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

FÁBIO FEDRIZZI BERNARDON

Function Statistics Applied to Volume Rendering: Transfer Functions Design and Computational Issues on Discrete Functions

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

Prof. Dr. João Luiz Dihl Comba
Advisor

Prof. Dr. Cláudio Silva
Coadvisor

Porto Alegre, July 2008

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Bernardon, Fábio Fedrizzi

Function Statistics Applied to Volume Rendering: Transfer Functions Design and Computational Issues on Discrete Functions / Fábio Fedrizzi Bernardon. – Porto Alegre: PPGC da UFRGS, 2008.

112 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2008. Advisor: João Luiz Dihl Comba; Coadvisor: Cláudio Silva.

1. Volume Rendering. 2. Transfer Function. 3. Volume Simplification. 4. Image Processing. I. Comba, João Luiz Dihl. II. Silva, Cláudio. III. Title.

“That is all, folks! ”
— LOONEY TOONS

ACKNOWLEDGMENTS

I would like to acknowledge the following people and institutions for supporting this work.

The Brazilian National Council of Scientific and Technological Development (CNPq) and the Scientific Computing and Imaging Institute (SCI) from the University of Utah for partially founding this research.

Bruno Notrosso (Eletricite de France) for the Super Phoenix (SPX) dataset.

Steve Callahan (University of Utah) for the HAVS code and several datasets.

João Comba, Cláudio Silva and Mike Kirby for the guidance when I needed.

My friends from the CG group,

Carlos Dietrich, Christan Pagot, Leandro Fernandes and Renato Silveira, for the support and productive discussions along the development of this work;

Rafael Torchelsen for all great barbecues;

Vitor Pamplona for organizing weekly soccer matches;

André Spritzer, Marcos Slomp, Leonardo Schmitz, Rodrigo Barni, Giovani Kuhn, Denison Linus and Sérgio Fuji for the fun and happy hours.

And specially

Sonia Lugo for the love and understanding;

and my parents, Lurdes and Clemir Bernardon for all the love, care and guidance through my live, and the freedom to take my own choices and follow my own path.

Thank you.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS 9

LIST OF FIGURES 11

ABSTRACT 13

RESUMO 15

1 INTRODUCTION 17

2 VOLUME VISUALIZATION 21
2.1 Datasets 22
2.2 The Visualization Pipeline 23
2.2.1 The Graphics Pipeline 26
2.3 Cell Projection 28
2.3.1 Splatting 30
2.4 Ray Casting 30
2.5 Texture-Based Direct Volume Rendering 32
2.6 Time-Varying Techniques 34
2.7 Discussion 36

3 TRANSFER FUNCTIONS 37
3.1 Introduction and Related Work 37
3.2 Histograms for Transfer Function Interaction 40
3.3 Transfer Function Ensembles 42
3.3.1 Blending Strategies 43
3.4 Time Varying Transfer Functions 45
3.5 Discussion 46

4 DATA SIGNATURE ANALYSIS 49
4.1 Introduction and Related Work 49
4.2 Gradient 50
4.3 Synthetic Case Study 51
4.3.1 Ideal Boundary 52
4.4 Sampling Issues 53
4.4.1 Structured Datasets 54
4.4.2 Unstructured Datasets 54
4.4.3 Influence of Boundary Width 56
4.5 Approximation Methods 56

4.5.1 Central Differences . . . 57
4.5.2 Least Squares . . . 58
4.5.3 Reconstruction Analysis . . . 60
4.6 Smoothing . . . 64
4.6.1 Other Unstructured Grids Issues . . . 69
4.7 Discussion . . . 71

5 APPLICATIONS . . . 73
5.1 Feature Extraction . . . 73
5.2 Transfer Functions Revisited . . . 75
5.3 Discussion . . . 76
5.4 Future work . . . 76
5.4.1 Other Boundary Models . . . 76
5.4.2 Dataset Simplification . . . 77

6 CONCLUSION . . . 79

REFERENCES . . . 81

APPENDIX A THE TRANSFER FUNCTION DESIGN PROGRAM . . . 89
A.1 Scalar Range Mapping . . . 89
A.2 Evaluation . . . 93

APPENDIX B ESTATíSTICAS DE FUNÇÕES APLICADAS A VISUALIZAÇÃO VOLUMÉTRICA: CUIDADOS ESPECIAIS EM FUNÇÕES DISCRETAS . . . 95
B.1 Visualização Volumétrica . . . 95
B.2 Funções de Transferência . . . 96
B.2.1 Histogramas . . . 97
B.2.2 Agrupamentos de Funções de Transferência . . . 98
B.2.3 Dados Dinâmicos . . . 99
B.3 Análise da Assinatura de Dados . . . 100
B.3.1 Amostragem . . . 102
B.3.2 Aproximação . . . 104
B.3.3 Suavização . . . 107
B.4 Aplicações . . . 109
B.5 Trabalhos Futuros . . . 111
B.6 Conclusão . . . 111

LIST OF ABBREVIATIONS AND ACRONYMS

CAT	Computer-Assisted Tomography
CoV	Coefficient of Variation
CPU	Central Processor Unit
DVR	Direct Volume Rendering
GP	Geometry Processor
GPU	Graphics Processor Unit
HAVS	Hardware-Assisted Visibility Sorting
HRC	Hardware-Based Ray Casting
LOD	Level-of-Detail
LS	Least Squares
MLS	Moving Least Squares
MPVO	Mesh Polyhedra Visibility Ordering
MRI	Magnetic Resonance Imaging
pdf	Probability Density Function
pixel	picture element
TFE	Transfer Function Ensemble
voxel	volume element
VP	Vertex Processor
WLS	Weighted Least Squares
WSG	Weighted Sweep Graph

LIST OF FIGURES

Figure 2.1: Example of structured datasets. 23
Figure 2.2: Example of unstructured datasets. 23
Figure 2.3: The volume pipeline. 24
Figure 2.4: Example of LOD 24
Figure 2.5: Composition ordering 25
Figure 2.6: Diagram of the current Graphics Pipeline. 26
Figure 2.7: Projected Tetrahedra Decomposition 28
Figure 2.8: Example of cycle 29
Figure 2.9: Ray Casting on structured and unstructured grids 31
Figure 2.10: Depth peeling for ray casting 32
Figure 2.11: Different strategies for sampling texture-based DVR 33
Figure 2.12: 2D representation of a TSP-Tree 35

Figure 3.1: Example of widgets for TF interaction 38
Figure 3.2: Histograms used to provide users with underlying dataset information 40
Figure 3.3: TJet volume rendering using TF specified with different histograms . 41
Figure 3.4: Gradient-Magnitude histogram for structured and unstructured datasets 41
Figure 3.5: Extrusion of 1D to a 2D transfer function 43
Figure 3.6: Dimensional reduction from a 2D transfer function to a 1D one. . . . 43
Figure 3.7: Blending strategies 44
Figure 3.8: Time-varying user interface. 45

Figure 4.1: Gradient vectors computed over a 2D scalar field 50
Figure 4.2: Gradient as the normal of an iso-line 51
Figure 4.3: Synthetic dataset used as case study 51
Figure 4.4: Error function and its respective 1st and 2nd derivatives 52
Figure 4.5: Analytic evaluation of f versus f' and f versus f'' with random samples 53
Figure 4.6: Gradient-magnitude histograms of the blurred structured sphere dataset 54
Figure 4.7: Regular distribution of samples on an unstructured blurred sphere . . 55
Figure 4.8: Irregular blurred sphere with samples spread in homogeneous regions 55
Figure 4.9: Spread of samples in the boundary of the unstructured blurred sphere 56
Figure 4.10: Influence of boundary width on the histogram quality 57
Figure 4.11: Prewitt and Laplacian filters 57
Figure 4.12: Prewitt and Laplace histograms of f' and f'' in structured grids . . . 58
Figure 4.13: LS gradient-magnitude histograms for structured grids 59
Figure 4.14: LS gradient-magnitude histograms for unstructured grids 60
Figure 4.15: Gradient-magnitude deviation for LS and Prewitt in regular grids . . . 62
Figure 4.16: Dot product between analytic and reconstructed vectors in regular grids 63

Figure 4.17: Equivalence of different weighting parameters for LS 63
Figure 4.18: Gradient-magnitude deviation for LS and WLS in unstructured grids . 64
Figure 4.19: Dot product of analytic and approximated vectors in irregular grids . 65
Figure 4.20: Features identified over different concentration of points in the Engine. 66
Figure 4.21: Engine gradient-magnitude histogram with several blurring levels. . . 66
Figure 4.22: Body gradient-magnitude histogram with several blurring levels. . . . 67
Figure 4.23: Gradient-magnitude histograms of several F117 blurring levels. . . . 68
Figure 4.24: Gradient-magnitude histogram for San Fernando blurring levels. . . . 68
Figure 4.25: Gradient-magnitude histogram for SPX blurring levels. 69
Figure 4.26: Use of statistical filters to improve histogram quality for TJet dataset. 70
Figure 4.27: Use of statistical filters to improve histogram quality for Jets dataset. . 70

Figure 5.1: Viewer that correlates the histogram and spatial domains. 74
Figure 5.2: Viewer of histogram samples along with volume rendering. 75
Figure 5.3: Comparison between standard and enhanced histograms. 75
Figure 5.4: Transfer functions for unstructured grids. 76
Figure 5.5: Different boundary shape in Fighter dataset. 77

ABSTRACT

Transfer function design is an important problem that receives much attention from the visualization community. Several researches have inspired the creation of better tools and techniques to deal with volumetric datasets. There are two major classes of datasets, namely structured and unstructured grids. Most of the previous work has only addressed structured data. This work presents two groups of contributions of different nature. The first contribution is related to the general problem of transfer function design. It introduces the concept of ensembles, which are complex transfer functions created from standard types. It also presents a key-frame based approach to handle time-varying sequences. The second group of contributions is related with a study on several characteristics of unstructured data. Problems have been discovered and addressed to allow a seamless integration of classical structured grids tools to unstructured data. This work includes results that show improvements on a statistical analysis of the data, as well as the developed transfer function design system. Further work is suggested to take advantage of the enhanced version of the gradient-magnitude histogram, and explore different boundary model.

Keywords: Volume Rendering, Transfer Function, Volume Simplification, Image Processing.

Estatísticas em Funções Aplicadas a Visualização Volumétrica: Detalhes Computacionais em Funções Discretas

RESUMO

O projeto de funções de transferência é um interessante problema que recebe muita atenção da comunidade de visualização. Diversas pesquisas tem sido conduzidas para criar melhores ferramentas e técnicas que trabalham com dados volumétricos. Existem duas grandes classes de dados: volumes estruturados e volumes não-estruturados. A maioria dos trabalhos anteriores apenas se refere a dados estruturados. Este trabalho possui dois grupos de contribuições. O primeiro diz respeito ao problema clássico de especificação de funções de transferência. Primeiramente é desenvolvido o conceito de *Ensembles*, que são funções de transferência desenvolvidas a partir da combinação de funções anteriores e mais simples. Também é apresentada uma abordagem de *key-framing* para manipular dados que variam no tempo. O segundo grupo de contribuições é um estudo aprofundado sobre o comportamento de dados não-estruturados. Problemas críticos foram descobertos e tratados para permitir uma integração quase perfeita de ferramentas usadas para dados estruturados em dados não-estruturados. Os resultados mostram a melhoria de qualidade de histogramas, e também o sistema de desenvolvimento de funções de transferência. Trabalhos futuros são sugeridos para utilizar a versão melhorada do histograma de gradiente-magnitude, assim como a exploração de novos modelos de bordas.

Palavras-chave: Visualização Volumétrica, Funções de Tranferência, Simplificação de Volumes, Processamento de Imagens.

1 INTRODUCTION

The continuous advance in technology has made available unprecedent amounts of data. Scanning equipments are able to capture lots of information from the real world, and computer simulations are incredibly complex and sophisticated. With so much information, researchers need advanced techniques to efficiently analyze and understand this information.

Among the scientific areas that take advantage of digital technology to improve the quality of its work two must be emphasized. The first one is Medicine, that utilizes techniques such as Magnetic Resonance Imaging (MRI) and Computer-Assisted Tomography (CAT) to acquire information about patients. This data is then analyzed and helps doctors to better plan their course of action. They are used to diagnose diseases and plan surgeries, among other uses.

Another area that takes advantage of digital technology is Engineering, that uses physical modeling and simulation techniques. Fast processors are used to generate simulations that help scientists to understand natural phenomena. Other simulations are used to help design products, offering a safe environment where engineers can test their hypothesis without risking people's live and the environment, as well as reducing significantly the investment necessary to develop these new solutions.

Visualizing these large volumes of data is a powerful way to easy their comprehension. Visual exploration may result in an intuitive understanding of the data properties, if performed correctly. But providing insightful visualizations is a difficult task by itself and even today it is subject to many research.

Different visualization methods have been developed to deal with specific types of data (SILVA et al., 2005). For the areas described above, the information usually presents itself in the continuous 3D space, usually as a function $f : \Re^3 \rightarrow \Re$, and may have an additional temporal characteristic associated. For this type of data, the interesting features may be located over its extern boundary or anywhere inside it, or even in specific temporal locations. Volumetric visualization was developed to enable scientists to visualize the inner structures of these datasets, by for instance rendering the whole volume with semi-transparent colors.

Several methods for enhancing interesting issues of these volumes have been developed. Among them one that has great impact and has also been subject of many researches is the specification of transfer functions. Transfer functions are mappings that associate data characteristics (temperature, density, etc) with optical properties (usually color and opacity). These mappings are used by rendering methods to convert data properties into visual features that can be easily identified by researchers, providing them information about its spatial location, shape, density and so on. Just as in the visualization case, several approaches have been proposed to easy the creation of transfer functions (PFIS-

TER et al., 2001). Some of these techniques do not provide any information to users, leaving them in a try-and-error environment. Other techniques rely on data signature to automatically detect and enhance interesting data features. A third class of methods uses information of the volumes to guide users when they create these mappings.

Volumetric datasets must balance two conflicting characteristics: the accuracy of its information and its raw size. While we want to keep data as accurate as possible, we also want the dataset to be as small as possible, so it can be easily manipulated on commodity computers. Information that can be approximated using more elementary data is often excluded from the normal representation, providing a way to balance accuracy and size. If the suppressed data is required, one can approximate it using reconstruction methods.

The quality of an approximated information is directly dependent on two principal characteristics of the datasets: sample location and the reconstruction method used. Intuitively one can say that we need more samples where the interesting features are located, while homogeneous non-interesting regions can be sparsely sampled. In spite of being important, this also impacts the quality of approximation methods and restrict which ones can be used.

Different approximation methods can be used to reconstruct data from discrete samples. Here a trade-off between accuracy and computational cost of using a given method must be faced (MAVRIPLIS, 2007). Also, according to some characteristics of the datasets, some methods may have similar quality to more robust solutions.

There are several applications that use this type of information. One example is the already mentioned transfer function design. It also can be applied to guide the simplification of unstructured datasets, by reducing the amount of data stored while keeping the required information to compute the data for analysis purpose.

This work address these areas, and its contributions are basically divided in two categories. First, for transfer function specification methods, it presents the concept of Ensembles, that allow combining existing transfer functions to produce a new one that enhance or deemphasize issues of the previous ones. Ensembles can be used to perform boolean operations during dataset visualization, through the manipulation of how data are mapped to optical properties. This work also presents a key-framing approach to help the visualization of time-varying datasets. In the second category, this dissertation presents a study on function statistics. It details the impact of different reconstruction methods and sampling distributions. It also presents how this analysis helps transfer function design, as well as simplification techniques for datasets.

The structure of this dissertation follows. A brief review of volume rendering techniques is presented in Chapter 2, since they are the main application for transfer functions and serve as the main motivation for this work. Firstly the visualization pipeline is presented, with all stages usually implemented by visualization techniques. An overview of all major paradigms for volume visualization is presented next, with a review of the state-of-the-art for volume rendering.

Transfer function design is explored in Chapter 3. The contributions to transfer function specification is explained, with detailed description of Ensembles and the solution to visualize time-varying data. This discussion also motivates chapters of this work that follow.

Some issues found during the transfer function work led to the exploration of data signatures, as explained in Chapter 4. This work exposes an explanation about Function Statistics and what information they expect to provide. A synthetic case study is developed to isolate the source of errors and correctly analyze how sampling and approximation

affect the result. This chapter exposes an analysis about the issues on how to compute and present Function Statistics. Details are presented on how different characteristics affect the resulting quality of computations, as well as how to improve it.

Once the computation of Function Statistics satisfy certain quality requirements the transfer function framework is revisited, now fixing the issues previously found. This is discussed in Chapter 5, which also describes how to use the computed information to perform mesh simplification. Other applications and further research are also commented in Chapter 5.

Final conclusions about the work are presented in Chapter 6, summarizing up all contributions and discussing possible extensions of this work.

Appendix A shows the transfer function design system and some characteristics that have not been previously mentioned in this document. Appendix B has a summary of this dissertation written in the Portuguese language.

2 VOLUME VISUALIZATION

The exploration of inner structures of objects in volumetric data is required to fully understand and analyze certain classes of problems. Questions as how does the heat spread inside an engine or what is the spatial relationship of tissues inside a person's foot require a visualization technique that shows the spatial placement of materials with different properties. Volume visualization has been developed to display the inner region of a material and its border (DREBIN; CARPENTER; HANRAHAN, Avg. 1988). It is a huge area of computer graphics, responsible for developing methods to allow the exploration of data that presents the aforementioned characteristics.

The main objective of volume rendering is to compute how light behaves inside a semi-transparent object. At any position inside the volume, the light intensity may be affected by three distinct types of interaction:

- Emission: the material emits light, increasing the amount of luminous energy inside the volume.
- Absorption: the material absorbs light, thus reducing the luminous energy propagated through the volume.
- Scattering: the material changes the direction of the light, causing it to spread. Spread light will be reflected and/or absorbed by other elements inside the volume other than the source of spread, unless the scattered light leaves the volume.

Most volume visualization techniques take into account only the first two items. This is the frequently called Emission-Absorption Optical Model (also known as Density-Emitter Model). It is widely used when performing volume rendering due to its computational efficiency and good quality result. The shape of the Volume-Rendering Integral, as shown in equation 2.1 (HEGE; HöLLER; STALLING, 1993; MAX, 1995), is a result of the use of this optical model:

$$I(s_k) = I(s_{k-1})e^{-\int_{s_{k-1}}^{s_k} k(s)ds} + \int_{s_{k-1}}^{s_k} q(s)e^{-\int_{s}^{s_k} k(s)ds}ds \tag{2.1}$$

In this equation, $k(s)$ is the absorption component, $q(s)$ is the emission component, $I(s_k)$ is the light intensity at position s_k. The *optical depth*

$$\tau(s_0, s_1) = \int_{s_0}^{s_1} k(s)ds \tag{2.2}$$

is interpreted as the distance that light may travel before it is absorbed. This way $e^{-\tau(s_0,s_1)}$ represents the transparency of the volume material.

Several methods have been developed to efficiently compute the volume-rendering integral. These techniques implement the visualization pipeline, as we explain in Section 2.2. In order to provide interactivity to volume visualization many of the proposed solutions considered restrictions imposed by the traditional rendering pipeline, as we show in Section 2.2.1. The classical graphics pipeline has been developed for many years and was primarily designed to support the rendering of the boundary of 3D objects. Although not fully designed to address volume visualization requirements, its development incorporated several stages and characteristics that we can use to accelerate volume visualization.

Before looking at the volume visualization pipeline one must understand how the volumetric data is represented. There are basically two different approaches to store this information, and each requires visualization methods specifically developed for it. Section 2.1 addresses the main differences between these two data representation: Structured and Unstructured datasets.

According to the data representation chosen different sets of rendering methods will be available. For unstructured datasets one can use the Cell Projection technique, described in Section 2.3. This technique process one data element at a time, projecting them against the image. The Ray Casting method addressed in Section 2.4 is well suited for both classes of datasets. This technique process every pixel of the final image independently, allowing an efficient parallel implementation. Section 2.5 presents a final technique called texture-based direct volume rendering. It was designed specifically for a type of structured dataset, namely 3D textures.

2.1 Datasets

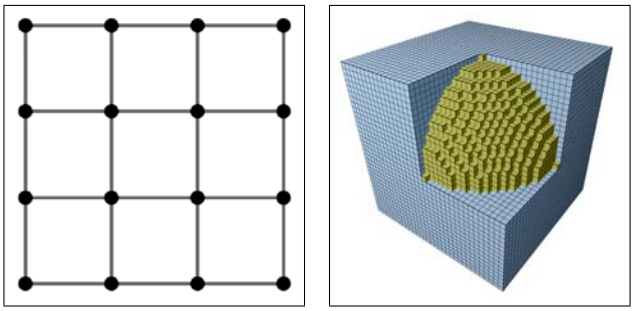
A scalar field is a function $\phi : \Re^3 \rightarrow \Re$ that assigns one or more scalar value to positions in a volumetric space (ENGEL et al., 2006). Although being continuously defined, a scalar field usually results from computational simulations or scanning procedures. Both approaches usually use and/or result in a discrete 3D field, thus requiring methods to represent and store this information.

Scanning methods have been used in Medical areas for some time. Procedures such as Magnetic Resonance Imaging (MRI) and Computer-Assisted Tomography (CAT) have been used to acquire volumetric information from patients (DREBIN; CARPENTER; HANRAHAN, Avg. 1988). These methods frequently scan the 3D space on a regular pattern, producing discrete samples stored as Structured Grids.

The most common type of structured dataset (also called regular grids) is the 3D image. An image in 2D space is commonly represented by a collection of squared elements called *pixels* (short from "picture elements"), regularly disposed in a rectangular region. Analogously for 3D, a 3D image is represented by a collection of *voxels* (short for "volume elements"), regularly distributed inside a box-shaped region. Voxels can be seen as cubes that have a given attribute associated to a single point (Figure 2.1(a)) or to its entire volume (Figure 2.1(b)).

Structured datasets do not need to explicitly store connectivity information for their elements since they have a regular, well-defined pattern. The only information needed is how is the regular placement, which is implicit for most common representations.

More flexible volume representations are required in other situations. Computational Fluid Dynamics (CFD) and Finite Element Modeling (FEM) require a data representation that adapts itself to the problem, avoiding the waste of computational resources (GAR-

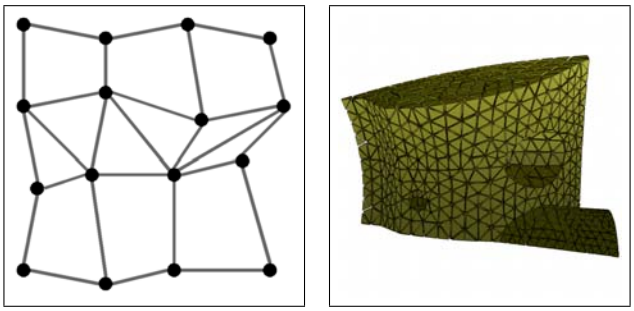


(a) Voxel-centered (b) Volumetric voxel

Figure 2.1: Example of structured datasets.

RITY, 1990). Unstructured Datasets are employed to model the scalar field used in most computer simulations in these areas.

Unstructured datasets (also known as irregular grids) allow gross representation for uninteresting or homogeneous regions while increasing data samples in regions of interest. This flexibility in distributing sample points is what makes these datasets widely used in computer simulations.



(a) Irregular grid (b) 3D tetrahedral mesh

Figure 2.2: Example of unstructured datasets.

This type of dataset requires the explicit definition of the connectivity between the sample points since it may have any possible connection scheme (as illustrated by Figure 2.2(a)). Even being this flexible, irregular grids are often formed by the same type of element, usually the simplex of the space where they are defined (like tetrahedral meshes in the 3D space, as shown in Figure 2.2(b)).

The simplicity of structured datasets allows them to be easily manipulated. They have the fastest rendering methods, and lots of techniques have been developed and shown to work around regular grids (although the fundamentals of such techniques can also be applied to unstructured datasets).

The scalar values for the spatial points not explicitly represented in both classes of datasets are approximated using an interpolation method while computing the volume rendering integral. The next section explains how the Visualization Pipeline is used to perform its computation using different types of datasets we just discussed.

2.2 The Visualization Pipeline

The volume visualization pipeline is a series of steps that are usually taken to compute the volume rendering integral presented in equation 2.1. The pipeline described here is

one of several possible pipelines used to perform volume rendering. It is a free interpretation of a pipeline presented by Engel et. al. (ENGEL et al., 2006). Figure 2.3 illustrates these steps.

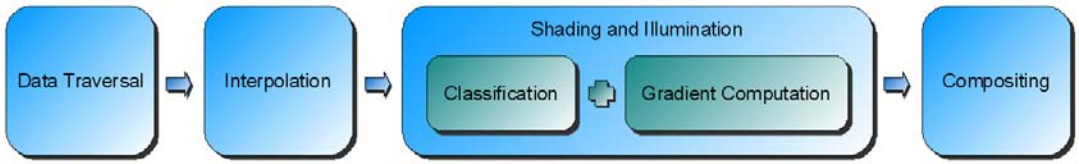


Figure 2.3: The volume pipeline.

The first stage in this pipeline, Data Traversal, defines how to sample the volume data. The way that sampling is performed will determine how the continuous rendering integral will be discretized and solved. The sampling strategy varies according to the type of the dataset, visualization method and the purpose of the visualization. Structured datasets frequently have an associated geometry responsible for sampling its scalar field. Unstructured datasets, on the other hand, frequently use its own geometric information as a sampling structure. Some rendering techniques may have a sampling strategy independent of the data representation, like Ray Casting (Section 2.4).

Another factor that determines how the scalar field will be sampled is the purpose of the visualization. Reducing the visual accuracy enables faster visualizations, resulting in a technique called Level-of-Detail (LOD). This method often drops some elements from the sampling geometry in order to reduce the computational cost of evaluating the volume visualization integral at a point. It assumes that the variation of the scalar field at that position can be interpolated from the previous and next evaluated points, as illustrated by Figure 2.4.

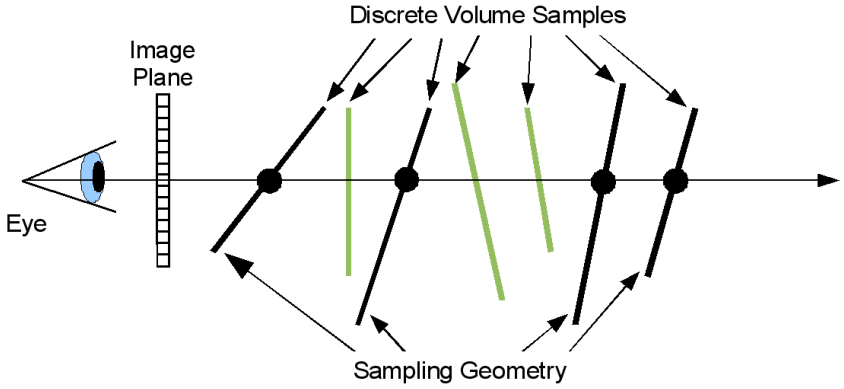


Figure 2.4: Example of LOD: only the black lines are used to sample the volume, while the green lines are skipped to increase performance.

If the sample positions do not correspond to the elements of the discrete volume, an interpolation technique (or filtering) must be used to approximate such values. This is the second stage of the visualization pipeline. Filters also have to balance accuracy and speed. High-order approximation methods, such as B-Spline curves (PARENT, 2001) have a computational cost significantly higher than less precise methods, like trilinear interpolation, and are less frequently used.

Once defined how to sample the scalar field, the evaluation of the volume rendering integral can take place. For every two consecutive points over the same viewing ray,

the color and opacity contributions are computed. This is the third stage of the pipeline, Shading and Illumination. This stage can be further refined into two different procedures:

- Classification: this stages classify the volume material, assigning visual properties to it. This stage frequently uses a custom mapping that converts the values of the scalar field to a color and opacity. This is called a Transfer Function, and will be further detailed in Chapter 3
- Gradient Computation: the gradient is a vector that indicates the direction of the scalar variation inside the volume. It is associated with a scalar value and is often used for computing illumination in volume rendering. Although being widely used for illumination, it holds other important properties as explained in Chapter 4. Section 4.2 further describes the gradient and Section 4.5 presents several methods to compute it.

Once the data has been classified and the illumination component evaluated, they are added into a single contribution element that must be combined with the previously evaluated portions of the integral. Compositing is the last stage of the pipeline and is responsible for it. There are basically two different implementations possible to the compositing stage, selected accordingly to the visualization method used, and are illustrated in Figure 2.5:

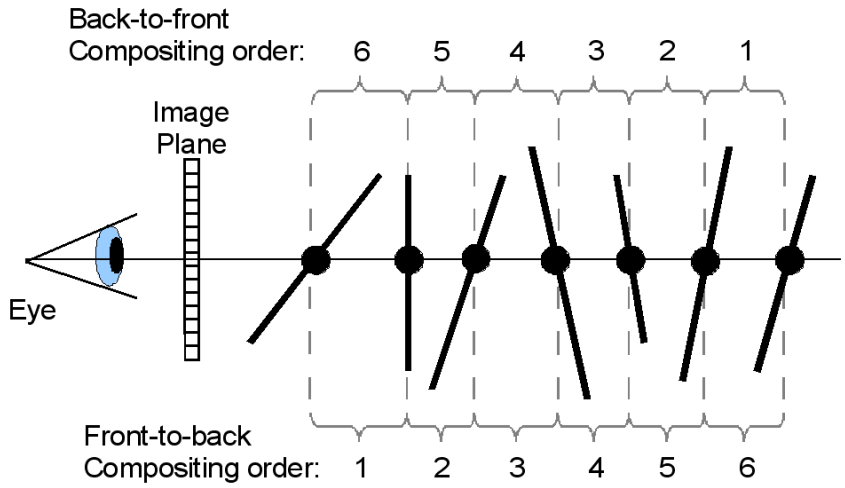


Figure 2.5: The two possible composition orders to produce a volume rendered image.

- Back-to-front: these methods combines the integral portions starting from the one farthest away from the eye position successively up to the closest one. The Texture-Based Direct Volume Rendering (Section 2.5) method frequently uses this composition method.
- Front-to-back: starting from the eye position, the integral portions are combined consecutively up to the farthest position from the eye. Ray Casting (Section 2.4) is an example of visualization technique that uses this composition scheme.

Not all stages presented in this pipeline are required. The gradient computation for illumination is an example of a stage that can be dropped to further increase performance.

This pipeline has not a direct implementation in commodity graphics cards. Researches have done much effort to implement it on rendering methods that take advantage of resources presented in the graphics pipeline. The next section explains what features are available in the graphics pipeline. Some of these features are important to accelerate volume visualization techniques.

2.2.1 The Graphics Pipeline

The graphics pipeline has been developed for many years. From time to time the graphics pipeline is revisited to incorporate changes that increase its flexibility and add new technologies. Just like the visualization pipeline, there are more than a single representation of the graphics pipeline. The one presented in this work has most of the recent technologies added to it, since it allows faster volume visualization. The diagram in Figure 2.6 shows the stages of this pipeline.

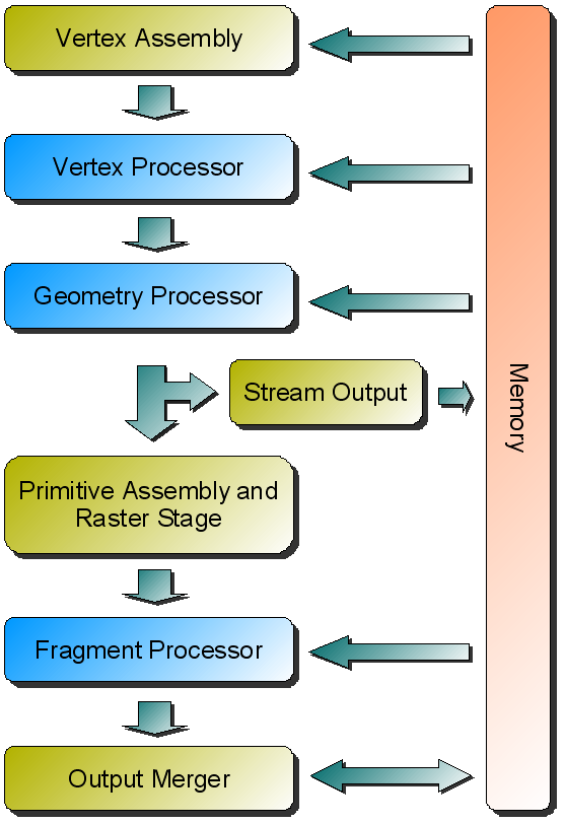


Figure 2.6: Diagram of the current Graphics Pipeline.

The Memory block (Red) is not a pipeline stage, but it represent the graphics memory. The arrows pointing to it indicate what stages have write access to the memory. Similarly, arrows leaving it indicate what stages have memory read access. Yellow stages are only configurable, so users have control over a predefined set of states, while the blue stages are programmable and execute custom programs written by developers.

The first stage of the pipeline is the Vertex Assembly. Its function is to gather all required vertex elements to start the rendering process. These elements include the vertex position, normal, color and texture coordinates among others. Once a vertex is assembled it is dispatched to the Vertex Processor.

The Vertex Processor (VP) is the second stage in the graphics pipeline, but the first

programmable one. Its function is to apply geometric transformations to vertices, modify its position, normals, etc. It can also be used to compute per-vertex illumination like Gouraud Shading (AKENINE-MöLLER; HAINES, 2002), but its use for this purpose has been significantly reduced by the implementation of higher quality procedures.

The Geometry Processor (GP) is the second programmable stage. It has been recently added to the graphics pipeline, and enables the creation and removal of vertices, since it has information about its neighborhood. It also can change the attributes of vertices, just like the VP does. Currently it only works with the same primitive type it received as input, and supports a very limited set of primitives.

The fourth stage is called Stream Output. It is a optional stage, and acts as a shortcut for programs that intend to update only a buffer in memory, avoiding the overhead of going through the remaining stages of the pipeline. It can write to memory, but can not read from it.

The Primitive Assembly stage is responsible for gathering the vertices emitted by the VP or the GP and assembling the geometric primitive being rendered. It also performs a series of optimizations to reduce the computational cost of rendering, as well as some other operations:

- Face culling: remove primitives whose vertices do not conform with a user-specified order in the clipping space, usually clockwise or counter-clockwise.

- Clipping: remove vertices that are outside the view frustum, replacing them with other vertices over the frustum border. The same procedure is used to handle user-defined clipping planes.

- Projection: Before generating the fragments, all of the vertices are mapped to the same projection image plane through the division of all its components by the homogeneous coordinate.

- Early-Z test: to reduce the overhead of processing fragments that are occluded by other fragments already processed a depth test is performed in this stage. If a fragment program is set to modify the fragment's depth position then this test is disabled.

The primitive rasterization occurs before the early-z test, right after projecting the primitives to the same image plane. It produces the fragments that are sent further down the pipeline.

The last programmable stage in the pipeline is the Fragment Processor. It usually is the unit with the fastest operation frequency. This is a very important unit for several volume rendering methods, since it has hardware support for 3D textures, with different interpolation filters available. It also has fast memory access due to efficient cache policies, increasing its use among techniques that require lots of memory reads.

The last stage in the pipeline is the Output Merger. It is the only stage with allowed read-write operations to memory. Among the operations it performs before writing the fragment to the rendering buffer are the following:

- Depth test: check the depth value of the current fragment against the one previously written in the same position, using a comparison function. If the test fails, the current fragment is discarded.

- Stencil test: similar to the depth test, but here the comparison is against an arbitrary mask, that is updated accordingly to a configured state.
- Alpha test: discard fragments whose color alpha value fail to pass a preset function.
- Blending: performs the color combination of the current fragment with the previously written one to a given position. It is another important stage for volume rendering, and corresponds to the composition stage in the volume rendering pipeline (Figure 2.3). It has several parameters that can be configured to match the desired composition function.

Besides not being developed with volume visualization as a primary goal, the graphics pipeline implemented in most Graphics Processor Units (GPU) has several stages that can be used to increase the performance of volume visualization techniques. The following sections will discuss different strategies to render volumetric datasets.

2.3 Cell Projection

The visualization of volumetric data requires the rendering of internal features of objects, but the traditional rendering pipeline described in Section 2.2.1 was optimized for the rendering of boundary triangular meshes.

A volume visualization techniques designed to fit in this rendering model is the Projected Tetrahedra algorithm (SHIRLEY; TUCHMAN, 1990). The volume data for this method is discretized in a tetrahedral mesh. Each tetrahedra is classified according to its projection in the view plane, and decomposed into triangles that are efficiently processed by the graphics hardware (see Figure 2.7). The color and opacity contributions are computed for each vertex and interpolated for the remaining elements of the dataset. This method is best suited for smoothly-changing data, since it has some visual artifacts resulting from the linear interpolation of the colors. The visual quality can be improved by the use of texture tables, reducing the amount of visual artifacts (STEIN; BECKER; MAX, 1994), perspective-correct interpolation and pre-integration (KRAUS; QIAO; EBERT, 2004; MORELAND; ANGEL, 2004).

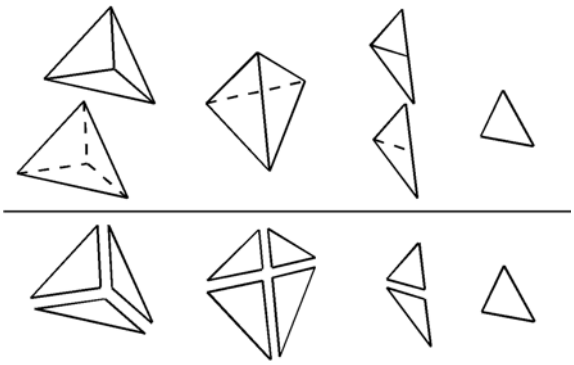


Figure 2.7: Decomposition of cells used by the Projected Tetrahedra method. The tetrahedra projection cases are shown above the line, and the associated triangles are shown below.

The cell projection method needs the elements to be drawn in a specific order, usually back-to-front (see Figure 2.5). The Mesh Polyhedra Visibility Ordering (MPVO) algorithm enables a fast ordering of a convex set of convex polyhedra (WILLIAMS, 1992),

creating a Direct Acyclic Graph (DAG) for a given viewpoint on a particular mesh. An extended version of the MPVO (XMPVO) handles non convex meshes and even disconnected meshes without using heuristics previously used on the original MPVO that did not ensure order accuracy. The view-dependent DAG can be replaced by a view-independent structure that further increases the performance of sorting mesh elements (COMBA et al., 1999). A layer-based technique can also be used to sort the primitives using successive rendering steps, similar to a depth-peeling approach (KRISHNAN; SILVA; WEI, 2001; EVERITT, 1999).

None of the previous techniques are robust enough to deal with mesh cycles without the use of other algorithms. Figure 2.8 illustrates a cycle problem, where an order of the letters “S”, “C” and “I” can not be automatically established.

Figure 2.8: Example of a cycle: an ordering can not be implicitly established for the letters “S”, “C” and “I”, since each letter overlaps another one and is overlapped by the remaining letter.

The ZSWEEP algorithm (FARIAS; MITCHELL; SILVA, 2000) handle cycles in meshes and is faster than previously described methods. It uses a sweep plane to process each vertex from the one closest to the view plane up to the farthest one. When the plane encounters a vertex all faces incident to it that extend themselves away following the sweep direction are projected, creating a list of intersections with its respective z-coordinate. After this, the mesh is projected and the color contribution is finally computed. This technique is efficiently parallelized, using a tile-based approach (FARIAS; SILVA, 2001). The image plane is divided in several tiles that are independently processed. When a tile is done, the processor moves to the next available one. This process is repeated until all tiles are processed.

Another solution for cycles is the use of an additional buffer to solve the cycle problem for each polygon individually, reducing the contribution of the obstructing region to the final image (KRAUS; ERTL, 2001). These images are rendered interlaced with the polygons to mask out the obstructed region, thus leading to a correct color composition.

A recent approach for rendering unstructured grids is the Hardware-Assisted Visibility Sorting (HAVS) (CALLAHAN et al., 2005). This technique considers the faces of the volume elements. For a face shared by two elements, only one instance is processed. The faces are partially sorted by their centroids on the CPU before rendering, so they are rendered in nearly optimal order. The elements that are slightly out of their position are placed in the correct order by a final sort that takes place in a per-fragment basis on the GPU, using a structure called the k-buffer which can correct elements that are out of their place by up to k positions.

The HAVS method can be used along with LOD techniques (CALLAHAN et al.,

2005) that allow interactive manipulation of large datasets, balancing performance and render quality. It works by discarding internal faces from the dataset, reducing the amount of elements to sort and render. There is no need to store topological information since no connectivity is required. Different metrics for assigning importance to faces can be used, and an user can select one that is best suited for a particular dataset.

Another extension to HAVS allows it to handle time-varying data (BERNARDON, 2005; BERNARDON et al., 2006). The time-varying data is hierarchically decomposed and compressed using vector quantization (LINDE; BUZO; GRAY, 1980), taking advantage of the temporal coherence between consecutive time steps. The quality of this compression approach can be increased by expanding the quantization tables, allowing a flexible compromise between storage space and reconstruction quality. This framework was extended to take advantage of multiple cores, trying to maximize the efficiency of the massively parallel architecture of current commodity computers (BERNARDON et al., 2007).

A somewhat similar rendering technique to Cell Projection is to project the volume sampling point itself against the image plane, as described next.

2.3.1 Splatting

The rendering of a geometric primitive fills a region in the image place correspondent to its projection. The continuous volume is discretized into sampling points, which are non-dimensional elements. To render such elements, a data structure that associated such points with geometric data to produce the final image is used. The idea behind Splatting is to associate a "region of effect" for every point (WESTOVER, 1990).

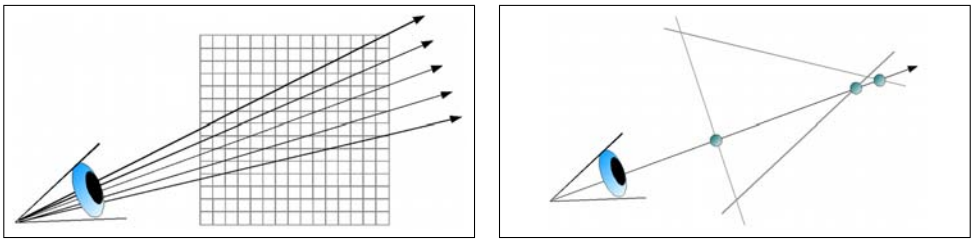
It works by sorting the samples for projection, similar to the standard Cell Projection technique. But instead of rendering a geometry associated to the sampling points, a reconstruction kernel is used to fill a region around the projection of the point.

This method is robust enough to handle both classes of datasets. Structured grids can be traversed voxel-by-voxel in the order required to render, eliminating the cost of explicitly sorting its elements. Unstructured datasets require a sorting method since its samples are arbitrarily scattered. Splatting techniques have become a large and active area of research (GROSS; PFISTER, 2007).

2.4 Ray Casting

Another method commonly used to render geometry instead of raster polygonal faces is the Ray Casting method. This method is a variation of a more complex technique, called Ray Tracing (WHITTED, 1980; PURCELL et al., 2002). Ray tracing works by launching a ray for every pixel in the image plane, starting from the eye position. When a ray hits a surface, it can be reflected and refracted in one or more directions, simulating scattering effects. After a number of iterations the final color is produced and stored in its associated pixel. This technique is particularly useful for computing specular highlights and inter-object reflections.

The Ray Casting method is a simplified version of Ray Tracing. It does not allow ray reflection or refraction, keeping the ray direction constant, and usually terminates its computation after finding the first intersection point against the geometry. For volumetric visualization, though, a ray is processed until its color contribution results in an opaque value. Figure 2.9 shows an example of rays being casted through structured (2.9(a)) and unstructured (2.9(b)) volumes.



(a) Structured grid (b) Unstructured grid

Figure 2.9: Example of Ray Casting through both structured (a) and unstructured (b) grids.

The first implementations of Ray Casting for volume visualization did not traverse the scalar field directly. Instead, it traverses a regular color and opacity volumes created using a transfer function that maps scalar data to color and opacity (LEVOY, 1988, 1989).

A technique to traverse unstructured data was developed shortly after the first visualizations of structured grids (GARRITY, 1990). It was used on datasets with elements that share faces (like tetrahedra). For two elements that share a face, only one is considered, eliminating errors caused by duplicated faces.

Regular grids can be efficiently stored using a multiresolution representation based in wavelets (WESTERMANN, 1995). It exploits the spatial coherence of data to reduce the size of a compressed volume. This technique was posteriorly revisited and a variation was proposed where vector quantization was used to select the most representative difference vector for each hierarchical level, except the base level (SCHNEIDER; WESTERMANN, 2003). Its decompression phase has the advantage of executing inside modern graphics hardware, reducing the rendering time.

Before modern commodity GPUs were able to efficiently manipulate regular datasets a technique was developed to convert them into unstructured grids (LüRIG; GROSSO; ERTL, 1997). It has the advantage that homogeneous regions can be merged together into larger elements. Since ray integration is performed from face to face the merging procedure reduces the amount of data to be processed, reducing the cost of ray integration. Non-convex and non-connected meshes can be processed using a plane-sweep approach to capture re-entrance points for rays that leave a connected mesh region (SILVA; MITCHELL, 1997).

An alternative implementation of ray casting for unstructured meshes stores a list of faces for each pixel (BUNYK; KAUFMAN; SILVA, 1997). The final color contribution for a pixel is only computed after listing all faces for that pixel, that are sorted and traversed in a predefined order.

Object importance can be used to selectively render a feature of a dataset with context information around it, but without obstructing the feature (VIOLA; KANITSAR; GRÖLLER, 2004). In a pre-processing phase the maximum importance for each ray is found. Rays traverse the material without accumulating the color and opacity until it reaches the region with higher importance. Once the region is reached, the ray integration is normally performed as usual.

High performance Ray Casting computation can be achieved using a GPU implementation (WEILER et al., 2003). The Hardware-based Ray Casting (HRC) technique encodes mesh geometry into textures that are retrieved and processed in the fragment processor. Each fragment corresponds to a ray, where the only information available is the cell the ray is currently hitting. Using this information, the remaining description of

the cell is accessed using texture fetches and the exit point for that ray is computed, and the next cell determined. An invalid element index is used to mark when a ray left the mesh.

A problem with the original HRC approach was handling non-convex meshes. In fact, it only handled convex (or convexified) meshes. This problem was solved using a technique called depth-peeling (EVERITT, 1999). Through a preprocessing step the algorithm captures the re-entrances of convex regions, as shown in Figure 2.10 (WEILER et al., 2004; BERNARDON et al., 2006).

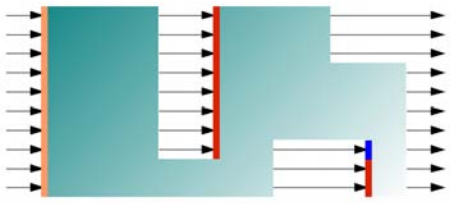


Figure 2.10: The use of depth peeling allows the Hardware-based Ray Casting to handle non-convex meshes.

More efficient structures where used to increase the computational efficiency of the algorithm (KRUEGER; WESTERMANN, 2003; BERNARDON, 2005):

1. mesh encoding: reduce memory access cost for HRC, using a more compact and efficient data structure based on 2D textures.

2. tiles: divide the screen in tiles, producing fragments (and consequently rays) only for portions of the screen that are still processing.

3. early ray termination: stop processing rays before they leave the mesh if their accumulated opacity exceeds the opaque threshold.

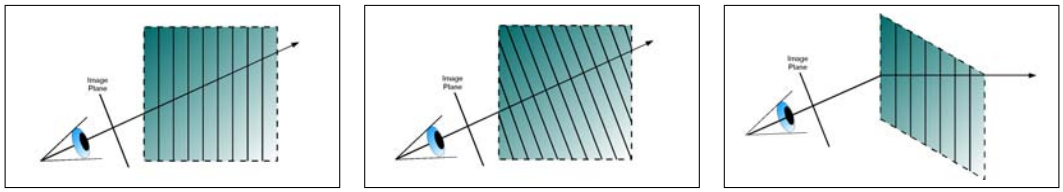
Another use for Ray Casting is the computation of iso-surfaces. Iso-surfaces are surfaces created in regions of the scalar field with a specific value (called iso-value). Using a technique called Volumetric Depth-Peeling, a Ray Casting program can efficiently render interior and exterior iso-surfaces in a single rendering pass.

Although being robust enough to handle unstructured and structured grids, ray casting is not the most widely-used rendering method. For structured datasets, texture-based direct volume rendering is the most widespread technique. It will be discussed in the next section.

2.5 Texture-Based Direct Volume Rendering

Texture-Based Direct Volume Rendering is a class of rendering methods developed exclusively for regular grids. Data is represented using a 3D texture (as the name suggests), or a set of 2D textures with the same dimension. The methods under this category use planes to traverse and sample the scalar field. The planes can be aligned with the dataset axis, the viewing direction or transformed to different spaces, as illustrated in Figure 2.11.

The first techniques to use these methods pre-classified the volume materials, aligned it with a viewing axis and then orthogonally projected it into the image plane, as shown in



(a) Axis-aligned planes (b) View direction aligned planes (c) Shear space planes

Figure 2.11: Different strategies for sampling the scalar field: (a) axis-aligned planes, (b) view direction aligned planes and (c) shear space planes.

figure 2.11(a) (DREBIN; CARPENTER; HANRAHAN, Avg. 1988). It produces artifacts when changing the axis the planes are aligned to, and changing the axis is necessary to avoid planes parallel to the viewing direction (which causes holes in the final image).

Another method to sample regular grids represented as 3D textures is to draw planes always aligned with the viewing direction, as shown in figure 2.11(b). The Weighted Sweep Graph (WSG) generates slicing polygons aligned with the viewing direction (DIETRICH et al., 2004). It is used to correctly generate vertices and texture coordinates for sampling the scalar field, resulting in geometry only inside the volume dimensions.

The shear-space technique has a simple and efficient projection of slicing planes, as shown in figure 2.11(c) (LACROUTE; LEVOY, 1994). For orthogonal projection only a translation of the slice planes is necessary. For perspective projection, a scale must be applied along with the translation. An intermediate image aligned with the volume is used as a base plane, where all slicing planes will be projected in a direction perpendicular to the image plane. This image is then warped to the final image plane.

Several methods have been developed to further improve the rendering efficiency. Wavelet-based compressed hierarchical representation of regular grids can be used to reduce the size of datasets at the cost of accuracy, so larger datasets can be efficiently processed and visualized (GUTHE et al., 2002). It allows different decompression levels according to a LOD controlled by the user. An extension to this approach performs a hierarchical volume decomposition coupled with a covariance analysis and vector quantization to minimize memory usage to store regular grids (SCHNEIDER; WESTERMANN, 2003). The compressed data is stored as textures, which allows fast decompression using the GPU. Other optimizations include volume subdivision and skipping of transparent regions, efficient use of transfer functions, clipping and lighting (LI; MUELLER; KAUFMAN, 2003; ROETTGER et al., 2003).

Direct rendering of compressed volumes may result in lower quality images not only due to the loss resulting from the compression method, but also for lacking operations available to non-compressed ones, like filtering. To solve this problem one can use a deferred filtering algorithm (FOUT et al., 2005). This method decompress two to four consecutive texture slices in a first pass, and render sampling slices between them in a second pass, taking advantage of hardware interpolation in the decompressed textures and manual interpolation from these to the sampling point.

Direct volume rendering can be combined with other rendering methods (such as iso-surfacing and non-photorealistic rendering) to enhance the final image, in an approach known as Two-Level volume rendering (HADWIGER; BERGER; HAUSER, 2003). It uses multiple render passes for each visualization method, and compensates occlusion using a shared depth buffer priorly computed. The images produced by the individual methods are blended together at a final stage, producing a final image that combines the

different rendering methods.

Another method proposed the combination of characteristics using different fusion algorithms (MANAGULI; YOO; KIM, 2005). These methods were developed to handle different types of ultrasound imaging:

1. Composite Fusion: rendering of RGB values, in contrast with the other two fusion techniques. Good depth cue but assign artificial colors to the volume.

2. Post Fusion: volumes rendered independently and fused in a post-process using alpha blending. Good correlation between materials, but lack depth information.

3. Progressive Fusion: volume composition occurs at several stages, during and after rendering. Has depth information and preserves the color, but results in more opaque volumes than the two other techniques.

Fusion methods are going to be explored in chapter 3 in the context of transfer function creation instead of direct volume rendering.

Texture-based DVR also has methods developed for High Dynamic Range Volume Visualization (YUAN et al., 2005, 2006). High precision structures (such as textures and buffers) are used during rendering to produce a final image with a high range of values. Tone mapping is applied to adjust the image range so standard display devices can display details. High resolution transfer functions must be used to avoid missing details and producing mapping artifacts (such as the degradation that usually results from color quantization).

Texture-based DVR probably is the most used visualization method. It is simple yet robust to handle structured data. This technique and the others we have previously described in sections 2.3 and 2.4 can also handle time-varying data, as next section shows.

2.6 Time-Varying Techniques

All three previously described volume rendering methods can be extended to handle time-varying data. The main challenge faced when using multiple time instances is to handle the massive amounts of data. Brute force methods can be easily implemented, but they usually hit bottlenecks faster than more advanced techniques.

A data structured developed to efficiently solve some problems of handling time-varying data is the Time-Space Partitioning (TSP) tree (SHEN; CHIANG; MA, 1999). This structure address the problem of visualizing multiple time instances of regular grids. It uses an octree that stores in each node the spatial and temporal information of the dataset. The temporal information is stored in a Binary Space Partitioning (BSP) tree, that is located in each node of the octree. Octree leaves are independently rendered using a ray casting (though it could use another rendering method) and merged together in a latter stage. Rendering speed can be improved if consecutive time steps have relatively homogeneous data and the user has not changed the viewing properties. Figure 2.12 presents the concept of the TSP-tree, using an analogous structure in 2D space.

Another data structure is based solely on the octree (MA; SHEN, 2000). Spatial quantization is performed to compress the dataset information. Consecutive time steps are stored as a difference from the previous one, further compressing the data. Incremental rendering is used in a similar fashion to the one of the TSP-Tree, reusing previous rendering in temporal homogeneous regions.

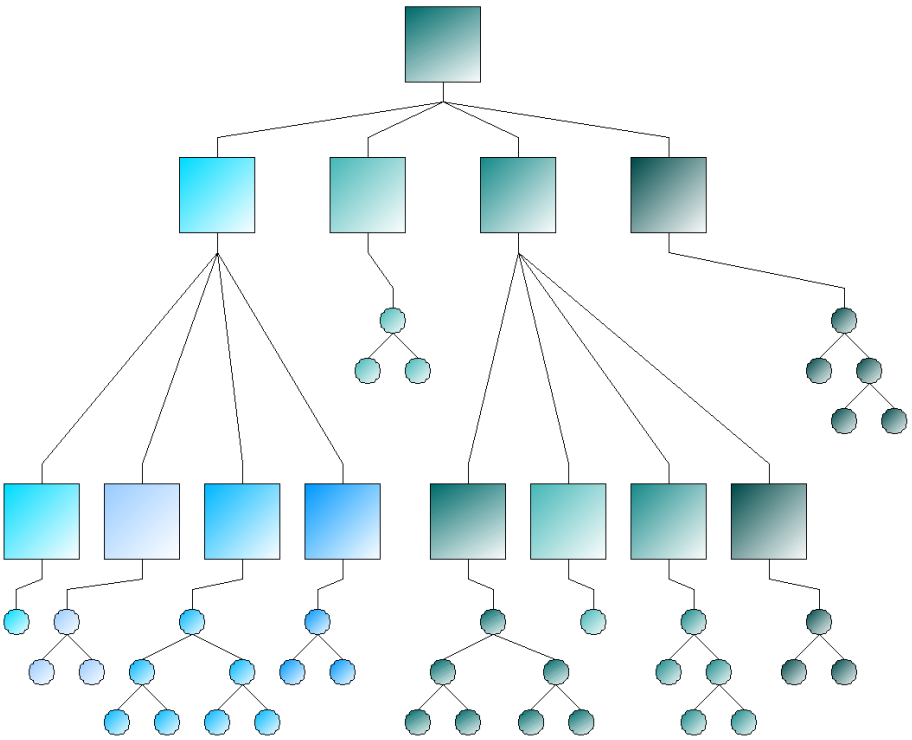


Figure 2.12: A TSP-tree is an octree (quadtree in this representation) created over the spatial domain of the dataset, where each leaf has an associated BSP-tree that stores temporal information.

Data quantization was also combined with hierarchical decomposition to render static data (SCHNEIDER; WESTERMANN, 2003). Dataset information is spatially compression using vector quantization, and the quantized lookup tables are reused in consecutive time-steps.

Unstructured datasets received little attention of researchers concerning time-varying data. A recent work presented an approach to compress unstructured scalar data that exploits temporal coherence (BERNARDON, 2005). Hierarchical decomposition is applied to consecutive time-steps, and the difference vectors are compressed using vector quantization. Two different rendering algorithms were modified to support this extension. A texture-encoded scheme was used to integrate this approach to a Hardware-based Ray Casting, decompressing the necessary information on-the-fly. The HAVS algorithm was also modified to incorporate these changes, using the CPU to decompress the data and send it to the GPU (BERNARDON et al., 2006).

The previously described system only handles time-varying scalar fields, while the grid is the same for all time steps. A solution for time-varying geometry is to use a key-frame based approach (DOLEISCH et al., 2005). Significant time steps are chosen and the position of the intermediate meshes is assumed to vary linearly. No solution to efficiently handle time-varying topology besides a brute force approach has been found during the research of this work.

Strategies to efficiently exploit I/O and parallelism (multithreading and distributed computing) using time-varying data have been recently developed for both structured and unstructured datasets (YU; MA; WELLING, 2004; BERNARDON et al., 2007).

The analysis of time-varying data is constantly increasing according to the available computational resources. It is a fairly important area for volume visualization since most

phenomena in the real life vary in time.

2.7 Discussion

This chapter presented a large area of computer graphics: volume visualization. The correct comprehension of its properties and its usefulness are fundamental to understand the remaining chapters of this work.

Several techniques have been developed around different ideas to explore (or not, according to the goals of the research) the graphics pipeline to solve the volume rendering integral. Section 2.2.1 presented a quick review of some volume rendering methods, and larger explanations and surveys can be found on the literature (SILVA et al., 2005).

Cell projection techniques have good rendering performance, since most of them exploit the benefits of traditional GPUs to solve the volume rendering integral. Ray casting is largely recognized by the quality of its result, with high quality images. It is powerful enough to handle both classes of datasets, but it is generally not as fast as other rendering methods. The texture-based DVR is probably the most used rendering method. Medical data generally comes in a format similar to 3D textures which are supported by modern GPUs and results in straightforward implementations.

Similar to all these techniques is the use of Transfer Functions to assign optical properties to different characteristics of the volume. Transfer function design will be described in the following chapter, and it holds the first set of contributions presented in this work.

3 TRANSFER FUNCTIONS

3.1 Introduction and Related Work

The previous chapter described which methods exist to perform volumetric visualization and suggested the need of Transfer Functions to produce the final images. Datasets are sets of scalar values measured over one or more conditions of a phenomena people want to study. Transfer Functions are mappings that translate such scalar information (density, temperature, etc) into optical properties. They are very important for volume visualization, since they are very flexible and allow the exploration of features from the underlying data (KINDLMANN, 2002).

The various methods for designing transfer functions can be classified under one of four paradigms (PFISTER et al., 2001):

1. Trial and error: users have no cue or information extracted from the dataset while specifying the transfer function;

2. Data centric without data model: create the transfer function automatically, without assuming an underlying model for the features;

3. Data centric with data model: (semi)-automatically creates transfer functions, assuming a specific data model, such as border detection.

4. Image centric: users interact directly with the final product of volume rendering, created with automatically generated transfer functions.

The first and last paradigms are less common since they require too much or too little knowledge from the users, respectively. Data centric models are more studied, since there is a good compromise between user effort and automation.

Trial and error techniques leave users alone to explore the entire transfer function space. The image centric paradigm presents images to users, and use them to refine transfer function creation. Design galleries (MARKS et al., 1997) is an image centric technique that uses genetic algorithms to improve the transfer function. Users select images produced using some initially guessed mappings, and the respective transfer functions are used as input to a refinement process. Since users have no explicit control over transfer function creation, some features may never be enhanced if the combining routine was poorly conceived.

For data centric methods that do not assume an underlying model the histograms try to present simple information to users, without trying to capture specific features. Scalar histogram shows how the scalar data is distributed over the entire scalar range. Often, this histogram presents an accumulation over one (or a few) scalar regions, while the largest

part of the histogram is presented as a thin line. One can use a logarithmic scale instead of a linear one to reduce this problem (POTTS; MÖLLER, 2004).

The paradigm that assumes an underlying data model is widely used. The definition of a boundary model is important for this work, and will be detailed in Chapter 4. This boundary model was previously used to define a semi-automatic procedure to create transfer functions (KINDLMANN; DURKIN, 1998), where the gradient-magnitude histogram is used to guide users about the location of boundaries. A similar approach was also used to define a three material transition model for virtual colonoscopy (SERLIE et al., 2003). It was used to classify and increase the quality of the visualization.

Several techniques that use histograms as a dual domain for the scalar field have been developed to easy transfer function specification (KNISS; KINDLMANN; HANSEN, 2001, 2002; MULTIDIMENSIONAL TRANSFER FUNCTIONS FOR VOLUME RENDERING, 2005). Users can draw and edit widgets over the histogram to specify multidimensional mappings, similar to those as shown in Figure 3.1. Transfer functions created with widgets can be applied to traditional rendering using the emission-absorption model or be combined with non-photorealistic rendering (SVAKHINE; EBERT; STREDNEY, 2005).

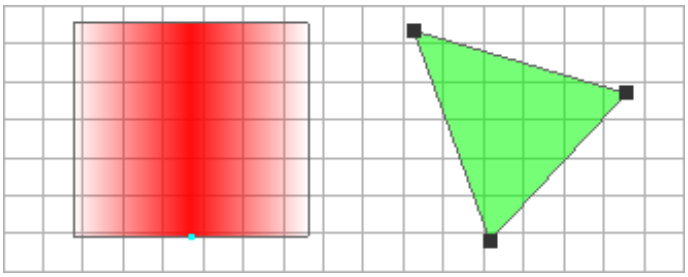


Figure 3.1: Example of two different widgets available in the system developed that are used to interact with histograms.

Histograms may present the same information with different properties. While the gradient-magnitude histogram presents transitions between materials as arc-like shapes, the LH (Low-High) histogram shows them as a region with high concentration of points (SERLIE et al., 2003; SEREDA et al., 2006). In the LH histogram, a value lies inside a material or in the boundary of two materials, and it does not differentiate these two situations. Also, its use is not recommended to enhance boundaries formed by three or more materials.

A drawback of working with the gradient-magnitude histogram is that it does not preserve the information about the spatial location of features, since it is basically a 2D structure. But one can use the color of the histogram to represent the spatial information (ROETTGER; BAUER; STAMMINGER, 2005), resulting in an easier interpretation of how the histogram features correspond to the 3D scalar field.

Besides being common, TF creation using widgets is not the only option. A high-level interaction approach (TZENG; LUM; MA, 2005) allows users to select a region of the dataset rendered using maximum intension projection. This region is then expanded using region growing, in a process similar to image segmentation.

Algorithms for transfer function specification for HDR datasets have been developed along with HDR volume visualization (SCHULZE; CHOURASIA, 2006), as presented in section 2.5. It is fairly important so transfer functions do not impose a limit for features that have a high precision domain.

Transfer functions have also been used to compute lighting between two materials, leaving the opacity to illustrate the thickness of homogeneous regions (LUM; MA, 2004). Other application is its use to simulate illustrations for education using non-photorealistic rendering and annotations (BRUCKNER; GRöLLER, 2005). These methods will not be extensively described since they are considered to be outside the scope of this work.

Another important subject when dealing with transfer functions is time-varying data. Some works have addressed this issue, but they are far less than the ones describing solutions for static data.

Time-varying series can be classified into statistically static datasets or statistically dynamic ones (AKIBA; FOUT; MA, 2006). Statistically static datasets present persistent histograms along the time, with little variation, allowing the use of a single transfer function for all time series. On the other hand, statistically dynamic datasets may require a specific transfer function for each time step, presenting high frequency features along the temporal line.

One of the first works on this topic proposes to analyze a dataset to extract features that vary in time (SILVER; WANG, 1998). Features that meet a certain criteria are selected and a region growing algorithm is used to expand the selection to an entire dataset feature. Maximums of the current time are used to accelerate convergence in consecutive time steps, increasing computational efficiency when handling statistically static datasets. Due to the fact that this approach uses a possibly different transfer function for every time step, it can also be used for statistically dynamic datasets.

Different strategies to handle transfer function creation for statistically static data have been proposed on the literature (JANKUN-KELLY; MA, 2001). The following strategies have been developed to produce a single transfer function intended to be used for all time series:

- average: calculates the average of all opacity maps;
- union: unite all individual transfer functions, capturing features that exist only over small time intervals;
- single representative: a TF created for a single time step is used during the whole time series;
- coherence-based: favor opacity values that are homogeneous in time over features that suffer high fluctuation, smoothing out high frequency features. Uses the Coefficient of Variation to perform the temporal analysis.

The coefficient of variation was also used to control the change of multiple transfer functions in the same time series. It uses interval thresholds to verify if the current transfer function is well suited for that time step or if a new one must be used, handling statistically dynamic data.

A histogram that fairly illustrates what happens in a time-varying sequence is the time histogram. It is the concatenation of a series of 1D scalar histograms, and as a result shows how the scalar field behaves over time. This histogram can be used to classify a time-varying dataset in one of the previously described categories.

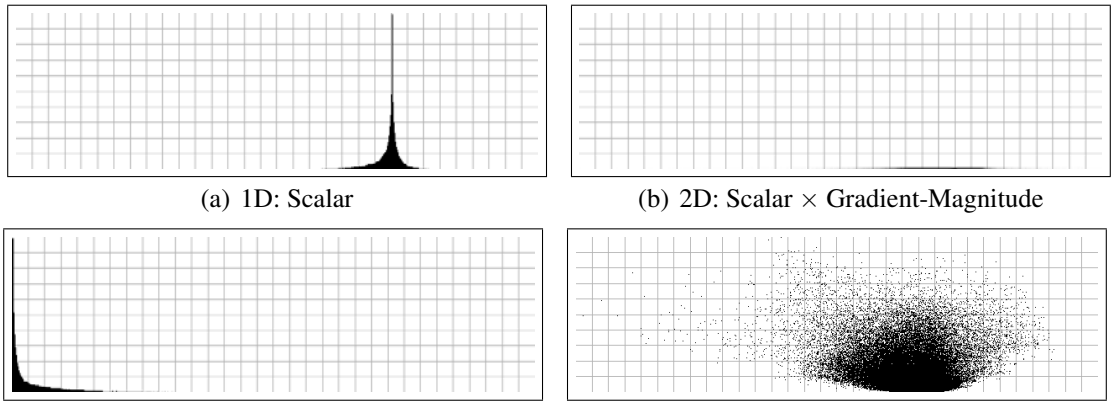
The transfer function design tool proposed in this work was designed around the concepts of histograms that try to capture underlying data information, as described in the next section, and widgets to provide interaction with the transfer function domain. Using this paradigm as a basic approach the concept of Ensembles that combine different

transfer functions is developed in section 3.3. This work also proposes a method for handling time-varying datasets using a key-framing approach, a method commonly used for standard computer animation, described in section 3.4.

3.2 Histograms for Transfer Function Interaction

As described in the introduction of this chapter, this work uses histograms to capture underlying data information and display it to users. This approach has been chosen so users are not left alone to explore the transfer function space while not imposing too many restrictions on how they can interact with it.

Four different histograms have been used to present information to users. These histograms are intended to capture different information and display them to users. Figure 3.2 illustrates these histograms computed over the TJet unstructured dataset.



(a) 1D: Scalar (b) 2D: Scalar × Gradient-Magnitude

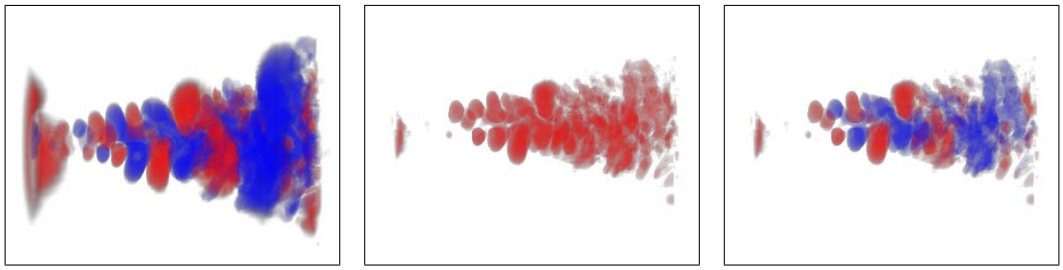
(c) 1D: Coefficient of Variation (d) 2D: Scalar × Coefficient of Variation

Figure 3.2: Four different histograms have been used to provide users with underlying dataset information: scalar histogram (a) shows scalar concentration; scalar × gradient-magnitude shows transitions between materials; coefficient of variation (c) shows scalars accumulation on different expectations; scalar × coefficient of variation (d) shows which scalars have higher variation. These histograms were computed over the TJet dataset.

Each histogram is used to quantify different properties of the dataset. The scalar histogram shows which scalars have greater concentration of values, and how they are distributed (figure 3.2(a)). It is useful to trivially isolate materials with high concentration of samples. An example of a volume rendering using a transfer function created over this type of histogram is shown in Figure 3.3(a).

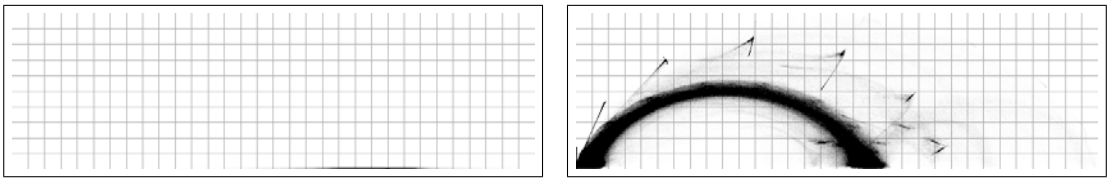
The gradient-magnitude histogram is widely used for enhancing transitions between regions of relatively homogeneous materials. These transitions are identified as arcs in this histogram. The maximum value of the arc represents the scalar value of the transition region, along with its neighborhood. The intuition behind this fact will be explained in Chapter 4. Figures 3.2(b) and 3.4(a) shows the gradient-magnitude histogram of the TJet dataset. Contrary to what would be expected this histogram does not present arc shapes like the one in figure 3.4(b). It rather displays a line in the bottom of the histogram. This does not happens only with the TJet dataset, but also with other unstructured datasets. These issues are further studied in Chapter 4.

The coefficient of variation (CoV) histogram is useful for checking the distribution of scalars through time, verifying if the amount of scalars with time-varying properties is



(a) Scalar (b) Coefficient of Variation (c) Scalar × Coefficient of Variation

Figure 3.3: Volume rendering of the TJet dataset using transfer functions created with different histograms.



(a) TJet (b) Engine

Figure 3.4: Scalar versus Gradient-Magnitude histograms for an unstructured and structured datasets.

significant compared to static ones. It is computed over scalars of the entire time series. It has already been used to create transfer functions for time-varying data (JANKUN-KELLY; MA, 2001; AKIBA; FOUT; MA, 2006), but they used an approach different from the one presented in this work. To correctly understand the meaning of the CoV, one must understand the concept of *Expectation* and *Variance of Expectation*.

Consider n time-steps for sample values s of an independent random variable X. The expectation (or mean) E at that position is expressed as

$$E[X] = \sum_{1}^{n} s(P\{X = s\}) \tag{3.1}$$

where $P\{X = s\}$ is the probability of picking a random value in X, and whose value in this case is $1/n$. The variance of the expectation Var can now be expressed as

$$Var[X] = E[X^2] - E^2[X] = \sum_{1}^{n} \left(\frac{s^2}{n}\right) - \left(\sum_{1}^{n} \frac{s}{n}\right)^2 \tag{3.2}$$

which is the spread of scalars around their mean. The CoV can now be computed as the ratio between the standard deviation to the expectation:

$$C_v[X] = \frac{\sqrt{Var[X]}}{E[X]} \tag{3.3}$$

The CoV is useful to measure the dispersion of probability distributions with widely differing means. It is a dimensionless measure, and can not be used to construct confidence intervals to the mean. These characteristics allow the CoV to be used to compare all values against each other.

The first histogram used with the CoV is a simple 1D histogram, depicted in Figure 3.2(c). This histogram can be used to enhance time-varying features independently of its relation to the scalar value, as illustrated in figure 3.3(b).

The last histogram correlates the scalar value with the CoV, as shown in Figure 3.2(d). It is useful to enhance time-varying features with different optical properties, according to the associated scalar value. Figure 3.3(c) shows the volume rendering of the TJet dataset using a transfer function created using this histogram. Besides the result being similar to the scalar histogram, it is easier to understand where the time-varying features are.

Histograms have been used here so users can initially define transfer functions using a widely known and used method. After creating initial and relatively simple mappings, more advanced tools can be used to produce more complex transfer functions. The transfer function ensembles technique explained in the next section is an example of such advanced tools

3.3 Transfer Function Ensembles

Users can draw widgets over different histograms to create transfer functions, as described in the previous section. But a single histogram is often used to enhance a particular feature of the dataset, like the boundary of relatively homogeneous regions. Users should be able to display in a single transfer function features that are easily enhanced by two or more types of histograms. This is the concept behind Transfer Function Ensembles (TFE): the composition of transfer functions to create a single, complex transfer function.

The combination of transfer functions has been mentioned in the literature, and some attempts have been made to use it. It has been used before to combine different nodes of a graph that stores a visualization history in a system (MA, 1999). Operations such as addition, difference and intersection are mentioned but not described. Other method suggests fusing features of different transfer function through interaction with final images (WU et al., 2006a). It allows the selection of regions in the final image that will be preserved while combining different transfer functions. Different from previous works, the idea presented here is that TFEs provide a versatile and robust method for dataset exploration. Using different blending operations, transfer functions are combined in a boolean-like manner, resulting in interesting visualization boolean operations.

To perform transfer function composition, transfer functions must be defined in common spaces (like the scalar space). Up to 3 different dimensions can be used, but this constraint is only applied due to limitations of the GPU (since the resulting transfer function is stored as 1D, 2D or 3D textures).

A lower-dimensional transfer function can be combined with higher-dimensional ones by using extrusion to fill the missing space. For instance, to combine a TF defined over a scalar histogram (1D) with a gradient-magnitude histogram (2D), one can replicate the scalar TF table for all values of the other dimension. Figure 3.5 illustrates this idea.

When the objective is to produce a transfer function with a lower dimensionality one can apply a dimensional reduction (FODOR, 2002). The application of dimensional reduction usually loses information. This work uses a simple dimensional reduction, where the intensity of the color and opacity for a given region of the transfer function is modulated accordingly to the amount of samples that project into the same region. For instance, if two samples will project into the same bin in the final transfer function and only one of them is affected by the 2D transfer function, the final color and opacity will be modulated by half, as shown in Figure 3.6.

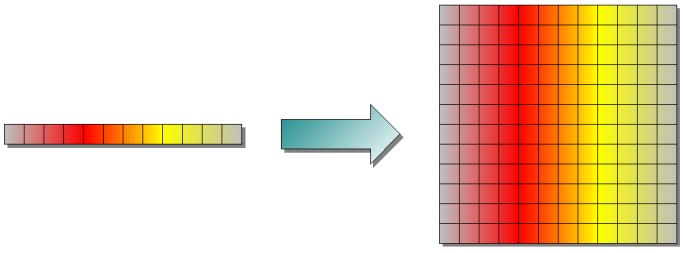


Figure 3.5: Extrusion of a 1D transfer function defined over a scalar range (left) to a 2D transfer function (right). This process allows the blending of 1D transfer functions with 2D (or 3D) ones.

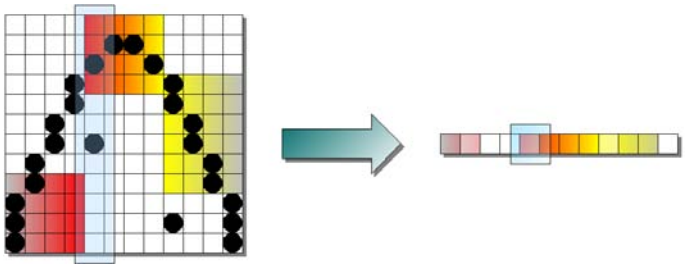


Figure 3.6: Analogously to extrusion, a transfer function can suffer dimension reduction to enable blending with lower dimensional transfer functions. We modulate opacity and color intensities according to the ratio between the amount of samples covered by the transfer function and the total amount of samples at that region.

Once all transfer functions share the same space we can combine them to produce a single transfer function that (de)emphasizes features of all original mappings. This composition is performed using a blending strategy, each one with different properties and useful in different scenarios.

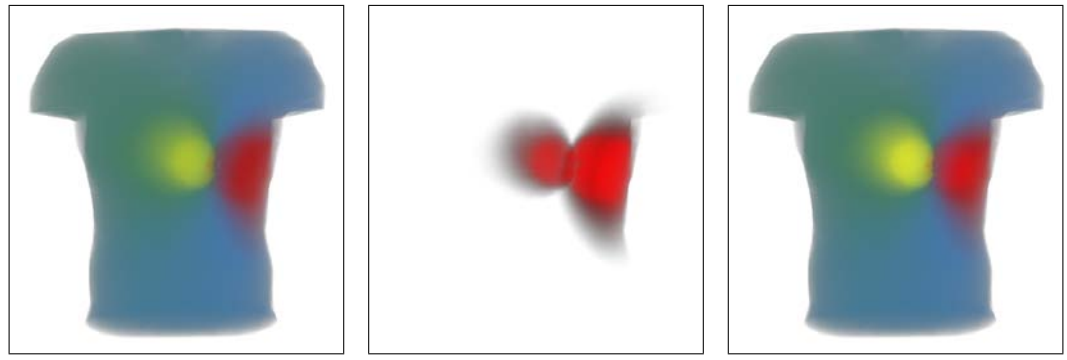
3.3.1 Blending Strategies

The tool this work presents for transfer function specification allows three different blending operations, though more can be added with easy. Common to all blending strategies is a weighting factor individual to each transfer function. This weight represents the importance of that transfer function to the final composition. It is a user-controlled value that multiplies the color and opacity prior to the composition. Figure 3.7(f) illustrates the importance of the weighting factor to emphasizes a feature and preserve context information.

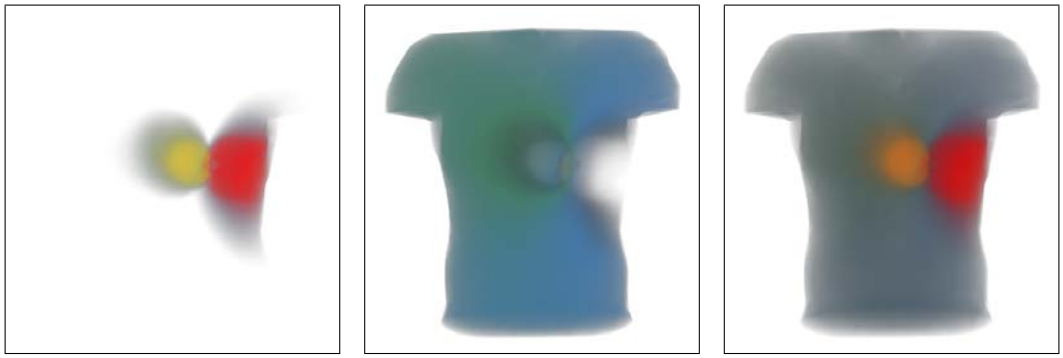
The following blending description applies to two transfer functions at a time (figures 3.7(a) and 3.7(b)). If more transfer functions are to be added, the blending operations are applied sequentially.

1. **ADD**: this blending operation results in a transfer function that combines features of both input transfer functions. The result *r* for color *C* and opacity α of a lookup table entry *i* for transfer functions 1 and 2 is the following:

$$\begin{aligned} C_r(i) &= C_1(i) + C_2(i) \\ \alpha_r(i) &= \alpha_1(i) + \alpha_2(i) \end{aligned}$$



(a) TF1 (b) TF2 (c) TF1 *add* TF2



(d) TF1 *and* TF2 (e) TF1 *xor* TF2 (f) 0.5 X TF1 *add* 1.0 X TF2

Figure 3.7: The combination of transfer function (a) and (b) using diverse blending strategies: simple addition in (c), AND operation in (d), XOR operation in (e). The effect of weighting is shown in (f).

This mode is useful to enhance interesting regions while preserving the context. Figure 3.7(c) shows the result of simply adding two transfer functions. Figure 3.7(f) shows the result of applying different blending weights before combining both transfer functions.

2. **AND**: this operation produces a transfer function that emphasize features enhanced by both input mappings. The result can be expressed using the same notation as above:

$$\begin{aligned} C_r(i) &= Max(C_1(i), C_2(i)) \\ \alpha_r(i) &= Min(\alpha_1(i), \alpha_2(i)) \end{aligned}$$

The maximum of each color channel is used to keep the color intensity, and the minimum alpha removes areas where both transfer functions are less common, as illustrated by figure 3.7(d). Again one ca use standard alpha-blending methods to combine the color channels.

3. **XOR**: this is a complementary operation related to the previous one:

$$\begin{aligned} C_r(i) &= (C_1(i) \wedge \overline{C_2(i)}) \vee (\overline{C_1(i)} \wedge C_2(i)) \\ \alpha_r(i) &= (\alpha_1(i) \wedge \overline{\alpha_2(i)}) \vee (\overline{\alpha_1(i)} \wedge \alpha_2(i)) \end{aligned}$$

If a features is enhanced by both transfer functions, it will be left out or the resulting transfer function (as shown in figure 3.7(e)). This is very useful for removing features from a more complex transfer function, as demonstrated in Figure 3.7(e).

Combining a set of predefined transfer functions can be a powerful tool. In medical visualization, for instance, if a doctor has several predefined transfer functions that enhance different organs (heart, bones, muscles, etc), he can easily combine them to produce a single transfer function that emphasizes all features he wants to using the ADD operation. Similarly, he can use the subtraction capabilities of AND and XOR to produce unique transfer functions from existing ones.

Ensembles have been developed to easy data exploration and transfer function creation primarily for static scalar fields. When one deal with time-varying data there are other situations that must be considered when creating a meaningful visualization, as described int the next section.

3.4 Time Varying Transfer Functions

The beginning of this chapter described techniques for transfer function creation, and Section 3.3 described how one can combine them to produce new functions and explore the scalar field. These methods have primarily addressed static scalar fields. Considering that most interesting phenomena in real life have variation in time, time-varying transfer function specification should receive more attention from researchers than it has received so far.

The use of single transfer functions for the whole time series or individual transfer functions per time step presents some drawbacks. In the first case, a single transfer function may not capture important information from specific time steps if it presents transient information not captured initially by the transfer function. The use of different transfer functions per time-step, on the other hand, may result in visual discontinuities and can potentially confuse users.

This work advocates the use of key-framing (PARENT, 2001) to easy the specification of transfer functions for dynamic datasets. The approach developed allows users to create a series of transfer functions and assign them to specific time steps, creating intervals. The remaining time steps will use a custom transfer function corresponding to the interpolation of the transfer functions that limits its interval. This interpolation is controlled by a user-defined transition curve presented in the left of Figure 3.8.

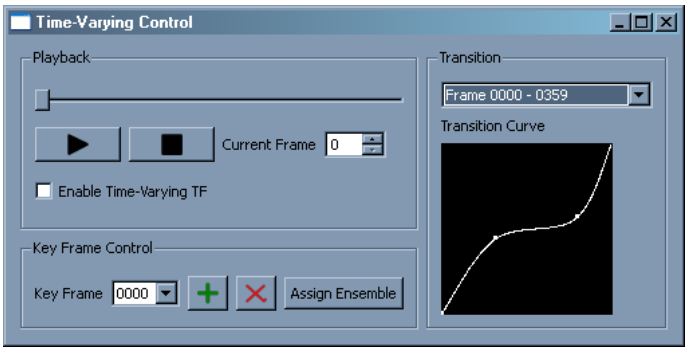


Figure 3.8: User interface allows user interaction and control over dataset playback and transfer function manipulation. Transition curve (on the right) enables customized transition between key-frames.

This approach is intended to minimize the problems previously described. It allows single transfer functions to be used for determined intervals whose features do not vary dramatically. When a new transfer function must be used, users can determine how much

time it will take for moving from the current function to the next one. Visual discontinuities caused by transfer function changes are minimized by the customizable interpolation scheme.

A key-frame based approach was also used to animate features on static datasets (WU et al., 2006b). It is used to increase focus in certain areas of the dataset. This technique has been developed in parallel with the one presented in this work, and illustrates the use of key-framing for volume visualization.

3.5 Discussion

Transfer functions are very useful for volume visualization since they are mainly responsible for assigning optical properties to data values. Users must be able to interact with the transfer function space to explore the dataset and identify interesting features.

This work chose to use histograms to guide users by presenting underlying data, as described in section 3.2. Different from other approaches that do not try to help users or impose restrictions to their interaction, the histogram approach presents underlying data and offers direct interaction with the transfer function space.

Although widely used, histogram-based methods can still be difficult to create more complex transfer functions. To easy this burden, this work contributes by introducing the concept of ensembles in section 3.3, and defining operations required to produce them. Ensembles combine different transfer functions into a single and more complex transfer function using a blending operation. Section 3.3.1 described three different methods for combining the transfer functions to create new mappings. The ADD operation is useful to combine features enhanced by both transfer functions. AND is useful to focus on specific features enhanced simultaneously by both transfer functions. Opposed to AND, the XOR operation is used to remove features of a transfer function, removing details enhanced by both.

When features that users are analyzing vary in time they may require more than one transfer function to effectively capture features. The second contribution of this work explains how to use key-framing to manage multiple transfer functions in a time-varying sequence. It has the advantage of minimizing visual discontinuities that occur when the transfer function changes by mixing both transfer functions of an interval. This combination is controlled by users, using a customizable transition curve.

These contributions have been integrated into a tool for designing transfer functions and interacting with volume visualization. The system is described in Appendix A, including some contributions not related to this document. Other systems have been proposed on the literature, like the VolumePro (KÖNIG; GRÖLLER, 2001), but there is no consensus on the adoption of a common interface. The development of the system described in Appendix A may be considered a contribution itself, although more validation is required to fully verify its efficacy and efficiency with end users. Expert reviews (TORY; MÖLLER, 2005; TORY; POTTS; MÖLLER, 2005) have been performed with some users, and they found it suitable for its purpose.

While developing the transfer function tool, the histogram did not behave as expected. The gradient-magnitude histogram for unstructured datasets did not present the arc structure that characterizes transitions between homogeneous regions. Figure 3.4 illustrates the gradient-magnitude histogram computed over an unstructured dataset (figure 3.4(a)) and over a structured one (figure 3.4(b)). Notice the clear arc shape on the structured dataset contrasting with the line at the bottom of the unstructured one. This behavior was

a big surprise, since it is based on a solid theory. The following chapter presents a study on why this happens and how one can deal with it to present meaningful information.

4 DATA SIGNATURE ANALYSIS

4.1 Introduction and Related Work

In the previous chapters we presented approaches to help users explore scalar fields. One important limitation concerning the computation of the traditional gradient-magnitude histogram was presented in section 3.2: unstructured datasets did not present the clear and characteristic arc-like shape similar to structured grids. This chapter is going to address the main differences between these two classes of datasets in the context of extracting underlying data information.

This work defines data signatures as the meaningful information that can be computed from a dataset. This information presents important characteristics directly related to its source. It is useful to better understand and analyze the problem.

Most of the techniques for computing data signature have been developed considering only datasets with regular sampling patterns. These datasets usually present desired properties, like a large amount and regular placement of samples. On the other hand, unstructured datasets are often composed by a reduced amount of samples irregularly distributed in space when compared with its respective counterpart. The extension of previously developed methods is not as intuitive as someone might expect. Characteristics of these datasets must be considered when one develops methods for computing and extracting statistical measures in unstructured grids.

Data signature has been extensively used for data analysis. The Contour Spectrum (BAJAJ; PASCUCCI; SCHIKORE, 1997) is a data signature formed by scalar data and contour attributes extracted from the dataset, such as surface area, volume and gradient integral. It is used in multidimensional unstructured grids, providing underlying data information without an assumed model to help users while designing a transfer function.

High Order Moments (HOM) are also model-independent statistical signatures (TENGINAKAI; LEE; MACHIRAJU, 2001). They have been used as estimators of central tendency of data distributions. By measuring the chance of a distribution to cluster around a particular value, it identifies possible salient iso-values, used to extract an iso-surface from a volumetric dataset. The Mean Shift procedure (SHAMIR, 2003) was similarly used for clustering. It combines functional and spatial domains for feature extraction, data exploration and partitioning, identifying the spatial location of a feature selected in the functional domain.

Data signature was used in previous works to enhance user experience while manipulating volumetric datasets. It is concerning that few works have mentioned difficulties in handling unstructured data for feature extraction and analysis. Most works regarding the use of unstructured grids come from finite elements simulations. They state that the shape of elements is crucial for good results in reconstruction techniques (SHEWCHUK, 2002,

2003), and discretization errors are closely linked to interpolation errors. The influence of one bad element is usually felt at locations near the bad element.

The remaining of this chapter will analyze possible sources of errors that exist and degrade the quality of data signature. The gradient is probably the most used signature method, and a brief introduction to it is presented in section 4.2.

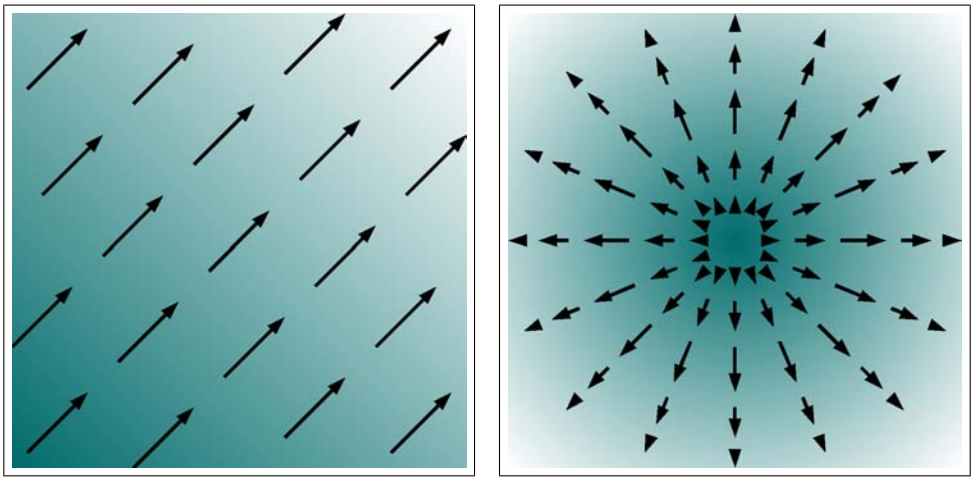
Section 4.3 presents the synthetic case study used to isolate sampling issues analyzed in section 4.4 from approximation issues, discussed in section 4.5. Section 4.6 explores the effect of smoothing samples prior to histogram computation. Section 4.7 discusses results from this work, and presents possible new models in section 5.4.1.

4.2 Gradient

This work has previously described the use of the gradient-magnitude histogram for transfer function specification (section 3.2) and the gradient itself, but has not yet presented a formal definition. The gradient of a scalar field is the directional derivative of a scalar field. For a scalar field $f(x, y, z)$ in the Euclidean space it is defined as

$$\nabla f(x, y, z) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \tag{4.1}$$

When applied to a point p in the scalar field, the gradient results in the directional derivative of the scalar field at p. It is important to notice that the magnitude of the gradient vector indicates the rate of change of the scalar towards the gradient vector direction. Figure 4.1 shows two examples of gradient vectors in 2D.



(a) (b)

Figure 4.1: Gradient vectors computed over two different 2D scalar fields: (a) linear scalar variation results in gradient vectors with constant direction and magnitude; and (b) spherical non-linear scalar variation results in gradient vectors with different directions and magnitudes.

When the gradient vector is computed over a point on a line or surface associated with an iso-value, it corresponds to the normal of that point regarding the line or surface. Figure 4.2 illustrates the correspondence of the gradient vector to the normal of an iso-line.

Another useful measure is the second derivative of the scalar field, the Laplacian:

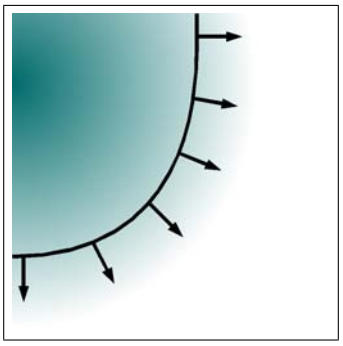


Figure 4.2: For lines (surfaces) defined over an iso-value, the gradient results in the normal of the line (surface).

$$\nabla^2 f(x,y,z) = \left(\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2} \right) \tag{4.2}$$

The Laplacian represents the divergence of the gradient. It is used along with the magnitude of the gradient for data analysis, and is important for the definition of a boundary, as described in the next section

4.3 Synthetic Case Study

A synthetic dataset was created for analysis purposes. The reason for using it is that it provides full control over the environment of the experiments, representing an ideal situation. This control allows the separation of sampling and approximation issues.

The dataset is composed of a sphere inside a box, whose centers coincide, as illustrated in figure 4.3. The box has dimensions ranging from 0 to 1 in all three main axis of the Cartesian coordinate system. The radius of the sphere is $\frac{1}{3}$, so the boundary of the sphere does not intersect the bounding box. The function value inside the sphere is 0, outside the sphere is 1 and the boundary has a smooth transition between these values. The dataset domain is limited by the surrounding box.

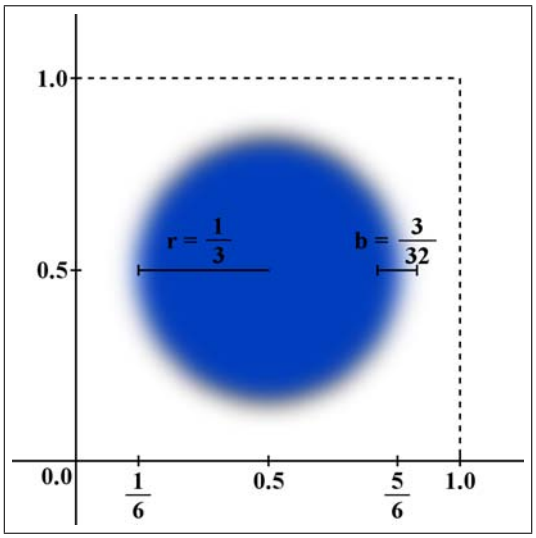


Figure 4.3: 2D representation of the synthetic dataset used as case study in this work.

The reason for using a sphere as the study case is that it presents gradients pointing to all directions. Different samples over the boundary region will present different directional derivatives. The description of an ideal boundary is fundamental to understand the shapes expected in the histograms, and is described next.

4.3.1 Ideal Boundary

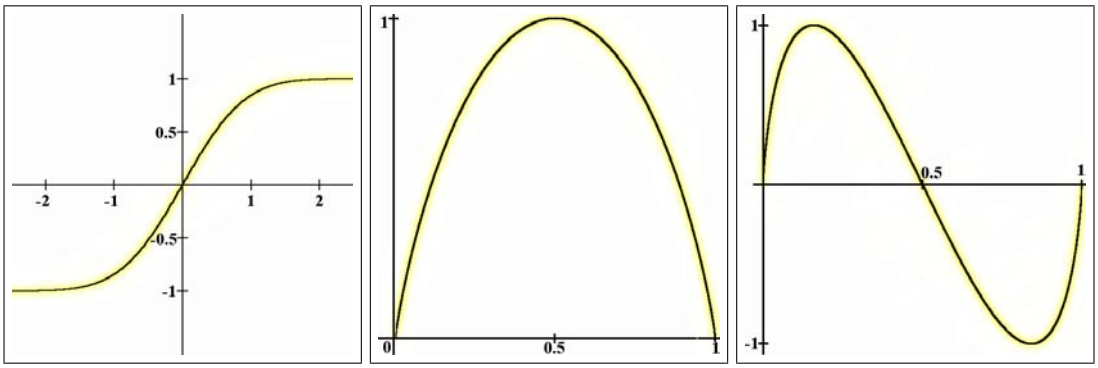
The analysis of this work, just like the one presented by Kindlmann and Durkin (KINDLMANN; DURKIN, 1998), is based in the characterization of an ideal boundary. A smooth transition must be present to provide different gradient magnitudes and allow the resulting histogram to present a particular shape.

The use of a step function to represent the boundary transition presents two major drawbacks. First and most important, real datasets suffer from low boundary quality inherent to the acquisition process. Second, if boundaries were well defined, one would be able to precisely identify them without the need for statistical measures.

To better simulate real datasets the Error Function (erf) is used to represent the boundary behavior. Equation 4.3 and figure 4.4(a) show it. The yellow region of the histograms presented in this chapter is used only to provide context information.

$$erf(x) = \frac{2}{\sqrt{\pi}} \times \int_0^x \exp^{-t^2} dt \tag{4.3}$$

The erf is the result of convolving a Step function with a Gaussian function. Its use is straightforward since it represents what happens with real datasets. The erf is always sampled inside the interval $[-6..6]$ because these limits result in values close enough to -1 and 1.



(a) $f(x)$ (b) $f(x) \times f\prime(x)$ (c) $f(x) \times f\prime\prime(x)$

Figure 4.4: Error function used as boundary behavior ($f(x)$) (a) and it plotted against the magnitude of its first directional derivative ($f'(x)$) (b) and second derivative ($f''(x)$) (c) (plots b and c are normalized).

Important for this analysis is that the analytic solution for the first and second derivatives of the erf are also known. The first derivative is the normal distribution, a Gaussian function of the form:

$$erf'(x) = \frac{2}{\sqrt{\pi}} \times \exp^{-x^2} \tag{4.4}$$

and its plot is shown at figure 4.4(b). The second derivative is the combination of a Gaussian function with an Hermite polynomial:

$$erf''(x) = \frac{4}{\sqrt{\pi}} \times (-x) \times \exp^{-x^2} \tag{4.5}$$

and its plot is shown in figure 4.4(c).

In the discussion that follows this work evaluates the impact of sampling and approximation issues when detecting boundary transitions.

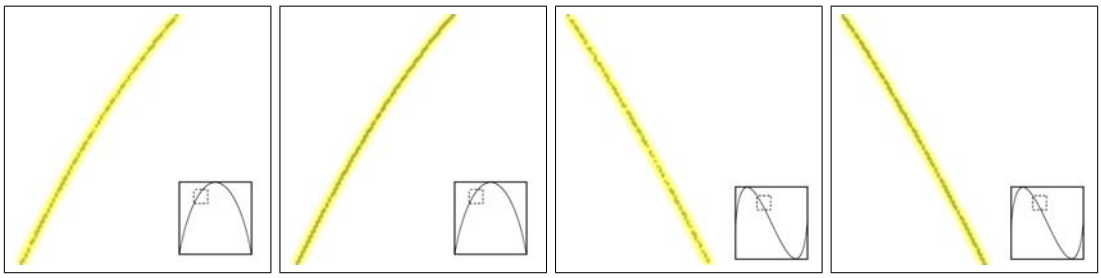
4.4 Sampling Issues

It is important to disconnect sampling from other possible distortion sources to precisely identify how it affects the histogram. Since the expected boundary has been defined as a known erf function (section 4.3.1), it is possible to find all required sample values analytically, thus avoiding negative effects from other error sources.

Consider $f(x)$ the function that will be analyzed, x as a coordinate in an arbitrary space (this work will focus on the $\Re^3$ space). $f'(x)$ represents the magnitude of the directional derivative of $f(x)$ (the gradient of $f(x)$). $f''(x)$ is the second directional derivative of $f(x)$.

Every sample in the domain Ω of the function $F(x) = (f(x), f'(x), f''(x))$ ($[0..1]^3 \in \Re^3$ for this work) has an assigned value for $f(x) \in [a_1, a_2]$, $f'(x) \in [b_1, b_2]$ and $f''(x) \in [c_1, c_2]$. As the function is well defined for all its domain, it can be seen as the process of a continuous random variable moving through Ω. Thus, the histogram represents a discretized form of a probability density function (pdf) of the random process F, whose quality depends of M_a, M_b and M_c bins for $[a_1, a_2]$, $[b_1, b_2]$ and $[c_1, c_2]$ respectively, and the number of samples used to measure the function.

A sampling method must be used to approximate the pdf of F because there is no prior known information about it. One can use the Monte Carlo sampling approach (L'ECUYER, 2003), since it is well known and useful to solve these problems. Figure 4.5 shows the resulting histograms of first (a, b) and second (c, d) derivatives with 100.000 and 400.000 samples, respectively.



(a) (b) (c) (d)

Figure 4.5: Analytic evaluation of f versus f' (a, b) and f versus f'' (c, d) on the sphere dataset with 10^5 and 4×10^5 samples. Notice the absence of gaps when enough samples are used.

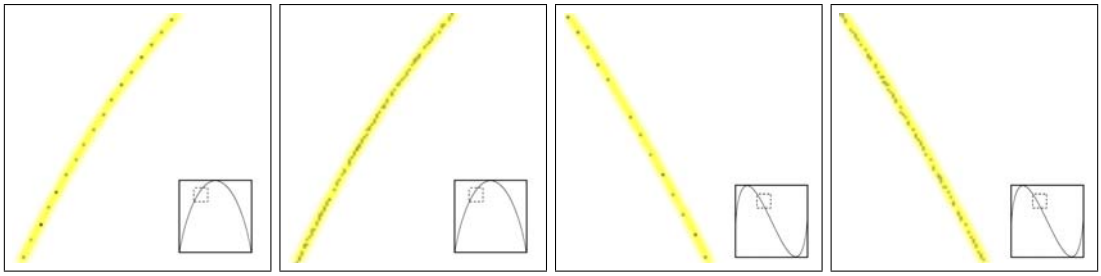
The analysis of emphasized regions presented in figures 4.5 demonstrates that 100.000 samples were not enough to produce a histogram without gaps, while 400.000 samples resulted in a continuous shape for histograms with resolution of 512^2.

Sampling behaves differently according to the disposition of samples in a dataset. Sections 4.4.1 and 4.4.2 analyze how sampling behaves in structured and unstructured

datasets, respectively. Section 4.4.3 analyzes how different boundary widths affect the quality of produced histograms.

4.4.1 Structured Datasets

Quasi-Monte Carlo methods are commonly used to accelerate convergence and increase computational efficiency of numerical methods. These methods take advantage of specially chosen collection of points, using some specified metric. The uniform distribution of points in a regular grid is a form of Quasi-Monte Carlo. Previous works have benefited from the regular lattice of structured datasets, generating useful histograms with resolutions between 80^2 and 256^2.



(a) (b) (c) (d)

Figure 4.6: Gradient-magnitude of the blurred sphere dataset. Histograms were created using approximately 10^5 samples inside regular grids with resolutions of 128^3 (a, c) and 512^3 (b, d).

The experiment with structured grids computed histograms with resolution of 512^2. Figures 4.6a and 4.6c show the resulting histograms of sampling the function using a regular distribution of points inside a volume of 128^3. This sampling shows the problems with insufficient representation of a domain, since large gaps appear in both histograms. Figures 4.6b and 4.6d present histograms computed with samples regularly distributed inside a volume with resolution of 512^3. Notice the better quality achieved by choosing a good sampling strategy for a domain. Both histograms where generated using 100.000 samples.

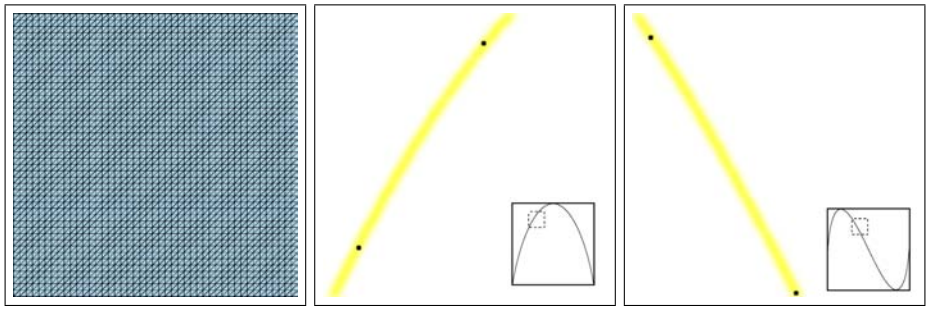
While structured grids usually present a large amount of samples with a regular distribution, unstructured datasets usually offer less samples irregularly placed. Next section analyses this class of datasets.

4.4.2 Unstructured Datasets

Scientists working with unstructured datasets are binded to a fixed number of samples corresponding to the quantity of vertices in the mesh. Rarely these vertices present regular or semi-regular patterns, so there is no control on the sampling points of the function to build a pdf. To analyze how sampling affects these datasets three distinct distribution of points have been used, all of them with 100.000 samples.

The first distribution used is a regular pattern. Despite the fact it is an unusual distribution, it is interesting to notice how it behaves when comparing to structured grids. A mesh with 46^3 vertices regularly distributed was built inside the procedural volumetric dataset. Figure 4.7a shows a 2D slice of the 3D mesh. By observing figures 4.7b and 4.7c one will notice that the same problems presented by structured datasets are present: the accumulation of points in a few bins and long gaps between samples. Just like in the

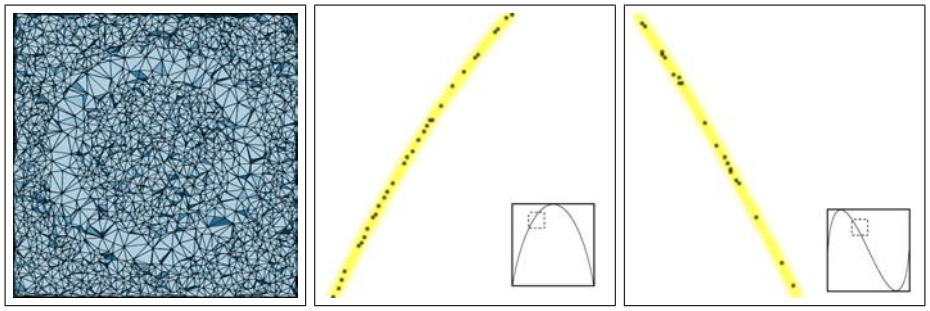
structured case, this is directly related to the resolution of the mesh used. Unfortunately, increasing the number of samples (vertices) will inflate considerably the total size of the unstructured datasets.



(a) (b) (c)

Figure 4.7: Regular mesh created over the sphere dataset (left) with 97k samples, its gradient-magnitude histogram (center) and second derivative histogram (right). This regular pattern presents the same problems as structured grids.

The second distribution used is a mesh with no regular pattern. In this case the samples are concentrated in homogeneous regions, leaving transition regions (edges) with coarse points. See figure 4.8a for an image of a slice of this dataset. Histograms presented in figures 4.8b and 4.8c show that the gap length between samples on the histogram have been reduced, mostly due to the irregular distribution of samples, but it still does not correspond to the well defined shape this work aims for.

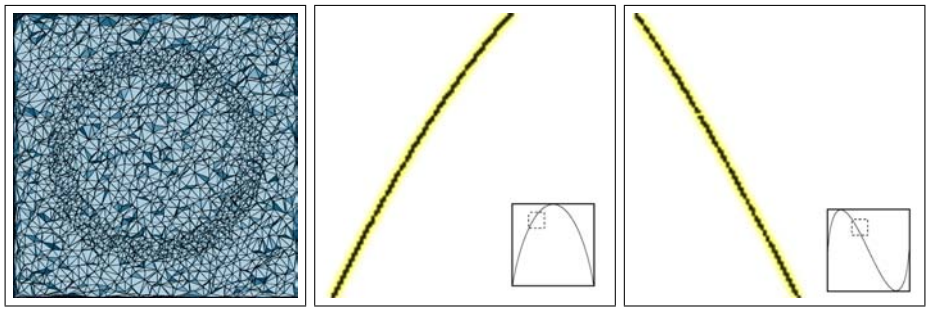


(a) (b) (c)

Figure 4.8: Sphere mesh refined at homogeneous regions (left) with 100k samples, its gradient-magnitude histogram (center) and its second derivative histogram (right). Irregularly samples reduce gap lengths on histograms.

The last distribution used produces a mesh refined at boundary transitions. Homogeneous regions now present coarse resolution. Figure 4.9a shows a slice of this dataset. Histograms generated by this distribution present a higher quality level than the other two previous strategies, as figures 4.9b and 4.9c show. This is due to the high concentration of samples in regions that present the interesting characteristic this work tries to identify.

The three previously described scenarios show the “average”, “worst” and “best” case respectively. One could argue that the second dataset produced better samples, since they were irregularly placed. Although it is true for this particular test case, in a more general scenario there are greater chances of a regular distribution present samples across different boundary zones.



(a) (b) (c)

Figure 4.9: Sphere mesh refined at boundary regions (left) with 100k samples, its gradient-magnitude histogram (center) and its second derivative histogram (right). Notices how this distribution provided good histograms.

Fortunately, the "best" scenario also is the most common one. Simulation researches usually produce meshes whose geometry captures sharp gradients. This way, the most common source of unstructured meshes naturally attempt to create and refine sampling densities at sharp features. Features that this work attempts to identify and enhance.

Although unstructured meshes usually present less samples than regular grids, these samples will often present better distribution. The quality of the samples will be similar, if not better, than the one of structured grids.

4.4.3 Influence of Boundary Width

The characterization of an ideal boundary assumed in this work was presented in section 4.3.1. Figure 4.3 presents a 2D version of the dataset we have been using this far. This section shows results on varying the boundary width b and verifies how it affects histograms.

It is important to notice that the sampling range of the erf ($[-6..6]$) has not been changed. The only change is how the area of influence of this result is mapped over the boundary of the dataset. Mathematically what happens is a modification of the variance in the Gaussian function that we use to convolve the step function.

Examining figure 4.10 one can notice that an increase in boundary width is associated with an increase in histogram quality. It demonstrates that the boundary width is an important factor when computing histograms. Larger boundaries will be easily detected, while thinner boundaries will be challenging to deal with.

Real datasets may not present a large boundary, but a thin one. For these situations, one can apply a blurring filter to increase the boundary width. This has been used in previous works (KINDLMANN; DURKIN, 1998) to increase the quality of histograms and easy the identification of model features.

4.5 Approximation Methods

After discussing which sample distributions produce acceptable histograms, this work will focus on how to compute (or approximate) data to create the histograms.

According to the data format (a lattice or irregular structure) different methods can be used to compute approximations of f' and f''.

Next section explains a method commonly used for computing the gradient vector in structured grids. Section 4.5.2 presents the Least Squares technique, robust enough

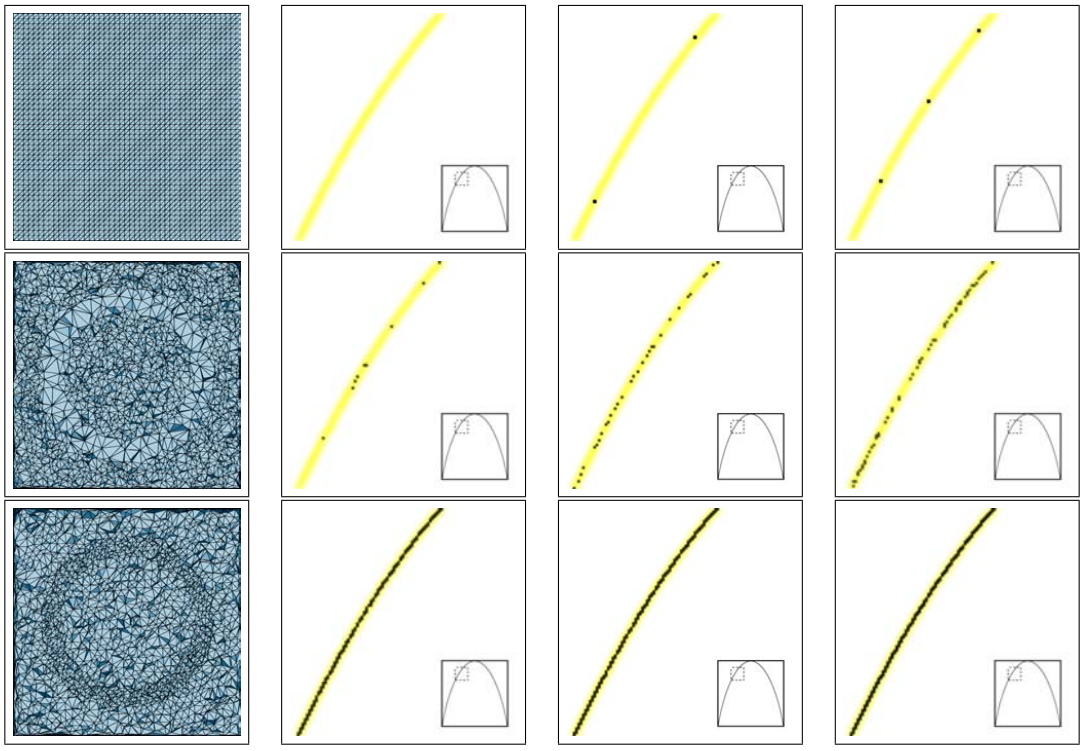


Figure 4.10: Effects of different boundary widthes for regular (1st row), homogeneous (2nd row) and edge (3rd row) distributions: 0.05 (2nd column), 0.09375 (3rd column) and 0.2 (4th column) boundary width. Larger boundaries produce better histograms.

to deal with structured and unstructured datasets. These two methods are analyzed in section 4.5.3.

4.5.1 Central Differences

The lattice nature of the first class of datasets makes it suitable for applying classical image processing algorithms. A relatively fast method for computing different measures on these structures is the use of convolution kernels with some well known operator.

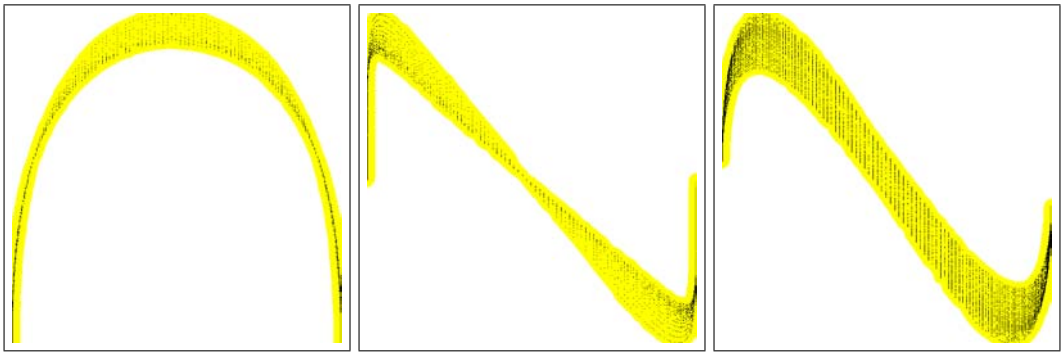
Central differences are easily applied using a 3D version of the Prewitt mask, whose 2D horizontal version is shown in figure 4.11(b) (GONZALEZ; WOODS, 2007). This filter results in good approximations of f', as presented in Figure 4.12(a). To compute f'' one can apply the previous filter twice, using the resulting values of the first computation as input for the second pass (Figure 4.12(b)). Another approach is to use the Laplacian filter presented in figure 4.11(d) (GONZALEZ; WOODS, 2007) for direct f'' computation, as illustrated on Figure 4.12(c).

$$\begin{vmatrix} -1 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{vmatrix} \qquad \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

(a) 1D Prewitt (b) 2D h. Prewitt (c) 2D v. Prewitt (d) Laplacian

Figure 4.11: 2D representation of the horizontal (b) and vertical (c) Prewitt filters for computing f', extended from a 1D representation (a). The Laplacian filter (d) is used for direct computation of f''. Both filters can be applied in structured grids.

The filters presented in figure 4.11 have been extended to 3D as follows. Both masks can be seen as a cube with with 27 elements. In the Prewitt case, all elements whose index in the axis being analyzed is smaller than the current one receive weight -1. All elements whose index are greater than the current one receive weight 1. All elements with the same index as the current one receive weight 0. In the case of the Laplacian filter, the current value has a weight of 26, while all surrounding values receive weight -1. Results using these configurations are presented in figure 4.12.



(a) Prewitt $f \times f'$ (b) Prewitt $f \times f''$ (c) Laplacian $f \times f''$

Figure 4.12: Gradient-magnitude histograms using approximations of f' and f'' using Prewitt, Prewitt applied over previous result, and Laplacian operators, respectively. Sphere regular grid with 128^3 samples.

Since unstructured grids may not have well distributed neighbors these filters are not directly applied to them. Instead it employed the Least Squares method, used for computing f' for both regular and irregular structures. This technique is explained in the following section.

4.5.2 Least Squares

The Least Squares (LS) technique is widely known for finding a good fit for a set of points. It has been used on the literature to reconstruct the gradient of scalar fields and perform filtering (HASELBACHER, 2001).

There are two types of least squares: linear and non-linear. This work will only use the linear least squares, since we want to find a direction (represented by a straight line). It works by finding adjustable values in a vector β such that the residual R^2 is a minimum:

$$R^2 = \sum_{i=1}^{p} \left(y_i - f(x_i, \beta)\right)^2 \tag{4.6}$$

where the dependent variable y_i is associated to the independent variable x_i through the function f. To be a minimum, R^2 must satisfy

$$\frac{\partial(R^2)}{\partial r_j} = 0, j \in [1..n] \tag{4.7}$$

In the case of linear least squares, the parameters of β are linearly combined with x_i, such that

$$f(x_i, \beta) = \sum_{j=1}^{n} x_{ij}\beta_j \tag{4.8}$$

Combining equations 4.6, 4.7 and 4.8 resultis in p linear equations

$$\frac{\partial(R^2)}{\partial r_j} = -2\sum_{i=1}^{p} x_{ij}\left(y_i - \sum_{j=1}^{n} x_{ij}\beta_j\right) \tag{4.9}$$

$$\sum_{i=1}^{p} x_{ik}y_i = \sum_{i=1}^{p}\sum_{j=1}^{n} x_{ik}x_{ij}\beta_j \tag{4.10}$$

Equation 4.10 can be written in matrix notation as

$$X^T y = \left(X^T X\right)\beta \tag{4.11}$$

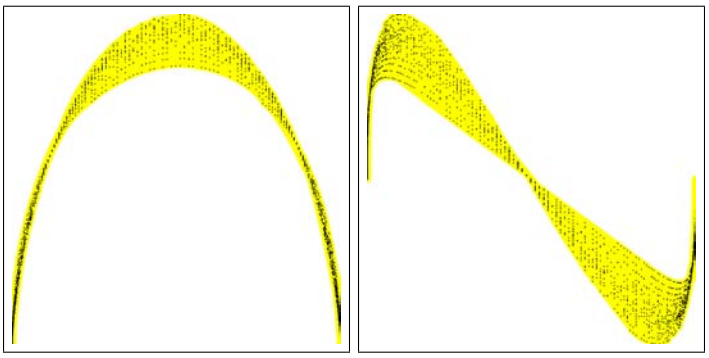
This work uses equation 4.11 to find the gradient vector β using sample points distributed around the one analyzed.

Besides the classification into linear and non-linear, the least squares method can use a weighting parameter to try to compensate some distortions, like the distribution of samples. This results in the Weighted Least Squares (WLS):

$$R^2 = \sum_{i=1}^{p} \left(y_i - f(x_i, \beta)\right)^2 \times w_i \tag{4.12}$$

where w_i is a weighting parameter associated to the data point x_i. When the weighting parameter is applied to a specific sample, biasing the calculation toward that value it is called Moving Least Squares (MLS). Several researches suggest that the use of weighted least squares produces better results than the unweighted version (MAVRIPLIS, 2003, 2007). Such works also suggest the use of the inverse of the distance between samples as a good choice for the weighting parameter.

Six neighbors have been used in the fitting process for each sample of a regular scalar field. Since all samples are taken at a fixed distance from the central sample both LS and WLS present the same result (the distance to the central sample was used as the weighting metric for the MLS method). Resulting histograms can be seen on Figure 4.13. Notice the presence of gaps due to the disposition of samples, as discussed in section 4.4.1.

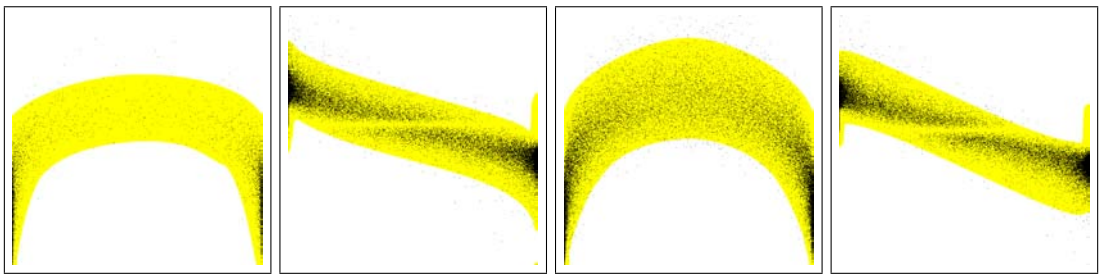


(a) $f \times f'$ (b) $f \times f''$

Figure 4.13: Gradient-magnitude histograms using approximations of f' and f'' using Least Squares. Sphere regular grid with 128^3 samples.

The computation for unstructured grids uses a prefixed amount of neighbors around the one being analyzed, to ensure the best fidelity while approximating the original function at that sample and a reasonable computational cost. The dataset used for these exper-

iments was the sphere with samples concentrated in the boundary of the sphere, as illustrated in figure 4.9. of section 4.4.2. Several works rely on this technique to compute the gradient for unstructured grids (MAVRIPLIS, 2007; SHEWCHUK, 2002). Figure 4.14 shows the result of using both LS and WLS with 32 neighbors per sample.



(a) LS $f \times f'$ (b) LS $f \times f''$ (c) WLS $f \times f'$ (d) WLS $f \times f''$

Figure 4.14: Gradient-magnitude histograms using approximations of f' and f'' with least squares and moving least squares. Sphere unstructured grid with dense boundary and $100k$ samples.

Analyzing figures 4.14(a) and 4.14(c) one can see that it is easier to recognize the arc-like shape of the histogram with MLS than with LS, since there are more samples scattered in the top of the arc, although both methods are not as clear as the structured case. Computed over the result of f', the histogram for f'' accumulates that error twice, resulting in bad shapes for both techniques.

The quality of the (M)LS will be further analyzed in the next section, along with the Prewitt method for structured grids.

4.5.3 Reconstruction Analysis

Sections 4.5.1 and 4.5.2 presented different methods to approximate f' and f''. This section discusses the quality of these reconstruction methods.

In the moving least squares method, both the simple inverse of the distance $\frac{1}{d_i}$ and the normalized inverse of the distance $\frac{d_c}{d_i}$ have been analyzed, where d_c is the distance of the point closest to the analyzed sample.

Four metrics have been used to compare the reconstruction methods with the analytic solution:

- Difference in magnitudes (DM): corresponds to the module of the difference between the reconstructed and analytic solutions. Small differences represent more accurate and robust solutions;
- Ratio between magnitudes (RM): compute *analytic* over *reconstructed* solutions. Better methods results in ratios close to *1*;
- Length of difference in the gradient vector (LDV): compute the difference vector of reconstructed and analytic gradients. Small values represent more accurate answers;
- Dot product (DP): measure the angle difference between analytic and reconstructed normalized gradient vectors. More accurate solution should present values closer to *1*.

Table 4.1 presents the maximum and minimum values for the above metrics computed for structured and unstructured techniques. This comparison was performed for the computation of f', the gradient vector. Least squares and weighted least squares with both weights yield the same results for structured datasets, so they have been displayed in a single column.

Table 4.1: Result of the comparison between the analytic and reconstruction methods according to several metrics: difference in magnitude of the gradient vector(DM), ratio between the magnitudes (RM), length of the difference vector of gradients (LDV) and the dot product between the gradient vectors (DP). Results measured in the structured sphere dataset with 128^3 samples and the unstructured sphere dataset with dense boundary and $100k$ samples.

	Structured		Unstructured		
	(M)LS	Prewitt	LS	MLS ($\frac{1}{d_i}$)	MLS ($\frac{d_c}{d_i}$)
DM_{Min}	0.000000	0.000000	0.000000	0.000000	0.000000
DM_{Max}	0.156699	0.252082	0.623909	0.549086	0.549126
RM_{Min}	0.000000	0.000000	0.000000	0.000000	0.000000
RM_{Max}	1.185967	1.101218	6.009077	3.970824	2.799576
LDV_{Min}	0.000000	0.000000	0.000000	0.000000	0.000000
LDV_{Max}	0.162731	0.252089	0.661233	0.572888	0.572865
DP_{Min}	0.636256	0.669197	-0.999983	-0.999462	-0.999609
DP_{Max}	1.000000	1.000000	1.000000	1.000000	1.000000

This work now separates the analysis of structured and unstructured grids for better comprehension. The next section analyzes the methods described for regular grids, while section 4.5.3.2 analyzes unstructured grids results.

4.5.3.1 Structured Grids

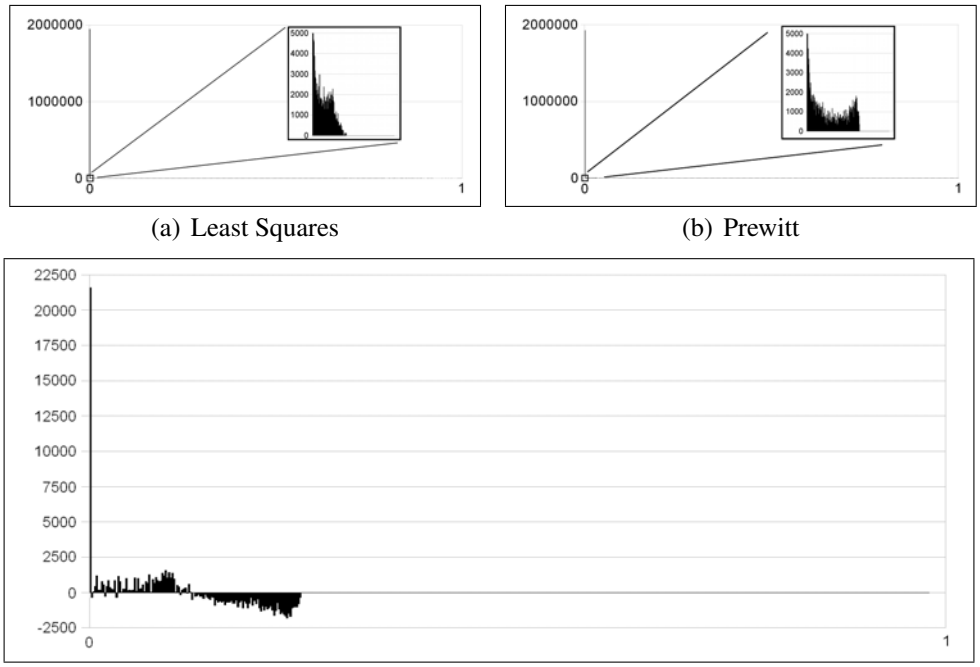
The values presented in table 4.1 for structured grids have two interesting outcomes. First, the least squares method seems to better approximate the magnitude of the gradient, while the Prewitt technique better approximates the gradient vector itself.

The histograms presented in Figures 4.15(a) and 4.15(b) show the deviation of the magnitude of the gradient from the analytic solution for the Least Squares and Prewitt methods. Figures 4.16(a) and 4.16(b) show the cosine of the angle between the analytically computed and approximated gradient. The difference-histogram between these two reconstruction methods can be seen in Figures 4.15(c) and 4.16(c).

The first set of histograms (figure 4.15) confirms the superior quality of the least squares method for approximating the magnitude of the gradient. The LS technique presents a significant larger amount of samples with small deviation from the analytic solution than the Prewitt method. So, for the case of transfer function specification the LS method presents an arc-like shape closer to the analytic solution than the Prewitt method.

On the other hand, the second set of histograms (figure 4.16) show the better approximation of the analytic gradient vector produced by the Prewitt filter. It has an expressive amount of samples whose dot product is closer to 1 than the LS method. If an application (such as fluid dynamics simulations) is more concerned with the gradient direction than with its magnitude, one should use the Prewitt method instead of LS.

Besides the quality of the result, one may be interested in the computational cost of



(a) Least Squares (b) Prewitt

(c) Least Squares - Prewitt

Figure 4.15: Difference between analytic and approximated solutions in the magnitude of the gradient vector: Least squares in (a) and Prewitt in (b). The difference between the Prewitt and Least Squares methods is shown in (c). Here one can see that LS results in better magnitude approximation than Prewitt, due to the concentration of positive samples in the beginning of the difference histogram. Sphere regular grid with 128^3 samples.

using Prewitt or LS. Due to the fact that the Prewitt filter can be applied using a convolution kernel its computation is performed faster than LS, that needs to compute a matrix inversion (if it is possible) to find the gradient vector. Applications that must run in real-time and tolerate a deviation from the solution, the Prewitt filter offers good accuracy and performance.

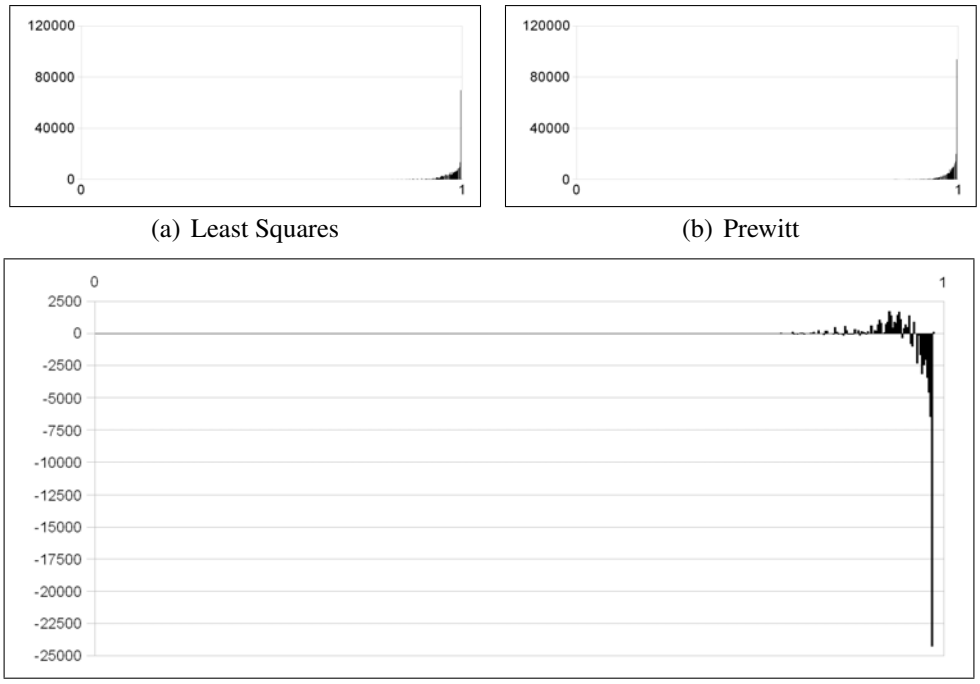
4.5.3.2 Unstructured Grids

Due to the irregular distribution of samples in the unstructured datasets, this work does not have an unstructured version of the Prewitt mask. Instead, it just compares the Moving Least Squares method with the original one, and verify the equivalence of two different weighting factors.

The beginning of section 4.5.3 presented both weighting parameters analyzed in this work: $\frac{1}{d_i}$ and $\frac{d_c}{d_i}$. The maximum and minimum values for all four metrics presented in table 4.1 have been similar, with the exception of the magnitude ratio. A closer look to those values actually demonstrate that all metrics present a very similar distribution of values, as displayed in figure 4.17.

All histograms presented in figure 4.17 oscillate inside a small range of less than 70 elements. The difference between the ratio of the magnitudes, that is the one with the largest divergence between weighting parameters, oscillates in a 10 element range. This clearly demonstrates that there is not a significant difference between both weightings for this dataset. For the remaining of this work, both weighting techniques will be considered the same and only moving least squares in general will be addressed.

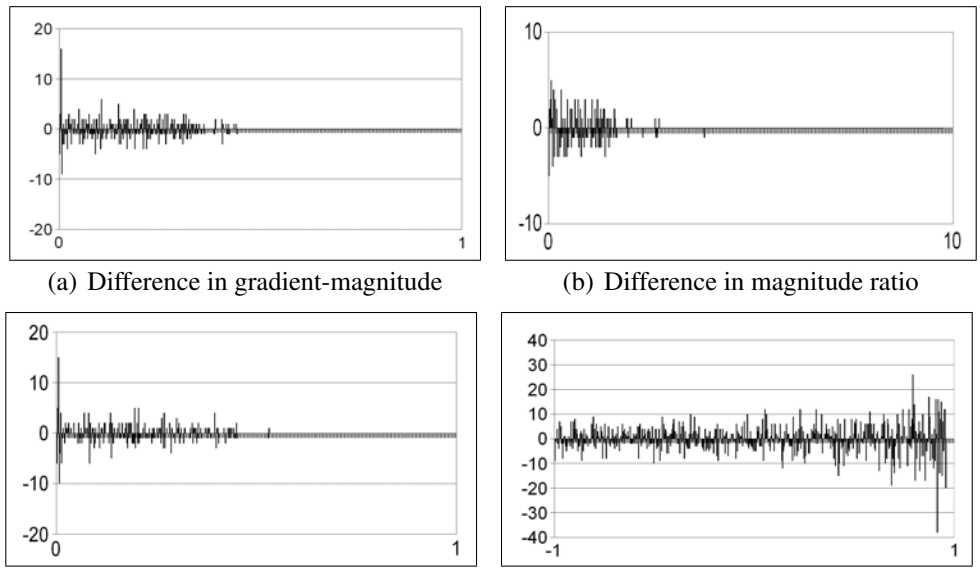
The moving least squares technique has shown more accurate results than the original



(a) Least Squares (b) Prewitt

(c) Least Squares - Prewitt

Figure 4.16: Cosine of the angle between approximated gradient and the analytic solution: Least squares in (a) and Prewitt in (b). The difference between the Least Squares and Prewitt methods is presented in (c), which shows that the Prewitt method results in better approximation of the direction of the gradient over the least squares method, since values are concentrated in the negative region closer to 1. Sphere regular grid with 128^3 samples.

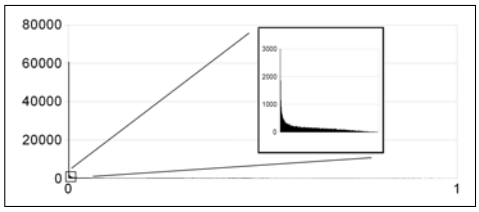


(a) Difference in gradient-magnitude (b) Difference in magnitude ratio

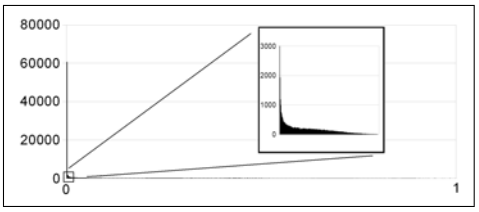
(c) Difference in gradient vector (d) Difference in direction

Figure 4.17: These four images demonstrate the similarity of results using both weighting parameters $\frac{1}{d_i}$ and $\frac{d_c}{d_i}$. Notice that all histograms oscillates between both techniques in a small range (less than 70 elements). Since these weighting parameters are so similar, the remaining of this work will only address MLS in general.

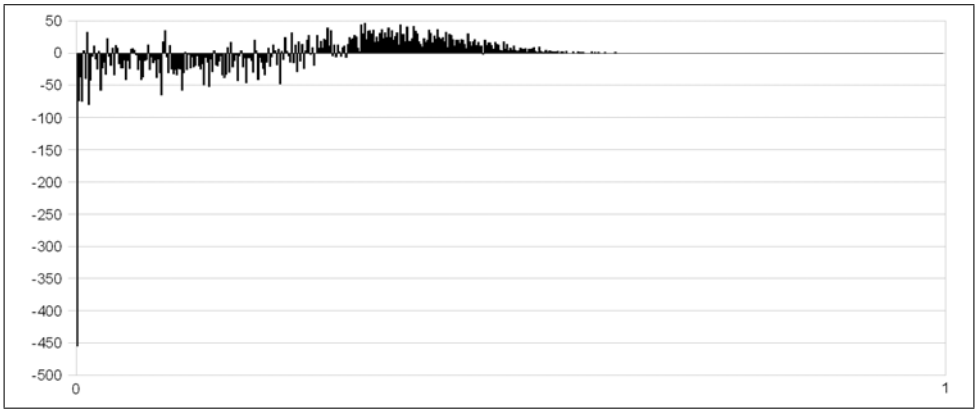
least squares method according to table 4.1. But only analyzing the distribution of the values one can confirm the superiority of one method. Figures 4.18 and 4.19 shows the distribution of values for two of the analyzed metrics, and a comparison between the LS and MLS solutions.



(a) Least Squares



(b) Weighted Least Squares



(c) Least Squares - Weighted Least Squares

Figure 4.18: Difference between the analytic and approximated solutions in the magnitude of the gradient vector: Least squares in (a) and Moving Least Squares in (b). The difference between the Least Squares and Moving Least Squares methods is presented in (c). Here one can see that MLS results in better magnitude approximation than LS. Unstructured sphere dataset with 100k samples.

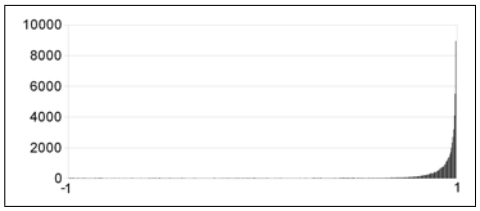
The quality of the gradient-magnitude for unstructured datasets is better approximated by the MLS method. This is comproved by looking at the high concentration of negative values near zero, and how it is sustained for the first quarter of the histogram, as shown in figure 4.18.

The accumulation of positive values near 1 in the histogram of figure 4.19 shows that the standard least squares approach produces better directions for the sphere dataset. Although one may argue that the advantage of LS only extends for a small amount of the histogram, the following sequence of good MLS results also extends for a fairly short range.

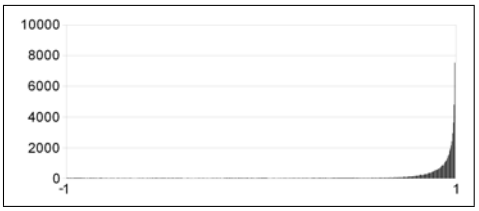
These results suggest that MLS produces more accurate magnitudes while lacking a bit of accuracy for gradient direction. MLS appears to be more robust than standard LS, since several parameters can be used for modulating the sampling points, and like many works point out (MAVRIPLIS, 2003, 2007).

4.6 Smoothing

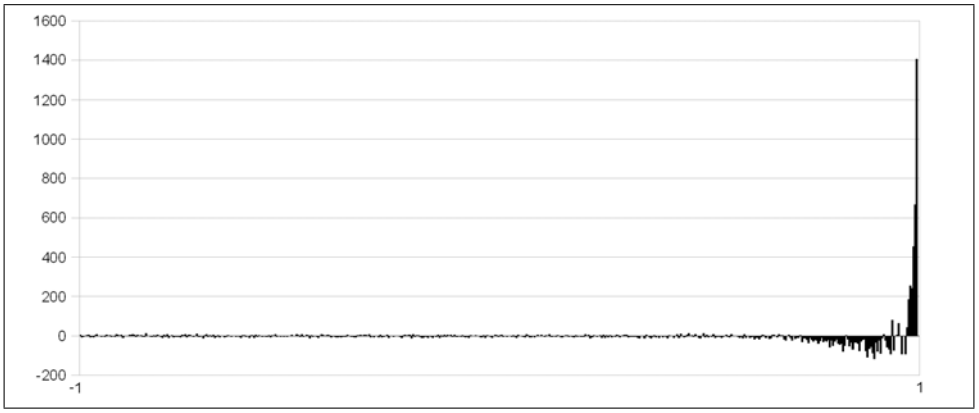
One important factor that affects the quality of gradient-magnitude histograms is the smoothness of the boundary. This is due to two reasons:



(a) Least Squares



(b) Moving Least Squares



(c) Least Squares - Moving Least Squares

Figure 4.19: Cosine of the angle between approximated gradient and the analytic solution: Least squares in (a), Moving Least Squares in (b). The difference between the LS and MLS methods is presented in (c). Although its worse value is better than the LS value, the MLS technique actually performs slightly worse in general than LS for direction approximation. Unstructured sphere dataset with 100k samples.

- smoother boundaries in a discretized domain will be larger than abrupt boundaries, resulting in more samples over it. This will result in gap reduction between samples in the $f(x)$ dimension of the histogram;

- smoother transitions will produce a wide range of gradient magnitudes (according to the boundary model presented in section 4.3.1). This will reduce the gap between histogram samples in $f\prime(x)$ and $f\prime\prime(x)$ dimensions.

An example of the use of smoothing to increase the boundary width and the quality of resulting histograms is the original work of semi-automatic transfer function generation (KINDLMANN; DURKIN, 1998). The authors of said work force smoothing by blurring the scalar field prior to computing any histogram.

Some care must be taken when analyzing a histogram computed over a blurred scalar field. Figure 4.20 shows what would be several materials densely concentrated in the side of the Engine dataset.

Figure 4.20 shows nothing more than an effect caused by interpolation and low sampling: the small amount of samples coupled with a reduced amount of gradient-magnitudes produces “islands” of points. These islands can be mistakenly taken for distinct boundaries in this dataset. Increasing the amount of blur applied in the scalar field results in the replication of islands, increasing the arc quality, as figure 4.21 shows. Eventually it would result in a single arc representing the transition of air to the Engine material.

A drawback of applying Gaussian blur to increase the arc quality is its characteristic of smoothing out small features (GONZALEZ; WOODS, 2007). Fast transitions in the

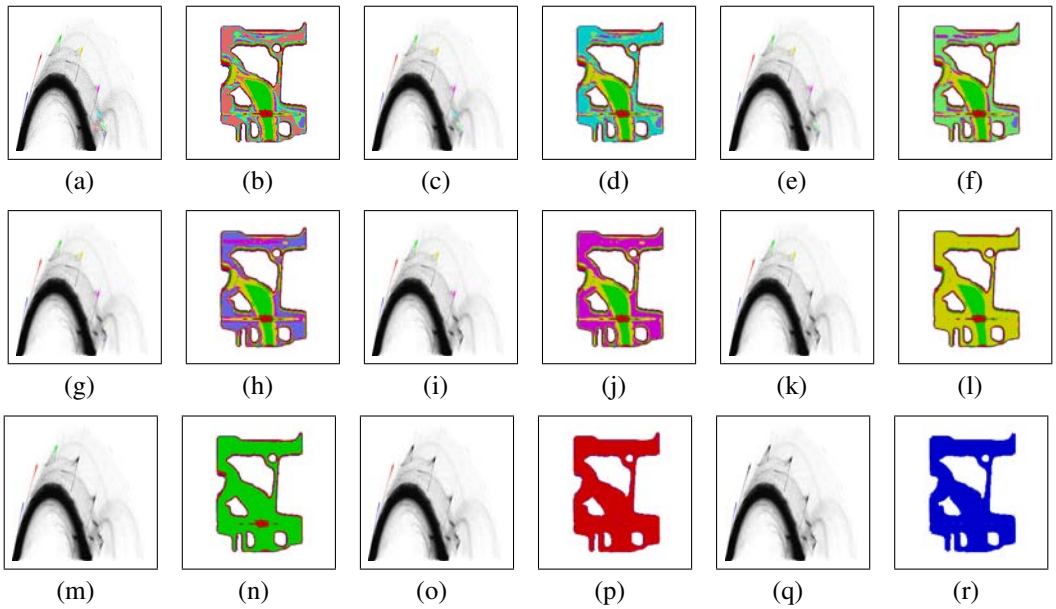


Figure 4.20: Different concentration of points may suggest that there are several different borders in the side of the Engine dataset, as presented above. This effect actually corresponds to a single boundary that lack enough samples to produce a single arc, as discussed in section 4.4.

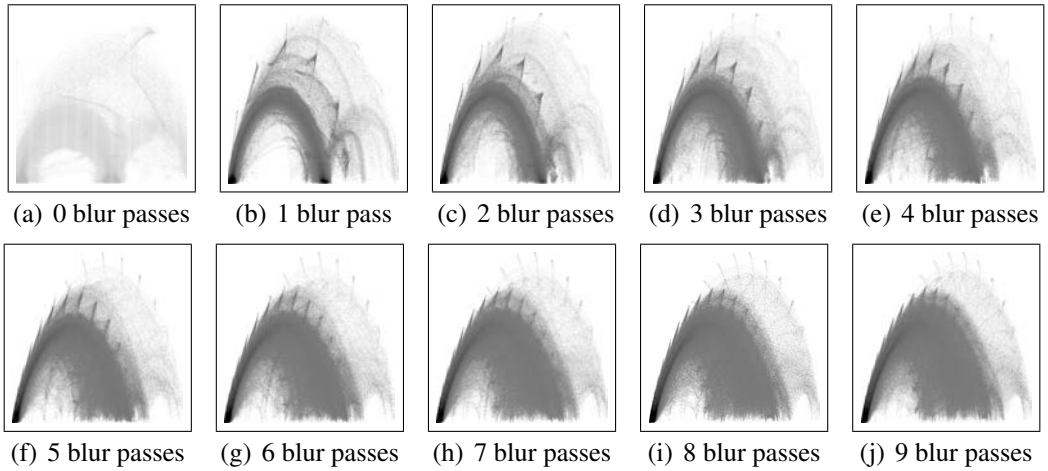
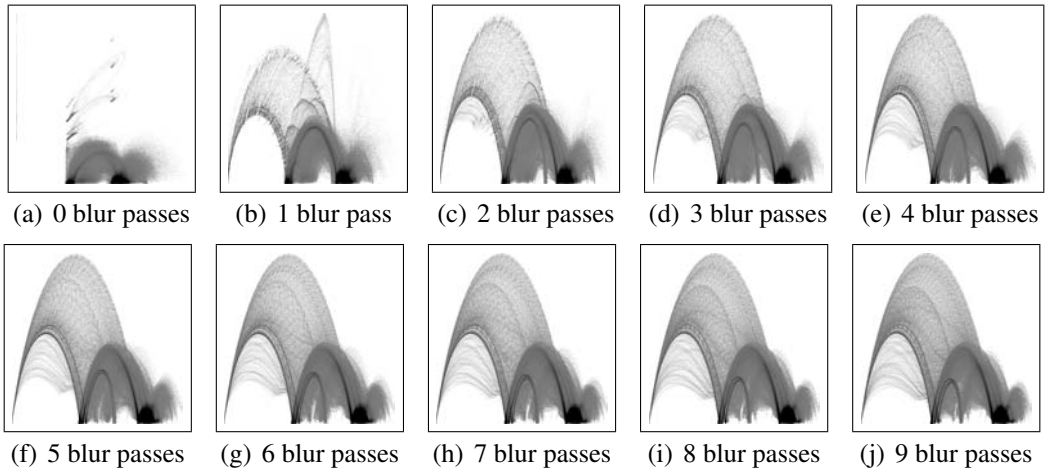


Figure 4.21: Gradient-magnitude histograms computed over the Engine dataset under different blurring levels. In (a) the original scalar field was used, and all following histograms have been computed over an incrementally blurred scalar field. Notice how an outer arc shape becomes more defined with more blurring levels. The Gaussian blur filter used considered an inflection point at 2 samples away and a cut off limit 3 samples away from the central sample.

dataset may be completely missed if their scalar value is close to the one of its neighbors. Another problem is the production of artifacts just like the ones presented above.

The Body dataset (figure 4.22)is another good example of how blur can help enhance present boundaries. Notice that the histogram computed over the original scalar field presents only one identifiable arc structure, contrasting with several shapes easily recognized in the blurred histogram. Artifacts also appear, just like in the engine dataset, although they are not as prominent as the previous ones.



(a) 0 blur passes (b) 1 blur pass (c) 2 blur passes (d) 3 blur passes (e) 4 blur passes

(f) 5 blur passes (g) 6 blur passes (h) 7 blur passes (i) 8 blur passes (j) 9 blur passes

Figure 4.22: The standard gradient-magnitude histogram for the Body dataset in (a) presents only one identifiable arc structured. When filtering the volume one can readily identify several other shapes (b - j). Similar to the Engine dataset, it also presents artifacts when increasing the amount of blur applied.

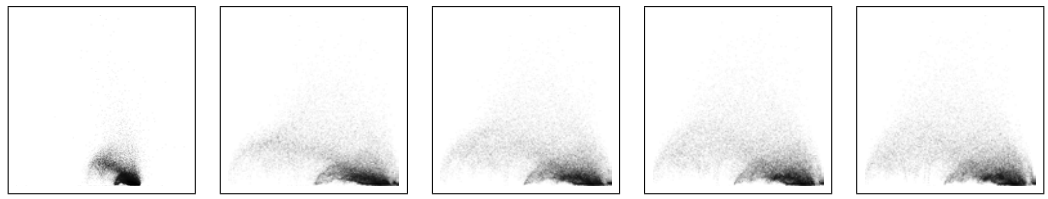
Unstructured datasets also benefit from blurring, although the application of a blurring filter is not as straightforward as in structured grids.

The blurring filter used analytically solves the Gaussian equation for each neighbor of a sample, with a maximum quantity of 32 neighbors per sample. The Gaussian curve is centered in each sample, and has an inflection point at a distance of $2\times$ the dataset mean edge and cut off threshold at $3\times$ the mean edge. Since datasets may have different scales, the individual mean edge was used to translate dataset-independent parameters to the dataset domain.

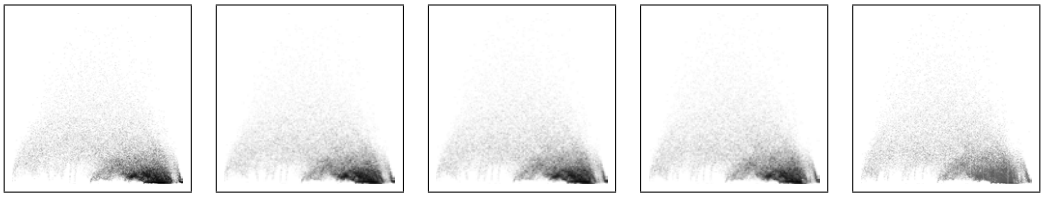
Figures 4.23, 4.24 and 4.25 show examples of blurring effects in different unstructured datasets. One can notice in all images an increased quality in the arc shapes following the increase in the blurring level.

Just like in the structured case, unstructured datasets are also subject to artifacts. The San Fernando earthquake simulation dataset, for instance, initially presents some identifiable independent concentration of points. With the increase of blur these features break into smaller pieces, suffering from the same quantization problem already stated for the Engine dataset (figure 4.20).

Blurring is a powerful method for increasing the quality of the arc structure, according to the boundary model adopted. The major drawback of using it is the appearance of issues in the resulting histograms that may be misinterpreted. This work has identified one of the possible artifacts, that will likely occur at thin boundaries that lack enough samples to produce an arc without gaps.



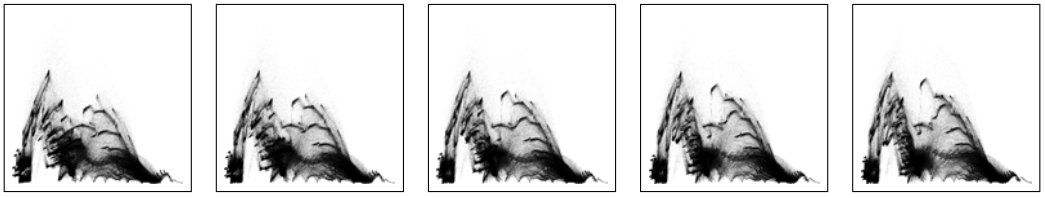
(a) 0 blur passes (b) 1 blur pass (c) 2 blur passes (d) 3 blur passes (e) 4 blur passes



(f) 5 blur passes (g) 6 blur passes (h) 7 blur passes (i) 8 blur passes (j) 9 blur passes

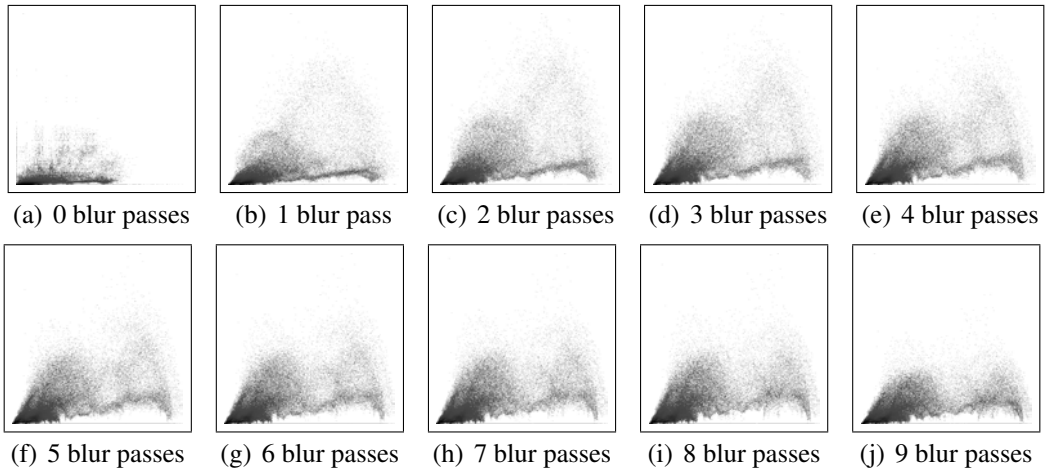
Figure 4.23: Example of how blurring helps increasing the quality of arc-like shapes in gradient-magnitude histograms for unstructured datasets. The original histogram for F117 dataset in (a) presents lots of samples concentrated in the middle of the image. Consecutive blurring steps help eliminating small sources of distortions and increase the quality of recognizable boundaries ((b) to (j)). A maximum quantity of 32 neighbors near the sample have been used to blur the scalar field, with the gaussian inflection point at $2 \times MeanEdge$ and cut off threshold at $3 \times MeanEdge$.

(a) 0 blur passes (b) 1 blur pass (c) 2 blur passes (d) 3 blur passes (e) 4 blur passes



(f) 5 blur passes (g) 6 blur passes (h) 7 blur passes (i) 8 blur passes (j) 9 blur passes

Figure 4.24: Gradient-magnitude histograms of the San Fernando earthquake simulation dataset under several blurring levels. The original gradient-magnitude histogram in (a) has its main boundaries enhanced through the application of blurring in the scalar field prior to computing it (images (b) to (j)). Blurring is applied with 32 samples per vertex with the gaussian inflection point at $2 \times MeanEdge$ and cut off threshold at $3 \times MeanEdge$.



(a) 0 blur passes (b) 1 blur pass (c) 2 blur passes (d) 3 blur passes (e) 4 blur passes

(f) 5 blur passes (g) 6 blur passes (h) 7 blur passes (i) 8 blur passes (j) 9 blur passes

Figure 4.25: The standard gradient-magnitude histogram in (a) for the Super Phoenix (SPX) dataset barely has identifiable arc shapes. Successive blurring filter in (b) to (j) increase the quality of identifiable boundaries using the model described in section 4.3.1. The gaussian blur filter used has inflection point at $2 \times MeanEdge$ and cut off threshold at $3 \times MeanEdge$, with a maximum of 32 closest neighbors used for each sample.

4.6.1 Other Unstructured Grids Issues

One problem identified while processing several unstructured grids was the concentration of samples in the bottom of the gradient-magnitude histogram. This is due to some samples that are too far from the other ones in both gradient and spatial domains.

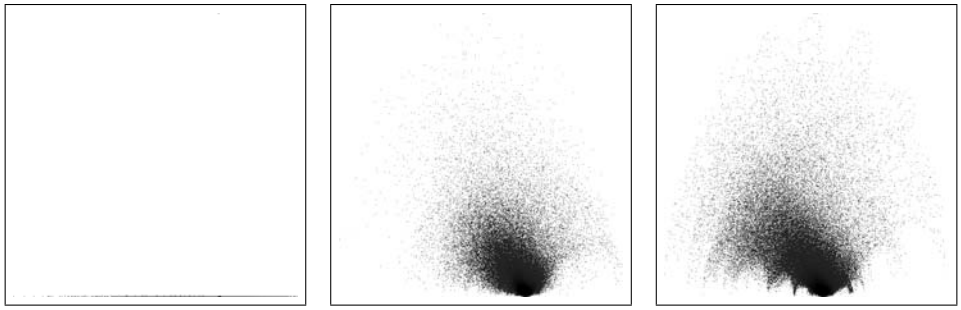
The large difference between the magnitude of the gradient of certain samples can be largely avoided with the use of a Gaussian blur to remove high frequencies. However, the implementation of the Gaussian blur filter of this work discards neighbor samples distant more than 3 times the mean edge of the dataset. If a dataset has degenerated samples with distant neighbors, the effects of blurring will be small or even insignificant.

In such situation this work proposes the use of statistical measures to discard some data samples that are far different from the others. This approach works by computing the mean gradient-magnitude for the entire dataset, as well as the standard deviation. Next, all samples whose deviation from the mean is larger than *n* times the variance are discarded. Figure 4.26 shows the use of this technique to compute the gradient-magnitude histogram for the Turbulent Jet dataset. Notice how there are shapes clearly identifiable in the histogram, contrasting with the original image. To produce these images all samples whose distance to the mean are larger than 10 times the variance have been discarded.

The mean edge for the TJet is approximately 6.26, while the maximum edge has length of 39.97. This results in a ratio of about 6.37 between the mean edge and the maximum edge, so the solely application of blur does not help improving the histogram quality.

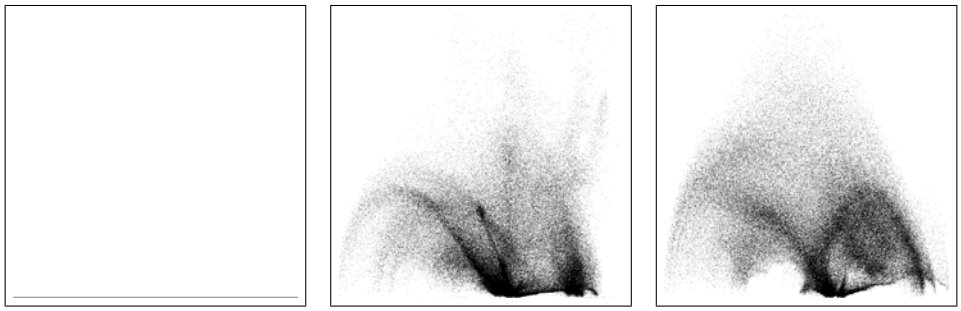
Another unstructured dataset that presents this same issue is the Jets dataset. This dataset is a simulation of combustion of a rocket engine. Figure 4.27 shows the gradient-magnitude histogram prior to the statistical filter and after applying it. This dataset has a mean edge length of 7.99 and maximum edge length of 40.90, also a factor larger than 3 times between these two measures.

Different datasets may require specific values to correctly discard distorted values while preserving good samples. If the variance multiplier parameter is too small standard values may be dropped from the final histogram along with erroneous ones.



(a) (b) (c)

Figure 4.26: Only blurring the dataset is not enough to produce good results for the turbulent jet dataset. The original gradient-magnitude histogram is presented in (a). Notice how almost all samples are concentrated at the bottom of the histogram. (b) shows the same histogram using the statistical filter to discard distorted samples. The same procedure applied in a blurred version of the dataset is presented in (c).



(a) (b) (c)

Figure 4.27: Just like the TJet case, the Large Jets dataset suffers from distorted samples that negatively affect the histogram. Just blurring the dataset is not enough again, and statistical filters must be used to discard erroneous samples. The original gradient-magnitude histogram with almost all samples concentrated at its lower part is presented in (a). The same histogram is shown in (b), this time using the statistical filter to discard distorted samples. (c) shows a blurred version of this dataset along with the statistical procedure.

4.7 Discussion

The analysis of statistical signatures (aka, the gradient-magnitude histogram specifically addressed in this work) can reveal insightful information about the features of dataset.

Through a controlled environment, this work demonstrated what problems may arise from different sampling strategies for both structured and unstructured datasets. Approximation issues have also been analyzed, and includes a comparison of the quality of different approximation schemes against the analytical solution.

Blur has already been used in the literature to preprocess datasets and increase the quality of histograms. Although its relatively widespread use, this practice has not been a direct subject in researches, and issues caused by it have not been fully studied and reported. This work recognize its benefits and also identified a relatively common source of errors in the presence of thin boundaries.

The next chapter will discuss more direct applications of statistical signatures applied to both datasets feature extraction and transfer functions specification.

5 APPLICATIONS

As already stated in section 4, statistical data signature have been extensively used for feature detection and analysis throughout the literature. It is a powerful method for unveiling underlying data features that may be hard to manually identify and enhance.

There is a large amount of applications that can benefit from function statistics. This chapter is going to address two specific applications that use it for enhancing features in volume rendering. Section 5.1 is going to specifically address the problem of finding features in the volumetric dataset, while section 5.2 will explain how to solve the problem identified in section 3.2 concerning the development of transfer functions for unstructured volume rendering.

5.1 Feature Extraction

The first application for function statistics is feature detection and enhancement. An user that wants to analyze a dataset without prior knowledge of both its features and where such features are located will face two great problems: what do these features mean and how to locate them.

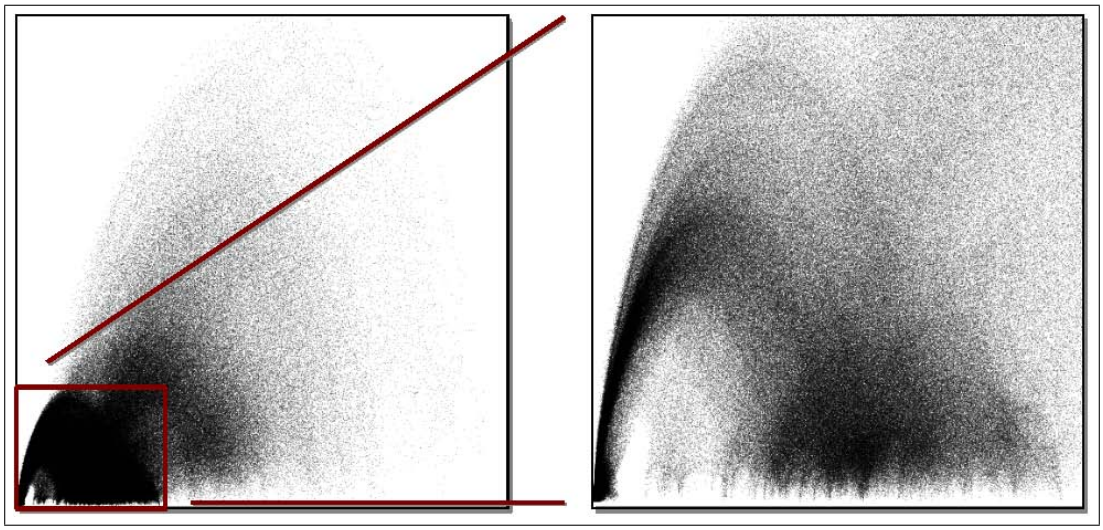
The first problem is subjective and, actually, it is the greatest challenge the user must face. It is inherent to the problem the user is studying.

The second difficulty can be eased by function statistics, that capture underlying data information using a specific data model. Through a correspondence between the histogram space and the respective spatial position users can quickly identify the location of possibly interesting features.

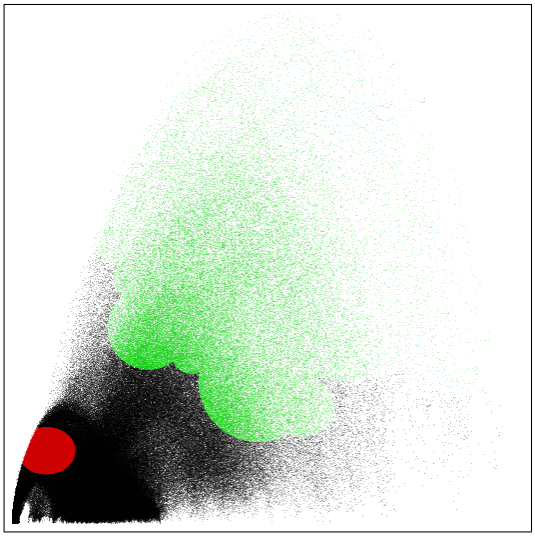
Figure 5.1 shows a program developed to display the relationship of histogram samples with the spatial domain. Users can assign different colors in the histogram and see the associated points in the function domain. Program features include a zooming tool and opacity control for the histogram. Using this tool users can quickly identify where the features of interest are located in the spatial dataset domain.

Users can additionally specify a transfer function to be used with volume rendering within the spatial domain, as illustrated in figure 5.2. This is interesting to demonstrate the correlation between a designed transfer function and the respective histogram samples, as well as to perform a possible focus+context visualization.

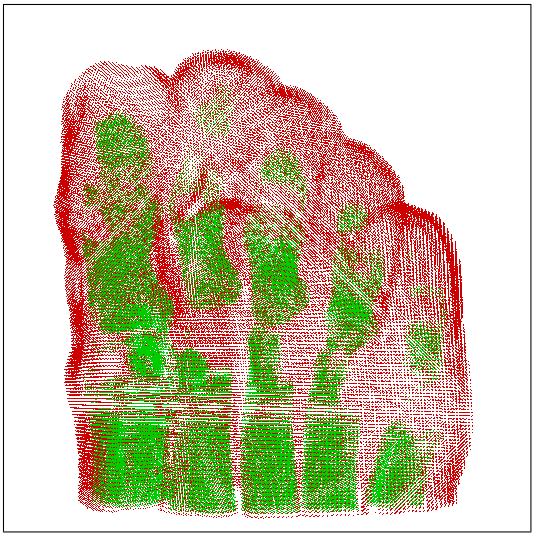
This may be an interesting functionality to analyze the dataset. But to actually analyze the features of a volumetric dataset, more robust visualization techniques are required. Next section is going to revisit the transfer function approach presented in section 3.2, explaining how one can deal with specific issues of unstructured datasets.



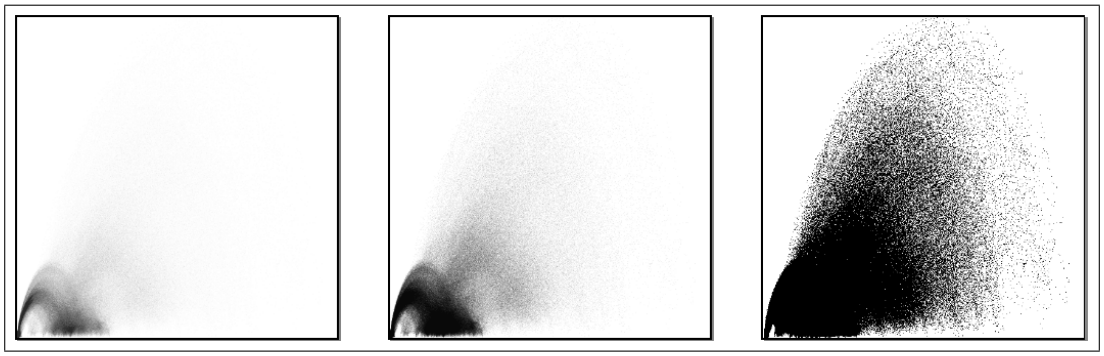
(a) Zooming tool



(b) Histogram space

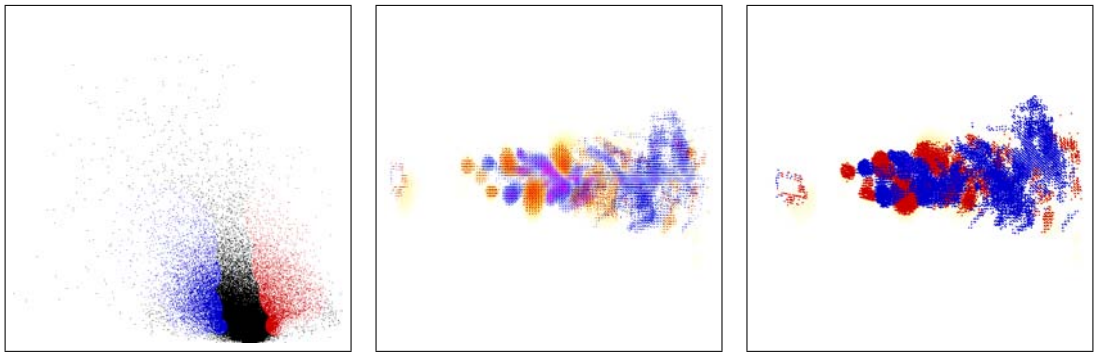


(c) Main visualization



(d) Opacity control

Figure 5.1: This dataset viewer correlates the histogram space with the spatial domain. Samples selected in the histogram have its respective counterparts enhanced in the 3D space. In (a) is shown an example of the importance of zoom to distinguish the boundary shapes on areas of high concentration of points. The histogram samples selected in (b) have its spatial position displayed in (c). The opacity level can be interactively modified to explore high concentration of samples, as presented in (d). This is a reduced version of the Foot dataset, a CT scan of a human foot.



(a) (b) (c)

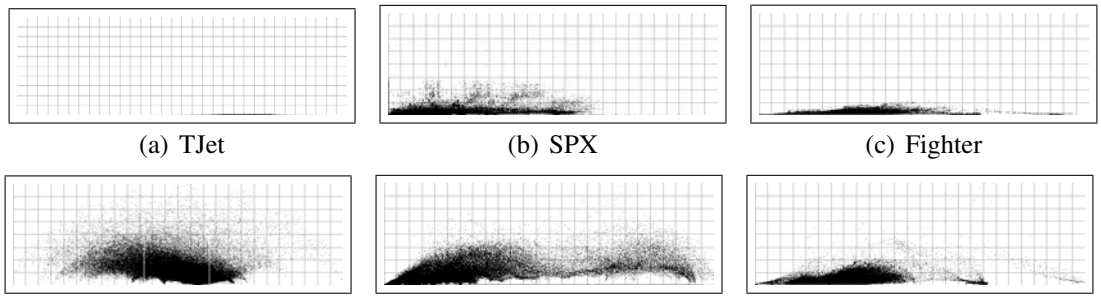
Figure 5.2: The viewer can also display a volume rendering of the dataset along with the sample positions. (a) shows the gradient-magnitude histogram for the TJet dataset with selected regions that are enhanced in (b) and (c) along with a volume rendering using a custom transfer function.

5.2 Transfer Functions Revisited

When firstly introduced in section 3.2, the use of histograms for guiding user choices when designing a transfer function presented issues in histograms for unstructured grids. These issues led to an analysis of the properties in the classical boundary model for structured datasets and its extension to unstructured grids, described in chapter 4.

Once all issues concerning unstructured grids have been sorted out we can finally revisit the proposed solution and verify the usefulness of gradient-magnitude histograms for unstructured grids.

Figure 5.3 shows three different unstructured datasets that previously had no identifiable shape on the gradient-magnitude histogram. Notice how the quality of the gradient-magnitude histograms have been consistently improved concerning the adopted boundary model.



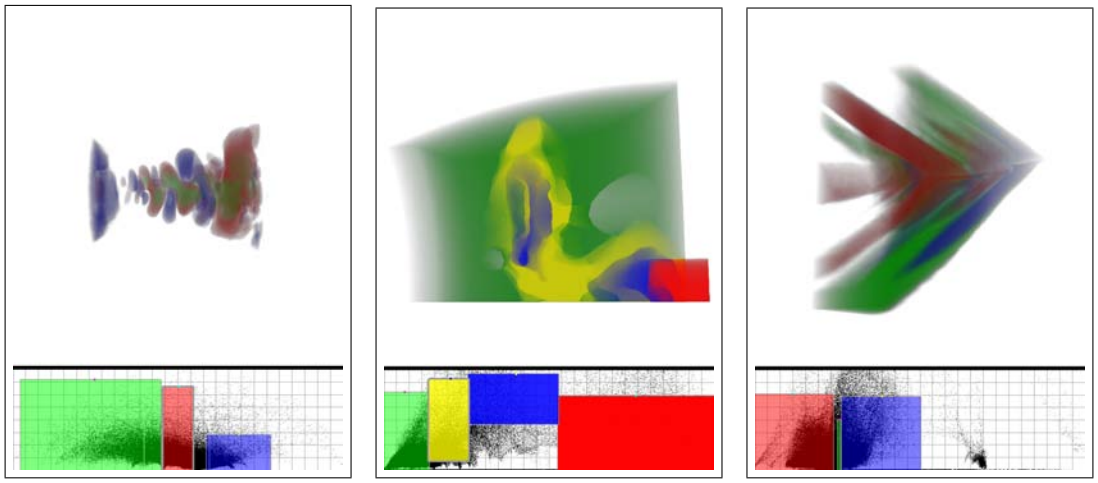
(a) TJet (b) SPX (c) Fighter

(d) TJet (e) SPX (f) Fighter

Figure 5.3: Comparison between standard computation of the gradient-magnitude histogram and the enhanced version (blurred scalars and distorted samples discarded). The TJet dataset (a) and (d), the SPX dataset (b) and (e) and the Fighter dataset (c) and (f).

The transfer function design program introduced in chapter 3 has been used to produce transfer functions using these histograms, as presented in figure 5.4. Using the enhanced gradient-magnitude histogram users have more information about where widgets should be placed to capture interesting features of those datasets.

With the specific issues of unstructured grids sorted out, the usefulness of gradient-magnitude histograms is now fully available for irregular grids as well.



(a) TJet dataset (b) SPX dataset (c) Fighter dataset

Figure 5.4: Example of unstructured grids volume rendering with associated transfer functions created using the gradient-magnitude histogram.

5.3 Discussion

Using the enhanced version of the gradient-magnitude histogram users can now take advantage of tools previously created for structured datasets. Since several applications rely on irregular grids, this extension has high potential for practical use.

This chapter described two applications for unstructured grids. Using a viewer to inspect where sample points of the histogram correspond to spatial positions, users can quickly identify interesting regions.

Another important application is transfer function design. Now that the problems identified on the gradient-magnitude histogram have been sorted out, several works previously developed on structured grids can be applied to irregular meshes.

More applications than the ones here described can be developed. The next section will explore interesting applications for the use of the gradient-magnitude on both structured and unstructured datasets.

5.4 Future work

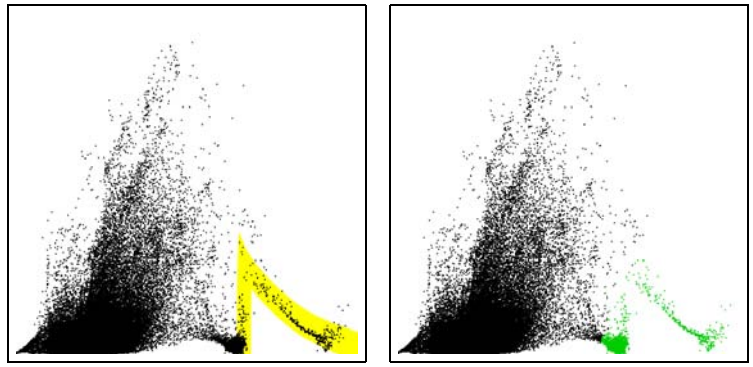
There are several possible applications for function statistics. This work has previously described two different uses in sections 5.1 and 5.2, but others are possible.

The following applications have been briefly explored and need further study before reaching a definitive conclusion. Section 5.4.1 describes the definition and use of other boundary models. Section 5.4.2 presents a new metric to be used for unstructured dataset simplification that preserves function statistics for latter analysis and study.

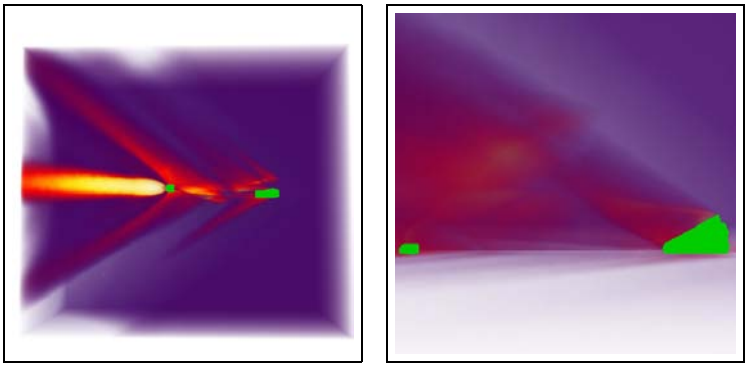
5.4.1 Other Boundary Models

This work has been using a boundary model based in the convolution of a *step* with a *Gaussian* function, just like many other works do (KINDLMANN; DURKIN, 1998; KINDLMANN, 2002; SERLIE et al., 2003). This results in an *erf* as the model of the boundary, whose first and second derivatives are well known, as explained and motivated in section 4.3.1. Although widely used, the *erf* boundary model is not a rule and there are other possible boundary shapes.

Different shapes have been identified in the gradient-magnitude histogram that suggest the presence of distinct boundary models. Figure 5.5 shows the gradient-magnitude histogram of the Fighter unstructured dataset along with the respective data points enhanced in the 3D volume space.



(a) Pattern different from arc. (b) Selected boundary shape.



(c) Visualization of (a). (d) Visualization of (b).

Figure 5.5: Example of a different boundary shape in the Fighter dataset (a). The selected points in the gradient-magnitude histogram in (b) are enhanced in the dataset in (c) and (d).

Certain types of datasets may present a predominant specific shape associated with a given boundary model. Other works have already stated that ark-like shape is the predominant boundary model in medical datasets (KINDLMANN; DURKIN, 1998), since there is a well-defined transition between relatively homogeneous tissues. This property of medical data is the reason why the boundary model described in section 4.3.1 was firstly proposed and is widely accepted.

Although widely used, the assumption that boundaries are a smoothed out step function should not be the only alternative considered for boundary models. While it is true for relatively large transitions between regions of homogeneous materials, boundaries with other shapes have been identified in real datasets. Whether these boundaries represent useful information depends on the subject in study and its characteristics. Other boundary models are an interesting subject for future research and development.

5.4.2 Dataset Simplification

Another possible application for function statistics specifically for unstructured grids is a metric for data simplification.

Dataset simplification is often used to reduce the size of a mesh, reducing its memory footprint and saving processing time (WALTER; HEALEY, 2001). Techniques that

perform this kind of operation usually support a given metric to preserve a specific mesh characteristic. The approach for using function statistics for dataset simplification is discussed next.

Firstly the gradient-magnitude histogram (or another histogram suited for capturing underlying data features) is computed. Then all features to be preserved must be selected in the histogram. This can be performed manually by an user or automatically, using an algorithm that searches for specific patterns, such as a boundary model.

Next, the actual simplification process takes place. Vertices are sequentially removed from the mesh if:

- removing such vertex do not propagates an error to its n nearest neighbors such that they leave the zone selected in the histogram;
- they represent a point inside a selection zone.

Tests of using the Hough transform (ILLINGWORTH; KITTLER, 1988; FERNANDES; OLIVEIRA, 2008) for automatically detecting arc-like shapes have been performed, but its conclusion has been delayed due to time constraints.

Dataset simplification is another possible application for function statistics. Preserving interesting features in simplified data is important for posterior analysis of data, as well as to increase the performance for manipulating volumetric datasets.

6 CONCLUSION

Volume rendering is an important tool for scientists to explore the 3D domain of several problems. Although much research has been previously performed on this subject, many challenging problems remain open.

The contributions of this work can be classified into two major groups. Firstly, for the field of volume visualization, the development of new solutions to transfer function design. The concept of Transfer Functions ensembles has been introduced in section 3.3, where previously developed transfer functions can be combined to create completely new ones. This is a powerful method for fast volume exploration, allowing users to create complex transfer functions from simple ones.

Another contribution for volume visualization is a key-frame based approach for time-varying transfer functions. Users can set independent transfer functions for different time instances of a volumetric dataset. The transition between these key-frames is performed using a user-defined transition curve, that controls the speed of a merge between these maps. This a robust solution for both statistically static and statistically dynamic datasets. A drawback of this technique is a possible visual discontinuity due to transfer function changes. It is minimized by the configurable transition curve, but not completely avoided. If it happens, users may become disoriented.

The second group of contributions consists of a deep study on boundary behavior. When applying the classical boundary model on unstructured grids in an ad hoc fashion a completely degenerated gradient-magnitude histogram was produced. To better understand what was happening, this work analyzed the two main sources of errors: the sampling and approximation issues. Sampling issues have been identified mostly in regular datasets, although it can happen in any class. The use of blurring to increase the boundary width prior to the computation of the histogram led to better results, although it can introduce this same error it was trying to circumvent.

Approximation methods also have been analyzed. This study verifies the accuracy of the Weighted Least Squares regression method as the best choice for gradient reconstruction in unstructured datasets. In regular grids, both WLS and central differences presented good results.

Regardless of blurring, some unstructured grids also presented an abnormal concentration of samples at the bottom of the gradient-magnitude histogram, as described in section 4.6.1. This is due to samples too far from the other, what contributes to nullify any high frequency removal that the Gaussian blur offers. To solve this problem, the variance of the gradient-magnitude is computed, and samples whose variance is larger than a given amount are discarded. This resulted in significantly better histograms, but the definition of the variance multiplier is subjected to the analyzed dataset. Although using subjective parameters, this approach makes over a decade of research on transfer

functions specification available for general unstructured grids.

Using the concepts previously developed, two application have been developed. The first one correlates samples in the histogram with its counterpart in the 3D space. This is useful to quickly explore the dataset and find the location of interesting histogram features. The second application is the transfer function program developed for the first set of contributions. This program also benefits from the teachings of the second set of contributions. Other techniques that may also benefit from the enhanced gradient magnitude histogram include a simplification algorithm for unstructured meshes and the proposition of different boundary models, that may occur and be common in certain classes of problems.

Without correctly understanding what issues are presented by different classes of datasets, a common approach for both of them will not succeed. Just after discovering how to deal with such issues developers can truly elaborate a common solution for both structured and unstructured grids. The use of the methods described in this work can also improve the user experience while interacting with transfer function design tools.

REFERENCES

AKENINE-MöLLER, T.; HAINES, E. **Real-Time Rendering** - 2nd ed. [S.l.]: A. K. Peters, 2002.

AKIBA, H.; FOUT, N.; MA, K.-L. Simultaneous Classification of Time-Varying Volume Data Based on the Time Histogram. In: IEEE VGTC/EUROGRAPHICS SYMPOSIUM ON VISUALIZATION, 2006. **Proceedings...** [S.l.: s.n.], 2006.

BAJAJ, C. L.; PASCUCCI, V.; SCHIKORE, D. R. The contour spectrum. In: IEEE VISUALIZATION, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.173, 539.

BERNARDON, F. et al. Interactive Volume Rendering of Unstructured Grids with Time-Varying Scalar Fields. In: EUROGRAPHICS SYMPOSIUM ON PARALLEL GRAPHICS AND VISUALIZATION, 2006. **...** [S.l.: s.n.], 2006. p.51–58.

BERNARDON, F. F. **Ray Casting de Volumes Não-Estruturados com Dados Dinâmicos usando Hardware Gráfico**. Projeto de Diplomação em Ciência da Computação – Instituto de Informática. UFRGS, Porto Alegre, RS.

BERNARDON, F. F. et al. GPU-based Tiled Ray Casting using Depth Peeling. **Journal of Graphics Tools**, [S.l.], v.11, n.3, p.23–29, July 2006.

BERNARDON, F. F. et al. An adaptive framework for visualizing unstructured grids with time-varying scalar fields. **Parallel Comput.**, Amsterdam, The Netherlands, v.33, n.6, p.391–405, 2007.

BRUCKNER, S.; GRöLLER, E. VolumeShop: an interactive system for direct volume illustration. In: IEEE VISUALIZATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.671–678.

BUNYK, P.; KAUFMAN, A.; SILVA, C. Simple, Fast, and Robust Ray Casting of Irregular Grids. In: DAGSTUHL, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.30–36.

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, 2005.

CALLAHAN, S. P. et al. Interactive Rendering of Large Unstructured Grids Using Dynamic Level-of-Detail. In: IEEE VISUALIZATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.199–206.

COMBA, J. L. et al. Fast Polyhedral Cell Sorting for Interactive Rendering of Unstructured Grids. **Computer Graphics Forum**, Amsterdam, v.18, n.3, p.C–369–C–376, 1999. Work presented in EUROPGRAPHICS, 1992.

DIETRICH, C. A. et al. Real-time interactive visualization and manipulation of the volumetric data using GPU-based methods. In: SPIE MEDICAL IMAGING 2004 - VISUALIZATION, IMAGE-GUIDED PROCEDURES AND DISPLAY, 2004. **Proceedings...** [S.l.: s.n.], 2004. v.5, p.181–192.

DOLEISCH, H. et al. Interactive Feature Specification for Simulation Data on Time-Varying Grids. In: CONFERENCE ON SIMULATION AND VISUALIZATION, SIMVIS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.291–304.

DREBIN, R. A.; CARPENTER, L.; HANRAHAN, P. Volume Rendering. **Computer Graphics**, New York, v.22, n.4, p.65–74, Aug. 1988. Work presented in SIGGRAPH, 1988.

ENGEL, K. (Ed.). **Real-Time Volume Graphics**. [S.l.]: A. K. Peters, 2006.

EVERITT, C. **Interactive Order-Independent Transparency**. [S.l.]: NVIDIA Corporation, 1999. White Paper.

FARIAS, R.; MITCHELL, J. S. B.; SILVA, C. T. ZSWEEP: an efficient and exact projection algorithm for unstructured volume rendering. In: IEEE SYMPOSIUM ON VOLUME VISUALIZATION, 2000, New York. **Proceedings...** New York: ACM Press, 2000. p.91–99.

FARIAS, R.; SILVA, C. T. Parallelizing the ZSWEEP Algorithm for Distributed-Shared Memory Architectures. In: JOINT IEEE TCVG AND EUROGRAPHICS WORKSHOP, VOLUMEGRAPHICS, 2001, Wien. **Proceedings...** [S.l.: s.n.], 2001. p.181–194.

FERNANDES, L. A. F.; OLIVEIRA, M. M. Real-time line detection through an improved Hough transform voting scheme. **Pattern Recogn.**, New York, NY, USA, v.41, n.1, p.299–314, 2008.

FODOR, I. **A Survey of Dimension Reduction Techniques**. Livermore: Center for Applied Scientifc Computing, Lawrence Livermore National Laboratory, 2002.

FOUT, N. et al. High-Quality Rendering of Compressed Volume Data Formats. In: EUROVIS, 2005. **Proceedings...** [S.l.: s.n.], 2005.

GARRITY, M. P. Raytracing Irregular Volume Data. **Computer Graphics**, New York, v.24, n.5, p.35–40, Nov. 1990. Work presented in the Workshop on Volume Visualization, 1990, San Diego, US.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing Digital Image Processing**. 3rd ed. [S.l.]: Prentice Hall, 2007.

GROSS, M.; PFISTER, H. (Ed.). **Point-Based Graphics**. [S.l.]: Morgan Kaufmann, 2007. 552p.

GUTHE, S. et al. Interactive rendering of large volume data sets. In: IEEE VISUALIZATION, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.53–60.

HADWIGER, M.; BERGER, C.; HAUSER, H. High-Quality Two-Level Volume Rendering of Segmented Data Sets on Consumer Graphics Hardware. In: IEEE VISUALIZATION, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.301–308.

HANSEN, C.; JOHNSON, C. (Ed.). **Multidimensional Transfer Functions for Volume Rendering**. [S.l.]: Academic Press, 2005. p.189–210.

HASELBACHER, A. **Discrete Filtering on Unstructured Grids Based on Least-squares Gradient Reconstruction**. Urbana, IL: Center for Simulation of Advanced Rockets, University of Illinois at Urbana-Champaign, 2001.

HEGE, H.-C.; HöLLER, T.; STALLING, D. **Volume Rendering - Mathematical Models and Algorithmic Aspects**. [S.l.]: ZIB, 1993.

ILLINGWORTH, J.; KITTLER, J. A survey of the Hough transform. **Comput. Vision Graph. Image Process.**, San Diego, CA, USA, v.44, n.1, p.87–116, 1988.

JANKUN-KELLY, T.; MA, K.-L. A Study of Transfer Function Generation for Time-Varying Volume Data. In: VOLUME GRAPHICS WORKSHOP, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.51–65.

KINDLMANN, G. **Transfer Functions in Direct Volume Rendering**: design, interface, interaction. [S.l.]: Scientific Computing and Imaging Institute, School of Computing, University of Utah, 2002.

KINDLMANN, G.; DURKIN, J. W. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In: IEEE SYMPOSIUM ON VOLUME VISUALIZATION, 1998. **Proceedings...** [S.l.: s.n.], 1998. p.79–86.

KITWARE INC. **Paraview**. Disponível em: `http://www.paraview.org/`. Acesso em: 2008.

KNISS, J.; KINDLMANN, G.; HANSEN, C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: IEEE VISUALIZATION, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.255–262.

KNISS, J.; KINDLMANN, G.; HANSEN, C. Multi-Dimensional Transfer Functions for Interactive Volume Rendering. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.8, n.3, p.270–285, July 2002.

KÖNIG, A.; GRÖLLER, E. Mastering Transfer Function Specification by Using VolumePro Technology. In: SPRING CONFERERENCE ON COMPUTER GRAPHICS, 2001, Budmerice, Slovakia. **Proceedings...** Los Alamitos, CA: IEEE Computer Society, 2001. p.279–286.

KRAUS, M.; ERTL, T. Cell-projection of cyclic meshes. In: IEEE VISUALIZATION, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.215–222.

KRAUS, M.; QIAO, W.; EBERT, D. S. Projecting Tetrahedra without Rendering Artifacts. In: IEEE VISUALIZATION, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.27–34.

KRISHNAN, S.; SILVA, C.; WEI, B. A Hardware-Assisted Visibility-Ordering Algorithm With Applications to Volume Rendering. In: DATA VISUALIZATION, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.233–242.

KRUEGER, J.; WESTERMANN, R. Acceleration Techniques for GPU-based Volume Rendering. In: IEEE VISUALIZATION, 2003. **Proceedings...** [S.l.: s.n.], 2003.

LACROUTE, P.; LEVOY, M. Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1994. **Proceedings...** New York: ACM, 1994. p.451–458.

L'ECUYER, P. Quasi-monte carlo methods in practice: quasi-monte carlo methods for simulation. In: CONFERENCE ON WINTER SIMULATION, WSC, 2003. **Proceedings...** New York: ACM, 2003. v.35, p.81–89.

LEVOY, M. Display of Surfaces from Volume Data. **IEEE Computer Graphics and Applications**, [S.l.], v.8, n.3, p.29 – 37, May 1988.

LEVOY, M. **Display of Surfaces from Volume Data**. 1989. Thesis (PhD on Computer Science) — University of North Carolina at Chapel Hill.

LI, W.; MUELLER, K.; KAUFMAN, A. Empty Space Skipping and Occlusion Clipping for Texture-based Volume Rendering. In: IEEE VISUALIZATION, 2003. **Proceedings...** [S.l.: s.n.], 2003.

LINDE, Y.; BUZO, A.; GRAY, R. An algorithm for vector quantizer design. **IEEE Transactions on Communications**, [S.l.], v.1, p.84–95, Jan. 1980.

LÜRIG, C.; GROSSO, R.; ERTL, T. Implicit Adaptive Volume Ray-Casting. In: GRAPHICON, 1997. **Proceedings...** [S.l.: s.n.], 1997. p.114–120.

LUM, E. B.; MA, K.-L. Lighting Transfer Functions Using Gradient Aligned Sampling. In: IEEE VISUALIZATION, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.289–296.

MA, K.-L. Image Graphs - A Novel Approach to Visual Data Exploration. In: IEEE VISUALIZATION, 1999. **Proceedings...** [S.l.: s.n.], 1999. p.81–88.

MA, K.-L.; SHEN, H.-W. Compression and Accelerated Rendering of Time-Varying Volume Data. In: INTERNATIONAL SYMPOSIUM WORKSHOP ON COMPUTER GRAPHICS AND VIRTUAL REALITY, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.82–89.

MANAGULI, R.; YOO, Y. M.; KIM, Y. Multi-Volume Rendering for Three-Dimensional Power Doppler Imaging. **IEEE Ultrasonics Symposium**, [S.l.], v.4, p.2046–2049, 2005.

MARKS, J. et al. Design Galleries: a general approach to setting parameters for computer graphics and animation. **Computer Graphics**, [S.l.], v.31, p.389–400, 1997.

MAVRIPLIS, D. J. **Revisiting the Least-squares Procedure for Gradient Reconstruction on Unstructured Meshes**. [S.l.]: NASA Langley Research Center, 2003.

MAVRIPLIS, D. J. Unstructured Mesh Discretizations and Solvers for Computational Aerodynamics. In: AIAA COMPUTATIONAL FLUID DYNAMICS CONFERENCE, 2007, Miami, FL, USA. **Proceedings...** [S.l.: s.n.], 2007.

MAX, N. Optical models for direct volume rendering. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.1, n.2, p.99–108, June 1995.

MORELAND, K.; ANGEL, E. A Fast High Accuracy Volume Renderer for Unstructured Data. In: IEEE SYMPOSIUM ON VOLUME VISUALIZATION AND GRAPHICS, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.9–16.

NOKIA CORPORATION. **Qt**. Disponível em: <http://www.qtsoftware.com/>. Acesso em: 2008.

PARENT, R. **Computer Animation - Algorithms and Techniques**. [S.l.]: Morgan Kaufmann, 2001.

PFISTER, H. et al. The Transfer Function Bake-Off. **IEEE Computer Graphics and Applications**, [S.l.], v.21, n.3, p.16–22, 2001.

POTTS, S.; MÖLLER, T. Transfer Functions on a Logarithmic Scale for Volume Rendering. In: GRAPHICS INTERFACE, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.57–63.

PURCELL, T. J. et al. Ray Tracing on Programmable Graphics Hardware. **ACM Transactions on Graphics**, [S.l.], v.21, n.3, p.703–712, July 2002.

ROETTGER, S.; BAUER, M.; STAMMINGER, M. Spatialized Transfer Functions. In: IEEE VGTC/EUROGRAPHICS SYMPOSIUM ON VISUALIZATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.271–278.

ROETTGER, S. et al. Smart Hardware-Accelerated Volume Rendering. In: EG/IEEE TCVG SYMPOSIUM ON VISUALIZATION, VISSYM, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.231–238.

SCHNEIDER, J.; WESTERMANN, R. Compression Domain Volume Rendering. In: IEEE VISUALIZATION, 2003. **Proceedings...** [S.l.: s.n.], 2003.

SCHULZE, J. P.; CHOURASIA, A. **A user interface for high dynamic range transfer function design**. In: ACM SIGGRAPH, 2006. Research Poster. New York: ACM, 2006.

SEREDA, P. et al. Visualization of boundaries in volumetric datasets using LH histograms. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.12, n.2, p.208–218, March/April 2006.

SERLIE, I. et al. Computed Cleansing for Virtual Colonoscopy Using a Three-Material Transition Model. In: INTERNATIONAL CONFERENCE OF MEDICAL IMAGE COMPUTER–ASSISTED INTERVENTION, MICCAI, 2003, Montreal, Canada. **Proceedings...** Berlim: Springer, 2003. p.175–183.

SHAMIR, A. Feature-Space Analysis of Unstructured Meshes. In: IEEE VISUALIZATION, 2003, Seatle, Washington. **Proceedings...** [S.l.]: IEEE Computer Society, 2003. p.25.

SHEN, H.-W.; CHIANG, L.-J.; MA, K.-L. A Fast Volume Rendering Algorithm for Time-Varying Field Using A Time-Space Partitioning (TSP) Tree. In: IEEE VISUALIZATION, 1999, San Francisco, CA. **Proceedings...** New York: IEEE, 1999. p.371–377.

SHEWCHUK, J. R. **What Is a Good Linear Finite Element? - Interpolation, Conditioning, Anisotropy, and Quality Measures**. Berkley, CA: Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2002.

SHEWCHUK, J. R. **Constrained Delaunay Tetrahedralizations, Bistellar Flips, and Provably Good Boundary Recovery**. Talk slides. Available from author's web page at: <http://www.cs.berkeley.edu/~jrs/papers/>. Visited on: 2008.

SHIRLEY, P.; TUCHMAN, A. A Polygonal Approximation to Direct Scalar Volume Rendering. **Computer Graphics**, New York, v.24, n.5, p.63–70, Nov. 1990.

SILVA, C.; MITCHELL, J. The Lazy Sweep Ray Casting Algorithm for Rendering Irregular Grids. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.3, n.2, p.142–157, 1997.

SILVA, C. T. et al. A Survey of GPU-Based Volume Rendering of Unstructured Grids. **Brazilian Journal of Theoretic and Applied Computing (RITA)**, [S.l.], v.12, n.2, p.9–29, 2005.

SILVER, D.; WANG, X. Tracking scalar features in unstructured datasets. In: CONFERENCE ON VISUALIZATION, VIS, 1998. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 1998. p.79–86.

STEIN, C. M.; BECKER, B. G.; MAX, N. L. Sorting and hardware assisted rendering for volume visualization. In: SYMPOSIUM ON VOLUME VISUALIZATION, 1994. **Proceedings...** New York: ACM, 1994. p.83–89.

SVAKHINE, N.; EBERT, D. S.; STREDNEY, D. Illustration Motifs for Effective Medical Volume Illustration. **IEEE Computer Graphics and Applications**, [S.l.], v.25, n.3, p.31–39, 2005.

TENGINAKAI, S.; LEE, J.; MACHIRAJU, R. Salient iso-surface detection with model-independent statistical signatures. In: CONFERENCE ON VISUALIZATION, 2001. **Proceedings...** Washington, DC: IEEE Computer Society, 2001. p.231–238.

TORY, M.; MÖLLER, T. Evaluating Visualizations: do expert reviews work? **IEEE Computer Graphics and Applications**, [S.l.], v.25, n.5, p.8–11, 2005.

TORY, M.; POTTS, S.; MÖLLER, T. A Parallel Coordinates Style Interface for Exploratory Volume Visualization. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.11, n.1, p.71–80, 2005.

TZENG, F.-Y.; LUM, E. B.; MA, K.-L. An Intelligent System Approach to Higher-Dimensional Classification of Volume Data. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.11, n.3, p.273–284, 2005.

VIOLA, I.; KANITSAR, A.; GRÖLLER, E. Importance-Driven Volume Rendering. In: IEEE VISUALIZATION, 2004. **...** [S.l.: s.n.], 2004. p.139–145.

WALTER, J. D.; HEALEY, C. G. Attribute preserving dataset simplification. In: CONFERENCE ON VISUALIZATION, 2001. **Proceedings...** Washington, DC: IEEE Computer Society, 2001. p.113–120.

WEILER, M. et al. Hardware-Based Ray Casting for Tetrahedral Meshes. In: IEEE VISUALIZATION, 2003. **Proceedings...** [S.l.: s.n.], 2003. p.333–340.

WEILER, M. et al. Texture-Encoded Tetrahedral Strips. In: IEEE SYMPOSIUM ON VOLUME VISUALIZATION AND GRAPHICS, 2004. **Proceedings...** Washington, DC: IEEE Computer Society, 2004. p.71–78.

WESTERMANN, R. Compression Domain Rendering of Time-Resolved Volume Data. In: IEEE VISUALIZATION, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.168–174.

WESTOVER, L. Footprint evaluation for volume rendering. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, SIGGRAPH, 1990. **Proceedings...** New York: ACM, 1990. p.367–376.

WHITTED, T. An improved illumination model for shaded display. **Commun. ACM**, New York, NY, USA, v.23, n.6, p.343–349, 1980.

WILLIAMS, P. L. Visibility-Ordering Meshed Polyhedra. **ACM Transactions on Graphics**, [S.l.], v.11, n.2, p.103–126, Apr. 1992.

WU, Y. et al. Fusing Features in Direct Volume Rendering Images. In: INTERNATIONAL SYMPOSIUM ON VISUAL COMPUTING, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.273–282.

WU, Y. et al. Focus + Context Visualization with Animations. In: IEEE PACIFIC-RIM SYMPOSIUM ON IMAGE AND VIDEO TECHNOLOGY, 2006. **Proceedings...** [S.l.: s.n.], 2006. p.1293–1302.

YU, H.; MA, K.-L.; WELLING, J. I/O Strategies for Parallel Rendering of Large Time-Varying Volume Data. In: EUROGRAPHICS SYMPOSIUM ON PARALLEL GRAPHICS AND VISUALIZATION, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.31–40.

YUAN, X. et al. High dynamic range volume visualization. In: IEEE VISUALIZATION, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.327–334.

YUAN, X. et al. HDR VolVis: high dynamic range volume visualization. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.12, n.4, p.433–455, July/August 2006.

APPENDIX A THE TRANSFER FUNCTION DESIGN PROGRAM

The techniques described in chapter 3 have been integrated into a larger system developed to help users on designing transfer functions for unstructured grids. Later in the development, the support for structured grids was integrated and it is functional for the features described in the section 3. This system was not solely developed by the main author of this document, so the features described in this appendix were left out of the main text.

This program was developed to become a Transfer Functions Design Engine, so its modules have been projected to be independent of the user interfar. To this end, the system has a core module that retains all the functionality of the program, an interface module that must be implemented along the user-interface code and a user-interface module written in QT (TROLLTECH, 2008) and presented in figure A.1.

The following section explains the concept of range-mapping, developed by another researcher for this visualization system.

A.1 Scalar Range Mapping

Volume data produced from scientific simulation typically contains a high dynamic range (HDR) of floating point scalar values. In addition, a high percentage of the scalar values are often contained in a small range of the histogram (see Figure A.2(a)). Consequently, to expose details that may be contained in these small regions, a large number of control points and a high resolution lookup table are required. There are two main issues with traditional transfer function design when dealing with HDR data. First, the narrow range of values makes specification difficult due to the low resolution of the features on the histogram interface. Second, the limited resolution of the color and opacity lookup table in graphics hardware is not sufficient to fully represent all the unique scalar values in the data.

To overcome the resolution limitations of the histogram interface, tools like ParaView (KITWARE, 2008) incorporate user-controlled zooming widgets to assist with transfer function specification over small regions of the data. Yuan *et al.* (YUAN et al., 2006) recently introduced a 1D fish-eye visualization of the histogram based on a *focus and context* concept, which allows simultaneous representation of global (context) and detail (focus) information on the same histogram display. These approaches, based on magnifying the range of interest in the user interface, greatly assist the user with HDR transfer function design. However, the second issue is still a challenge. Ideally, the number of entries in the color and opacity lookup table should correspond to the number of

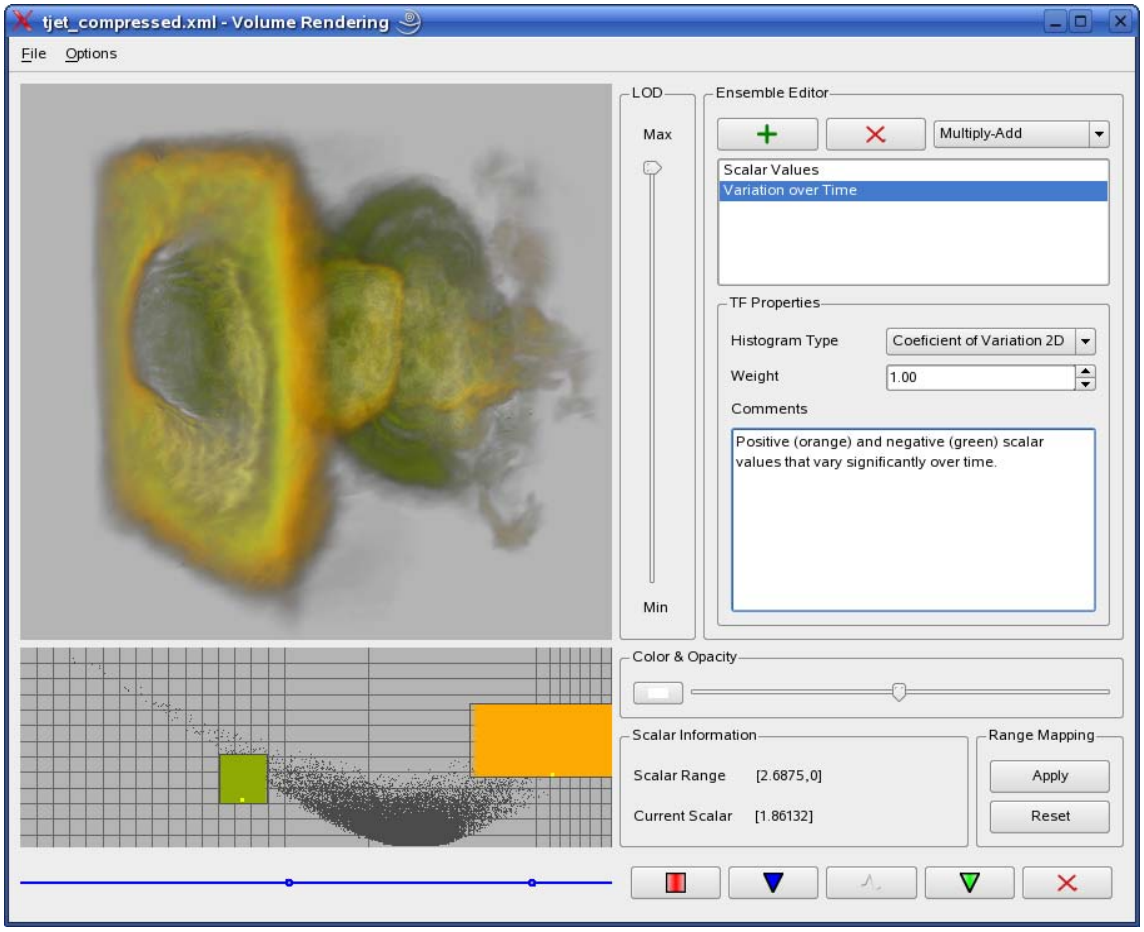


Figure A.1: The user interface for interactive transfer function specification is shown for the time-varying Turbulent Jet dataset using a 2D time histogram.

unique scalar values in the volume. Yuan *et al.* (YUAN et al., 2006) leverage tone mapping and specialized high-precision graphics hardware to handle the high precision of texture based volume rendering. With limits in texture size, this is not always sufficient and may result in many scalar values being assigned to one entry in the table (see Figure A.2(b)). Instead, this work proposes *range mapping*, which redistributes the scalar range non-linearly to spread the regions of interest more evenly across the lookup table (see Figure A.2(c)). Range mapping is related to histogram equalization and is a common approach in image processing for handling low contrast images. This feature facilitates the design process by allowing focus and context zooming effects, while avoiding resolution issues of a fixed-size lookup table. The result is a tool that is naturally capable of extracting detailed features in the data, as shown in Figure A.3.

Based on the observation that the transfer function design difficulties of HDR data are mainly due to the non-uniform distribution of scalar data, the proposed solution is to redistribute the scalar range. This can be done automatically by performing histogram equalization, which spreads out the clustered regions. Mathematically, histogram equalization is performed by introducing a cumulative density function (CDF) as a sum of probability density functions (PDFs) over normalized scalar inputs:

$$CDF(x_i) = \sum_{x_j < x_i} PDF(x_j).$$

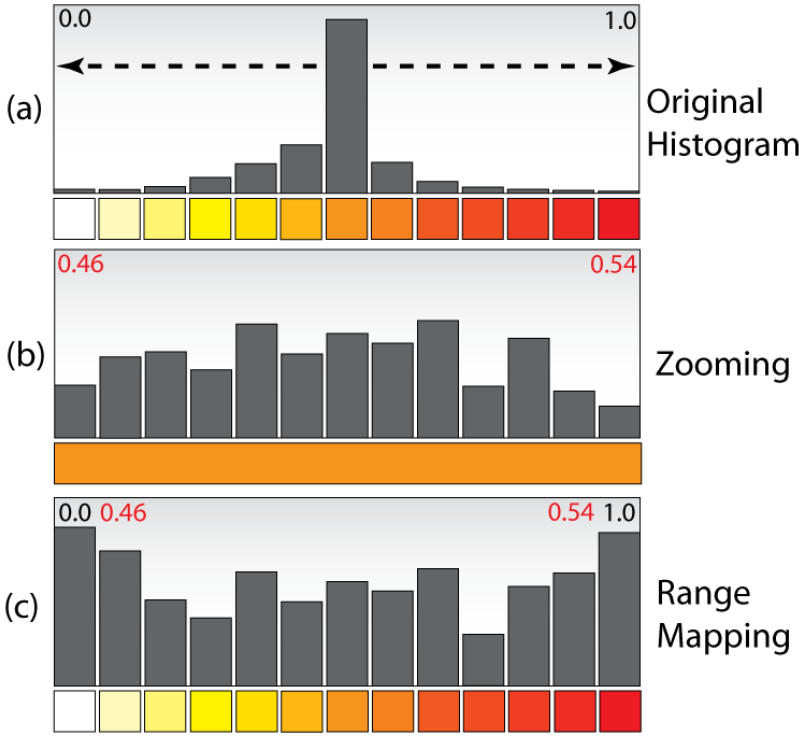


Figure A.2: (a) A high-dynamic range histogram (above) is shown with a corresponding lookup table (below). (b) Zooming into the dense region of the histogram does not change the resulting image due to the static resolution of the lookup table. (c) By range mapping the scalar values, the high-dynamic range elements of the mesh can be spread more evenly across the static lookup table, enhancing hidden features in the data.

Then, a simple mapping is performed on the normalized scalar input value x that yields a new uniformly distributed normalized output y:

$$y = CDF(x).$$

Due to its speed and simplicity, it is common to use a discrete histogram equalization and perform this mapping with a lookup table. This approach is automatic, but gives the user very little control over the redistribution process and tends to break the continuities of the scalar range. Range mapping, a generalization of histogram equalization, is based on piecewise linear mapping functions and provides more control while maintaining the continuity of the scalar range. The range mapping functions that map the input scalars $[x_0 \dots x_n]$ to a new scalar range $[y_0 \dots y_n]$ are a class of piecewise continuous functions f over the input range that satisfy the following conditions: f is a monotonically increasing function, $f(x_0) = y_0$, and $f(x_n) = y_n$. Similar to histogram equalization, the new scalar value y is computed as:

$$y = f(x).$$

Given this definition, the function can arbitrarily redistribute the scalar range while maintaining the order and continuity.

In practice, this work use the linear range mapping functions that combine many line segments, each of which performs mapping from a specific range $[x_i \dots x_{i+1}]$ to $[y_i \dots y_{i+1}]$ by applying the linear mapping equation:

$$y = \frac{x - x_i}{x_{i+1} - x_i}(y_{i+1} - y_i) + y_i.$$

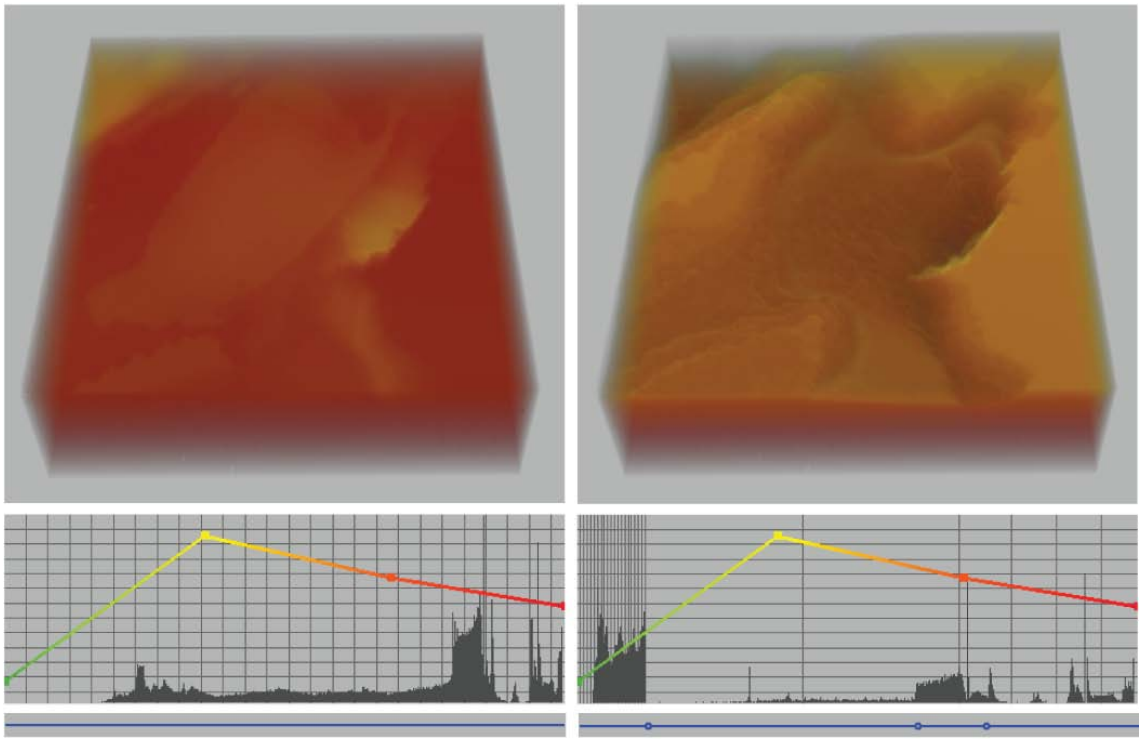


Figure A.3: Volume exploration on the San Fernando earthquake simulation through range mapping. A predefined transfer function (left) is used to explore the data by remapping the scalars (right). Only a non-linear remapping can enhance features that are hidden in multiple spikes of the data.

These linear function are sufficient to represent all range mappings since any function can be approximated using many piecewise linear functions.

Because the cost of the linear interpolation is relatively low, one can perform range mapping interactively while the user is manipulating control points for the range. The remapping process is performed in hardware by storing a 1D texture that contains one entry for every control point of the remapping. When the mapping changes, to minimize CPU to GPU transfer, only the new mapping texture is sent to the GPU. During rendering, the normalized scalar values can then be remapped to normalized scalar values using a single texture lookup with linear interpolation enabled. Thus, this extra remapping step impacts the rendering performance very little and is flexible enough to be used in a variety of volume rendering algorithms.

As illustrated in Figure A.2, range mapping yields a magnification effect that is different from a normal zooming effect, since the actual shape of the histogram changes non-linearly. This helps the user exploit the real data distribution in narrow clusters of the scalar range. Even without transfer function widgets, range mapping can be a powerful exploration tool. Figure A.3 shows how range mapping can be used to explore the data using a simple, pre-defined transfer function.

Creating a user interface that can fully exploit the power of range mapping is a challenge.

The proposed solution is a simple, intuitive interface that allows the user to choose the range by adding control points and extend the range by dragging two control points away from each other. This allows the user to continue adding control points between the previous points to further probe important regions. The histogram and volume rendering

change interactively during this control point manipulation to provide visual feedback of the remapping. In addition, the range and scalar value under the cursor are displayed to the user to facilitate specification when there is *a priori* knowledge of the data. Figures A.1 and A.3 show snapshots of this interface.

A.2 Evaluation

An important consideration for a transfer function specification tool is that it does not introduce additional computational overhead and thus adversely impact interactivity. There is no measurable performance penalty when using the transfer function specification tools. Thus, the interactivity of the rendering remains the same as the original volume renderer.

To evaluate the usefulness of the proposed techniques, an informal expert evaluation of the system has been performed. Expert reviews have been shown to be a useful means of evaluation, and require fewer reviewers than standard user studies (TORY; MÖLLER, 2005). Comments and suggestions were collected from four experts with different relevant backgrounds. The first expert develops open source visualization software. The second performs research in the area of volume visualization. The third expert has used existing volume visualization software in a clinical setting. Finally, the fourth expert is a specialist in bioengineering and concentrates mostly on biomedical computing. These experts were given a demonstration of the proposed system along with ParaView (KITWARE, 2008), a freely available system that has some basic transfer function specification abilities such as a zooming interface. The experts were then given the opportunity to perform their own explorations using both systems and asked a series of questions about their experience with the system compared to ParaView and other systems that they have used in the past.

Overall, the feedback was very positive and the reviewers feel that the proposed system is useful for quick data exploration and that existing visualization systems would benefit from some of the components introduced in the system. The reviewers also provided many suggestions that can be incorporated into the system. This work summarizes some of the main advantages and disadvantages that the reviewers pointed out.

Advantages:

- The system provides fine-grain control of the data due to the resolution control that range mapping provides.
- Interactive histogram information significantly improves the ability to place widgets and to explore the volume.
- The histograms that update over time provide more information about the volume than other systems provide.
- The ability to interpolate between transfer functions over time is very useful for contextualizing the data.

Disadvantages:

- The interface could use some work to consolidate the concept of ensembles and make it more intuitive.

- Though some of the features are more powerful, they may require longer to learn to use if the user is unfamiliar with transfer function specification.
- Currently there is no undo for operations.

Beyond these general comments, some interesting comments about the usefulness of the system from the reviewers unique perspectives was received. The first reviewer mentioned that when the application he develops moved from 8-bit data to higher precision, he noticed problems associated with limits in the lookup table precision, though he had not found a reasonable solution for the problem. After the evaluation, his plan is now to add range mapping to his system. He also really liked the idea of creating transfer functions using combinations of other transfer functions and is evaluating this addition to his system as well. The third reviewer worked extensively with time-varying data that changes substantially over time. He commented that the ability to define different transfer functions for time steps and interpolate between them through keyframing would have saved him enormous amounts of time. He also mentioned that the ability to keyframe the range mapping would be a useful feature for these datasets. This feature is planned to be added to the system as soon as possible. The fourth reviewer stated that he would like to see the features presented in this system incorporated into the biomedical simulation software that he and his collaborators use because it would facilitate the process of analysis for time-varying volumes.

APPENDIX B ESTATÍSTICAS DE FUNÇÕES APLICADAS A VISUALIZAÇÃO VOLUMÉTRICA: CUIDADOS ESPECIAIS EM FUNÇÕES DISCRETAS

Cada vez mais dados são utilizados por cientistas para estudar os mais diversos problemas. Simulações físicas são utilizadas por engenheiros para projetar novas soluções, assim como imagens de ressonância magnética (MRI), entre outras técnicas, são utilizadas por médicos para avaliar as condições de saúde de pacientes. Uma importante ferramenta que acelera a compreensão desses tipos de dados é a visualização das informações.

Por tratarem-se de dados em 3D, a visualização desse tipo de dado é complexa e depende diretamente do formato dos dados. A seção B.1 explica quais tipos de dados existem e métodos que podem ser utilizados para visualizá-los.

O primeiro conjunto de contribuições deste trabalho é apresentado na seção B.2. O sistema apresentado introduz o conceito de Agrupamento de funções, que são combinações de funções de transferência relativamente simples para a produção de novos mapeamentos mais complexos. Além disso, uma solução simples porém robusta para tratar dados dinâmicos é descrita.

O segundo grupo de contribuições é, na verdade, um amplo estudo sobre a extração de informações em conjuntos de dados volumétricos, apresentados na seção B.3. Técnicas antes apenas utilizadas em uma das classes de dados são estendidas para funcionarem com ambas. Além disso, a aparição de determinados padrões é estudada e sua origem explicada.

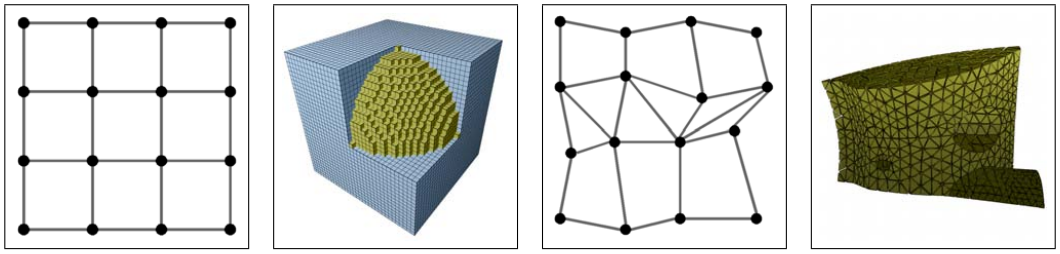
O estudo realizado na seção B.3 possibilitou o desenvolvimento de aplicações descritas na seção B.4. Outras possíveis aplicações são descritas na seção B.5, e a conclusão do trabalho é apresentada na seção B.6.

B.1 Visualização Volumétrica

Visualização volumétrica é o nome dado ao tipo de visualização realizada em conjuntos de dados que possuem informações sobre seu interior, e objetivam justamente realçar essas características. Para isso, técnicas especiais são utilizadas para desenhar esses objetos, e funções de transferência são utilizadas para mapear as diferentes propriedades desses dados para cor e opacidade. Funções de transferência serão descritas na seção B.2.

Existem basicamente duas classes de dados volumétricos: malhas estruturadas e malhas não-estruturadas. A principal diferença entre esses tipos de dados é a forma como os seus componentes estão organizados. Malhas estruturadas (ou regulares) possuem um padrão regular na distribuição das suas amostras, e não precisam de informações sobre a conectividade destes elementos. Volumes não-estruturados ou irregulares, por outro

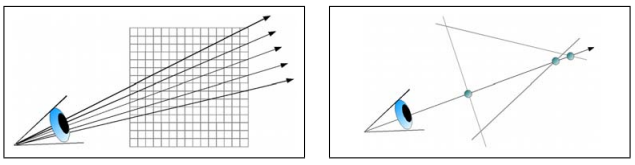
lado, apresentam amostras com diferentes distribuições espaciais, possuindo regiões com maiores e menores concentrações de dados. Por ter essa distribuição irregular, dados não-estruturados precisam armazenar as informações a respeito de sua topologia (conexões entre seus elementos). A figura B.1 apresenta exemplos destes tipos de dados.



(a) (b) (c) (d)

Figure B.1: Exemplo de dados estruturados em (a) e (b), e dados não-estruturados em (c) e (d).

Diferentes métodos foram desenvolvidos para a visualização destes dados. A técnica conhecida por *Ray Casting* (LEVOY, 1988, 1989) pode ser utilizada em ambos os tipos de dados. Seu funcionamento consiste em emitir raios partindo do centro de uma câmera virtual através dos *pixeis* da imagem, e calcular a intersecção desses raios com o conjunto de dados, acumulando de forma seqüencial as contribuições de cor e opacidade dos elementos interseccionados. Figura B.2 mostra o princípio de funcionamento dessa técnica.



(a) Volume regular (b) Volume irregular

Figure B.2: Exemplo do método de *Ray Casting* aplicado para dados estruturados em (a) e dados não-estruturados em (b).

A técnica de projeção de células (SHIRLEY; TUCHMAN, 1990) foi desenvolvida para lidar exclusivamente com dados não-estruturados. Neste método, as amostras estão organizadas na forma de células, geralmente tetraedros. Estas células são subdivididas em triângulos, ordenadas do ponto de vista da câmera virtual, e projetadas ordenadamente no plano de imagem.

Uma técnica amplamente utilizada para visualizar volumes regulares é a Visualização Volumétrica Direta Baseada em Texturas (*Texture-Based Direct Volume Rendering*). Este método usualmente cria planos de amostragem paralelos ao plano de projeção, e estes são utilizados para amostrar o volume de dados que é armazenado como uma textura (DREBIN; CARPENTER; HANRAHAN, Avg. 1988). Figura B.3 ilustra essa técnica.

Estes métodos de visualização utilizam funções de transferência para mapear propriedades dos dados para cor e opacidade. Estas funções são explicadas a seguir.

B.2 Funções de Transferência

Os dados utilizados em visualização volumétrica são formados por algum tipo específico de medida (como a densidade dos materiais). Para visualizá-los, é necessário utilizar

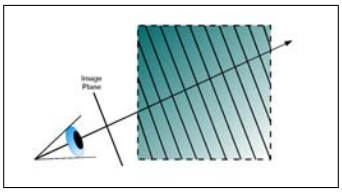


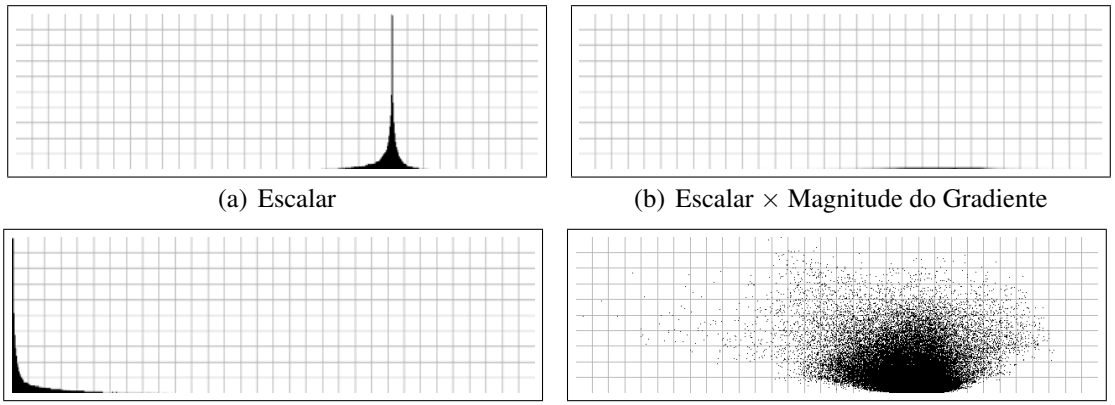
Figure B.3: Em visualização volumétrica direta, a prática mais comumente utilizada é desenhar planos alinhados ao plano de projeção para amostrar o volume regular.

um mapeamento do espaço do volume para um espaço de cor e opacidade. Esse é o objetivo das funções de transferência.

Diversos métodos foram e ainda são desenvolvidos para auxiliar usuários na tarefa de criar funções de transferência (PFISTER et al., 2001). Este trabalho advoga o uso de *widgtes* desenhados sobre histogramas (seção B.2.1) para definir funções de transferência básicas que podem ser combinadas em Agrupamentos (seção B.2.2). Agrupamentos (ou funções simples) podem ser utilizadas para criar visualizações de dados dinâmicos, como apresentado na seção B.2.3.

B.2.1 Histogramas

Histogramas podem revelar diferentes informações sobre os dados nos quais eles foram computados. Este trabalho apresenta quatro diferentes histogramas usados para guiar os usuários quando estes definem as funções de transferência. A figura B.4 exemplifica estes histogramas.



(a) Escalar (b) Escalar × Magnitude do Gradiente

(c) Coeficiente de Variação (d) Escalar × Coeficiente de Variação

Figure B.4: Diferentes histogramas foram utilizados para guiar as escolhas dos usuários quando estes definem funções de transferência: o histograma dos valores escalares em (a), o escalar × magnitude do gradiente em (b), o coeficiente de variação para dados dinâmicos em (c) e o escalar × coeficiente de variação em (d).

O histograma de valores escalares mostra como estes estão distribuídos no volume, permitindo identificar uma possível concentração de amostrar ao redor de algum valor.

O histograma da magnitude do gradiente é amplamente utilizado para identificar transições entre materiais relativamente homogêneos. Nesse caso, uma forma de arco é claramente identificada no histograma. O conjunto de dados TJet não apresentou essa característica, contrário ao esperado. Esse comportamento será avaliado na seção B.3.

O coeficiente de variação é uma medida útil para verificar se a quantidade de valores dinâmicos no volume é significante quando comparada a totalidade total de valores.

Essa medida já foi utilizada em outros trabalhos relacionados a funções de transferência (JANKUN-KELLY; MA, 2001). O primeiro histograma a utilizá-la é útil para salientar amostras com determinada variação temporal independentemente do seu valor escalar associado. Por outro lado, o histograma de escalares $\times$ coeficiente de variação é útil para salientar diferentes características dinâmicas utilizando diferentes característica ópticas.

Funções de transferência definidas utilizando esses histogramas podem ser combinadas para produzir mapeamentos mais complexos. Esta técnica é explicada a seguir.

B.2.2 Agrupamentos de Funções de Transferência

O conceito de agrupamentos em funções de transferência é produzir novos mapeamentos a partir de funções já existentes. Outros trabalhos existentes já propuseram a combinação de diferentes funções (MA, 1999; WU et al., 2006a). Todavia esses trabalhos não demonstram completamente o seu funcionamento ou o aplicam a casos bastante específicos. Neste trabalho são explicadas diferentes formas para combinar as funções de transferência, além se serem sugeridos exemplos de aplicações.

Para combinar diferentes funções, elas devem estar definidas sobre um mesmo domínio. Duas situações derivam dessa situação. Primeiro, para produzir uma função cuja dimensão é a máxima entre as dimensões das duas funções, um processo de extrusão pode ser aplicado para igualar as duas dimensões antes da combinação. Já no segundo caso, onde o resultado deve possuir a mesma dimensão da função com menor dimensionalidade, um processo de redução dimensional deve ser aplicado (FODOR, 2002). Neste trabalho uma simples redução dimensional foi utilizada, onde a cor e opacidade para uma dada região da função de transferência é modulada de acordo com a quantidade de amostras que são projetadas na mesma região da função final. A figura B.5 demostra esse conceito.

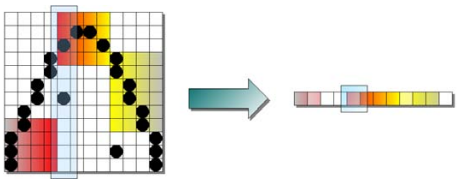


Figure B.5: Redução dimensional utilizada para converter uma função de dimensão mais alta para uma mais baixa. O exemplo acima mostra a redução de um mapeamento 2D para um mapeamento 1D.

Depois de levar as funções para um espaço em comum pode-se realizar a combinação delas. Para isso, três diferentes estratégias de combinação foram desenvolvidas:

1. **ADD**: combina as características realçadas pelas duas funções, segundo a fórmula

$$\begin{aligned} C_r(i) &= C_1(i) + C_2(i) \\ \alpha_r(i) &= \alpha_1(i) + \alpha_2(i) \end{aligned}$$

   Este modo é útil para realçar características destacadas pelas duas funções.

2. **AND**: este modo destaca características que duas funções tem em comum:

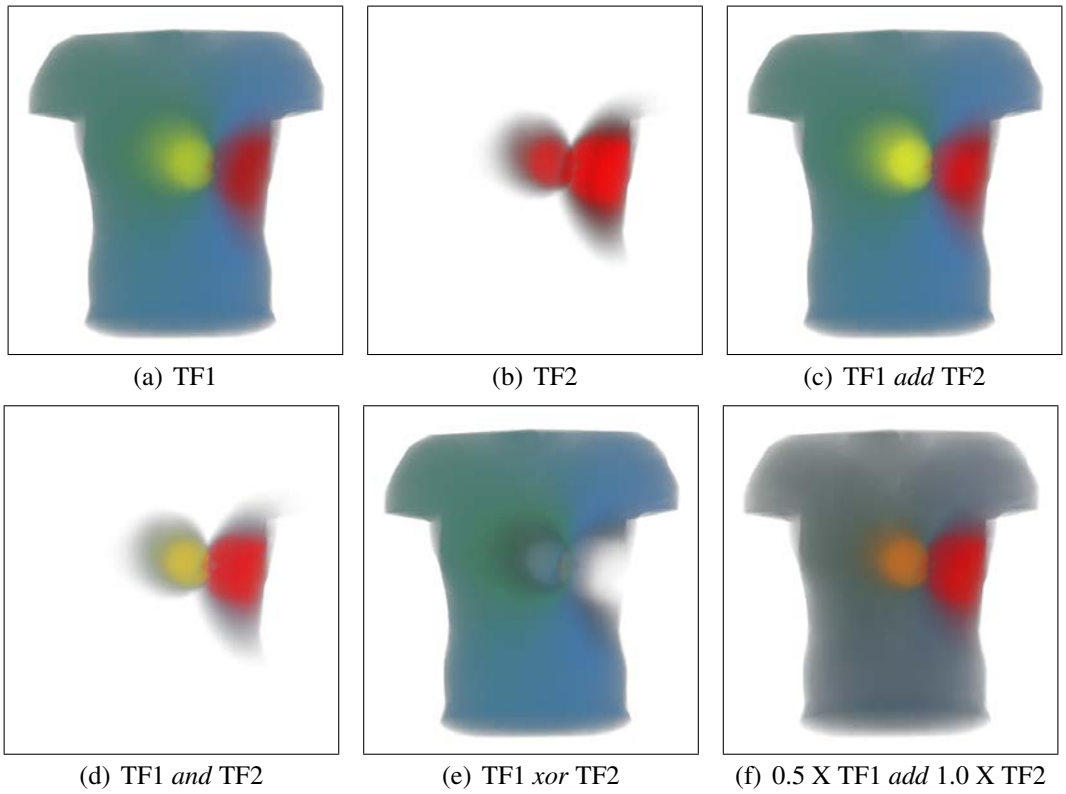
$$\begin{aligned} C_r(i) &= Max(C_1(i), C_2(i)) \\ \alpha_r(i) &= Min(\alpha_1(i), \alpha_2(i)) \end{aligned}$$

3. **XOR**: este modo é complementar ao anterior:

$$C_r(i) = (C_1(i) \wedge \overline{C_2(i)}) \vee (\overline{C_1(i)} \wedge C_2(i))$$
$$\alpha_r(i) = (\alpha_1(i) \wedge \overline{\alpha_2(i)}) \vee (\overline{\alpha_1(i)} \wedge \alpha_2(i))$$

Este modo remove as características que ambas as funções apresentam em comum.

Aplicações para os agrupamentos de funções permitem uma rápida manipulação de dados através da combinação de funções específicas, como demonstra a figura B.6. No caso de um conjunto de dados médicos, por exemplo, funções de realce para órgãos individuais podem ser combinadas para permitir a visualização de diversos órgãos simultaneamente.



(a) TF1 (b) TF2 (c) TF1 *add* TF2

(d) TF1 *and* TF2 (e) TF1 *xor* TF2 (f) 0.5 X TF1 *add* 1.0 X TF2

Figure B.6: Agrupamentos das funções de transferência (a) e (b) utilizando diversas estratégias de combinação: adição simples (*ADD*) em (c), operação *AND* em (d) e operação *XOR* em (e). O efeito de utilizar pesos diferenciados para cada função é mostrado em (f).

Agrupamentos são bastante úteis para desenvolver funções complexas partindo de funções mais simples. Eles também podem ser utilizados com dados temporais como mostra a próxima seção.

B.2.3 Dados Dinâmicos

Os métodos anteriormente descritos para a criação de funções de transferência foram desenvolvidos pensando primariamente em dados temporalmente estáticos. Dados com variação temporal tem recebido uma menor atenção dos pesquisadores.

Dados dinâmicos podem ser classificados em dois diferentes grupos (AKIBA; FOUT; MA, 2006):

- estatisticamente estáticos: possuem histogramas persistentes ao longo do tempo, com pouca variação, e uma mesma função de transferência pode ser utilizada para todas as instâncias de tempo.

- estatisticamente dinâmicos: este tipo de dado apresenta bastante variação temporal. Neste caso, mais de uma função de transferência é necessária para capturar as características do volume.

Poucos trabalhos anteriores apresentam métodos para lidar com ambas as classes de dados (SILVER; WANG, 1998; JANKUN-KELLY; MA, 2001). Em ambos os trabalhos, porém, diferentes funções de transferência devem ser utilizadas para tratar dados estatisticamente dinâmicos, mas utilizam uma simples troca dos mapeamentos utilizados. Esta quebra de continuidade visual pode desorientar o usuário.

Este trabalho advoga o uso de um sistema de *key-frames* para possibilitar que o usuário controle quantas funções de transferência serão utilizadas e como a troca entre elas será realizada, através de uma curva de interpolação customizável. O programa desenvolvido apresenta uma interface que possibilita ao usuário escolher quais funções correspondem a quadros da animação temporal, e como a mudança entre esses quadros deve ser feita. A figura B.7 mostra essa interface.

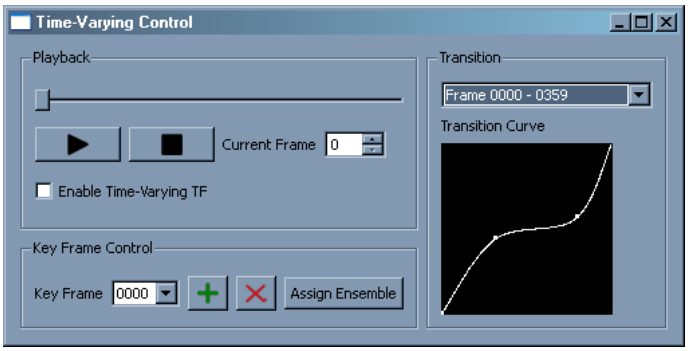


Figure B.7: Usuários podem customizar as funções de transferência que serão utilizadas em em volume dinâmico através da interface acima. Usuários podem associar funções com quadros-chave na animação - definidos pelo próprio usuário - e como fazer a interpolação entre esses quadros, usando a curva de transição (*transition curve*) apresentada na figura.

As funções utilizadas nos quadros-chave podem ser agrupamentos ou funções simples, desenvolvidas com qualquer método. Como explicado anteriormente, o sistema oferece suporte a funções definidas sobre histogramas. Como alguns dos histogramas não apresentaram as formas esperadas, um estudo sobre quais fatores influenciam a computação desses histogramas foi realizado, como mostra a próxima seção.

B.3 Análise da Assinatura de Dados

Assinatura de dados no contexto deste trabalho é a informação que pode ser computada a partir do conjunto de dados inicial. Um exemplo deste tipo de informação é o gradiente da função.

O gradiente de um campo escalar é um vetor que indica a direção de maior variação dentro do campo escalar. Para um campo escalar $f(x, y, z)$ no espaço euclidiano se gradiente pode ser definido como

$$\nabla f(x,y,z) = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right) \tag{B.1}$$

Ao aplicar o gradiente em um ponto p do conjunto de dados, o resultado será a derivada direcional do campo escalar em p. A magnitude do vetor gradiente representa a velocidade de variação do campo naquela direção.

A segunda derivada do campo escalar é o Laplaciano

$$\nabla^2 f(x,y,z) = \left(\frac{\partial^2}{\partial x^2}, \frac{\partial^2}{\partial y^2}, \frac{\partial^2}{\partial z^2}\right) \tag{B.2}$$

e também é bastante útil para a definição de borda utilizada neste trabalho.

Para avaliar corretamente as possíveis causas da baixa qualidade de alguns hitogramas é preciso isolar as diversas fontes de erro, como a amostragem da função e a aproximação dos resultados. Um conjunto de dados com soluções conhecidas foi desenvolvido para permitir a exploração de diferentes fatores individualmente. A figura B.8 ilustra o volume utilizado.

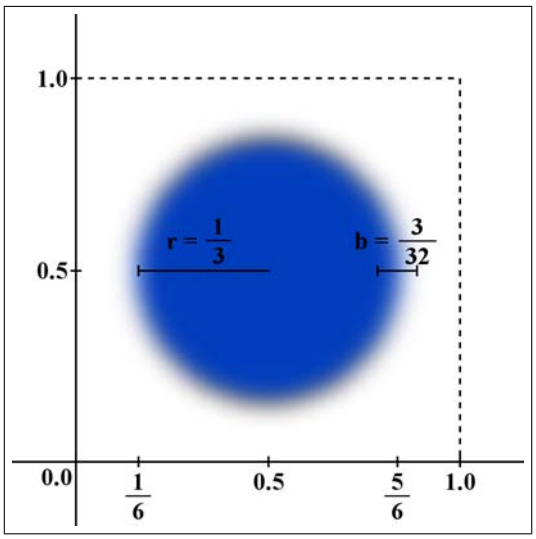


Figure B.8: Representação 2D do volume de dados utilizado como estudo de caso neste trabalho.

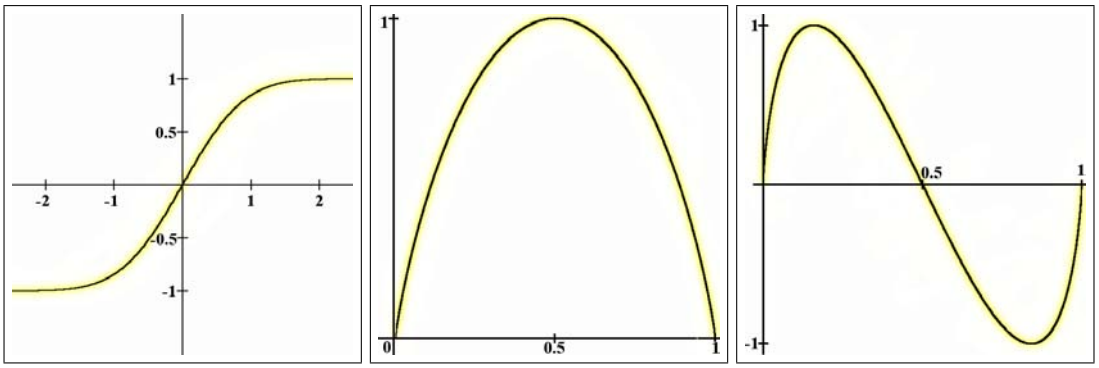
Como funções de transferência são bastante utilizadas para realçar bordas entre regiões relativamente homogêneas (KINDLMANN; DURKIN, 1998; PFISTER et al., 2001), é importante definir o formato da borda. Seguindo a linha apresentada por Kindlmann (KINDLMANN; DURKIN, 1998), uma borda ideal seria uma *step function*, uma função com uma transição praticamente instantânea de um valor relacionado a uma região homogênea para outra. Em situações reais, todavia, isso é muito difícil de ocorrer, devido em grande parte a imprecisões de equipamentos de medição. Essas imprecisões em geral acarretam em uma suavização da borda da função. Essa suavização pode ser simulado pela aplicação de um filtro gaussiano na borda, resultando na *error function*

$$erf(x) = \frac{2}{\sqrt{\pi}} \times \int_0^x \exp^{-t^2} dt \tag{B.3}$$

utilizada como modelo de borda ideal. Essa função, bem como suas primeira (equação B.4) e segunda (equação B.5) derivadas podem ser vistas na figura B.9.

$$erf'(x) = \frac{2}{\sqrt{\pi}} \times \exp^{-x^2} \tag{B.4}$$

$$erf''(x) = \frac{4}{\sqrt{\pi}} \times (-x) \times \exp^{-x^2} \tag{B.5}$$



(a) $f(x)$ (b) $f(x) \times f\prime(x)$ (c) $f(x) \times f\prime\prime(x)$

Figure B.9: A *error function* ($f(x)$) é utilizada como modelo ideal de borda, e mostrada em (a). Seu relacionamento com a magnitude de sua primeira ($f'(x)$) e segunda ($f''(x)$) derivadas direcionais é apresentado em (b) e (c), respectivamente (os gráficos em (b) e (c) estão normalizados).

Como as soluções de sua primeira e segunda derivadas são conhecidas, pode-se isolar problemas decorrentes de amostragens mal distribuídas de problemas relacionados a aproximação da solução real, como explicado nas próximas seções.

B.3.1 Amostragem

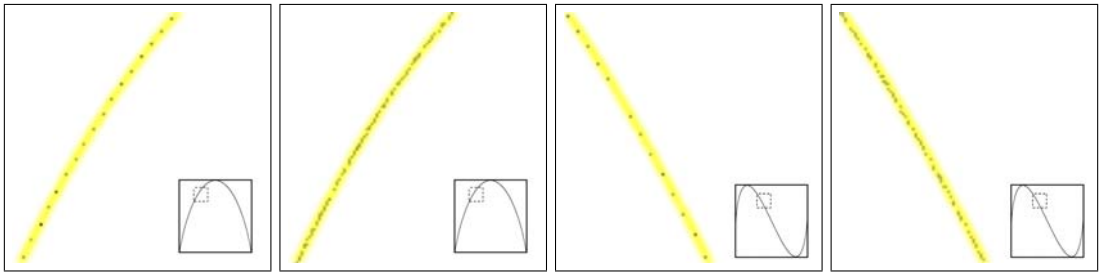
Para verificar como a amostragem afeta a computação dos histogramas a solução analítica de $f\prime(x)$ e $f\prime\prime(x)$ é utilizada sempre que esses valores devem ser consultados. A qualidade dos histogramas é diretamente afetada pela quantidade de amostras utilizadas e pela distribuição destas no domínio da função.

O método de *Quasi-Monte Carlo* (L'ECUYER, 2003) é bastante utilizado para acelerar a convergência em certos métodos computacionais. A distribuição regular de amostras em volumes estruturados é uma forma de Quasi-Monte Carlo. Trabalhos anteriores se beneficiaram deste fato para produzir histogramas. A figura B.10 mostra como diferentes estratégias de amostragem afetam a qualidade dos histogramas.

Analisando os histogramas da figura B.10 percebe-se que um determinado passo de amostragem afeta diretamente a qualidade do histograma. O conjunto de dados com 128^3 amostras mostra resultados nitidamente inferiores em termos de distribuição que o volume de 512^3. Nota-se particularmente a presença de diversas descontinuidades largas quando as amostras estão mais distribuídas.

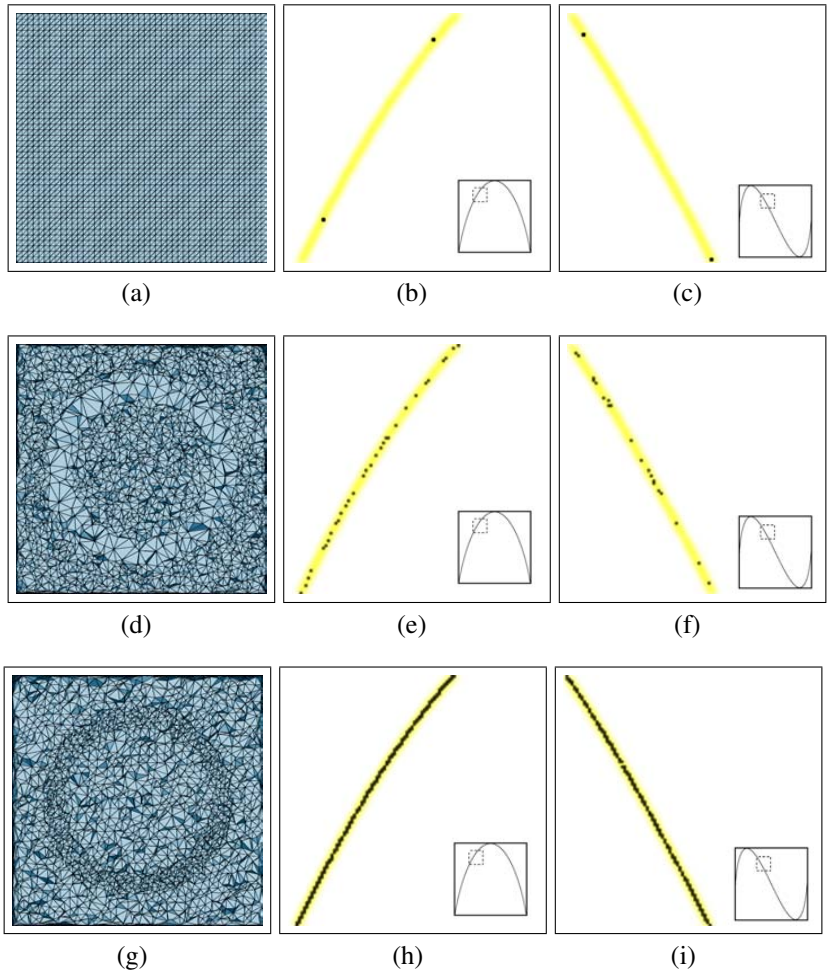
Volumes não-estruturados, por outro lado, geralmente apresentam uma menor quantidade de amostras. Para o estudo a seguir, malhas com aproximadamente 100.000 vértices foram utilizadas (46^3 no caso da distribuição regular). A figura B.11 mostra o comportamento de 3 diferentes estratégias de amostragem para volumes não-estruturados.

Amostras distribuídas em um padrão regular apresentam os mesmos artefatos que volumes regulares. Uma concentração maior de amostras em regiões homogêneas resulta em histogramas aparentemente mais povoados, porém ainda apresenta descontinuidades.



(a) (b) (c) (d)

Figure B.10: Gradient-magnitude of the blurred sphere dataset. Histograms were created using approximately 10^5 samples inside regular grids with resolutions of 128^3 (a, c) and 512^3 (b, d).



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure B.11: Diferentes estratégias empregadas para amostrar o volume da esfera utilizando malhas não-estruturadas. Em (a), (b) e (c) é utilizado um padrão similar aos volumes regulares, e nota-se a ocorrência dos mesmos problemas. Em (d), (e) e (f) a maioria das amostras está distribuída em regiões homogêneas. Note como uma distribuição irregular aparenta melhorar a qualidade dos histogramas, o que neste caso em particular é verdade. Em outras situações esta distribuição pode resultar em histogramas piores, já que o objetivo é identificar as bordas que receberam poucos pontos de amostragem. Já em (g), (h) e (i) as amostras concentram-se em regiões de transição, produzindo histogramas mais bem definidos que anteriormente.

Neste caso específico, o histograma fica mais definido. Porém, em outras situações, as amostras tem grandes chances de ficar de fora da região de transição, diminuindo a qualidade dos histogramas. Uma distribuição em regiões de transição resulta em histogramas bem mais definidos, com raras descontinuidades. Esse cenário é o ideal, e corresponde ao que ocorre na realidade, pois malhas não-estruturadas procuram identificar regiões de transição e características bem definidas.

Outro ponto relacionado com a amostragem de funções é a largura das bordas do volume. A figura B.12 o que ocorre quando a largura da borda é modificada, sem alterar a geometria da malha.

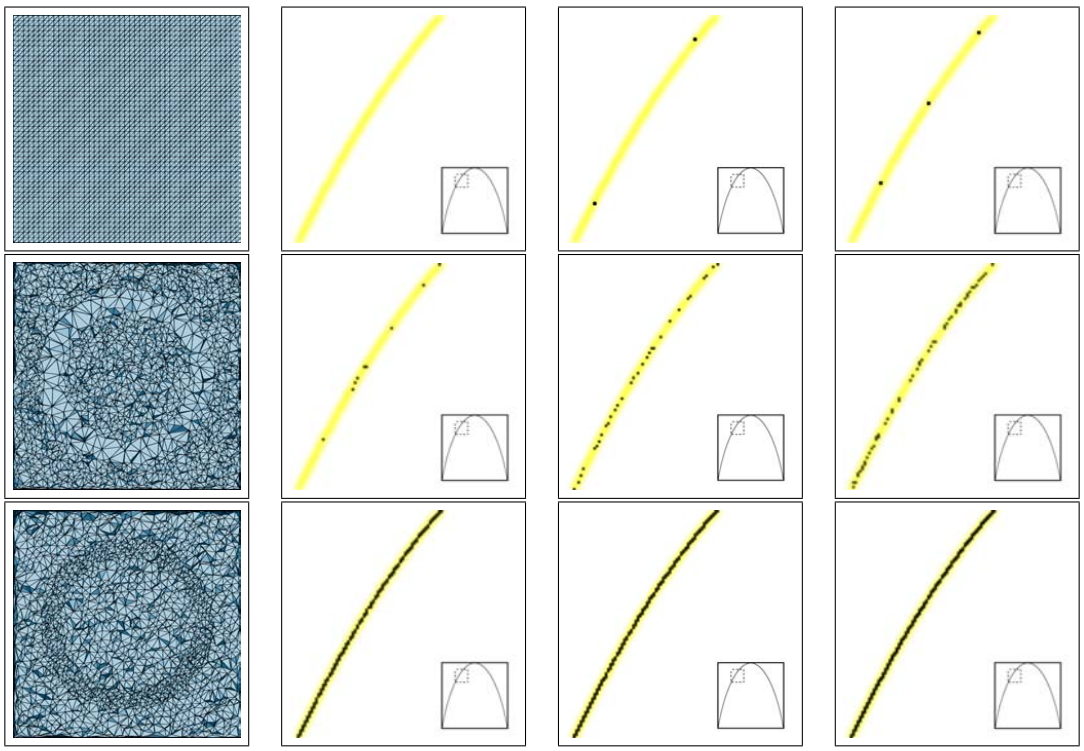


Figure B.12: Efeito de bordas com diferentes larguras para amostragens regulares (1a linha), homogêneas (2a linha) e em áreas de transição (3a linha): larguras de borda de 0.05 (2a coluna), 0.09375 (3a coluna) e 0.2 (4a coluna)(valores normalizados com relação a largura do volume). Bordas mais largas produzem histogramas mais definidos, já que mais amostras estão localizadas na região de transição.

Bordas largas produzem histogramas mais claros, o que facilita sua identificação. Dados reais podem possuir bordas finas, e um cientista pode fazer uso de técnicas de suavização para melhorar a qualidade do resultado. Esta técnica é descrita na seção B.3.3. Mas um fator determinante para a qualidade dos histogramas é o impacto da técnica de aproximação utilizada, como mostrado a seguir.

B.3.2 Aproximação

Diferentes métodos de aproximação podem ser utilizados, de acordo com o tipo de volume sendo utilizado. Dados estruturados possuem, em geral, métodos computacionalmente mais rápidos que dados não-estruturados.

Devido a sua estrutura regular, volumes estruturados beneficiam-se diretamente da extensão para 3D de filtros classicamente utilizados em processamento de imagens. Dentre

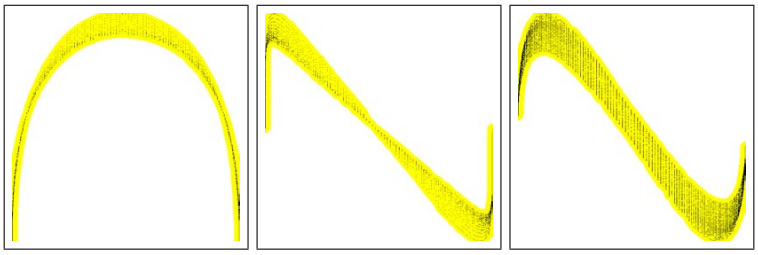
esses filtros, destacam-se Prewitt e Laplace (GONZALEZ; WOODS, 2007), que podem ser vistos na figura B.13.

$$\begin{vmatrix} -1 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix} \qquad \begin{vmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{vmatrix} \qquad \begin{vmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{vmatrix}$$

(a) 1D Prewitt (b) 2D h. Prewitt (c) 2D v. Prewitt (d) Laplacian

Figure B.13: Representação 2D para o filtro horizontal (b) e vertical (c) de Prewitt para computar f', estendido de uma representação 1D (a). O filtro Laplaciano (d) é usado para computação direta de f''. Ambos filtros podem ser utilizados em dados estruturados.

O resultado da aplicação desses filtros no volume da esfera com 128^3 amostras pode ser visto na figura B.14.



(a) Prewitt $f \times f'$ (b) Prewitt $f \times f''$ (c) Laplacian $f \times f''$

Figure B.14: Histograma da magnitude do gradiente com aproximações de f' e f'' utilizando Prewitt, Prewitt aplicado duas vezes, e o filtro Laplaciano, respectivamente. Volume da esfera com 128^3 amostras.

Uma outra técnica que pode ser utilizada tanto em dados regulares quanto dados irregulares é Mínimos Quadrados. Este método é bastante utilizado para reconstruir o gradiente e realizar filtragens (HASELBACHER, 2001). Este trabalho utiliza a versão matricial do método, apresentada na equação B.6.

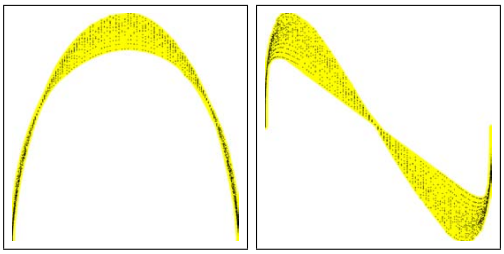
$$X^T y = \left(X^T X\right) \beta \tag{B.6}$$

O método de mínimos quadrados pode utilizar pesos diferenciados para cada amostra, para melhor refletir a importância de cada uma. A análise de ambas as versões da técnica (com e sem pesos) será mostrada no final deste capítulo.

Em volumes regulares, seis *voxels* foram utilizados para computar o gradiente de cada *voxel* central (um acima, um abaixo, um a esquerda, um a direita, um a frente e um atrás). O resultado pode ser visto na figura B.15.

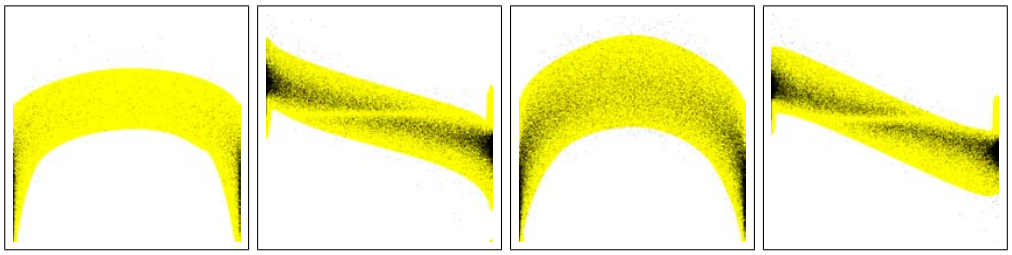
O cálculo do gradiente em volumes irregulares foi realizado através do uso dos 32 vizinhos mais próximos do ponto sendo analisado, a fim de manter o custo computacional a um nível aceitável. Foi utilizado somente o volume cujas amostras concentram-se na região de transição entre as áreas homogêneas. A figura B.16 mostra os histogramas produzidos.

A análise da figura B.16 mostra que o resultado de mínimos quadrados utilizando pesos possui uma forma de arco mais definida que mínimos quadrados tradicional. Os histogramas para f'' são computados pela aplicação consecutiva de mínimos quadrados (onde o resultado de uma primeira etapa é usado como entrada em uma segunda etapa).



(a) $f \times f'$ (b) $f \times f''$

Figure B.15: Histogramas da magnitude do gradiente utilizando mínimos quadrados para aproximar f' e f''. Volume regular da esfera com 128^3 amostras.



(a) LS $f \times f'$ (b) LS $f \times f''$ (c) WLS $f \times f'$ (d) WLS $f \times f''$

Figure B.16: Histograma da magnitude do gradiente utilizando aproximações de f' e f'' com os métodos de mínimos quadrados puro (LS) e com pesos (WLS). Volume irregular com bordas densas e 100.000 amostras.

Nota-se que os erros acumulados duas vezes praticamente removem a possibilidade de se identificar a forma desejada.

Quatro métricas foram utilizadas para analizar as técnicas de aproximação:

- Diferença das magnitudes (DM): é o módulo da diferença entre as soluções reconstruídas e analíticas.
- Razão entre as magnitudes (RM): calcula a solução *analítica* sobre *reconstruída*
- Comprimento da diferença do vetor gradiente (LDV): calcula o módulo do vetor diferença entre os gradientes analítico e reconstruído.
- Produto escalar (DP): mede o cosseno do ângulo entre as soluções analítica e reconstruída.

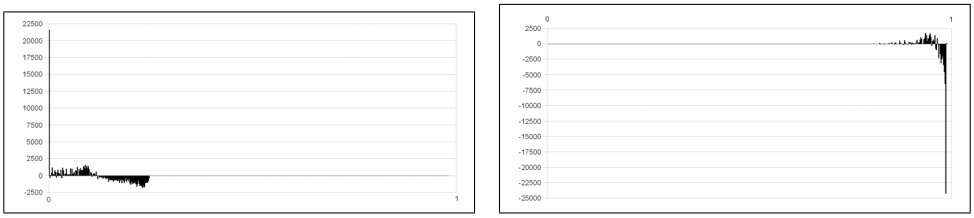
A tabela B.1 mostra o resultado dessas métricas calculadas tanto para malhas estruturadas como não-estruturadas.

Para entender corretamente o resultado, deve-se analisar o comportamento das amostras mais profundamente. A figura B.17 apresenta o resultado da análise para volumes estruturados. Ela mostra como mínimos quadrados aproximam melhor a magnitude do gradiente, enquanto Prewitt aproxima melhor a direção do mesmo. Isto leva a conclusão que uma técnica pode ser escolhida baseada na aplicação, aproximando melhor a magnitude ou a direção do gradiente.

O mesmo tipo de análise foi realizado com volumes não-estruturados, porém comparando mínimos quadrados tradicional com sua versão com pesos, utilizando a distância entre as amostras como fator de ponderação. A figura B.18 mostra como o uso de pesos para ponderar a técnica de mínimos quadrados melhora a aproximação da magnitude

Table B.1: Resultado das comparações entre os métodos de reconstrução e a solução analítica utilizando diversas métricas. O volume regular utilizado possui 128^3 amostras, e o irregular 100.000 amostras.

	Estruturados		Não-estruturados	
	(W)LS	Prewitt	LS	WLS
DM_{Min}	0.000000	0.000000	0.000000	0.000000
DM_{Max}	0.156699	0.252082	0.623909	0.549086
RM_{Min}	0.000000	0.000000	0.000000	0.000000
RM_{Max}	1.185967	1.101218	6.009077	3.970824
LDV_{Min}	0.000000	0.000000	0.000000	0.000000
LDV_{Max}	0.162731	0.252089	0.661233	0.572888
DP_{Min}	0.636256	0.669197	-0.999983	-0.999462
DP_{Max}	1.000000	1.000000	1.000000	1.000000



(a) DM: Mínimos quadrados - Prewitt (b) DP: Mínimos quadrados - Prewitt

Figure B.17: Resultado da reconstrução do gradiente de acordo com as métricas da diferença entre as magnitudes do vetor gradiente (DM) em (a) e o cosseno do ângulo entre os gradientes reconstruído e analítico (DP) em (b). Note como o primeiro histograma demonstra uma melhor qualidade de mínimos quadrados sobre Prewitt para reconstruir a magnitude, enquanto o segundo histograma mostra uma melhor aproximação da direção do gradiente por Prewitt. Volume regular da esfera com 128^3 amostras.

do gradiente. Quanto a direção, uma análise inicial aponta para uma melhor qualidade quando pesos não são utilizados. Todavia, como diferentes parâmetros podem ser utilizados para modular o resultado de mínimos quadrados com pesos, essa técnica tende a ser mais robusta (MAVRIPLIS, 2003, 2007).

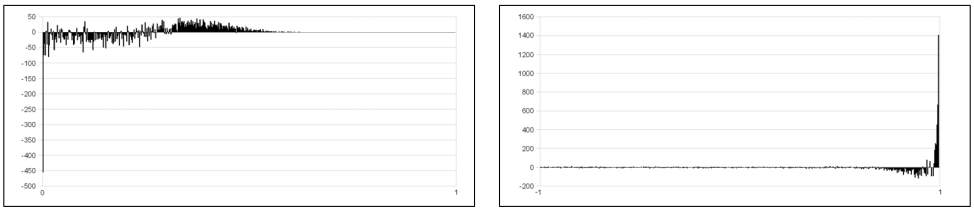
Após verificar a qualidade das técnicas de reconstrução do gradiente, um outro fator ainda é muito importante para determinar a qualidade dos histogramas da magnitude do gradiente. A suavização do volume de dados é discutida a seguir.

B.3.3 Suavização

A suavização das bordas entre as regiões de um volume de dados afeta diretamente a computação dos histogramas. Isto é devido a duas razões principais:

- bordas suaves em um domínio discreto serão mais largas que bordas abruptas, possuindo assim mais amostras. Isso diminuirá os espaçamentos entre as amostras na dimensão $f(x)$ do histograma.

- transições suaves produzirão uma gama maior de magnitudes do gradiente (utilizando o modelo de borda descrito na seção B.3). Isto reduz os espaçamentos entre as amostras nas dimensões $f\prime(x)$ e $f\prime\prime(x)$ do histograma.

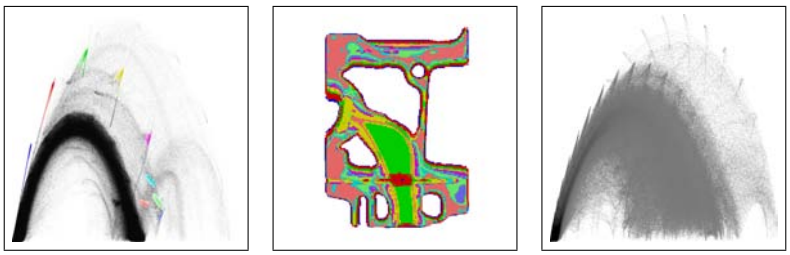


(a) DM: Least Squares - Weighted Least Squares

(b) DP: Least Squares - Weighted Least Squares

Figure B.18: Resultado da reconstrução do gradiente de acordo com as métricas da diferença entre as magnitudes do vetor gradiente (DM) em (a) e o cosseno do ângulo entre os gradientes reconstruído e analítico (DP) em (b). Note como o primeiro histograma demonstra uma melhor qualidade quando pesos são utilizados com mínimos quadrados para reconstruir a magnitude, enquanto o segundo histograma mostra que mínimos quadrados sem pesos aproxima melhor a direção do gradiente. Volume irregular da esfera com 100.000 amostras.

Trabalhos anteriores utilizam suavização para melhorar a qualidade dos histogramas computados (KINDLMANN; DURKIN, 1998). Porém cuidados especiais devem ser tomados para garantir uma correta interpretação dos resultados. A figura B.19 mostra o que seriam diversas bordas. Isso é, porém, o efeito uma borda que possui poucas amostras. A figura B.19(c) mostra o resultado da suavização do volume original antes de computar o histograma. Um filtro gaussiano de dimensões 3x3x3 foi utilizado para gerar essa figura, sendo aplicado 8 vezes em seqüência.

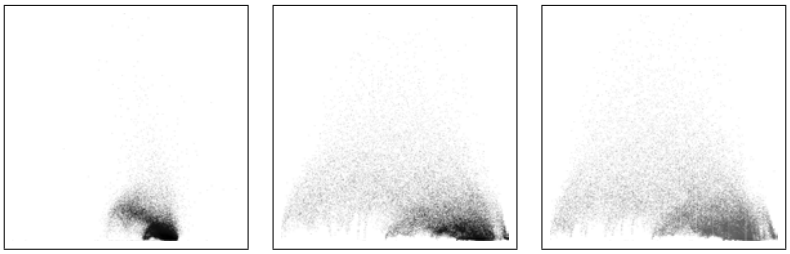


(a) (b) (c)

Figure B.19: Artefatos isolados não representam necessariamente diferentes bordas. No caso acima, os diferentes acúmulos de pontos em (a) fazem parte da mesma borda, que é estreita demais e não possui amostras suficientes para amostrá-la (b). A aplicação de suavização no volume em (c) melhora a qualidade deste arco, porém acaba misturando os demais arcos.

A aplicação de suavização para dados não-estruturados é mais complicada, pois sua disposição irregular impede a aplicação de um filtro clássico de processamento de imagens. Neste trabalho, 32 vizinhos foram utilizados para computar a suavização de cada pixel utilizando a própria equação gaussiana. A equação utilizada é única para o volume, e possui ponto de inflecção a $2\times$ a distância da aresta média do volume e ponto de corte a $3\times$ essa distância. A figura B.20 mostra os histogramas do volume irregular F117 com diferentes níveis de suavização. Versões do histograma com volume suavizado permitem uma identificação visual mais fácil das estruturas em arco.

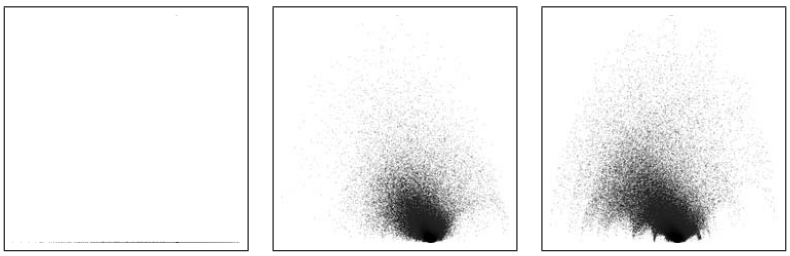
Alguns histogramas, todavia, continuam apresentando problemas. A técnica de suavização não consegue corrigir o histograma do volume TJET, mostrado na figura B.21(a) e apresentado na seção B.2.1. Essa distorção é causada por algumas amostras que são muito



(a) (b) (c)

Figure B.20: Exemplo de como suavização ajuda a melhorar a qualidade de histogramas de volumes não-estruturados. O histograma original (a) apresenta diversas amostras concentradas no meio da imagem. A aplicação sucessiva de suavização diminui as distorções e possibilita uma melhor identificação dos arcos (5 níveis em (b) e 9 níveis em (c)).

díspares com relação a maioria das amostras. Para corrigir o histograma nesse caso uma medida estatística é utilizada: todas as amostras cuja variação da magnitude do gradiente for maior que n vezes a variação média do volume são descartadas. A figura B.21 apresenta o resultado desse filtro no volume TJET, onde n é igual a 10.



(a) (b) (c)

Figure B.21: A suavização sozinha não corrige o volume TJET, cujas amostras concentram-se na parte inferior do histograma em (a). O uso de técnica proposta em (b) melhor a visualização da estrutura do histograma, que pode ser melhorada utilizando a suavização (c).

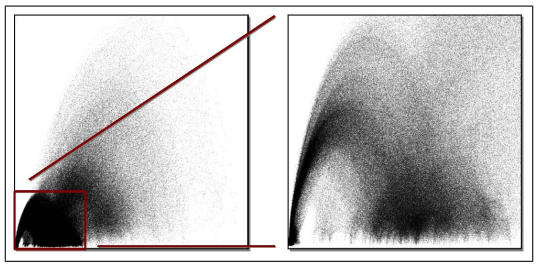
De posse dessas ferramentas, algumas aplicações foram desenvolvidas, como mostra a próxima seção.

B.4 Aplicações

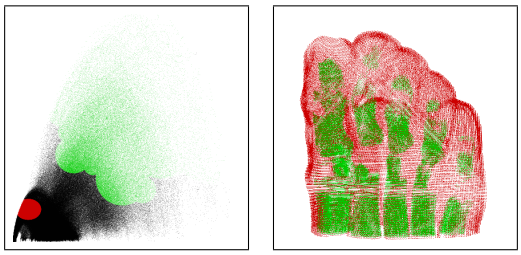
Duas aplicações foram desenvolvidas utilizando os conceitos previamente descritos neste trabalho. Primeiramente, uma ferramenta de análise para localização e realce de características foi desenvolvida. A ferramenta mostra a correspondência das amostras dos histogramas com o volume original no espaço 3D (figura B.22). A ferramenta permite a manipulação de *zoom*, controle de opacidade para o histograma e realiza a correlação de pontos do histograma no volume em 3D.

A outra aplicação é a própria ferramenta de criação funções de transferência apresentada na seção B.2. As soluções aqui apresentadas para dados não-estruturados foram incorporadas no sistema desenvolvido. A figura B.23 mostra diferentes volumes com suas respectivas funções de transferência, que foram desenvolvida no sistema apresentado.

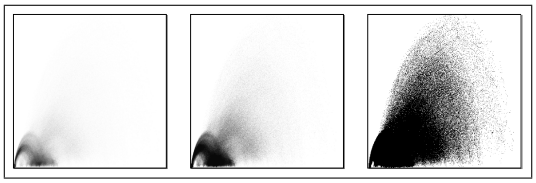
Fora essas aplicações desenvolvidas, outras aplicações podem beneficiar-se desta pesquisa, como exemplificado a seguir.



(a) Zooming tool

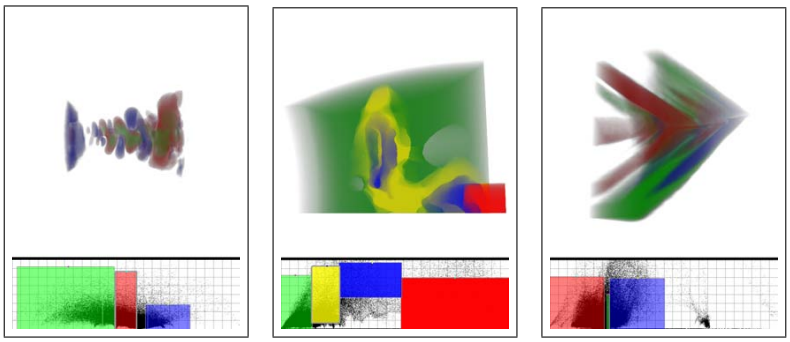


(b) Histogram space (c) Main visualization



(d) Opacity control

Figure B.22: Visualizador que correlaciona dados do histograma para dados do espaço 3D do volume. Zoom é demonstrado em (a), pontos selecionados no histograma em (b) são mapeados em (c), e diferentes opacidades são demonstradas em (d). Este volume é uma versão reduzida do CT de um pé humano.



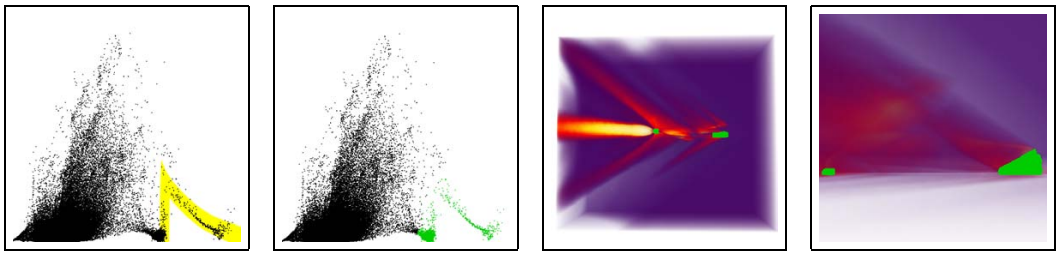
(a) TJet (b) SPX (c) *Fighter*

Figure B.23: Exemplo de visualização de volumes não-estruturados utilizando funções de transferência criadas com o histograma da magnitude do gradiente no sistema apresentado.

B.5 Trabalhos Futuros

Diversas aplicações podem beneficiar-se da análise desenvolvida anteriormente. A seguir serão mencionadas duas possibilidades de trabalhos futuros envolvendo estatísticas em funções discretas.

O modelo de borda assumido neste trabalho representa a transição ideal entre duas regiões homogêneas. Este fato é bastante comum para volumes médicos (KINDLMANN; DURKIN, 1998), porém em outras áreas modelos diferentes podem proporcionar melhor resultados. Como exemplo, na figura B.24, pode-se distinguir uma borda cujo formato não condiz com o modelo esperado de borda. É possível que modelos de borda adaptados para as condições dos volumes (simulação, digitalização, etc.) produzam melhores resultados que um modelo único.



(a) (b) (c) (d)

Figure B.24: Exemplo de um formato de borda diferente no volume *Fighter* (a). A correspondência dos pontos selecionados no histograma em (b) são mostrados em (c) e (d).

Outra possibilidade de exploração futura é utilizar histogramas como métrica para medir a qualidade de simplificações de volumes não-estruturados. A simplificação de volumes irregulares é uma prática comumente utilizada para reduzir o tamanho destes volumes, aumentando a velocidade de processamento dos dados e permitindo uma melhor interação. Para realizar isso, no entanto, é necessário utilizar uma forma automática de detectar a estrutura da borda no histograma. Testes iniciais foram realizados utilizando a transformada de *Hough* (ILLINGWORTH; KITTLER, 1988; FERNANDES; OLIVEIRA, 2008), porém mais análise é necessária antes de chegar a uma decisão definitiva sobre a utilidade desta nova métrica.

B.6 Conclusão

A visualização volumétrica é uma importante ferramenta para a exploração do domínio espacial 3D. Apesar de muitas pesquisas abordarem este tópico, muitos problemas ainda encontram-se abertos.

As contribuições apresentadas neste trabalho dividem-se basicamente em dois conjuntos. Primeiramente, novas propostas para a criação de funções de transferência foram apresentadas, utilizando-se agrupamentos para produzir mapeamentos complexos a partir de mapeamentos base mais simples, como explicado na seção B.2.2. Além disso, um método para tratar a visualização de dados com variação temporal utilizando uma técnica baseada em quadros-chave foi apresentada na seção B.2.3. Esta técnica possibilita a visualização de dados estatisticamente estáticos e dados estatisticamente dinâmicos. Uma limitação desta técnica, todavia, é que ela não elimina descontinuidades visuais quando funções de transferência diferentes são utilizadas. Ela apenas diminui sua percepção pela interpolação entre dois quadros-chave.

O segundo grupo de contribuições apresenta um estudo sobre o comportamento da borda em volumes de dados. Ao simplesmente aplicar o modelo de borda tradicional de volumes regulares em volumes não-estruturados, histogramas degenerados foram produzidos. Para compreender o porquê disso, estratégias de amostragem e de aproximação dos volumes foram estudadas. Artefatos gerados por má amostragem da função foram identificados em dados regulares, porém podem ocorrer em qualquer uma das classes. A suavização do volume antes de computar o histograma aumenta a qualidade dos arcos que representam bordas, mas também podem causar a aparição de mais artefatos similares aos que tenta evitar.

A precisão de técnicas de aproximação demonstrou que a utilização de mínimos quadrados utilizando pesos produz um resultado mais preciso quando se trabalha com volumes não-estruturados. No caso de volumes regulares, tanto mínimos quadrados quanto o filtro de Prewitt apresentaram bons resultados.

Apesar da aplicação de suavização, alguns volumes de dados irregulares apresentaram uma concentração anormal de amostras em determinadas regiões do histograma. Para solucionar este problema, amostras muito distorcidas da média do volume são descartadas quando o histograma é computado.

Duas aplicações foram desenvolvidas para auxiliar na visualização e na exploração de volumes tanto regulares quanto não-estruturados. Além disso, dois exemplos de trabalhos futuros que utilizam os conceitos desenvolvidos neste trabalho foram apresentados.

Sem correto entendimento de características apresentadas pelas diferentes classes de volumes uma solução geral para ambos os tipos não será alcançada. Apenas após compreender corretamente os problemas envolvidos é que desenvolvedores poderão propor soluções robustas o suficiente para lidarem com dados regulares e irregulares, reduzindo custos de desenvolvimento de soluções específicas e de treinamento de usuários em ferramentas também específicas.