection of the 0-level isosurfaces of three distinct scalar fields, as shown in Equation 4.2. Thus, one possible approach to extract the feature line is to locate two of the three possible isosurfaces and compute the curve generated by their intersections. This can be accomplished with a marching lines algorithm, such as the one proposed by (THIRION; GOURDON, 1996). Later on, at each point of the extracted curve, the third scalar field is tested for a zero. Points on the extracted curve that successfully evaluate to zero for the third isosurface can be considered as feature points.

This approach might suffer from some problems. The curve extracted in the first step is just a piecewise linear approximation of the actual curve. Thus, the test between the extracted curve and the third scalar field should account for, probably, large tolerance values that may lead to inaccurate feature lines and discontinuities. Additionally, isosurfaces forming sharp angles may generate numerically unstable tracing vectors.

4.4.2 Iteration on Cell Faces

Another possibility for the feature extraction is to use an iterative method on the cell faces to find the corresponding intersection points with the feature lines. These intersection points serve as *seeds* for a post feature tracing stage.

According to this method, the parallel vectors expression to be zeroed is

$$\mathbf{w} = \mathbf{u} \times \mathbf{z}, \quad (4.21)$$

and the vector valued function \mathbf{w} is defined within grid cells.

Before it starts, for each cell face (be it a quadrilateral or a triangle) a point is placed at its center. This point serve as the starting point for several Newton iterations restricted to

the supporting plane of the cell face. During the iterations, if the point is moved beyond the corresponding face limits, it is positioned back at the face border. If the value of \mathbf{w}^2 becomes smaller than a prescribed error tolerance, the iteration stops and the current seed location is stored.

Despite the accuracy regarding seed placements, the algorithm just delivers a set of seeds which are not related to one another. The reconstruction of the feature, by means of seed connection, must be accomplished using heuristics. Cases such as cells containing zero or two intersection points are trivially handled. However, problems emerge when the cell contains only one or more than two intersection points. In these cases several heuristics can be used, including seed proximity, direction of the tangent vector at seed positions, etc.

4.4.3 Analytic Solution at Triangular Faces

This algorithm explores the existence of an analytic solution for linear feature intersections on triangular cell faces. As the algorithm presented in Section 4.4.2, it provides a set of non-connected seeds which must be connected afterwards. The algorithm can be applied to quadrilateral faces since they can be broken into triangles. It explores the fact that a 3-dimensional linear vector field \mathbf{u} can be described in terms of the triangle local coordinate system through a matrix \mathbf{U} :

$$\mathbf{u} = \mathbf{U}\mathbf{x}, \quad (4.22)$$

where \mathbf{U} is a 3×3 matrix as is \mathbf{Z} and $\mathbf{x} = [s, t, 1]^T$ is a point in the triangle local coordinate system.

Two linear vector fields are parallel when

$$\mathbf{U}\mathbf{x} = \lambda\mathbf{Z}\mathbf{x}. \quad (4.23)$$

If \mathbf{Z} is invertible, the problem of finding the locus where both fields are parallel reduces to an eigenvector problem for a 3×3 matrix

$$\mathbf{Z}^{-1}\mathbf{U}\mathbf{x} = \lambda\mathbf{x} \quad (4.24)$$

If \mathbf{Z} is not invertible, but \mathbf{U} is, the expression can be swapped. If both vector fields are not invertible, then no solution exists.

4.4.4 Curve-following Methods

As observed with the methods presented in Sections 4.4.2 and 4.4.3, heuristics must be used during the connection of the seeds, and it can be the case that the connections generated do not represent the actual feature topology.

An alternative to reduce topological problems related to incorrect seed connections is to use curve-following algorithms to trace the features from the previously extracted seeds (BANKS; SINGER, 1994; HARALICK; SHAPIRO, 1992). Although not capable to guarantee topologically correct lines due the limited floating point precision available in computers, sufficiently small tracing steps usually lead to acceptable quality results. Additionally, since it is known that the value of the parallel vectors expression at a point lying on the feature line is zero for all its components (Section 4.1), one can also advocate the use of predictor-corrector-based tracing schemes for higher-quality feature tracing.

4.5 State of the Art in Parallel Vectors

Current research on parallel vectors feature extraction is mainly focused on more accurate feature tracing and extraction in higher dimensions. Sukharev *et al.* (SUKHAREV; ZHENG; PANG, 2006) propose a new line-type feature integration method based on analytical expressions for feature line tangents that allows for topologically correct extraction of smooth feature curves from linear data. The authors discuss two methods for seed finding that simultaneously evaluate the equations that compose the parallel vectors expression, respecting their linear dependence. Gelder and Pang (VAN GELDER; PANG, 2009), inspired by Banks and Singer (Section 4.3.2), propose *PVsolve*, an accurate parallel vectors line-type feature tracing method based on a predictor-corrector scheme. A stable feature flow field (WEINKAUF *et al.*, 2010), which guarantees converging behavior around feature lines, was also proposed. Although capable of accurately extracting features using larger step sizes than previously published tracing methods, these methods assume that seed points are given.

Several works discuss feature extraction in higher dimensional spaces. Bauer *et al.* (BAUER; PEIKERT, 2002) propose a method for parallel vectors feature extraction using scale-space techniques. Theisel *et al.* (THEISEL *et al.*, 2005) discuss the extraction of parallel vectors vortex core surfaces in time-dependent vector fields using a robust method for seed detection in linearly interpolated data. Their method is based on a recursive subdivision of the element's face and interior to ensure the detection of at least one seed on each feature line. The authors also present a general derivation for the FFF, which represents features of a general parallel vectors expression as streamlines. Our method extends seed finding and subdivision approaches for robust feature extraction in higher-order data.

Feature extraction from volumetric multi-cell higher-order data is discussed in related problems of isocontouring and direct volume rendering (MEYER *et al.*, 2007; NELSON; KIRBY, 2006; REMACLE *et al.*, 2006; SCHROEDER *et al.*, 2006; WALFISH, 2007; ÜFFINGER; FREY; ERTL, 2010).

Schindler *et al.* (SCHINDLER *et al.*, 2009) propose, as far as we know, the first attempt to extract parallel vectors features from higher-order data. However, their method is tailored to the specific case of SPH data, which does not provide cells. The work presented in this thesis is, to the best of our knowledge, the first attempt to handle the extraction of parallel vectors line-type feature lines from multi-cell higher-order data. For numerical accuracy, several works use interval arithmetic (IA) (MOORE, 1966) or its variants such as Affine Arithmetic (AA) (COMBA; STOLFI, 1993) and Reduced Affine Arithmetic (RAA) (GAMITO; MADDOCK, 2007). Knoll *et al.* (KNOLL *et al.*, 2009) present a method for extracting isosurfaces from implicits based on an IA-driven bisection approach. They achieve interactivity and less conservative results by utilizing an adaptation of the inclusion-preserving RAA approach proposed by Messine (MESSINE, 2002). Our method also benefits from the bounds given by IA for feature extraction.

4.6 Summary of Parallel Vectors Features

Table 4.1 presents a summary of the parallel vectors formulations presented so far. All formulations were devised to extract the 1-dimensional features embedded in \mathbb{R}^3 . The column *PV Criterion* presents the feature formulation in terms of the parallel vectors operator. The column *Feature* describes the type of feature to be extracted. The column

Author(s) presents the authors which developed the original feature definition, pointing the reader to the corresponding section where the feature, and its equivalent description in terms of parallel vectors, is further explained. Finally, the column *Description* presents a brief comment about the corresponding parallel vectors formulation. It also presents required post-filtering and classification criteria when the corresponding parallel vectors description is not sufficient to fully describe the feature.

	PV Criterion	Feature	Author(s)	Description
1	$\mathbf{v} \parallel (\nabla \mathbf{v}) \mathbf{v}$	vortex cores	Sujudi and Haimes (Section 4.3.1.1)	Jacobian at extracted point locations must present only one real eigenvalue.
2	$\mathbf{v} \parallel (\nabla \mathbf{v}) \mathbf{v}$	attachment and detachment lines	Kenwright <i>et al.</i> (Section 4.3.1.2)	Applies only on 2D vector field parameterizations (skin friction field). Further analysis of the eigenvalues of the Jacobian needed for proper feature classification.
3	$\nabla \times \mathbf{v} \parallel \nabla p$	vortex cores	Banks and Singer (Section 4.3.2)	Both eigenvalues of the Hessian of the pressure (parameterized on the plane perpendicular to the velocity) must be positive (minimum pressure).
4	$\mathbf{v} \parallel (\nabla \mathbf{v})^T \mathbf{v}$	vortex cores	Strawn <i>et al.</i> (Section 4.3.2)	Both eigenvalues of the Hessian of the pressure (parameterized on the plane perpendicular to the velocity) must be negative (maximum vorticity).
5	$\mathbf{g} \parallel \mathbf{H} \mathbf{g}$	ridge and valley lines	Eberly	Gradient of the scalar field parallel to eigenvectors of the Hessian. Needs post-filtering.
6	$\mathbf{v} \parallel \nabla \times \mathbf{v}$	vortex cores	Levy <i>et al.</i> (Section 4.3.4)	The PV formulation fully describes the feature.
7	$\mathbf{v} \parallel (\nabla \mathbf{a}) \mathbf{v}$	vortex cores	Roth and Peikert (Section 4.3.4)	Jacobian at extracted point locations must present only one real eigenvalue.

Table 4.1: Possible parallel vectors expressions as discussed by Roth and Peikert in (PEIKERT; ROTH, 1999). Some of the presented expressions extract raw features, and post filtering is necessary in order to isolate only the desired structures.

5 EXTRACTING FEATURES FROM HIGHER-ORDER CFD DATA

In this chapter, we present a technique for extraction of line-type parallel vectors features from higher-order (HO) data represented by piecewise analytical basis functions defined over structured and unstructured grid cells. The extraction uses parallel vectors in two distinct stages. First, seed points on the feature lines are placed by evaluating the inclusion form of the parallel vectors criterion with reduced affine arithmetic. Second, a feature flow field is derived from the parallel vectors expression in such a way it contains the features in the form of streamlines starting at the seeds. Our approach allows for guaranteed bounds regarding accuracy with respect to existence, position, and topology of the features obtained. The method is suitable for parallel implementation and we present results obtained with our GPU-based prototype. We apply our method to the discontinuous Galerkin higher-order datasets presented in Section 2.5.1. The contents of this chapter were published as "Efficient Parallel Vectors Feature Extraction from High Order Data" (PAGOT et al., 2011).

5.1 Background

To facilitate the understanding of the proposed technique, we start by introducing the feature flow field (FFF) and interval arithmetic concepts. Afterwards, the proposed method is presented and explained in detail.

5.1.1 Feature Flow Fields

Feature tracking methods usually work by detecting features of the flow in several time steps. Features are detected independently for each time step, being their correspondence and events established afterwards.

The correspondence among features in different time steps is traditionally investigated taking into consideration two criteria: region correspondence and attribute similarities. While region correspondence usually involves distances among features, attribute similarities involve feature sizes, volume and color among others.

Once the correspondence among features is established, their attributes are checked for significant changes over the time, that may indicate the occurrence of certain events. The following events, according to (THEISEL; SEIDEL, 2003), are considered:

- continuation
- birth (creation)

- death (dissipation)
- entry
- exit
- split (bifurcation)
- merge (amalgamation)

Until the introduction of the FFF concept, an usual approach was to track the features through isocontouring in a 4-dimensional space (WEIGLE; BANKS, 1998; BAUER; PEIKERT, 2002). Feature flow fields (THEISEL; SEIDEL, 2003) proposes an alternate approach for feature tracking that represents the dynamic behavior of feature not as higher dimensional isosurfaces, but as stream lines of a higher dimensional vector field. This approach benefits from the well-established techniques for numerical integration of stream lines commonly used in flow visualization.

5.1.1.1 General Feature Flow Field formulation

Assuming a 3-dimensional instationary vector field \mathbf{w} for which we want to track features:

$$\mathbf{w}(x, y, z, t) = \begin{pmatrix} k(x, y, z, t) \\ l(x, y, z, t) \\ m(x, y, z, t) \end{pmatrix}, \quad (5.1)$$

where t represents discrete time steps.

The idea is to construct a 4-dimensional vector field \mathbf{f} from \mathbf{w} in such a way that the dynamic behavior of features in \mathbf{w} can be represented by stream lines of \mathbf{f} . In this case \mathbf{f} is a vector field that points in the direction in which vectors of \mathbf{w} remain constant.

The direction of highest variation of k (Equation 5.1) at a given point happens in the direction of ∇k . Assuming a first order approximation, the value of k remains constant on the plane perpendicular to ∇k . The same holds for ∇l and ∇m . Since \mathbf{f} must point in the direction in which the vectors in \mathbf{w} remain constant, \mathbf{f} must be perpendicular to the gradients of all \mathbf{w} components simultaneously:

$$\begin{aligned} \mathbf{f} &\perp \nabla k \\ \mathbf{f} &\perp \nabla l \\ \mathbf{f} &\perp \nabla m \end{aligned} \quad (5.2)$$

Thus, from Equation 5.2, \mathbf{f} can be formulated as

$$\mathbf{f}(x, y, z, t) = \begin{pmatrix} \det(\mathbf{w}_y, \mathbf{w}_z, \mathbf{w}_t) \\ \det(\mathbf{w}_z, \mathbf{w}_t, \mathbf{w}_x) \\ \det(\mathbf{w}_t, \mathbf{w}_x, \mathbf{w}_y) \\ \det(\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z) \end{pmatrix}, \quad (5.3)$$

where \mathbf{w}_x , \mathbf{w}_y , \mathbf{w}_z and \mathbf{w}_t are the partial derivatives of \mathbf{w} .

5.1.1.2 PV features as Streamlines of the FFF

To represent the parallel vectors feature lines of the vector field \mathbf{w} (Equation 4.2) as streamlines of the FFF \mathbf{f} , the vectors of \mathbf{f} must point in the direction in which the vectors of \mathbf{w} neither change direction nor magnitude, i.e., once a streamline of \mathbf{f} is started where $\mathbf{w} = 0$, this holds along the streamline. Thus, the dynamic evolution of the features in \mathbf{w} is represented as stream lines of \mathbf{f} .

Since \mathbf{f} must point in the direction in which the vectors in \mathbf{w} remain constant, \mathbf{f} must be perpendicular to the gradients of all \mathbf{w} components simultaneously. Let \mathbf{f}_1 , \mathbf{f}_2 and \mathbf{f}_3 represent perpendicular directions defined as

$$\begin{aligned}\mathbf{f}_1(\mathbf{p}) &= \nabla k(\mathbf{p}) \times \nabla l(\mathbf{p}), \\ \mathbf{f}_2(\mathbf{p}) &= \nabla k(\mathbf{p}) \times \nabla m(\mathbf{p}), \\ \mathbf{f}_3(\mathbf{p}) &= \nabla l(\mathbf{p}) \times \nabla m(\mathbf{p}).\end{aligned}\tag{5.4}$$

As demonstrated in (THEISEL et al., 2005), \mathbf{f}_1 , \mathbf{f}_2 and \mathbf{f}_3 define collinear vectors for a given point \mathbf{p} and can be linearly combined to form a unique FFF for \mathbf{w} . However, we observed in our experiments that this approach can generate results with poor numerical stability. This can be caused by a combination of vectors pointing in opposite directions, leading to cancellation, or due to small angles between ∇k , ∇l and ∇m . These issues have been recently addressed by the stable FFF (WEINKAUF et al., 2010) at the cost of storing higher-order derivatives.

In our modified formulation we consider \mathbf{f}_1 , \mathbf{f}_2 , and \mathbf{f}_3 separately. At each integration step, we choose one as the current FFF, based on the angle formed by the gradient vectors at the current integration point \mathbf{p} . For example, one of the angles is computed by the following scalar product:

$$\alpha_1(\mathbf{p}) = \arccos \left(\frac{\nabla k(\mathbf{p})}{|\nabla k(\mathbf{p})|} \cdot \frac{\nabla l(\mathbf{p})}{|\nabla l(\mathbf{p})|} \right).\tag{5.5}$$

The other two angles, $\alpha_2(\mathbf{p})$ and $\alpha_3(\mathbf{p})$, are defined similarly for the pairs ∇k , ∇m , and ∇l , ∇m , respectively. The FFF chosen is the \mathbf{f}_i corresponding to the α_i closer to 90 degrees. For consistency during feature tracing, once a FFF is chosen, it is used for all evaluations required by the Runge- Kutta integration scheme for the computation of the current integration step. Figure 5.1 illustrates geometrically how \mathbf{f} is chosen. This formulation avoids vector cancellation that may occur in (THEISEL et al., 2005), improving the numerical accuracy of the FFF, at the same time that uses lower order derivatives than required by the stable feature flow field approach.

5.1.2 Interval Arithmetic

Interval arithmetic (IA), or interval analysis, is a tool conceived by Ramon E. Moore (MOORE, 1966) to automatically handle round-off truncation errors during computations with floating-point numbers and to robustly inspect the behavior of multivariate functions over domain intervals. The following sections present a brief overview of the original IA proposal, some existing IA variations and the reduced affine form adopted in this work for the representation of parallel vectors expressions.

5.1.2.1 Interval Arithmetic

According to IA definitions, a real quantity x is represented as an interval of floating-point numbers $\bar{x} = [x_L, x_U]$, where x_L and x_U represent the corresponding interval lower

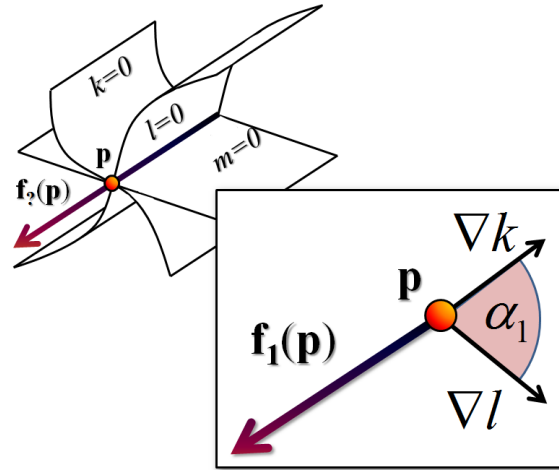


Figure 5.1: Selection of the current FFF based on the angle formed between the gradient vectors. In the above example we choose \mathbf{f}_1 as the FFF for the current feature integration step, since the angle α_1 , formed by its orthogonal gradient vector fields at the point \mathbf{p} is closest to 90 degrees.

and upper bounds. The actual value of x is known to lie somewhere in \bar{x} . Besides representing quantities as intervals of real values, IA also redefines the elementary arithmetic operators and functions such that results are guaranteed to contain the value the operation represents, a property known as inclusion property. Thus, for example, addition, subtraction and multiplication of intervals are defined as

$$\begin{aligned}\bar{x} + \bar{y} &= [x_L + y_L, x_U + y_U], \\ \bar{x} - \bar{y} &= [x_L - y_U, x_U - y_L], \\ \bar{x} \times \bar{y} &= [\min(x_L y_L, x_L y_U, x_U y_L, x_U y_U), \max(x_L y_L, x_L y_U, x_U y_L, x_U y_U)].\end{aligned}\tag{5.6}$$

Although it is not easy to find simple inclusion form descriptions for many functions, it is still possible to extend some functions to account for inclusion. That is the case, for example, with sine, cosine, power, square root, among others. Once the primitive operations and functions are redefined to account for inclusion, complex functions obtained by the combination of these operations will also preserve this property. Thus, any function $f : \Omega \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$ can be extended into its inclusion form \bar{f} such that

$$\bar{f}(\bar{x}) \supseteq f(\bar{x}) = \{f(x) : x \in \bar{x}\},\tag{5.7}$$

where $\bar{x} \subseteq \Omega$ is an interval box.

Since floating-point arithmetic presents limited precision, it may affect the actual interval limit values due rounding. Thus, in order to preserve the inclusion property at the cost of a more conservative result, it must be ensured that, at the generation of every interval quantity, the lower limit is always rounded down and the upper limit is always rounded up.

The main disadvantage of IA is the assumption that unknown values vary independently over the corresponding intervals. This assumption may lead to overly conservative results that are often impractical to use. One simple example is the expression $\bar{x} - \bar{x}$, for $\bar{x} = [-1, 1]$. Although the actual range for this expression is $[0, 0]$, according to the original IA definition it evaluates to $[-2, 2]$. The conservativeness problem becomes even

more evident when a longer sequence of arithmetic operations is performed. One possibility to reduce the conservativeness of IA is to subdivide the function domain into sub-domains and to evaluate the inclusion function on each sub-domain, combining the separate resulting intervals into a single one. However, the relative accuracy of IA is generally independent of the width of the input intervals.

5.1.2.2 Affine Arithmetic

As an alternative to the IA conservativeness, Comba and Stolfi (COMBA; STOLFI, 1993) proposed the affine arithmetic (AA). While still retaining the inclusion property of the original IA, AA attempts to reduce the conservativeness problem by keeping track of correlations between quantities during arithmetic operations. This is accomplished through the representation of interval quantities with affine forms, actually first degree polynomials, whose variables represent sources of uncertainty and coefficients their corresponding magnitudes. Thus, in AA an affine quantity \hat{x} takes the form

$$\hat{x} = x_0 + \sum_{i=1}^n x_i \varepsilon_i, \quad (5.8)$$

where x_0 is a real value representing the central value of the affine form, x_i are real coefficients indicating the magnitude of the corresponding partial deviation and ε_i are the noise symbols.

The noise symbols ε_i represent unknown values which are assumed to be in the closed interval $[-1, 1]$. Noise symbols can be shared among several AA quantities, and this is the key feature of the method. The sharing of certain noise symbols among several quantities indicates how they are related to each other.

As for IA, to compute operations between affine forms the elementary operations must be replaced by their AA counterparts. Affine operations, such as addition, subtraction of two affine forms, as well as addition and multiplication with a scalar value α , can be computed according to

$$\begin{aligned} \hat{x} + \hat{y} &= (x_0 + y_0) + (x_1 + y_1)\varepsilon_1 + \cdots + (x_n + y_n)\varepsilon_n, \\ \hat{x} - \hat{y} &= (x_0 - y_0) + (x_1 - y_1)\varepsilon_1 + \cdots + (x_n - y_n)\varepsilon_n, \\ \hat{x} + \alpha &= (x_0 + \alpha) + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n, \\ \alpha\hat{x} &= (\alpha x_0) + (\alpha x_1)\varepsilon_1 + \cdots + (\alpha x_n)\varepsilon_n. \end{aligned} \quad (5.9)$$

Non-affine operations, such as the multiplication of two affine forms, lead to non-affine results which must be approximated by affine forms. The main challenge is to choose affine forms which present small approximations errors.

After the execution of a non-affine operation, the approximation error is represented by a new noise symbol added to the resulting quantity. An example of non-affine operation is the multiplication of two affine forms \hat{x} and \hat{y} , as shown bellow

$$\begin{aligned} \hat{x} \times \hat{y} &= \left(x_0 + \sum_{i=1}^n x_i \varepsilon_i \right) \times \left(y_0 + \sum_{i=1}^n y_i \varepsilon_i \right) \\ &= x_0 y_0 + \sum_{i=1}^n (x_0 y_i + y_0 x_i) \varepsilon_i + z_k \varepsilon_k, \end{aligned} \quad (5.10)$$

where the coefficient z_k for the new noise symbol is computed according to

$$z_k = \sum_{i=1}^n |x_i| \times \sum_{i=1}^n |y_i|. \quad (5.11)$$

As for IA, errors may be inserted into resulting affine forms due rounding of floating-point values. However, differently from IA, rounding can destroy the relation of the current affine form quantity with respect to others. Thus, rounding and truncation errors are handled through the determination of an upper bound for the round-off errors committed in the computation of each coefficient and then adding this error to the linearization error coefficient.

Converting quantities between IA and AA forms is straightforward. Assuming that \hat{x} is a quantity represented in the affine form, its conversion to the IA form can be done according to

$$\bar{x} = [x_0 - \delta, x_0 + \delta], \quad \delta = \sum_{i=1}^n |x_i|. \quad (5.12)$$

The conversion from AA to IA form destroys any correlation information between quantities since the existing noise symbols are collapsed during conversion. Conversely, assuming that a quantity $\bar{x} = [x_L, x_U]$, in IA form, is to be converted to the corresponding AA form $\hat{x} = x_0 + x_k \varepsilon_k$, this can be accomplished according to

$$x_0 = \frac{x_U + x_L}{2}, \quad x_k = \frac{x_U - x_L}{2}. \quad (5.13)$$

5.1.2.3 Affine Arithmetic Extensions

Despite producing tighter bounds, AA is more expensive to compute and suffers from a growing number of error symbols each time non-affine operations occur, thus affecting overall performance and memory management. Messine (MESSINE, 2002) observed that and proposed some extensions to the original AA. One of these extensions is called the *first affine form* (AF1), and was introduced to attack the memory management problem. According to AA, for each variable x_i a corresponding noise symbol ε_i may be created. This allows the tracking for quantities correlations during computations. However, non affine operations introduce new noise symbols $\varepsilon_k, k \in \{n+1, n+2, \dots, n+p\}$, which may be generated by performing p non affine operations. Once these new symbols are created, they are stored, and never changed. It was observed that all these new terms could be added into a single new noise symbol ε_{n+1} , without losing the affine information related to the variables x_i . The AF1 could then be written as

$$\hat{x} = x_0 + \sum_{i=1}^n x_i \varepsilon_i + x_{n+1} \varepsilon_{n+1}, \quad (5.14)$$

where ε_{n+1} is the error symbol which represents the errors generated by all affine approximations executed so far, with $x_{n+1} \geq 0$.

In order to preserve the inclusion property, elementary operations are defined in such way that only positive definite operations are involved in the generation of x_{n+1} . Some

operations defined by AF1 are

$$\begin{aligned}
\hat{x} \pm \hat{y} &= (x_0 \pm y_0) + \sum_{i=1}^n (x_i \pm y_i) \boldsymbol{\varepsilon}_i + (x_{n+1} + y_{n+1}) \boldsymbol{\varepsilon}_{n+1}, \\
a \pm \hat{x} &= (a \pm x_0) + \sum_{i=1}^n x_i \boldsymbol{\varepsilon}_i + x_{n+1} \boldsymbol{\varepsilon}_{n+1}, \\
a \times \hat{x} &= (ax_0) + \sum_{i=1}^n ax_i \boldsymbol{\varepsilon}_i + |a|x_{n+1} \boldsymbol{\varepsilon}_{n+1}, \\
\hat{x} \times \hat{y} &= x_0 y_0 + \sum_{i=1}^n (x_0 y_i + x_i y_0) \boldsymbol{\varepsilon}_i + \left(|x_0| y_{n+1} + |y_0| x_{n+1} + \sum_{i=1}^{n+1} |x_i| \times \sum_{i=1}^{n+1} |y_i| \right).
\end{aligned} \tag{5.15}$$

5.1.2.4 Reduced Affine Arithmetic

The framework of AA is meant to be as general as possible, turning it into valuable tool for the solution of a wide range of problems. However, sometimes the problem is defined over a very restricted domain. In these cases, the original AA can be modified, and even simplified, in order to attend the problem specificities without loosing its original properties. This is the case with the reduced affine arithmetic (RAA) method proposed by Gamito and Maddock (GAMITO; MADDOCK, 2007) for the rendering of implicit fractal surfaces, that limits the number of error symbols the affine form can contain. The 3-term RAA form employs aggressive condensation executed each time non-affine operations are performed, reducing the correlation between error variables while still preserving the inclusion property under the specific problem circumstances.

Still in the context of implicit surface visualization, Knoll *et al.* (KNOLL *et al.*, 2009) revisited the work of Messine (MESSINE, 2002) and implemented a modified 3-term RAA formulation that, while executing condensation steps after each non-affine operation, ensures correct inclusion for all compositions of AA operations. Despite the increased conservativeness, the method present improved performance with respect to the original AA proposal.

Due to its generality and high performance, in this work we have used the RAA form by Knoll *et al.* (KNOLL *et al.*, 2009). Since we evaluate our expressions in \mathbb{R}^3 , we adopt the following 5-term RAA form

$$\hat{x} = x_0 + x_1 \boldsymbol{\varepsilon}_1 + x_2 \boldsymbol{\varepsilon}_2 + x_3 \boldsymbol{\varepsilon}_3 + x_c \boldsymbol{\varepsilon}_c, \tag{5.16}$$

where $x_c \boldsymbol{\varepsilon}_c$ represents the condensed error symbol.

5.2 PV feature extraction from HO data using RAA

Parallel vectors line-type feature extraction methods usually start by defining a collection of seed points that are used as starting points for subsequent feature reconstruction. In order to place a seed, it is common to use some data subdivision scheme together with an additional refinement step. While regular grids tend to require a large number of cells, adaptive refinement methods early discard data regions that do not contain solutions, directing computational effort towards relevant regions. One way to guide an adaptive spatial subdivision is to use the sign of the field at the element vertices as subdivision criterion. In the case of trilinearly interpolated data, different signs at element vertices indicate the presence of line-type features and that the spatial subdivision process

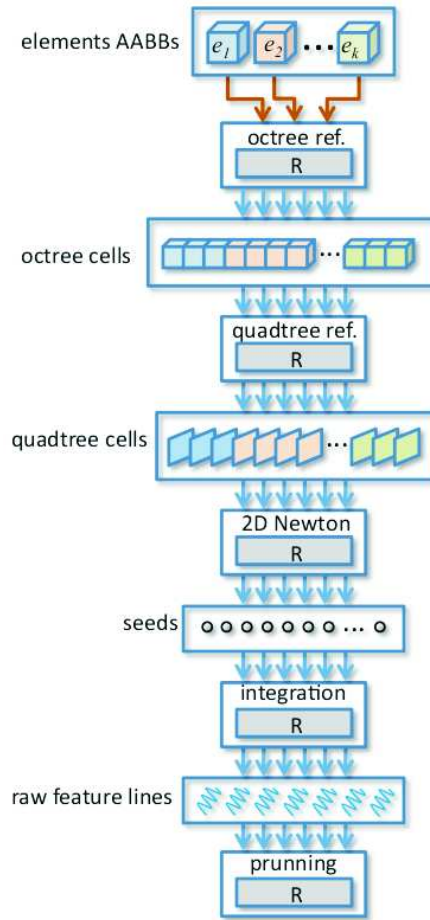


Figure 5.2: Computation pipeline for extracting features from higher-order data. At the end of each iteration step, for each stage, a reduction step (represented by the gray 'R' box) is applied over the primitives list in order to remove gaps.

should continue recursively. In case signs are equal, there cannot be features inside the current cell and subdivision is stopped. However, in the case of higher-order data this simple criterion no longer works: a feature line can intersect a cell even if all vertices have identical signs.

In this section we present an algorithm for parallel vectors feature extraction composed of two stages. The seed extraction stage executes octree and quadtree-based adaptive subdivision followed by Newton iterations to accurately locate seeds. The use of RAA to guide the spatial subdivision provides error bounds with respect to existence, position and topology of the features. In the second stage, features lines defined as streamlines of the FFF are traced from previously located seeds. Figure 5.2 illustrates the method's pipeline, which is explained in detail in the following sections.

5.2.1 Seed Extraction

The search for seeds starts with an adaptive subdivision scheme which narrows the search for features separately for each element (cell of the grid). Since elements can be represented by arbitrary polyhedral shapes, the estimation of bounds for the higher-order data might not be trivial. Instead, we estimate bounds using an axis-aligned bounding box (AABB) B_i that encloses each dataset element e_i .

Spatial refinement is performed in two successive steps. First, B_i is adaptively subdivided in an octree-fashion guided by the evaluation of the RAA form of the parallel vectors criterion. Subdivision stops when the octree cell sizes are within a minimum prescribed size (features down to this size are guaranteed to be intersected by the cells). This process generates a collection of candidate octree cells that might contain (intersected) features. For each face of the octree cells, we apply adaptive quadtree-based subdivision steps, again based on RAA, to better approximate seed locations. The center point of candidate quadtree cells are used as starting points for the final refinement step, where Newton iterations further improve seed placement. The next sections explain in detail each step of the seed extraction stage.

5.2.1.1 Octree Seed Refinement

Initially, each element e_i is tightly enclosed by an axis-aligned bounding box B_i that represents the initial cell of the octree subdivision process. The extents of the bounding box are represented in RAA form by \hat{x}, \hat{y} , and \hat{z} in the reference space of e_i . Adaptive subdivision is guided by the evaluation of the RAA form of the parallel vectors expression $\hat{\mathbf{w}}(\hat{x}, \hat{y}, \hat{z})$. If $0 \in \hat{\mathbf{w}}(\hat{x}, \hat{y}, \hat{z})$ for the current octree cell, it potentially contains features and must be further subdivided. This process is repeated recursively for child cells. If $0 \notin \hat{\mathbf{w}}(\hat{x}, \hat{y}, \hat{z})$ for a given cell, it does not contain features and can be safely discarded. During the subdivision process, due to the conservativeness of B_i with respect to the coverage of e_i , some cells may fall outside e_i . These cells are also discarded. The remaining cells after the subdivision process ends are candidates for containing features.

Since each box B_i may have a different size depending on the corresponding element e_i , the maximum octree depth, needed to capture features with a minimum prescribed size, must be computed separately for each element. For this purpose, we define the feature size as the length of the longest side of its axis-aligned bounding box. Thus, considering the minimum feature size ε_F , the maximum octree depth O_{D_i} for B_i is computed in terms of ε_F and the length l_i of the largest edge of B_i as

$$O_{D_i} = \lceil \log_2 \left(\frac{l_i}{\varepsilon_F} \right) \rceil. \quad (5.17)$$

The parameter ε_F allows us to extract feature lines at different levels of the octree. Smaller values for ε_F capture smaller features at the cost of decreased performance and increased memory consumption, whereas larger values result in more efficient feature extraction at the cost of possibly missing small features.

5.2.1.2 Quadtree Seed Refinement

After the octree refinement it is guaranteed that features larger than $\sqrt{3}\varepsilon_F$ intersect at least an octree cell face. However, it can be the case that a given face intersects multiple feature lines, or even multiple times the same feature. This second subdivision scheme further refines the search for, potentially multiple, seeds at the octree cell faces by computing an adaptive quadtree-based 2D subdivision of each face. Each rectangle representing an octree cell face is set as the root for the quadtree refinement step. As for the octree subdivision, the quadtree refinement is driven by the evaluation of a RAA form of the parallel vectors expression. However, since each face is perpendicular to one coordinate axis and hence represents a 2D space (Figure 5.3), only three specific parallel vectors RAA

expressions have to be defined:

$$\begin{aligned}\hat{\mathbf{w}}_{xy}(\hat{x}, \hat{y}, z), \\ \hat{\mathbf{w}}_{xz}(\hat{x}, y, \hat{z}), \\ \hat{\mathbf{w}}_{yz}(x, \hat{y}, \hat{z}),\end{aligned}\tag{5.18}$$

where \hat{x}, \hat{y} and \hat{z} are intervals representing the quadtree cell extents along the three axes, and x, y and z are constant real values representing the fixed coordinate. According to the alignment of a given quadtree cell, one of the expressions in Equation 5.18 is used for the evaluation of the parallel vectors operator. If the resulting interval encloses zero, it potentially contains a feature intersection and must be further subdivided.

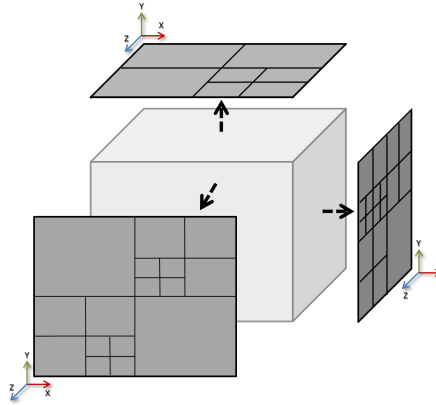


Figure 5.3: Quadtree cells aligned with octree cell faces reduce the dimension of the domain where parallel vectors criterion will be evaluated.

Considering that a minimum prescribed distance among seeds must be detected, and that the size of faces in an octree cell may vary, the maximum quadtree depth would be ideally computed separately for each face. Since the aspect ratio of a given cell usually has little variation, for simplicity, a maximum quadtree depth Q_{D_i} is defined for all octree cell faces generated for an element e_i . This leads to a more conservative subdivision that still preserves the accuracy thresholds and does not significantly affect performance. Q_{D_i} is computed in terms of minimum distance among seeds ε_S , the largest edge l_i of the B_i , and the maximum depth of the current octree:

$$Q_{D_i} = \lceil \log_2 \left(\frac{l_i}{\varepsilon_S} \right) \rceil - O_{D_i}.\tag{5.19}$$

Feature intersections over a quadtree face that are farther away than ε_S are considered as individual seeds by the quadtree refinement, whereas seeds with a distance smaller than ε_S are collapsed into a single point and hence only a single feature will be traced from there.

5.2.1.3 Newton Seed Refinement

The last step employs root finding to refine seed positions. Remaining quadtree cells are candidates for containing seeds, however, RAA conservativeness may lead to false positives. To locate the final seed positions, we use 2D Newton root finding for \mathbf{w} constrained to the cell supporting plane. The starting point for the Newton iterations is the center point \mathbf{p}_0 of the corresponding quadtree cell.

The quadtree cells are always perpendicular to one of the coordinate system axis, leading to three possible 2D Newton expressions:

$$\begin{aligned}\mathbf{p}^{k+1} &= \mathbf{p}^k - \mathbf{J}_{xy}(\mathbf{p}^k)^{-1} \mathbf{w}_{xy}(\mathbf{p}^k), \\ \mathbf{p}^{k+1} &= \mathbf{p}^k - \mathbf{J}_{xz}(\mathbf{p}^k)^{-1} \mathbf{w}_{xz}(\mathbf{p}^k), \\ \mathbf{p}^{k+1} &= \mathbf{p}^k - \mathbf{J}_{yz}(\mathbf{p}^k)^{-1} \mathbf{w}_{yz}(\mathbf{p}^k),\end{aligned}\tag{5.20}$$

where \mathbf{w}_{xy} , \mathbf{w}_{xz} , and \mathbf{w}_{yz} are parallel vectors expressions, \mathbf{J}_{xy} , \mathbf{J}_{xz} , and \mathbf{J}_{yz} their corresponding Jacobians, and k is the iteration step.

The point \mathbf{p} is considered as a seed if it has converged within the cell limits after the execution of a maximum number of Newton iterations N_q . For all our experiments, $N_q = 25$ and the convergence thresholds $|\mathbf{w}_x| \leq 10^{-3}$, $|\mathbf{w}_y| \leq 10^{-3}$, and $|\mathbf{w}_z| \leq 10^{-3}$ worked well. To be more generic, the thresholds could be scaled by the maximum of $|\mathbf{w}|$ at the corners of the quadtree cell.

5.2.2 Feature Tracing and Filtering

Seeds generated by previous refinement steps serve as starting points for feature tracing along the stream lines of the FFF. Depending on the feature type to be extracted, additional filtering must be applied to the extracted raw features. The next sections describe additional details regarding the feature tracing and additional filtering steps.

5.2.2.1 Feature Tracing

An accurate tracing strategy such as stable FFF (WEINKAUF et al., 2010) would greatly benefit from the analytical derivatives available in higher-order data. However, the computation of higher-order derivatives requires additional storage of the corresponding coefficients. To save high-performance cached memory for higher efficiency by simultaneous processing of a larger number of elements, we use a tracing scheme based on the one by Theisel et al. (THEISEL et al., 2005), which requires lower-order derivatives while providing reasonably accurate results.

Feature tracing starts at seed points and follows streamlines of the FFF. The feature flow field \mathbf{f} is constructed from the respective parallel vectors expression \mathbf{w} (Section 5.1.1). Our feature tracing method consists of a predictor-corrector scheme using a 4th-order Runge-Kutta scheme with fixed step size s . For all our experiments, we used $s = 10^{-3}$ (compare the extents of the datasets in the result section) and three corrector moves per predictor move.

Some prior predictor-corrector tracing schemes constrain corrector moves to a plane perpendicular to the predictor move. However, this plane can assume arbitrary orientation with respect to the reference system during the tracing process. This would require a re-parameterization of \mathbf{w} on the new correcting plane at each predictor-move step, which would degrade performance. Therefore, we only use corrector moves constrained to axis-aligned planes. This approach allows efficient re-parameterization of \mathbf{w} by simply keeping the coordinate related to the perpendicular axis constant. Thus, after each integration step, we check the angle between the predictor move and the three possible axis-aligned planes. The correcting plane is the one that forms the angle closest to 90 degrees (Figure 5.4). To compute the correcting moves, we apply the Newton method, searching for zeros of \mathbf{w} . Since the correcting planes are axis-aligned, we can use the same 2D Newton formulation used for seed refinement (Section 5.2.1.3).

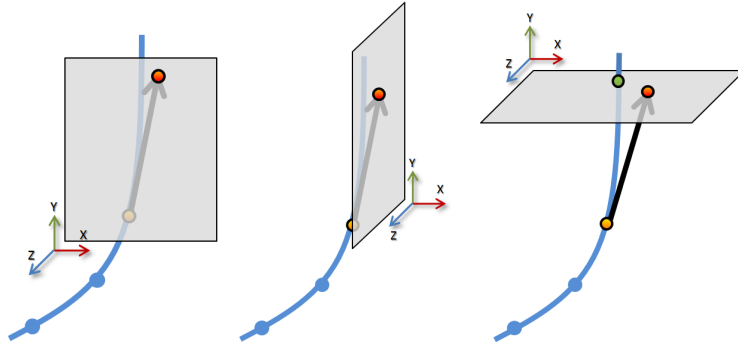


Figure 5.4: Selection of the correcting plane (gray) based on the angle formed with the predictor move (black arrow). Left and center: planes form smaller angles with the predictor move direction. Right: the ZX plane forms the angle that is the closest to 90 degrees, thus being elected to be the correcting plane.

Differently from the seed refinement stage, the point is assumed to have converged if, after the Newton iterations, it remains within the bounds of a quadrilateral centered at the predicted point. The size of the quadrilateral defines the maximum allowed angle formed between the predictor move and the feature for the given step size s . For our experiments, a quadrilateral with dimensions $6s \times 6s$ showed to work well. Additionally, for all our experiments, three Newton steps were sufficient for the correction step. Once feature lines are traced, redundant feature parts are removed, as well as features that extend beyond element boundaries (pruning) as shown in Figure 5.5.

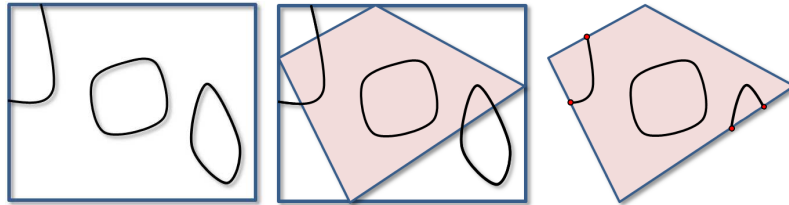


Figure 5.5: Pruning extracted feature lines against the actual element boundary.

5.2.2.2 Filtering

According to (PEIKERT; ROTH, 1999), there are several possibilities in the context of line-type features. One of the most powerful and often neglected ones is the angle between the feature tangent and the parallel vectors. This quantity has to stay small for a well-defined feature, as already mentioned by Eberly (EBERLY, 1996) in the context of ridges. It is also important in the context of the extraction of vortex core lines and lines of separation and attachment. Further, it is important to filter features by their strength. Here, the filter definition depends on the feature type. In case of vortex core lines, one can use the absolute value of the imaginary part of the complex eigenvalue of the velocity gradient. In case of ridges, one can filter the ridges by their height, i.e., by the value of the scalar field. We apply this filter for the results presented in Section 5.4.

Another criterion used is the length of the feature. It is often the case that short features are less important than long ones and more likely to arise due to noise. Although this filtering is often used, we were not able to use it in our discontinuous Galerkin re-

sults because the discontinuities at the cell boundaries disrupt features and make it even impossible in many cases to identify correspondence to features in adjacent cells. Further details on feature filtering are discussed in (PEIKERT; ROTH, 1999; PEIKERT; SADLO, 2008).

5.3 CUDA Implementation

The design of our algorithm lends itself for parallel computation of several elements. In this section we describe a CUDA (CUDA, 2010) implementation that aims at parallelizing computations as much as possible.

5.3.1 Ordering of Computation

The only pre-processing step that must be performed is the CPU-based element ordering based on the degree of the polynomials, since simultaneous processing of elements with same degree polynomials improves performance by reducing divergence when executing threads in parallel. Dataset elements are pushed through a pipeline that performs seed placement, feature tracing, and filtering. Each pipeline stage is implemented as a separate kernel. One important aspect to consider is data transfer: prior to the processing of each element, related data, including polynomial coefficients and element boundaries, must be loaded into the GPU. To minimize data transfers and improve cache locality, computations are performed on a per-element basis rather than on a per-pipeline-stage basis.

5.3.2 Polynomial Evaluation and Storage

All steps of the algorithm, with the exception of pruning, execute some sort of polynomial evaluation. Since each element field interpolation can be represented by an arbitrary degree polynomial, kernels must account for that. However, a kernel capable of evaluating a general degree polynomial with an arbitrary degree should contain a loop that could not be unrolled by the compiler. To increase performance, we decided to keep several versions of these kernels, each one targeted to a specific polynomial degree. For the polynomial evaluation itself we use static expressions generated by a multivariate Horner scheme approach (CEBERIO; KREINOVICH, 2004).

In our experiments the amount of data for each element is less than 8kB. Since the size of the shared memory of current GPUs is between 16kB and 64kB, more than one element can be processed in different threads without fetching data at each new pipeline stage.

The polynomial describing the field for each element is given in analytical form through an array of coefficients. Expressions such as the parallel vectors or the FFF are usually constructed from the n -th order derivatives of the original fields resulting in polynomial expressions of very high degrees and thousands of terms. However, large polynomial expressions reduce performance, increase data transfer and storage demands. Additionally, during the subdivision process the polynomials representing the parallel vectors criterion must be translated into the corresponding inclusion form, implying that operators and quantities are replaced by their (more complex) inclusion form counterparts.

In order to improve performance, and keep the necessary storage space within acceptable limits, instead of storing the parallel vectors and FFF expressions, we store only the coefficients of the n -th order derivatives used to compute them. This approach provides knowledge of the highest polynomial degree to be evaluated during the compu-

tations. Another advantage is that smaller expressions fit in CUDA’s high performance shared memory, while intermediate values generated during evaluations fit almost completely into registers, thus avoiding costly global memory access. Additionally, for all polynomial evaluations we take advantage of a multivariate Horner scheme (CEBERIO; KREINOVICH, 2004), which further reduces the number of necessary arithmetic operations.

5.3.3 Seed Extraction

The following sections examine the implementation of the seed refinement stages.

5.3.3.1 Octree Refinement

Before computation starts, we load into the GPU shared memory the polynomial and boundary shape information for the corresponding element, previously stored on GPU global memory. The octree subdivision kernel corresponding to the respective polynomial degree is loaded and refinement starts. As described in the previous section, octree cells that potentially contain features are identified by the evaluation of the RAA form of the parallel vectors expression.

The implementation of the adaptive octree refinement stage in CUDA suffers from limitations of the GPU’s with respect to dynamic memory allocation. Therefore, since we only evaluate octree cells to find regions that will be used as candidates for seed extraction, we do not need to store explicit links between a node and its descendants. Instead, we construct the octree incrementally by adding a new refinement level at each kernel invocation.

All octree leaves in the current level, that potentially contain features, are stored in a list. The octree subdivision is done in successive steps, each receiving a list of octree leaves from the previous step, and producing a list of octree leaves for the next level. After a new list of cells is produced, the previous list is discarded. This process is repeated until the maximum octree depth O_{D_i} for the current element is reached.

At the end of a refinement step, child cells that were evaluated as candidates for containing features are stored into a list, while cells that were discarded generate gaps in the list. At the end of each refinement step, gaps are removed by the execution of a specific condensation kernel that overwrites the previous list contents with a list of cells without gaps. The resulting list is used as input for the next octree refinement step, or as input for the quadtree refinement step, if the maximum octree depth was reached.

5.3.3.2 Quadtree Refinement

Initially a face list with the faces of each octree cell is created. Each face will serve as the starting cell for the quadtree refinement. The faces are perpendicular to one coordinate system axis and are related to an arbitrary degree polynomial. Thus, to improve efficiency, quadtree subdivision is implemented through several CUDA kernels, each one optimized for a specific combination of polynomial degree and face alignment.

The face list certainly presents arbitrary face alignment sequences, and it is likely that simultaneous execution of distinct kernels will happen, degrading performance. Ideally, cells presenting the same alignment should be processed in batches together. Since the size of the octree cells list is known, and that each cell generates six faces (two perpendicular to each axis), we generate an alignment-based grouped face list directly from the octree cell list (Figure 5.6). An advantage is that subdivided faces generate equally

aligned cells, which means that the alignment-based grouping is preserved along the entire quadtree subdivision stage.

Once the ordered face list is built, quadtree subdivision is computed in parallel for each face through successive steps, as for the octrees. At the end of one step a new quadtree cell list is produced, which replaces the previous list and will serve as input for the next subdivision step. At the end of a subdivision step there might be gaps due discarded cells. Again, a reduction step, implemented as a separate kernel, compacts the cell list. The reduction step does not affect the cell list alignment-based ordering. The subdivision process is repeated until the maximum quadtree depth level Q_{D_i} for the current element e_i is reached.



Figure 5.6: (1) Quadtree cell list presenting arbitrary alignment sequences. (2) Quadtree cell list presenting alignment-based grouping sequences. Alignment-based cell grouping improves performance by reducing divergence during kernel executions.

5.3.3.3 Newton Refinement

Candidate faces generated by the quadtree refinement stage are used to generate the initial list of candidate seed points. The points are obtained from the central coordinates of each face.

A fixed number of 2D Newton iterations, restricted to the corresponding face supporting plane, are computed for each point through a set of kernels optimized for specific combinations of polynomial degrees and face alignments. The number of iterations is fixed in order to keep threads synchronized. Additionally, through the use of several kernel implementations, this stage also benefits from the face alignment-based grouping imposed previously by the quadtree subdivision stage.

After the maximum number of iterations are reached, the resulting point list is written together with flags that indicates whether the point has converged to a seed or not. An additional reduction step is performed to remove divergent points from the list.

5.3.4 Feature Tracing and Classification

Features are traced in parallel from each seed through several passes (kernel invocations). In each pass a small number of integration steps are executed. This approach helps in reducing the overhead involved with excessive kernel invocations and additionally distributes the tracing workload over several threads. For all our experiments, the number of integration steps executed in each pass was 32 since this number showed to give better performances according to our hardware and software setup.

Raw features are represented by closed lines, and some features can be doubled traced. However, from our experiments the number of features to be traced showed to be small if compared to the parallel processing capacity of the GPU, meaning that the double integration presented no significant performance impact on performance.

After the integration stage, an additional CPU step is executed to remove the redundant features. The last step in the pipeline involves the classification and computation of the attributes of points on the extracted raw feature lines. The classification and computation

of the local properties depends on the type of feature to be extracted and on a series of constant-cost parallel computations particular to each point. In case of height ridge extraction, one has to decide if the feature represents a ridge or a valley. This can be done by checking the signs of the eigenvalues of the Hessian of the scalar field (Section 4.3.3).

5.4 Results

Performance measurements were obtained on a computer equipped with a Intel Core i7 960 3.2 GHz processor, 6GB of RAM, and a NVIDIA Geforce GTX 470. The CPU-side code was implemented in C++/OpenGL and the GPU-side in CUDA 3.2 and Thrust 1.3. Feature tracing used the predictor-corrector approach presented in Section 5.2.2.1. Fixed settings were used for feature tracing. Octree and quadtree maximum depths depend on the minimum feature size threshold and may vary across elements (Sections 5.2.1.1 and 5.2.1.2). The method was evaluated using the two higher-order discontinuous Galerkin datasets presented in Section 2.5.

Performance measurements and images presented in this section refer to the extraction of ridge and valley lines according to the parallel vectors formulation presented in Section 4.3.3.

In the left column of Figures 5.7 and 5.8, we present the performance measurements of our method with respect to feature extraction from the *sphere* and *shock channel* datasets respectively. Each row indicates the performance measurements according to the number of elements processed in parallel: 2 elements in top row, 4 elements in the middle row and 8 elements in the bottom row. The charts show performance timings against minimum feature size ϵ_F . The charts contain 5 different curves showing the timing breakdown across different stages of the algorithm: octree subdivision (orange), quadtree subdivision (yellow), seed refinement using 2D Newton steps (green), feature tracing (brown), and total time to extract raw features (blue).

Values for ϵ_F were chosen to assess the method’s performance in scenarios where finer refinement is demanded. Even though there is just a small cost associated with octree subdivision at the largest ϵ_F (fewer subdivisions), we observe that the cost related to feature tracing is the highest. This is explained by the fact that the number of feature lines inside a given element is usually small, leaving the GPU idle. As we decrease ϵ_F , more octree subdivisions start to occur and feature lines are broken into segments. As the number of segments approaches the maximum number of threads that can be executed in parallel on the GPU, the feature tracing cost reaches its minimum. We also observed that the cost of octree subdivision did not increase substantially as feature size diminished. Additionally, finer refinement leads to tighter intervals in the RAA evaluations.

Differently from our approach, Theisel et al. (THEISEL et al., 2005) proposed an octree refinement towards points on features lines. Their method has lower memory requirements, but is less robust to handle higher-order data since it assumes trilinear interpolation. As we discussed before, fewer seeds lead to poorer performance in parallel architectures. The right columns of Figures 5.7 and 5.8 present performance measurements for our RAA-based method using their octree refinement (towards points). As can be observed, octree refinement towards points on the feature lines leads to lower performance.

Figures 5.9 and 5.10 illustrate results for the shock-channel dataset. In Figure 5.9 we show all raw features in a given cell, as well the leaves of the octree at two different depths to demonstrate how the subdivision process. Figure 5.10 shows raw features extracted for

all cells and the main valley lines obtained after filtering.

In Figure 5.11 we compare valley lines extracted through our method with the lines extracted with Peikert and Roth (PEIKERT; ROTH, 1999) approach, for two cells of the *shock channel* data set. Since their method does not operate directly over higher-order data, we have resampled the two cells into a regular grid. Three sampling resolutions were used. Results for the coarser resolution (6x3x3 samples) are shown in Figure 5.11a. Figure 5.11b presents results obtained with a more refined sampling, consisting of 20x10x10 samples. Results for the highest resolution sampling (160x80x80) are shown in Figure 5.11c. Figure 5.11d presents the current state of the RAA-based octree refinement after 10 subdivision steps computed over the higher-order data represented by the two elements. It can be seen that, despite the high resolution used during resampling, in the left part of Figure 5.11c there are spurious line segments in regions that were robustly discarded by the RAA-based octree refinement used in Figure 5.11d. This can be explained by the introduction of errors due resampling and the use of finite-differences scheme for derivative computations, which may not be sufficiently accurate (in particular, regarding the second order derivatives used in the parallel vectors-based ridges and valleys definition). Finally, in Figure 5.11e we present the valley lines extracted directly from the higher-order data using our method. As can be seen, there is higher number of features without spurious lines and features tend to be represented by smoother and longer lines (despite discontinuities at the element boundaries due the nature of the data).

There are several criteria used for filtering (see caption of figure for details). Small feature lines are less important since they are usually related to noise. Thus, filtering out features by length becomes an important criterion. However, given the boundary discontinuities present in our discontinuous Galerkin data, filtering by length is not possible and local filtering criteria become of great importance. Figure 5.12 shows several stages of the feature filtering for the sphere dataset. Figure 5.12a presents unfiltered ridges (red), valleys (blue) and connector curves (white). Figure 5.12b shows only unfiltered valley lines. Figure 5.12c shows valley lines filtered by the angle between the feature tangent and the parallel vectors (≤ 2.5 degrees). Finally, in Figure 5.12d we have valley lines filtered by angle (≤ 2.5 degrees) and isovalue (≤ 0.998). Figure 5.12d also presents, for illustration, the overlay of the isosurface that corresponds to isovalue 0.998.

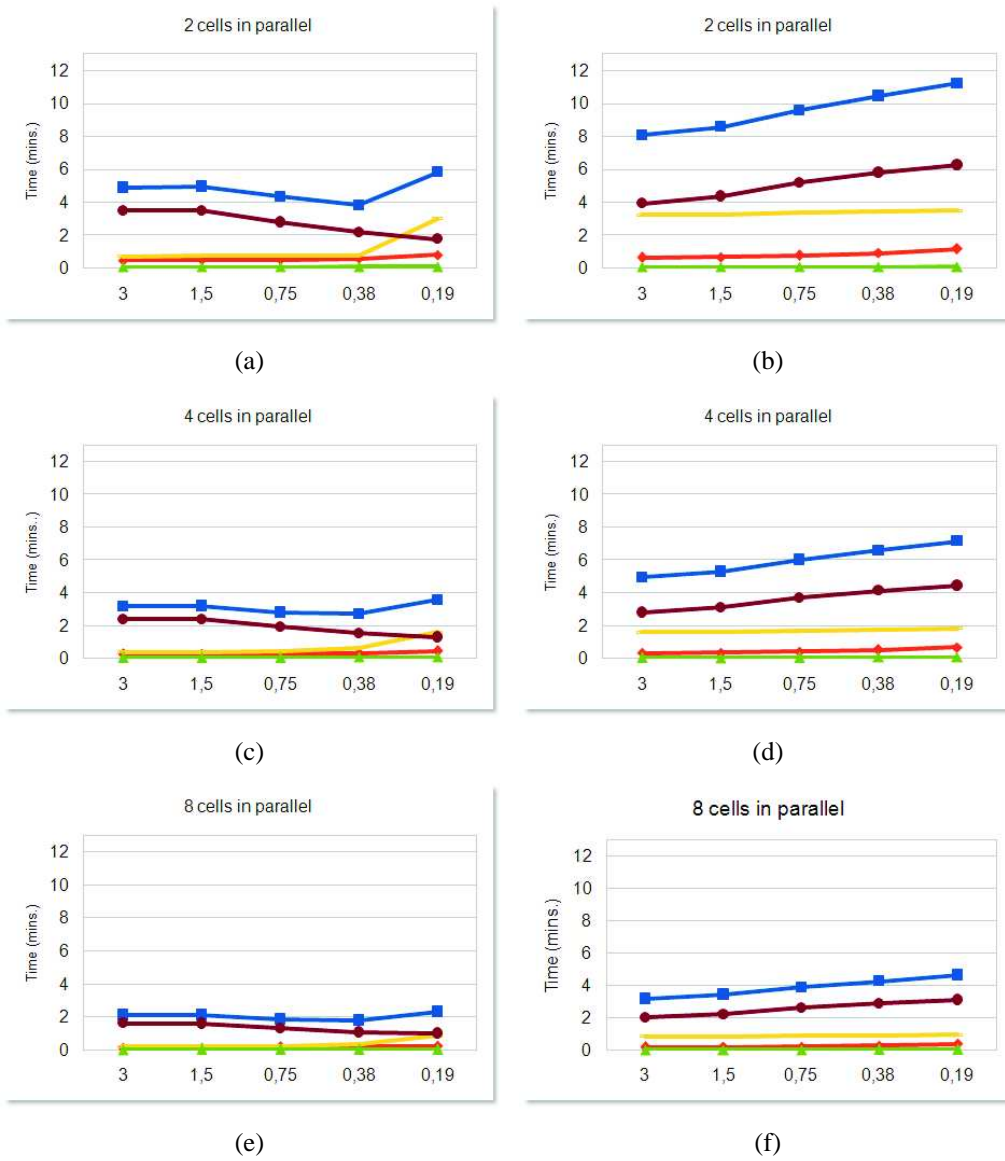


Figure 5.7: Performance measurements obtained with our feature extraction method for the *sphere* dataset (34,535 elements). Ordinate represents performance measurements in minutes while abscissa represents the value of ϵ_F (smaller ϵ_F implies higher octree refinement). For all tests $\epsilon_S = \epsilon_F/2$, to force quadtree subdivisions. Left column: Measurements obtained with octree refinement towards feature lines. Right column: Measurements obtained with octree refinement towards single points on closed feature lines (as proposed by Theisel et al. (THEISEL et al., 2005)). Lines present timings for the processing of 2 (top), 4 (middle), and 8 (bottom) dataset elements in parallel. Colored lines represent performance measurements for the octree subdivision (orange), quadtree subdivision (yellow), seed refinement with 2D Newton (green), feature tracing (brown), and total time to extract raw features (blue).

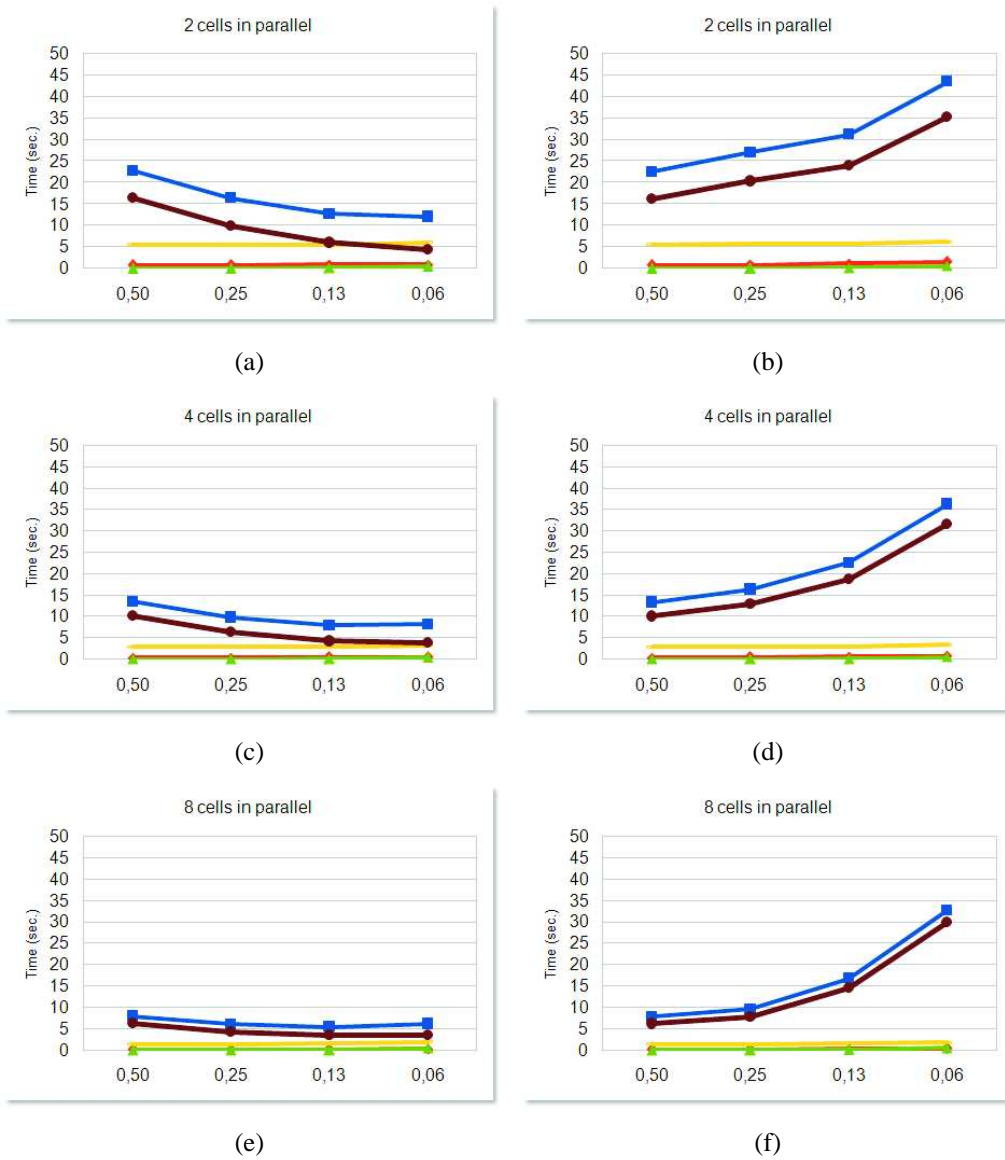


Figure 5.8: Performance measurements obtained with our feature extraction method for the shock channel dataset (119 elements). Ordinate represents performance measurements in seconds. For more details, see caption of Figure 5.7.

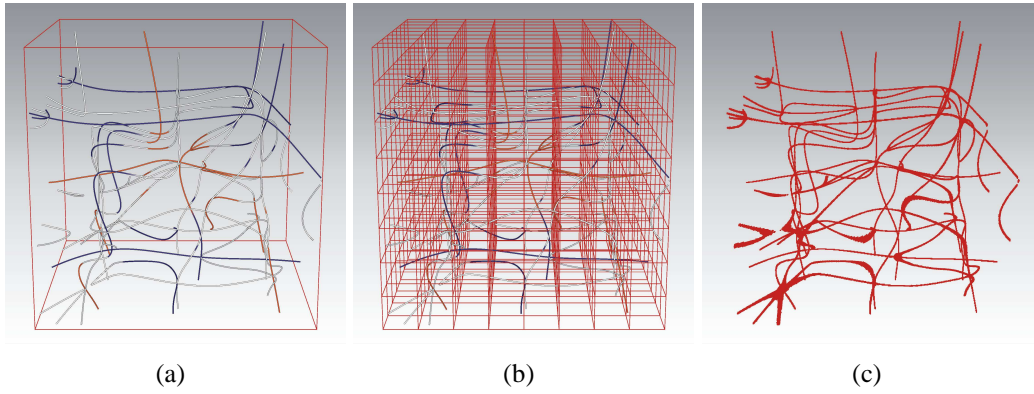
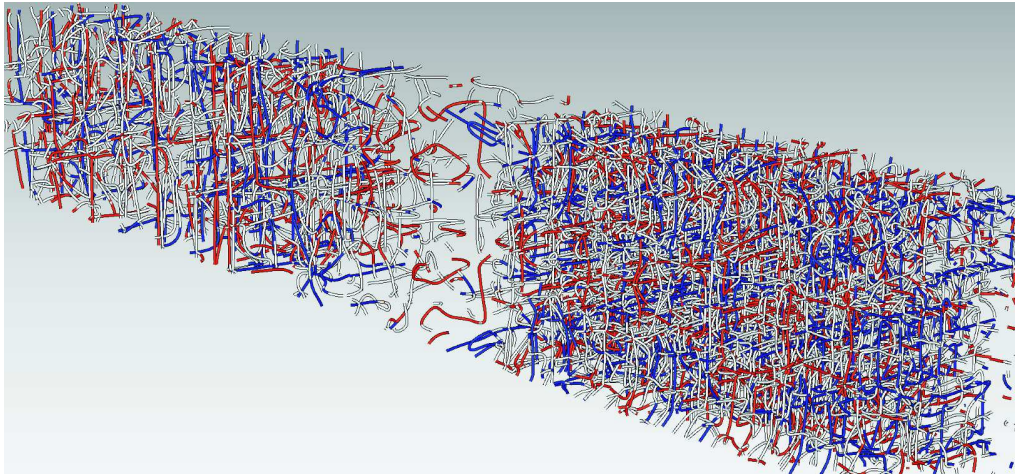
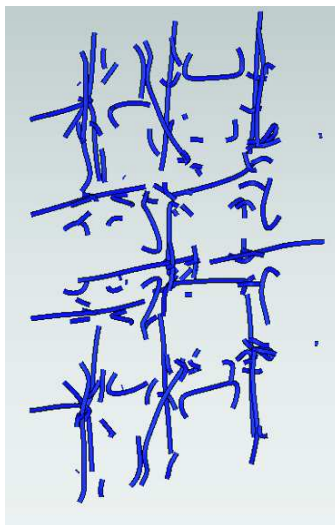


Figure 5.9: Shock-channel dataset. (a) Raw features extracted from a single element: ridges (red) and valleys (blue) ($\epsilon_F = 1$, $\epsilon_S = 0.01$). No octree subdivision. (b) Same as (a), with $\epsilon_F = 0.0625$, $\epsilon_S = 10^{-3}$. Octree leaves at depth 4. (c) Same as (a), with $\epsilon_F = 0.0019$, $\epsilon_S = 10^{-3}$. Octree leaves at depth 8.



(a)



(b)

Figure 5.10: Shock-channel dataset. (a) Connector curves (white) for all dataset elements ($\epsilon_F = 10^{-1}$, $\epsilon_S = 10^{-2}$). (b) Filtered valley lines from (d). Minimum scalar value = 1 and maximum = 1.9995. Angle between gradient and FFF tangent vector ≤ 27 degrees.

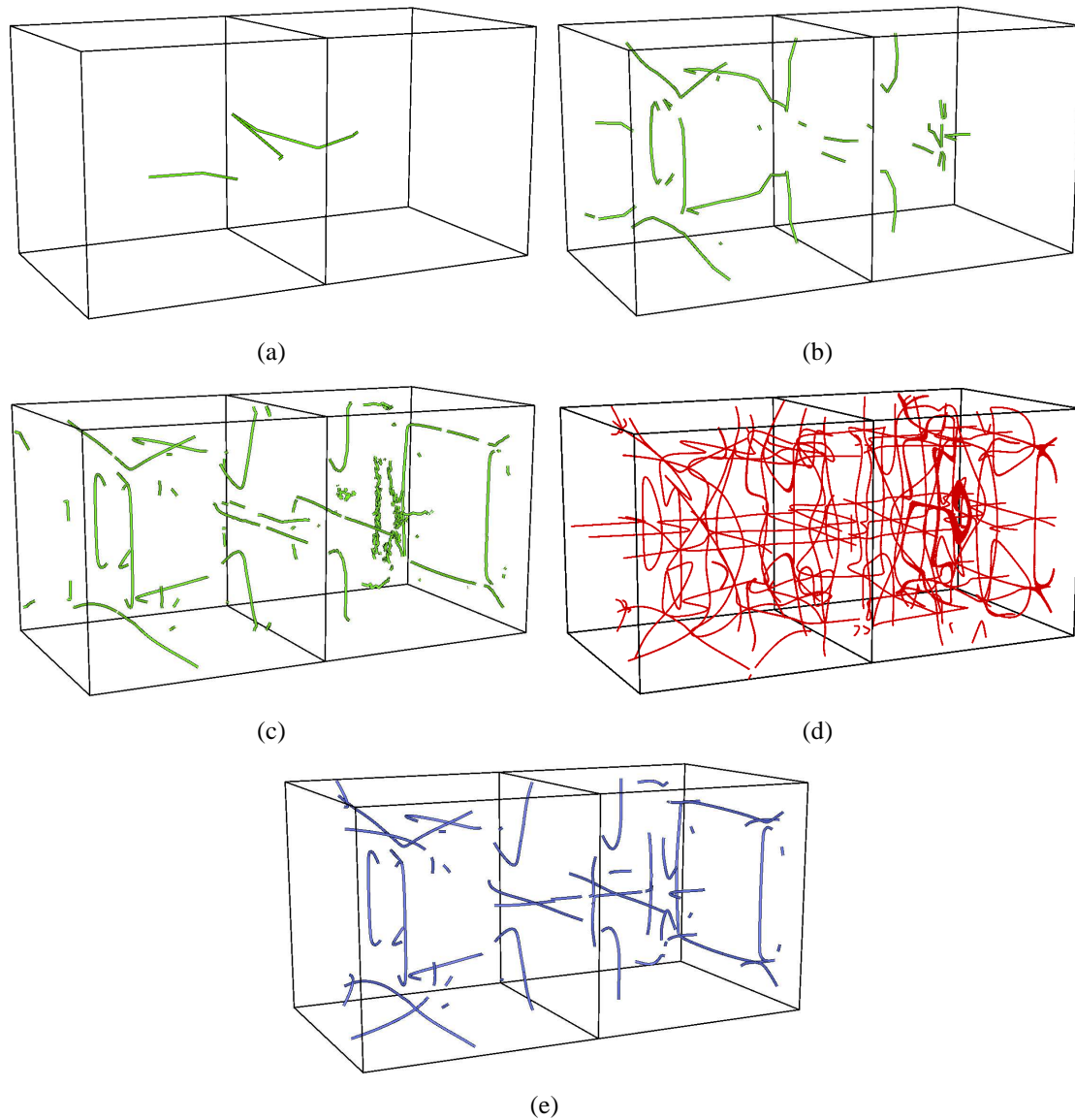
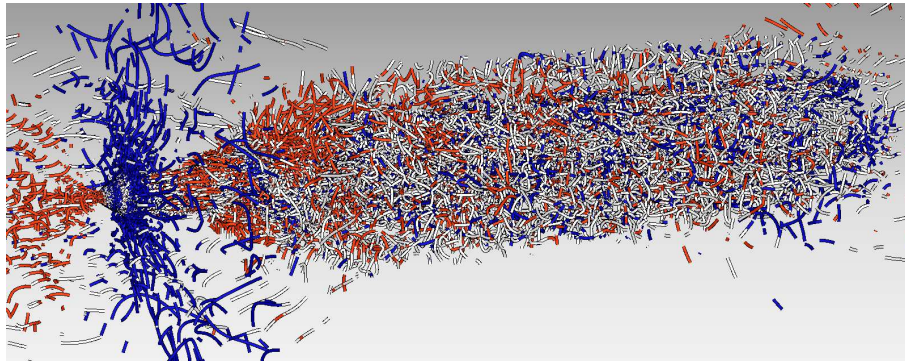
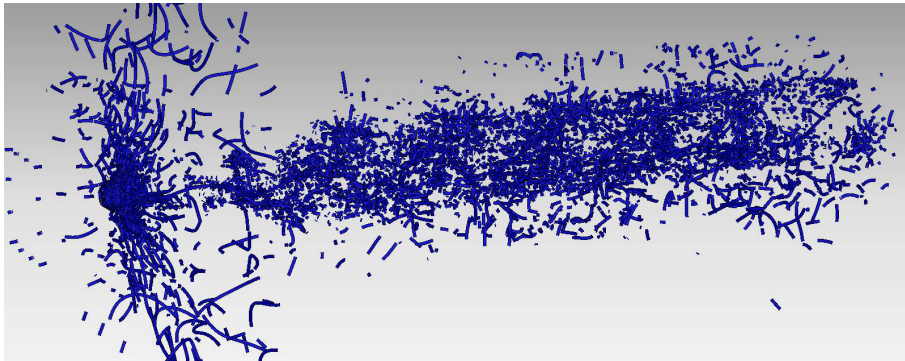


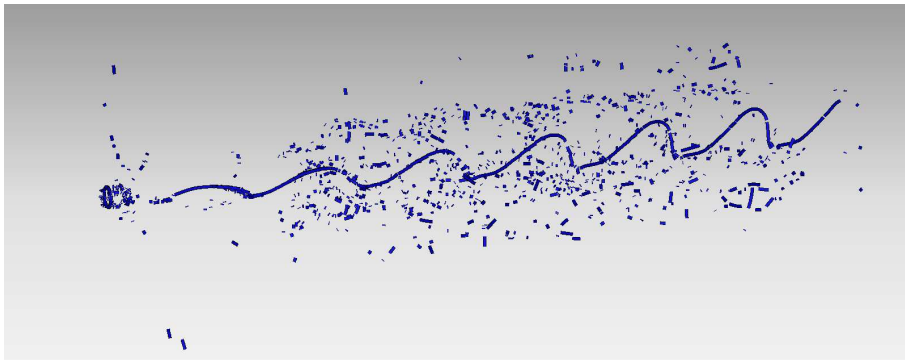
Figure 5.11: Comparison of valley lines extracted through our higher-order method with the lines extracted with the linear method by Peikert and Roth (PEIKERT; ROTH, 1999) for two cells of the *shock channel* data set. Figures (a),(b) and (c) were computed with the linear approach, while Figures (d) and (e) were computed directly over the higher-order data. a) Results for the coarser resolution (6x3x3 samples). b) Results obtained with a more refined sampling (20x10x10 samples). c) Results for the highest resolution sampling (160x80x80 samples). d) Current state of a RAA-based octree refinement after 10 subdivision steps computed over the higher-order data represented by the two elements. e) Valley lines extracted directly from the higher-order data using our method, resulting in a higher number of longer and smoother feature lines.



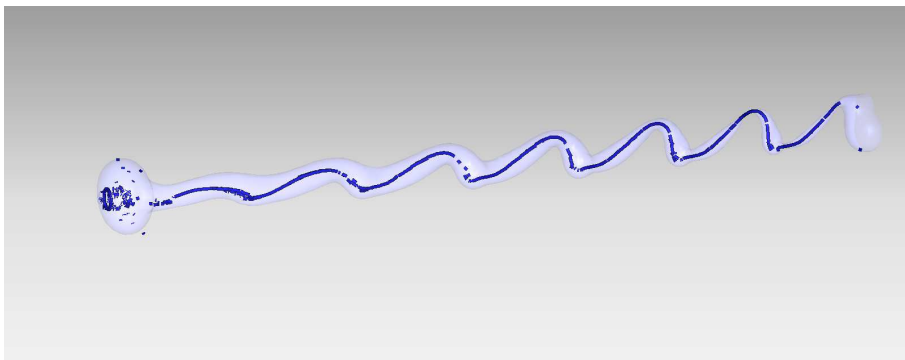
(a)



(b)



(c)



(d)

Figure 5.12: Extracted line-type features from sphere dataset under increasingly restricting filtering criteria. (a) Unfiltered ridges (red), valley (blue) and connector curves (white). (b) Only unfiltered valley lines (c) Valley lines filtered with the angle criterion (≤ 2.5 degrees). (d) Valley lines filtered with the angle (≤ 2.5 degrees) and isovalue (≤ 0.998) criteria.

6 CONCLUSIONS AND FUTURE WORK

The analysis of CFD data is a problem confronted by scientists and engineers every day. This analysis may involve the extraction of regions that present constant values for certain properties or the detection of data portions associated to relevant structures. The development of visualization and feature extraction methods have been explored and extended in many ways to allow exploration of various data types. This thesis has presented two new methods for the direct exploration of multi cell higher-order data. The proposed methods are efficient, scalable and do not rely in data resampling. This closing chapter presents a summary of the major points discussed in this dissertation. The chapter ends with a list of suggested areas for future exploration.

6.1 Isocontouring of Higher-Order Data

Existing isocontouring methods targeted at multi cell higher-order data usually rely on mesh extraction or image based techniques. Although allowing for interactive exploration of a extracted isosurface, mesh extraction techniques rely on intensive pre-computation that prohibits dynamic exploration of different isovalues. In the case of image based techniques, each image is generated from the scratch, leading to lower frame rates.

We propose a hybrid approach for the isocontouring of continuous or discontinuous higher-order data generated by *hp*-adaptive discretization methods. The method operates on higher-order data whose mapping function includes only affine transformations. Interactivity is achieved by splitting the contouring workload over computations in object and image spaces.

By using point sampling in the object space stage, we avoid the use of complex data structures and neighborhood information, thus making this stage suitable for parallel processing. Coverage of the isosurface is obtained through the generation of quads for each seed successfully projected onto the isosurface in the first stage. Points in these quads serve as starting points for the refinement stage based on ray casting, which also maps nicely on parallel architectures.

Since no neighborhood information is kept along the isocontouring pipeline, artifacts can emerge due subsampling. Although not focusing high accuracy, the proposed isocontouring method is capable of delivering results of reasonably quality. An additional heuristic, inspired on dynamical systems theory, is proposed to, at the cost of additional computational resources, pre-compute quads scaling factors resulting in better isosurface coverage.

The method counts with several parameters that can be adjusted in order to control the trade-off between efficiency and accuracy. It has been shown that one can explore both (accuracy and performance) by adjusting just a few parameters, keeping the remaining

parameters at fixed values.

6.2 Line-type Feature extraction from Higher-Order Data

Several techniques for parallel vectors feature extraction in trilinearly interpolated data have been already developed. Recent research in the field has focused mainly in more accurate feature tracing and feature extraction in higher dimensional spaces. Although some existing tracing strategies could be applied directly to higher-order data, the seed placement problem remained almost untouched in this context.

In this work we propose an efficient method for line-type parallel vectors feature extraction from multi cell higher-order data. The method operates on piecewise discontinuous higher-order data containing affine mappings between reference and world spaces. As in other research fields, such as isocontouring and optimization problems, the proposed method advocates the use of inclusion arithmetic to achieve efficiency and to guarantee bounds regarding accuracy with respect to existence, position, and topology of the features obtained.

Inclusion arithmetic (actually a RAA extension) is used during the seed placement stage to evaluate the parallel vectors expressions that drive the octree and quadtree-based adaptive data subdivisions towards feature lines and seed points. This approach improves performance by safely early discarding data portions that do not contain features and seeds. A compact 5-term RAA form used for the parallel vectors operator allows for efficient evaluation at the same time that avoids the conservativeness of the original interval arithmetic proposal. We also propose a more efficient octree subdivision scheme, towards feature lines, that is more efficient for parallel implementations. Performance improvement is also obtained through a processing pipeline carefully designed to process data in a per-element basis instead of per-stage basis, reducing the data read-back overhead. Additionally, we present a predictor-corrector based tracing scheme that efficiently re-parameterize the data field onto correcting planes perpendicular to the coordinate system axis.

6.3 Future Work

With respect to the proposed isocontouring system, some points should be addressed in future work. Despite the good results obtained with the FTLE-inspired heuristic for quads' scaling, the pre-processing stage involved in its computation could be removed through quad rescaling "on the fly". Another possibility regarding conservativity reduction during point sampling would be to remove points that have traveled through "long distances" inside the cell before getting projected onto the isosurface. The development of an improved initial point distribution heuristic for the first phase would be an alternative for improved surface sampling. Point-based isosurface approximation is view-independent and must be stored in order to be used by the image based phase. This may lead to higher storage demands during the isocontouring of large datasets. A multi-resolution approach for the isosurface approximation would allow the processing of larger datasets. Policies and strategies to discard seeds related to subpixel cells are also other possibilities for future research.

With respect to the parallel vectors line-type feature extraction technique, there are also several avenues for future investigations. More involved memory management methods may allow processing of a higher number of dataset elements in parallel. Simplifica-

tions in the tracing stage and storage of previous computations might allow progressive feature refinement. The extension of this method for the extraction of higher dimensional features is also a challenge under consideration. The simple inclusion of additional terms in the proposed RAA form, accounting for the combinations of pairs of independent variables, would result in less conservative results during the adaptive data spatial subdivision.

Both methods proposed in this thesis are targeted to higher-order data containing affine mappings between reference and world spaces. An interesting research avenue under consideration is the handling of higher-order data whose mapping functions are non-linear, thus leading to methods capable to handle the general case of higher-order CFD data.

APPENDIX A MATHEMATICAL DEFINITIONS

A.1 Differential Geometry Equations

A parametric curve in \mathbb{R}^3 can be expressed as

$$\mathbf{w} = \mathbf{w}(t) \quad (\text{A.1})$$

Defining $\mathbf{w}' = d\mathbf{w}/dt$ and $\mathbf{w}'' = d^2\mathbf{w}/dt^2$, the curvature vector of \mathbf{w} can be expressed as

$$\mathbf{c} = \frac{\mathbf{w}' \times \mathbf{w}''}{|\mathbf{w}'|^3} \quad (\text{A.2})$$

Thus, the curvature κ of \mathbf{w} can be written as

$$\kappa = |\mathbf{c}|. \quad (\text{A.3})$$

A.2 Directional Newton Method

Given a function f , such that

$$f(\mathbf{x}) = 0. \quad (\text{A.4})$$

The function f can be restricted to a line L

$$L = \{\mathbf{x}^0 + t\mathbf{d} : t \in \mathbb{R}\}, \quad (\text{A.5})$$

where it is a univariate function

$$F(t) = f(\mathbf{x}^0 + t\mathbf{d}). \quad (\text{A.6})$$

One Newton iteration for F at the point $t^0 = 0$ gives the next point

$$t^1 = -\frac{F(0)}{F'(0)}. \quad (\text{A.7})$$

Since $F(0) = f(\mathbf{x}^0)$ and $F'(0) = \nabla f(\mathbf{x}^0) \cdot \mathbf{d}$, from A.7 we can write

$$t^1 = -\frac{f(\mathbf{x}^0)}{\nabla f(\mathbf{x}^0) \cdot \mathbf{d}}. \quad (\text{A.8})$$

Given that

$$\mathbf{x}^1 = \mathbf{x}^0 + t^1 \cdot \mathbf{d}, \quad (\text{A.9})$$

we can write A.8 as

$$\frac{\mathbf{x}^1 - \mathbf{x}^0}{\mathbf{d}} = -\frac{f(\mathbf{x}^0)}{\nabla f(\mathbf{x}^0) \cdot \mathbf{d}}, \quad (\text{A.10})$$

or

$$\mathbf{x}^1 = \mathbf{x}^0 - \frac{f(\mathbf{x}^0)}{\nabla f(\mathbf{x}^0) \cdot \mathbf{d}} \mathbf{d}, \quad (\text{A.11})$$

which is the directional Newton method for the function f along the direction \mathbf{d} , according to (LEVIN; BEN-ISRAEL, 2002).

A.3 Convective Derivative

Assuming a vector function $\mathbf{w}(\mathbf{x}, t)$ and a scalar function $f(\mathbf{x}, t)$ defined over the same domain. The convective derivative D/Dt is defined as

$$\frac{Df}{Dt} = (\nabla f) \cdot \mathbf{w} + \frac{\partial f}{\partial t}, \quad (\text{A.12})$$

which represents the variation in f experienced by a particle that is at a particular place and time, being advected by the flow.

A.4 Strawn, Kenright and Ahmad Vortex Core Definition in Terms of the PV Operator

Section 4.3.2 introduced the PV form for the vortex core definition by Strawn, Kenwright and Ahmad (STRAWN; KENWRIGHT; AHMAD, 1998) as

$$\mathbf{w} \parallel \nabla(\mathbf{w}^2). \quad (\text{A.13})$$

According to the general vector identity

$$\nabla(\mathbf{u} \cdot \mathbf{v}) = (\nabla \mathbf{u})^T \mathbf{v} + (\nabla \mathbf{v})^T \mathbf{u}. \quad (\text{A.14})$$

Thus, for $\mathbf{u} = \mathbf{v}$, we have

$$\nabla(\mathbf{u}^2) = 2(\nabla \mathbf{u})^T \mathbf{u}. \quad (\text{A.15})$$

From A.15, we can write A.13 as

$$\mathbf{w} \parallel (\nabla \mathbf{w})^T \mathbf{w}. \quad (\text{A.16})$$

APPENDIX B ISOCONTOURING CODE

B.1 Unprojecting Fragments

Fragments in GLSL environment are described in normalized device coordinates (NDC). Thus, for a point $\mathbf{p} = (s, t, u, v)$, in the GLSL NDC space: $s, t \in [-1, 1]$; $u \in [0, 1]$ and $v = 1/w$, where w is the homogeneous coordinate.

```
float width_half = floor(float(viewport_w) / 2.0);
float height_half = floor(float(viewport_h) / 2.0);

vec4 ndc_coord;
ndc_coord.x = (gl_FragCoord.x * (1.0/width_half) - 1.0) * (1.0/↵
    gl_FragCoord.w);
ndc_coord.y = (gl_FragCoord.y * (1.0/height_half) - 1.0) * (1.0/↵
    gl_FragCoord.w);
ndc_coord.z = (gl_FragCoord.z * 2.0 - 1.0) * (1.0/gl_FragCoord.w);
ndc_coord.w = 1.0/gl_FragCoord.w;
vec4 unproj_coord = gl_ModelViewProjectionMatrixInverse * ndc_coord;
```


APPENDIX C IEEE VISUALIZATION 2008 POSTER

A Fast GPU Particle System Approach for Isocontouring on hp-Adaptive Finite Element Meshes

C. A. Pagot¹ J. E. Vollrath² J. L. D. Comba¹ D. Weiskopf²

¹Instituto de Informática, Federal University of Rio Grande do Sul, Brazil

²Visualization Research Center (VISUS), University of Stuttgart, Germany

ABSTRACT

The hp-adaptive finite element (FE) method is a discretization scheme which is increasingly finding application in numerical solvers. Most existent visualization tools, however, are based on linear graphic primitives and corresponding low order interpolation schemes. Consequently, pragmatic approaches in such a setting follow a resampling strategy that may introduce a loss of information, prohibitive increase of memory consumption, or additional complex data structures which are likewise difficult to handle for visualization. In this work we advocate the use of a GPU-based particle system for isocontouring of scalar quantities defined on hp-adaptive FE meshes. We present the status of our ongoing research which shows that such an approach fulfills three important requirements: i) preservation of the original data, ii) control over memory consumption and iii) interactive exploration.

Index Terms: I.3.1 [Computer Graphics]: Hardware Architecture - Graphics processors—Parallel processing; I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics data structures and data types

1 INTRODUCTION

Hp-adaptive FE discretization methods can be employed in scientific and engineering applications, such as discontinuous Galerkin formulations, ranging from aeroacoustics to hydrodynamics. By allowing the elements in the simulation to be refined, and the polynomial order within the element domain to be increased, these methods are able to handle complex geometries while offering good convergence properties. The visualization of the data produced by this method is essential for scientists to understand the simulations. Isocontouring methods such as Marching Cubes [3] are not directly applicable to high order data. Linear graphic primitives and low order interpolants available in visualization systems require elaborate resampling schemes to represent such data with sufficient accuracy and tolerable memory consumption.

In this work we propose a simple and effective particle system approach to interactively and accurately visualize isocontours of scalar quantities defined on hp-adaptive FE meshes. We focus our attention to arbitrary polyhedral elements and polynomials of degree up to 6, given analytically on a monomial basis in object space. Our solution has the following characteristics: a compact memory layout for polynomial coefficients; an efficient framework for polynomial evaluation based on a greedy approach to the multivariate Homer scheme; fast isocontouring formulated as a highly parallelized root finding algorithm in the CUDA [6] programming language; computational efficiency since the computation of the isocontour is view-independent for a given isovalue, as opposed i.e. to raycasting methods.

2 RELATED WORK

Remacle *et al.* [8] proposed an adaptive resampling scheme with guaranteed error bounds for the visualization of higher order finite elements and thus effectively transform the problem into one of adaptive mesh refinement (AMR). Figueiredo *et al.* [2] presented two physically based methods to generate triangle meshes from implicit surfaces. The first approach is based on a mass-string system while the second is based on a particle system. Witkin and Heckbert [10] developed an iterative technique, based on particle systems, to model and visualize implicit surfaces. When used as a modeling tool, the points are used as handles that deform the implicit surface. When used as a visualization tool, the particles are projected onto the surface and rendering is performed via splatting. The size and distribution of the splats over the surface take curvature into account and are controlled through several parameters.

Meyer *et al.* [4] presented a particle system based method to sample high order surfaces generated by FE simulations. Particle projection is based in the same procedure presented by Witkin and Heckbert. On the other hand, their approach is based on data sets where the high order functions must be evaluated in a reference space, which involves expensive mapping operations. In this algorithm the projection and repulsion of particles are interleaved until the system converges. Despite its effectiveness, the procedure is not interactive, taking several minutes to project thousands of points. Nelson and Kirby [5] proposed an isocontour raytracing solution for spectral/hp-adaptive FE which projects a polynomial onto the viewing ray of each pixel in the image plane with a subsequent root finding along each ray. To the best of our knowledge, Kooten *et al.* [9] presented the first interactive implicit surface visualization method based on particle systems that runs entirely on a GPU. Points are projected through the same procedure described by Witkin and Heckbert, but neighbor search required for inter-particle repulsion is accelerated using a spatial hash structure.

3 A GPU-BASED PARTICLE SYSTEM APPROACH

The hp-adaptive FE data used in this work originates from discontinuous Galerkin (DG) simulations in which the solution is given for each element based on its geometry (which can have an arbitrary polyhedral shape) and polynomials. Matters are simplified by the fact that polynomials are defined in object space. Therefore, an intricate mapping between object and reference space is unnecessary. We assume polynomials to be given on a monomial basis of the form:

$$P(\mathbf{x}) = \sum_{i+j+k < n} c_{i,j,k} x^i y^j z^k \quad (1)$$

with $i, j, k \in \mathbb{N}_0$, the maximum order n , the coefficients $c_{i,j,k}$ and the object space coordinates x, y, z . Given n , which is known beforehand and constant for one scalar quantity per element, we define an ordering rule for the list of coefficients C_l by the following algorithm:

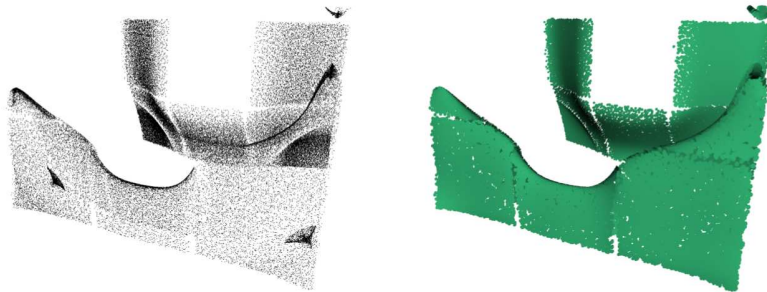


Figure 1: Point density appears to be higher in areas of high surface curvature (left). Rendering with discs and surface lighting (right).

```

l = 0
for k = n-1.0 do
  for j = n-k-1.0 do
    for i = n-k-j-1.0 do
      C_l = c_i,j,k
      l += 1
    end
  end
end
end

```

Algorithm 1: Ordering rule for polynomial coefficients

A straightforward strategy is to evaluate Equation 1 in the same fashion as Algorithm 1, computing the exponents of x, y, z explicitly. Instead, our solution uses the greedy multivariate Horner scheme of Ceberio and Kreinovich [1] which we have implemented in C++ and CUDA. Empirical results indicate that using this evaluation scheme, the resulting number of multiplications is equal to the number of summations, which we believe to be the theoretical optimum.

Isocontouring is formulated as a root finding problem with initially uniformly distributed particles per element. Elements are processed successively whereas particles of one element are processed in parallel by spawning a CUDA thread per particle solving $P(\mathbf{x}) - I = 0$ by successive Newton-Raphson iterations for a given isovalue I . Since this requires ∇P , derivatives in x, y and z direction are computed analytically in a preprocessing step. Coefficients of P, P_x, P_y and P_z are interleaved and stored in a vector of `float4` which resides in constant global device memory. This interleaving results in a minor memory overhead since the degrees of derivatives and P differ by one. However, the benefit of this layout is that it allows for a 16 byte stride which enables quick coalesced memory access on the GPU. A particle is modeled with a `struct` containing the position in space and a flag describing its state (three floats and one int). Particles can thus be arranged in a vector with 16 byte stride, which again allows for efficient memory access. Depending on the size of the data it is either possible to stream coefficients and particles to the GPU per element or to store the dataset entirely in device memory. Measurements show that with all data residing locally on device memory, the proposed framework is capable of performing 95 million Newton-Raphson iterations per second on an NVIDIA GeForce 8800 GTX card. Therefore, for a 10 million particle system and 10 Newton-Raphson iterations on average until convergence, the computation roughly lasts a second.

Many particle-based isocontouring methods employ inter-particle repulsion in order to distribute particles evenly on the surface, which is a time consuming task. In this work we propose to intentionally omit the repulsion step based on two observations: First, the projection of particles onto the isocontour can be done quickly on a GPU, as already observed by Kooten *et al.* [9]. Second, it appears that gradient based methods already distribute par-

ticles onto the isocontour with curvature-dependent density, as observed by Figueiredo *et al.* [2], but not further explored. Figure 1 shows some preliminary results.

4 DISCUSSION AND FUTURE WORK

The presented approach is subject of ongoing research which we have decided to share with the community before a subsequent detailed publication. In the immediate future we plan to further explore both the use of local surface curvature and parallelized inter-particle repulsion, for which an n-body approach along the lines of Lyland [7] appears promising.

ACKNOWLEDGEMENTS

The authors wish to thank Christoph Altmann and Claus-Dieter Munz of the Institut für Aerodynamik und Gasdynamik (IAG) at the University of Stuttgart for providing the simulation data.

REFERENCES

- [1] M. Ceberio and V. Kreinovich. Greedy Algorithms for Optimizing Multivariate Horner Schemes. *SIGSAM Bull.*, 38(1):8–15, 2004.
- [2] L. H. de Figueiredo, J. de Miranda Gomes, D. Terzopoulos, and L. Velho. Physically-based Methods for Polygonization of Implicit Surfaces. In *Proc. Graphics Interface*, pages 250–257, 1992.
- [3] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *Proc. ACM SIGGRAPH*, volume 21, pages 163–169. New York, NY, USA, 1987.
- [4] M. Meyer, B. Nelson, R. Kirby, and R. Whitaker. Particle Systems for Efficient and Accurate High-Order Finite Element Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1015–1026, Sept.-Oct. 2007.
- [5] B. Nelson, B. Nelson, and R. Kirby. Ray-tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering. *IEEE Transactions On Visualization And Computer Graphics*, 12(1):114–125, 2006.
- [6] NVIDIA Corporation. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 1.1*. NVIDIA Corporation, 2007.
- [7] L. Nyland, M. Harris, and J. Prins. Fast N-Body Simulation with CUDA. In *GPU GEMS 3*, chapter 31, pages 677–695. Addison-Wesley, 2007.
- [8] B. Rémacle, N. Chevaugnon, . Marchandise, and C. Geuzaine. Efficient Visualization of High Order Finite Elements. *International Journal for Numerical Methods in Engineering*, 69(4):750–771, 2006.
- [9] K. van Kooten, G. van den Bergen, and A. Telea. Point-Based Visualization of Metaballs on a GPU. In *GPU GEMS 3*, chapter 7. Addison-Wesley, 2007.
- [10] A. P. Witkin and P. S. Heckbert. Using Particles to Sample and Control Implicit Surfaces. In *Proc. ACM SIGGRAPH*, pages 269–277, 1994.

Figure C.1: IEEE Visualization 2008 poster.

APPENDIX D RESUMO EXPANDIDO (EXTENDED ABSTRACT IN PORTUGUESE)

Os fundamentos da pesquisa em dinâmica de fluidos computacional (DFC) foram estabelecidos na década de 60 com o artigo de Hess e Smith (HESS; SMITH, 1967). Desde então, o aumento no poder de processamento dos computadores, em conjunto com a necessidade de maior precisão nos cálculos, têm dado origem a métodos de DFC cada vez mais sofisticados. Esses novos métodos têm sido aplicados no estudo de uma vasta gama de problemas incluindo a aeroacústica (REYMEN et al., 2005; RICHTER; STILLER; GRUNDMANN, 2009), a dinâmica de gases (GALANIN; SAVENKOV; TOKAREVA, 2009; VAN DEN BERG, 2009), fluidos viscoelásticos (GUÉNETTE et al., 2008), turbo máquinas (SUN et al., 2010), transporte em meios porosos (AL-HAMAMRE; AL-ZOUBI; TRIMIS, 2010), entre outros. Entretanto, esta evolução também tem dado origem a conjuntos de dados de soluções mais complexos, cuja análise tem se tornado cada vez mais difícil. Neste contexto, as técnicas para extração de estruturas relevantes (*features*) e visualização passaram a desempenhar um papel essencial.

Métodos voltados para a extração de estruturas relevantes se concentram na análise, detecção e seleção automática de porções de dados, capturando estruturas significativas a partir de conjuntos de dados grandes e intrincados. No contexto da visualização de fluxos, exemplos de tais estruturas seriam as dobras (cristas e vales) encontradas em campos escalares, bem como as linhas de convergência, separação e centros de vórtices encontrados em campos vetoriais. Estes métodos podem reduzir significativamente a quantidade de dados a ser manipulada, permitindo que se concentre a atenção somente em regiões relevantes. As técnicas de visualização, por outro lado, fornecem uma maneira mais intuitiva e natural de inspecionar os dados através da atribuição de uma representação visual às estruturas selecionadas.

A evolução dos métodos de DFC levaram a soluções representadas analiticamente através de funções de alta ordem. Apesar de mais precisos, dados representados desta forma não são compatíveis com os métodos de visualização e de extração de estruturas desenvolvidos para operar sobre dados interpolados linearmente. Neste caso, a abordagem pragmática é a reamostragem dos dados de alta ordem, permitindo o uso dos métodos tradicionais de visualização e de extração de estruturas. Entretanto, a reamostragem não se mostra uma alternativa atraente pelo fato de poder introduzir erros relacionados a subamostragem e aumentar o consumo de memória, necessária ao armazenamento das amostras. De forma a superar essas limitações, mais atenção tem sido concentrada no desenvolvimento de métodos visualização e extração de estruturas capazes de operar diretamente sobre dados de alta ordem.

Nesta tese são propostos dois métodos que operam diretamente sobre dados de DFC

de alta ordem. O primeiro método consiste em um sistema de pré-visualização para a exploração interativa de isosuperfícies. A interatividade do método é obtida através da distribuição do esforço computacional em computações executadas nos espaços do objeto e da imagem. O segundo método consiste uma técnica eficiente para extração de estruturas lineares descritas pelo operador de vetores paralelos (*parallel vectors*). A técnica se baseia no uso de aritmética afim reduzida e processamento paralelo, desta forma permitindo o ganho de performance e garantias em relação a limites de precisão quanto à existência, posição e topologia das estruturas obtidas.

Ambos os métodos foram projetados para tirar vantagem do paralelismo do *hardware* gráfico. Resultados quantitativos e qualitativos são apresentados para ambos os métodos através de sua aplicação sobre dados sintéticos e dados gerados por simulações baseadas no método de Galerkin descontínuo. As próximas seções apresentam um resumo do funcionamento de cada um dos métodos.

D.1 Extração de Isosuperfícies

A representação aproximada de dados de alta ordem, geralmente obtida através de reamostragem, se apresenta como uma possibilidade no contexto da visualização de isosuperfícies por permitir o balanço entre a interatividade na exploração e a precisão da isosuperfície gerada. Algoritmos para a visualização de dados de baixa ordem, tais como o *marching cubes* (LORENSEN; CLINE, 1987), estão entre as soluções mais simples para este problema. Versões adaptativas deste algoritmo permitem a redução dos erros introduzidos pelo *marching cubes* ao mesmo tempo em que permitem a captura de estruturas mais complexas (REMACLE et al., 2006; SCHROEDER et al., 2006). Entretanto, abordagens baseadas na reamostragem dos dados de alta ordem introduzem erros e, em alguns casos, levam a um aumento considerável no consumo de memória, necessária ao armazenamento das amostras.

Métodos que extraem isosuperfícies diretamente a partir de dados de alta ordem normalmente formulam o problema em termos de localização de raízes ou descida de gradiente. Entretanto, a alta ordem dos dados pode inviabilizar o uso de soluções fechadas fazendo com que métodos numéricos, muitas vezes iterativos, sejam adotados. Alguns destes métodos, porém, apresentam um custo computacional elevado, degradando a performance da aplicação. Como forma de reduzir o impacto negativo na performance, algumas abordagens decidem por executar os cálculos numéricos em um estágio de pré-processamento. Este tipo de abordagem é utilizada em uma série de algoritmos para extração de malhas (REMACLE et al., 2006; SCHROEDER et al., 2005, 2006) e em algoritmos baseados em nuvens de pontos (FIGUEIREDO et al., 1992; WITKIN; HECKBERT, 1994; VAN KOOTEN; VAN DEN BERGEN; TELEA, 2007; MEYER et al., 2007). Embora estes métodos permitam a exploração interativa de uma isosuperfície computada no estágio de pré-processamento, eles não permitem a exploração dinâmica de isosuperfícies referentes a diferentes isovalores. A possibilidade de se explorar de forma interativa diferentes isovalores esta presente em alguns algoritmos baseados no traçado de raios (*ray tracing* ou *ray casting*) (WILEY et al., 2004; NELSON; KIRBY, 2006; KNOLL et al., 2009). Entretanto, a intensa avaliação de cálculos de intersecção executadas ao longo da geração das imagens leva a baixas taxas de quadros por segundo.

Desta forma, baseando-se nas características dos métodos existentes, observou-se que uma abordagem híbrida, capaz de calcular uma aproximação da isosuperfície no espaço do objeto, e rapidamente refiná-la através de computações executadas no espaço da im-

agem, poderia resultar em um método para extração de isosuperfícies interativo, sem que fosse necessária a reamostragem dos dados de alta ordem.

Sendo assim, propomos na primeira parte desta tese um algoritmo baseado em duas fases para a extração interativa e aproximada de isosuperfícies a partir de dados de alta ordem definidos sobre malhas compostas por células. Na primeira fase, partículas uniformemente distribuídas no interior do volume de dados são guiadas através do campo gradiente de forma a gerar uma amostragem inicial da isosuperfície no espaço do objeto. Na segunda fase, utiliza-se o traçado de raios no espaço da imagem, calculado a partir das vizinhanças das partículas, para refinar a representação da isosuperfície. Como a vizinhança das partículas tende a ser pequena, o traçado dos raios é iniciado já próximo à isosuperfície, levando a uma eficiente localização das raízes. A primeira fase do algoritmo, calculada no espaço do objeto, afeta a densidade das partículas podendo gerar problemas de amostragem da isosuperfície. Desta forma, também é proposta uma heurística, baseada na teoria dos sistemas dinâmicos, que adapta a vizinhança das partículas de forma a gerar uma melhor taxa de amostragem da isosuperfície.

D.2 Extração de Estruturas Lineares Descritas pelo Operador de Vetores Paralelos

Uma série de estruturas, tais como isosuperfícies e linhas de corrente, podem ser descritas por meio de equações algébricas e diferenciais. Isto permite a separação entre a descrição da estrutura e o seu procedimento de extração. Entretanto, para uma série de outras estruturas não tradicionais, esta separação entre a sua descrição e o seu método de extração não é simples de definir. Originalmente introduzido por Roth e Peikert (PEIKERT; ROTH, 1999), o operador de vetores paralelos consiste em uma ferramenta utilizada para identificar estruturas lineares em campos escalares e vetoriais. Através da formulação proposta, uma série de estruturas pode ser descrita analiticamente através de um conjunto de pontos onde dois campos vetoriais distintos se tornam paralelos.

De acordo com a proposta original, estruturas lineares são extraídas a partir de dados interpolados linearmente através da localização de suas intersecções com as faces das células da malha, que são posteriormente conectadas através de segmentos de reta. Este método é local (soluções são localizadas por célula), robusto e eficiente. Entretanto, esta abordagem não é precisa o suficiente dado que as estruturas são aproximadas através de segmentos de reta, e pode sofrer de problemas relacionados a ambiguidade topológica durante a fase de conexão dos pontos de intersecção. Neste contexto, o método de campos de fluxo de estruturas (*feature flow field*) (THEISEL; SEIDEL, 2003) fornece um método mais preciso para a extração de estruturas lineares. O campo de fluxo de estruturas foi utilizado com sucesso na extração de estruturas lineares descritas pelo operador de vetores paralelos através de um método de subdivisão espacial em dados interpolados linearmente (THEISEL et al., 2005).

Nossa abordagem pode ser vista como uma extensão deste trabalho objetivando a extração de estruturas a partir de dados de alta ordem. O método localiza estruturas lineares tais como centros de vórtices, linhas de separação e conexão, cristas, vales, etc., descritas analiticamente pelo operador de vetores paralelos. Além de permitir a separação entre a descrição das estruturas e o seu procedimento de extração, a forma analítica do operador permite sua representação no formato de inclusão. Desta forma, o método proposto localiza estruturas interessantes no contexto de visualização através da subdivisão adaptativa espacial dos dados, guiada através da avaliação do formato de inclusão do operador de

vetores paralelos. Após a fase de subdivisão espacial, um método de localização de raízes é utilizado para localizar com precisão pontos localizados sobre as estruturas. Estes pontos servirão como pontos iniciais para o traçado das estruturas, que é calculado a partir do campo de fluxo gerado para as estruturas.

D.3 Idéia Central

São propostos dois diferentes métodos para a exploração de conjuntos de dados compostos por dados de alta ordem descritos sobre malhas. Assume-se que as funções de mapeamento entre os espaços do objeto e do universo são compostas apenas por transformações afins. O primeiro método é uma técnica híbrida para a extração de isosuperfícies baseada na seguinte idéia:

A extração interativa aproximada de isosuperfícies a partir de dados de alta ordem é possível através da distribuição do processamento entre os espaços do objeto e da imagem. No espaço do objeto, uma aproximação da isosuperfície, independente do ponto de vista, é gerada. Esta aproximação irá reduzir o custo associado ao estágio de refinamento da isosuperfície, que será calculado no espaço da imagem.

Dois pontos devem ser considerados na construção de um algoritmo capaz de extrair isosuperfícies de forma eficiente neste contexto. O primeiro é o desenvolvimento de um método que aproxime a isosuperfície desejada no espaço do objeto. Este método deve ser eficiente ao mesmo tempo em que sua saída sirva como um bom ponto de partida para o passo de refinamento subsequente. O segundo ponto está relacionado ao método utilizado no espaço da imagem, que deve refinar a representação da isosuperfície baseando-se na aproximação calculada anteriormente.

O segundo método apresentado nesta tese consiste em uma técnica eficiente para extração de estruturas lineares descritas pelo operador de vetores paralelos a partir de dados de alta ordem. Este segundo método é baseado na seguinte idéia:

A representação do operador de vetores paralelos em sua forma de inclusão permite uma aproximação eficiente e precisa das reais estruturas lineares existentes em dados de alta ordem. O processamento do conjunto de dados elemento a elemento, ao invés de estágio a estágio, reduz demandas relacionadas a largura de banda necessária ao tráfego de dados ao mesmo tempo que permite o processamento dos elementos em paralelo.

Os pontos levantados em relação as declarações apresentadas incluem uma representação para a expressão do operador de vetores paralelos que seja utilizável, a escolha de uma forma de inclusão que possa ser eficientemente implementada e avaliada, e uma estratégia eficiente de subdivisão espacial do conjunto de dados baseada na avaliação da forma de inclusão do operador de vetores paralelos. A forma de inclusão do operador de vetores paralelos é representada por uma forma afim de 5 termos, derivada a partir de uma extensão da aritmética afim. A avaliação do formato de inclusão do operador de vetores paralelos é utilizada para guiar a subdivisão adaptativa do conjunto de dados, utilizando-se estruturas de dados baseadas em *octrees* e *quadtrees*. Adicionalmente, é proposta uma estratégia para a eficiente avaliação do operador de vetores paralelos que utiliza os coeficientes de seus componentes de menor ordem.

D.4 Resultados

As principais contribuições deste trabalho incluem:

- Um método híbrido para a extração eficiente de isosuperfícies a partir de dados de alta ordem;
- Um método para redimensionamento de *splats* baseado na teoria dos sistemas dinâmicos;
- Um método para extração de estruturas lineares a partir de dados de alta ordem gerados por DFC baseado na avaliação da forma de inclusão do operador de vetores paralelos;
- A análise experimental de uma estratégia eficiente de subdivisão espacial para o refino de estruturas lineares descritas pelo operador de vetores paralelos;
- Uma estratégia para a avaliação eficiente de dados de alta ordem em GPU.
- Um esquema de traçado de estruturas lineares baseado em movimentos preditores e corretores que eficientemente reparametrizam o campo no plano corretor.
- Uma estratégia para agrupamento de faces baseado em alinhamentos e que aumenta a performance do algoritmo de extração de estruturas em arquiteturas SIMD através da redução da divergência entre fluxos de execução.

D.5 Conclusões e Trabalhos Futuros

Este trabalho apresentou extensões à métodos de visualização e de extração de estruturas existentes e que permitiram a eficiente exploração de dados de alta ordem. Os métodos propostos são eficientes, escaláveis e não dependem de reamostragens. Esta seção apresenta um sumário dos principais pontos discutidos nesta tese. A seção termina com uma lista de tópicos considerados como possibilidades para futuras investigações.

O primeiro método apresentado trata da extração de isosuperfícies a partir de dados de alta ordem. O método opera sobre dados cujas funções de mapeamento contenham somente transformações afins, não conta com o uso de estruturas de dados complexas e não reamostra os dados originais. Embora não tenha sido projetado para gerar isosuperfícies com alta precisão, as superfícies resultantes apresentam muito boa qualidade. Uma série de parâmetros pode ser ajustada de forma a controlar o balanço entre a eficiência e a precisão do método, embora se possa explorar ambos (precisão e performance) através do ajuste de apenas alguns destes parâmetros.

O segundo método trata da extração de estruturas lineares descritas pelo operador de vetores paralelos. O método é composto por uma série de estágios que extraem estruturas dos dados de forma eficiente e em paralelo. Através do uso de aritmética afim reduzida podemos eficientemente subdividir o conjunto de dados de alta ordem, descartando porções de dados que seguramente não contém estruturas relevantes. Através da avaliação do campo de fluxo de estruturas, utilizando-se uma adaptação capaz de gerar vetores tangente mais estáveis numericamente, podemos reconstruir as estruturas com maior precisão. Adicionalmente, apresentamos uma estratégia para subdivisão espacial adaptativa cuja implementação em paralelo é mais eficiente que as existentes.

D.5.1 Trabalhos Futuros

Em relação ao método de extração de isosuperfícies, alguns pontos devem ser considerados em trabalhos futuros: a redução de computações redundantes durante a fase de refino das isosuperfícies, o desenvolvimento de uma heurística para uma melhor distribuição inicial de pontos, o tratamento de conjuntos de dados grandes e otimizações que considerem células menores que um pixel.

A técnica para extração de estruturas lineares também sugere uma série de possibilidades para trabalhos futuros. Um ponto a ser investigado é uma melhor comparação qualitativa entre os resultados gerados pelos métodos existentes e os gerados pelo método proposto. O melhor gerenciamento da memória pode permitir que um maior número de células possa ser processada em paralelo. Os efeitos, em termos de desempenho, do processamento assíncrono de elementos e o processamento simultâneo de células de tamanhos diferentes são possibilidades para estudos. Simplificações no estágio de traçado das estruturas pode permitir o refino progressivo destas estruturas. A extensão do método proposto, objetivando a extração de estruturas com dimensões maiores, também é considerado.

Um tópico adicional para pesquisa, e que poderia beneficiar ambos os métodos propostos, seria a manipulação de conjuntos de dados contendo funções de mapeamento não-lineares. A resolução deste problema tornaria os métodos capazes de tratar o caso geral dos dados de alta ordem. Outro tópico interessante para pesquisa é a remoção de discontinuidades no caso de dados descontínuos.

REFERENCES

- AL-HAMAMRE, Z.; AL-ZOUBI, A.; TRIMIS, D. Numerical investigation of the partial oxidation process in porous media based reformer. **Combustion Theory and Modelling**, [S.l.], v.14, n.1, p.91–103, 2010.
- BANKS, D. C.; SINGER, B. A. Vortex tubes in turbulent flows: identification, representation, reconstruction. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 1994. p.132–139.
- BAUER, D.; PEIKERT, R. Vortex tracking in scale-space. In: EUROGRAPHICS / IEEE TVCG SYMPOSIUM ON VISUALISATION. **Proceedings...** [S.l.: s.n.], 2002. p.233–ff.
- BLINN, J. F. Light reflection functions for simulation of clouds and dusty surfaces. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.16, p.21–29, July 1982.
- CABRAL, B.; CAM, N.; FORAN, J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In: VOLUME VISUALIZATION, 1994., New York, NY, USA. **Proceedings...** ACM, 1994. p.91–98. (VVS '94).
- CALLAHAN, S. et al. Hardware-assisted visibility sorting for unstructured volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.11, n.3, p.285–295, May-June 2005.
- CAMERON, G.; UNDERILL, P. Rendering Volumetric Medical Image Data on a SIMD Architecture Computer. In: THIRD EUROGRAPHICS WORKSHOP ON RENDERING. **Proceedings...** [S.l.: s.n.], 1992.
- CEBERIO, M.; KREINOVICH, V. Greedy algorithms for optimizing multivariate Horner schemes. **SIGSAM Bull.**, New York, NY, USA, v.38, n.1, p.8–15, 2004.
- CHAMP, P. E. B. de. Note sur les lignes de faite et de thalweg. **Comptes rendus hebdomadaires des séances de l'académie des Sciences**, [S.l.], v.39, p.647–648, 1854.
- CIGNONI, P. et al. Speeding up isosurface extraction using interval trees. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.3, n.2, p.158–170, 1997.
- COMBA, J. L. D.; STOLFI, J. Affine arithmetic and its applications to computer graphics. In: VI BRAZILIAN SYMPOSIUM ON COMPUTER GRAPHICS AND IMAGE PROCESSING. **Proceedings...** [S.l.: s.n.], 1993. p.9–18.

CRAWFIS, R. A.; MAX, N. Texture splats for 3D scalar and vector field visualization. In: IEEE VISUALIZATION CONFERENCE, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 1993. p.261–266.

CROSSNO, P.; ANGEL, E. Isosurface extraction using particle systems. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 1997. p.495–498.

CUDA. <http://www.nvidia.com>, Website.

CULLIP, T. J.; NEUMANN, U. **Accelerating Volume Reconstruction With 3D Texture Hardware**. Chapel Hill, NC, USA: [s.n.], 1994.

DIETRICH, C. et al. Edge Transformations for Improving Mesh Quality of Marching Cubes. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.15, n.1, p.150–159, 2009.

DREBIN, R. A.; CARPENTER, L.; HANRAHAN, P. Volume rendering. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.22, p.65–74, June 1988.

EBERLY, D. **Ridges in Image and Data Analysis**. [S.l.]: Kluwer Academic Pub., 1996. (Computational Imaging and Vision).

ENGEL, K.; KRAUS, M.; ERTL, T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In: ACM SIGGRAPH/EUROGRAPHICS WORKSHOP ON GRAPHICS HARDWARE, New York, NY, USA. **Proceedings...** ACM, 2001. p.9–16. (HWWS '01).

FIGUEIREDO, L. H. de et al. Physically-based methods for polygonization of implicit surfaces. In: GRAPHICS INTERFACE '92. **Proceedings...** [S.l.: s.n.], 1992. p.250–257.

GALANIN, M.; SAVENKOV, E.; TOKAREVA, S. Solving gas dynamics problems with shock waves using the Runge-Kutta discontinuous Galerkin method. **Mathematical Models and Computer Simulations**, [S.l.], v.1, p.635–645, 2009.

GAMITO, M. N.; MADDOCK, S. C. Ray casting implicit fractal surfaces with reduced affine arithmetic. **The Visual Computer**, Secaucus, NJ, USA, v.23, n.3, p.155–165, 2007.

GARLAND, M.; HECKBERT, P. S. Surface simplification using quadric error metrics. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 24., New York, NY, USA. **Proceedings...** ACM Press/Addison-Wesley Publishing Co., 1997. p.209–216. (SIGGRAPH '97).

GASSNER, G.; LÖRCHER, F.; MUNZ, C. D. A Discontinuous Galerkin Scheme based on a Space-Time Expansion II. Viscous Flow Equations in Multi Dimensions. **Journal of Scientific Computing**, New York, NY, USA, v.34, p.260–286, March 2008.

GAVRILIU, M. et al. Fast extraction of adaptive multiresolution meshes with guaranteed properties from volumetric data. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 2001. p.295–565.

GINGOLD, R.; MONAGHAN, J. SPH: theory and application to non-spherical stars. **Monthly Notices of the Royal Astronomical Society**, [S.l.], v.181, p.375–389, 1977.

- GUAN, S.; LIPES, R. G. **Innovative volume rendering using 3D texture mapping**. [S.l.]: SPIE, 1994. 382–392p. v.2164, n.1.
- GUÉNETTE, R. et al. An adaptive remeshing strategy for viscoelastic fluid flow simulations. **Journal of non-newtonian fluid mechanics**, [S.l.], v.153, n.1, p.34–45, 2008.
- HAASDONK, B. et al. Multiresolution visualization of higher order adaptive finite element simulations. **Computing**, New York, NY, USA, v.70, n.3, p.181–204, 2003.
- HALLER, G. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. **Physica D**, [S.l.], v.149, n.4, p.248–277, 2001.
- HARALICK, R.; SHAPIRO, L. **Computer and Robot Vision**. [S.l.]: Addison-Wesley, 1992. v.I.
- HESS, J.; SMITH, A. Calculation of potential flow about arbitrary bodies. **Progress in Aerospace Sciences**, [S.l.], v.8, p.1–138, 1967.
- KAJIYA, J. T.; VON HERZEN, B. P. Ray tracing volume densities. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.18, p.165–174, January 1984.
- KENWRIGHT, D.; HENZE, C.; LEVIT, C. Feature extraction of separation and attachment lines. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.5, n.2, p.135–144, 1999.
- KENWRIGHT, D. N. Automatic detection of open and closed separation and attachment lines. In: IEEE VISUALIZATION CONFERENCE, Los Alamitos, CA, USA. **Proceedings...** IEEE Computer Society Press, 1998. p.151–158. (VIS '98).
- KNOLL, A. et al. Fast Ray Tracing of Arbitrary Implicit Surfaces with Interval and Affine Arithmetic. **Computer Graphics Forum**, [S.l.], v.28, n.1, p.26–40, 2009.
- LACROUTE, P.; LEVOY, M. Fast volume rendering using a shear-warp factorization of the viewing transformation. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 21., New York, NY, USA. **Proceedings...** ACM, 1994. p.451–458.
- LEVIN, Y.; BEN-ISRAEL, A. Directional Newton methods in n variables. **Mathematics of Computation**, Boston, MA, USA, v.71, n.237, p.251–262, 2002.
- LEVY, Y.; DEGANI, D.; SEGNER, A. Graphical visualization of vortical flows by means of helicity. **AIAA Journal**, [S.l.], v.28, p.1347–1352, Aug. 1990.
- LORENSEN, W. E.; CLINE, H. E. Marching cubes: a high resolution 3D surface construction algorithm. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.21, n.4, p.163–169, 1987.
- LORENZ, E. N. A study of the predictability of a 28-variable atmospheric model. **Tellus**, [S.l.], v.17, p.321–333, 1965.
- LUCY, L. A numerical approach to the testing of the fission hypothesis. **Astronomical Journal**, [S.l.], v.82, p.1013–1024, December 1977.
- MCCORMICK, B.; DEFANTI, T.; BROWN, M. Visualization in Scientific Computing. **Computer Graphics**, [S.l.], v.21, n.6, p.2, 1987.

MESSINE, F. Extensions of Affine Arithmetic: application to unconstrained global optimization. **Journal of Universal Computer Science**, [S.l.], v.8, n.11, p.992–1015, 2002.

MEYER, M. et al. Particle Systems for Efficient and Accurate High-Order Finite Element Visualization. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.13, n.5, p.1015–1026, 2007.

MOORE, R. E. **Interval Analysis**. Englewood Cliffs, NJ: Prentice Hall, 1966.

NELSON, B.; KIRBY, R. M. Ray-Tracing Polymorphic Multidomain Spectral/hp Elements for Isosurface Rendering. **IEEE Transactions on Visualization and Computer Graphics**, Piscataway, NJ, USA, v.12, n.1, p.114–125, 2006.

PAGOT, C. et al. Interactive Isocontouring of High-Order Surfaces. In: **Scientific Visualization: Interactions, Features, Metaphors**. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010. (Dagstuhl Follow-Ups (to appear)).

PAGOT, C. et al. Efficient Parallel Vectors Feature Extraction from High Order Data. **Computer Graphics Forum**, [S.l.], v.30, n.3, p.751–760, 2011.

PAYNE, B.; TOGA, A. Surface Mapping Brain Function on 3D Models. **IEEE Computer Graphics Applications**, [S.l.], v.10, n.5, p.33–41, 1990.

PEIKERT, R.; ROTH, M. The parallel vectors operator: a vector field visualization primitive. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 1999. p.263–270.

PEIKERT, R.; SADLO, F. Height Ridge Computation and Filtering for Visualization. In: IEEE PACIFIC VISUALIZATION SYMPOSIUM. **Proceedings...** [S.l.: s.n.], 2008. p.119–126.

POVRAY. <http://www.povray.org>, Website.

REMACLE, B. et al. Efficient visualization of high order finite elements. **International Journal for Numerical Methods in Engineering**, [S.l.], v.69, n.4, p.750–771, 2006.

REYMEN, Y. et al. A 3D discontinuous Galerkin method for aeroacoustic propagation. In: INTERNATIONAL CONGRESS ON SOUND AND VIBRATION, 12. **Proceedings...** [S.l.: s.n.], 2005.

RICHTER, A.; STILLER, J.; GRUNDMANN, R. Stabilized High-Order Discontinuous Galerkin Methods for Aeroacoustic Investigations. In: **Computational Fluid Dynamics 2008**. [S.l.]: Springer Berlin Heidelberg, 2009. p.77–82.

ROETTGER, S. et al. Smart hardware-accelerated volume rendering. In: SYMPOSIUM ON DATA VISUALISATION. **Proceedings...** Eurographics Association, 2003. p.231–238. (VISSYM '03).

ROTH, M. **Automatic extraction of vortex core lines and other line-type features for scientific visualization**. 2000. Tese (Doutorado em Ciência da Computação) — ETH Zürich.

ROTH, M.; PEIKERT, R. A higher-order method for finding vortex core lines. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 1998. p.143–150.

SADLO, F.; PEIKERT, R. Efficient Visualization of Lagrangian Coherent Structures by Filtered AMR Ridge Extraction. **IEEE Transactions on Visualization and Computer Graphics**, Piscataway, NJ, USA, v.13, p.1456–1463, November 2007.

SAINT-VENANT, A. J. C. B. de. Surfaces à plus grande pente constituées sur des lignes courbes. **Bull. de la Soc. Philomath. de Paris**, [S.l.], p.24–32, March 1852.

SCHINDLER, B. et al. Predictor-Corrector Schemes for Visualization of Smoothed Particle Hydrodynamics Data. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.15, n.6, p.1243–1250, 2009.

SCHREINER, J.; SCHEIDEGGER, C.; SILVA, C. High-Quality Extraction of Isosurfaces from Regular and Irregular Grids. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.12, n.5, p.1205–1212, 2006.

SCHRÖDER, P.; STOLL, G. Data parallel volume rendering as line drawing. In: VOLUME VISUALIZATION, 1992., New York, NY, USA. **Proceedings...** ACM, 1992. p.25–32. (VVS '92).

SCHROEDER, W. J. et al. Framework for Visualizing Higher-Order Basis Functions. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 2005. p.43–50.

SCHROEDER, W. J. et al. Methods and Framework for Visualizing Higher-Order Finite Elements. **IEEE Transactions on Visualization and Computer Graphics**, Piscataway, NJ, USA, v.12, n.4, p.446–460, 2006.

SETTLES, G. S. Important developments in schlieren and shadowgraph visualization during the last decade. In: INTERNATIONAL SYMPOSIUM ON FLOW VISUALIZATION, 14., Daegu, Korea. **Proceedings...** [S.l.: s.n.], 2010.

SHIRLEY, P.; TUCHMAN, A. A polygonal approximation to direct scalar volume rendering. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.24, n.5, p.63–70, November 1990.

STRAWN, R. C.; KENWRIGHT, D. N.; AHMAD, J. Computer visualization of vortex wake system. **AIAA Journal**, [S.l.], v.37, n.4, p.511–512, 1998.

SUJUDI, D.; HAIMES, R. Identification of swirling flow in 3D vector fields. In: AIAA CFD CONFERENCE, 12., San Diego, CA. **Proceedings...** [S.l.: s.n.], 1995.

SUKHAREV, J.; ZHENG, X.; PANG, A. Tracing parallel vectors. In: VISUALIZATION AND DATA ANALYSIS. **Proceedings...** SPIE, 2006. v.6060, n.1, p.606011.

SUN, Z. et al. Efficient Finite Element Analysis/Computational Fluid Dynamics Thermal Coupling for Engineering Applications. **Journal of Turbomachinery**, [S.l.], v.132, n.3, p.031016, 2010.

THEISEL, H. et al. Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In: IEEE VISUALIZATION CONFERENCE. **Proceedings...** [S.l.: s.n.], 2005. p.631–638.

THEISEL, H.; SEIDEL, H.-P. Feature flow fields. In: EUROGRAPHICS / IEEE TVCG SYMPOSIUM ON VISUALISATION. **Proceedings...** [S.l.: s.n.], 2003. p.141–148.

- THIRION, J.; GOURDON, A. The 3D marching lines algorithm. **Graphical Models Image Processing**, Orlando, FL, USA, v.58, n.6, p.503–509, 1996.
- ÜFFINGER, M.; FREY, S.; ERTL, T. Interactive High-Quality Visualization of Higher-Order Finite Elements. **Computer Graphics Forum**, [S.l.], v.29, n.2, p.337–346, 2010.
- UPSON, C.; KEELER, M. V-buffer: visible volume rendering. In: **ACM SIGGRAPH Proceedings...** [S.l.: s.n.], 1988. p.59–64.
- VAN DEN BERG, A. "BLAST" : a compilation of codes for the numerical simulation of the gas dynamics of explosions. **Journal of loss prevention in the process industries**, [S.l.], v.22, n.3, p.271–278, 2009.
- VAN GELDER, A.; PANG, A. Using PVsolve to Analyze and Locate Positions of Parallel Vectors. **IEEE Transactions on Visualization and Computer Graphics**, Los Alamitos, CA, USA, v.15, n.4, p.682–695, 2009.
- VAN KOOTEN, K.; VAN DEN BERGEN, G.; TELEA, A. **Point-based visualization of metaballs on a GPU**. [S.l.]: Addison-Wesley, 2007. (GPU Gems 3).
- WALFISH, D. **Visualization for High-Order Discontinuous Galerkin CFD Results**. 2007. Dissertação (Mestrado em Ciência da Computação) — Massachusetts Institute of Technology.
- WALLIS, J. et al. Three-Dimensional display in nuclear medicine. **IEEE Transactions on Medical Imaging**, [S.l.], v.8, n.4, p.279–303, December 1989.
- WEIGLE, C.; BANKS, D. C. Extracting iso-valued features in 4-dimensional scalar fields. In: **IEEE SYMPOSIUM ON VOLUME VISUALIZATION**, 1998., New York, NY, USA. **Proceedings...** ACM, 1998. p.103–110. (VVS '98).
- WEINKAUF, T. et al. Stable Feature Flow Fields. **IEEE Transactions on Visualization and Computer Graphics**, Los Alamitos, CA, USA, v.17, n.6, p.770–780, 2010.
- WESTOVER, L. Footprint evaluation for volume rendering. **ACM SIGGRAPH Computer Graphics**, New York, NY, USA, v.24, n.4, p.367–376, August 1990.
- WILEY, D. F. et al. Ray Casting Curved-Quadratic Elements. In: **SYMPOSIUM ON DATA VISUALISATION. Proceedings...** [S.l.: s.n.], 2004. p.201–210. (VISSYM'04).
- WITKIN, A. P.; HECKBERT, P. S. Using particles to sample and control implicit surfaces. In: **ACM SIGGRAPH 1994. Proceedings...** [S.l.: s.n.], 1994. p.269–277.
- YAGEL, R.; KAUFMAN, A. Template-Based Volume Viewing. **Computer Graphics Forum**, [S.l.], v.11, n.3, p.153–167, 1992.
- ZHOU, Y.; GARLAND, M. Interactive Point-Based Rendering of Higher-Order Tetrahedral Data. **IEEE Transactions on Visualization and Computer Graphics**, Piscataway, NJ, USA, v.12, n.5, p.1229–1236, 2006.

“Feature Extraction and Visualization from High-Order CFD Data”

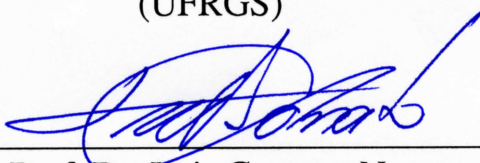
por

Christian Azambuja Pagot

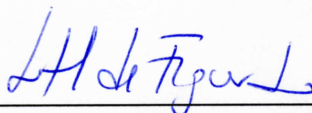
Tese apresentada aos Senhores:



Prof. Dr. Marcelo Walter
(UFRGS)



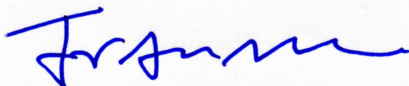
Prof. Dr. Luis Gustavo Nonato
(USP)



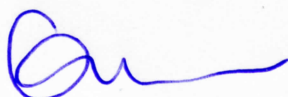
Prof. Dr. Luiz Henrique de Figueiredo
(IMPA)

Vista e permitida a impressão.

Porto Alegre, 28/09/11



Prof. Dr. João Luiz Dihl Comba
Orientador



Profa. Carla Maria Dal Sasso Freitas
Coordenadora Substituta do Programa de
Pós-Graduação em Computação-PPGC
Instituto de Informática-UFRGS