Ain Shams University

Faculty of Computer & Information Sciences

Department of Scientific Computing

Visualization of Time Varying Data on High Performance Computing

Ahmed Safwat Ali Youssef

B.Sc. in Computer and Information Sciences (2004)

Under Supervision of

Mohamed Fahmy Tolba

Professor, Department of Scientific Computing,
Faculty of Computer and Information Sciences,
Ain Shams University.

Ashraf Saad Hussein

Professor, Department of Scientific Computing,
Faculty of Computer and Information Sciences,
Ain Shams University.

Ahmed Hassan Youssef

Associated Professor, Department of Computer Engineering,
Faculty of Engineering,
Ain Shams University.

Cairo (2010)

ACKNOWLEDGEMENTS

I am heartily thankful to my supervisors, Prof. Mohamed Fahmy Tolba, Prof. Ashraf Saad Hussein and Dr. Ahmed Hassan Youssef, whose encouragements, guidance and support from the initial to the final level enabled me to develop an understanding of the subject. Many thanks to Dr. Amal Khalifa for the help she offered. I am also indebted to my colleagues at the Faculty of Computer and Information Sciences.

It is a pleasure to thank my family members who made this thesis possible.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Ahmed Safwat Ali.

ABSTRACT

Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights. In many fields it is already revolutionizing the way scientists do science. The goal of visualization is to leverage existing scientific methods by providing new scientific insight through visual methods.

In the modeling of many scientific and engineering problems, vector fields are used to describe moving fluids or changing forces, where a vector (i.e., a direction with magnitude) is assigned to each point in the space-time domain. Effective visualization of time-varying 3D vector fields is critical for the understanding of complex phenomena and dynamic processes under investigation. A typical time-varying dataset from a Computational Fluid Dynamics (CFD) simulation can easily require hundreds of gigabytes or even terabytes of storage space, which creates challenges for the consequent data-analysis tasks.

In this research, new techniques for visualization of extremely large time-varying vector data using high performance computing are presented. The high level requirements that guided the formulation of the new techniques are (a) support for large dataset sizes, (b) support for temporal coherence of the vector data, (c) support for distributed memory high performance computing and (d) optimum utilization of the computing nodes with multi-cores (multi-core processors). The challenge is to design and implement techniques that meet these complex requirements and balance the conflicts between them. The fundamental innovation in this work is developing efficient distributed visualization for large time-varying vector data. The maximum performance was reached through the parallelization of multiple processes on the multiple cores of each computing node. Accuracy of the proposed techniques was confirmed compared to the benchmark results with average difference of 5%. In addition, the proposed techniques exhibited acceptable speedup that reaches 700% for different data sizes with better scalability for larger ones. Finally, the utilization of the computing nodes was satisfactory for the considered test cases.

KEYWORDS

Visualization, Time Varying Data, High Performance Computing, Parallel, Vector Data, Stream-Lines, Path-lines, Stream Surface, Flow Field, Flow Volume.

TABLE OF CONTENT

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

1.01 MOTIVATION

The field of visualization focuses on creating images that convey salient information about underlying data and processes. In the past three decades, the field has seen unprecedented growth in computational and acquisition technologies [1] [2], which has resulted in an increased ability both to sense the physical world with very detailed precision and to model and simulate complex physical phenomena [2]. Given these capabilities, visualization plays a crucial enabling role in our ability to comprehend such large and complex data—data that, in two, three, or more dimensions, conveys insight into such diverse applications as medical processes, earth and space sciences, complex flow of fluids, and biological processes, among many other areas as shown in Figure 1.1.

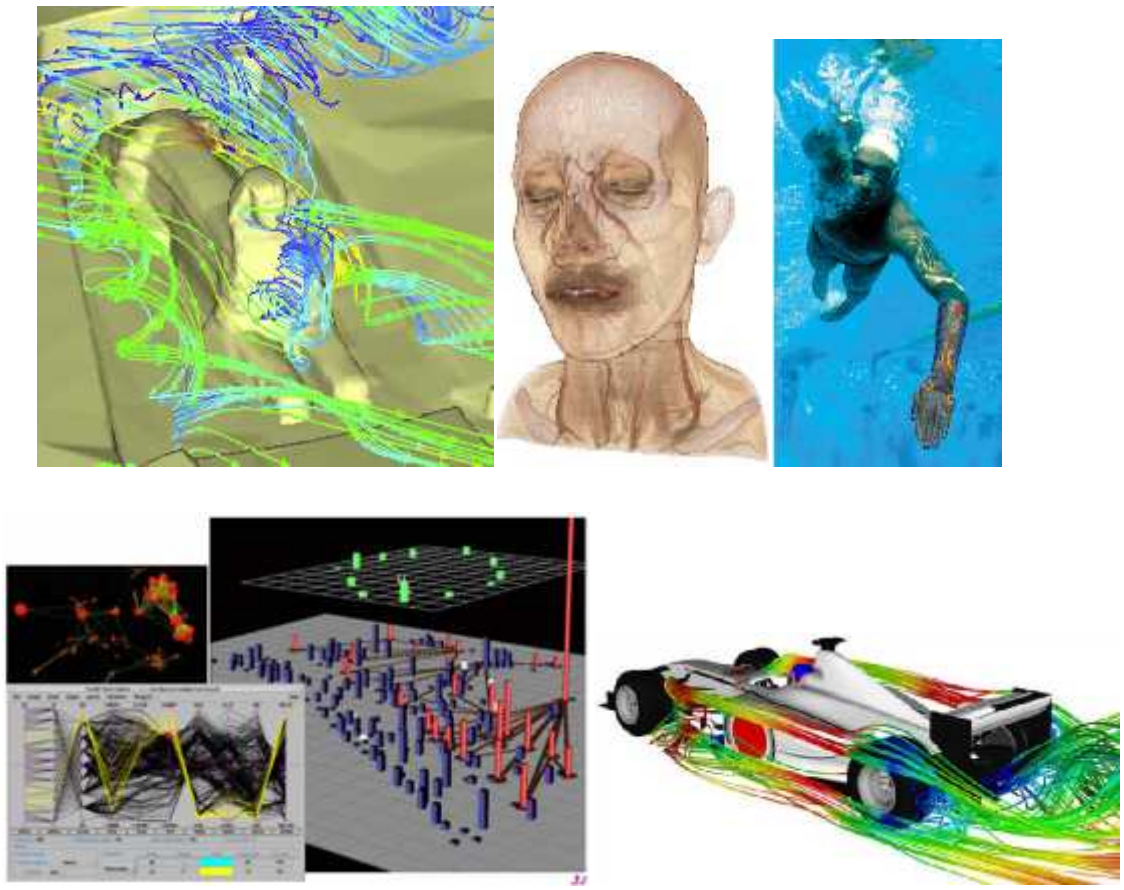

Figure 1.1 Visualization applications fields, [3]

The discipline of visualization as it is currently understood was born with the advent of scientific computing and the use of computer graphics for depicting computational data.

Simultaneously, devices capable of sensing the physical world, from medical scanners to geophysical sensing to satellite-borne sensing and the need to interpret the vast amount of data either computed or acquired, have also driven the field. In addition to the rapid growth in visualization of scientific and medical data, data that typically lacks a spatial domain has caused the rise of the field of information visualization [4].

Vector visualization is an important topic in scientific visualization and has been the subject of active research for many years [5] [6] [7]. Typically, data originates from numerical simulations, such as those of computational fluid dynamics, and needs to be analyzed by means of visualization to gain an understanding of the flow. The role of visualization in simulation pipeline is shown in Figure 1.2. With the rapid increase of computational power for simulations, the demand for more advanced visualization methods has grown. One of the most promising methods is the use of distributed architectures to improve the performance of the visualization algorithms [8] [9]. This raises a new research challenge in applying the distributed architecture on time varying data visualization.

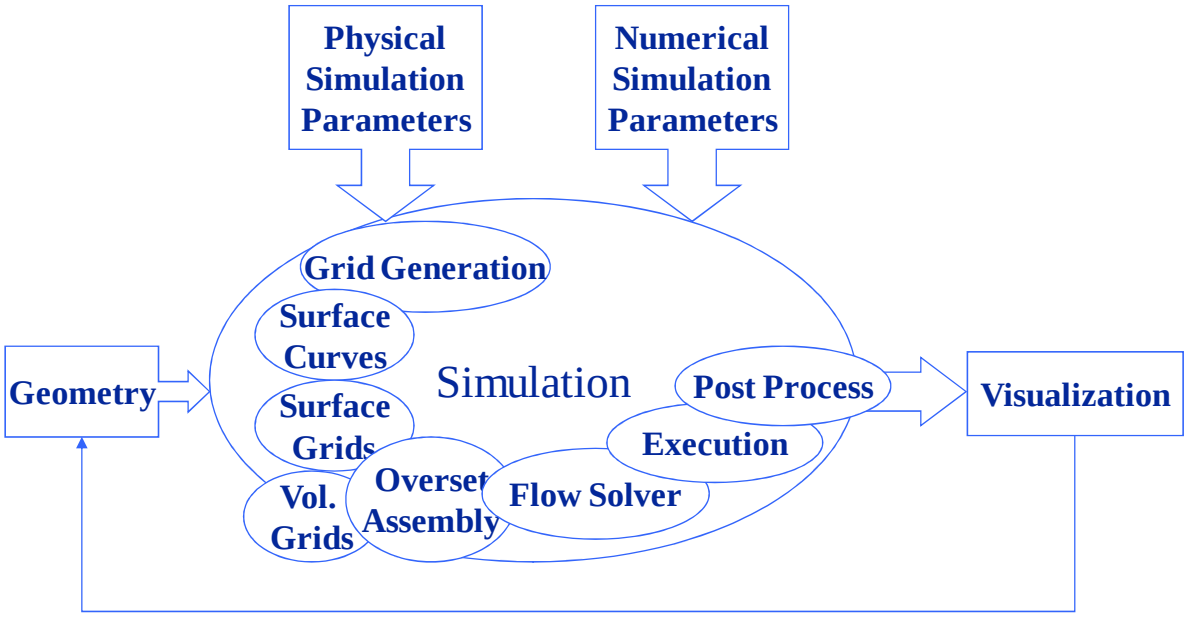

Figure 1.2 Role of visualization in simulation pipeline, [3]

1.02 THESIS CHALLENGES AND CONTRIBUTIONS

In computational flow visualization, integration based geometric flow visualization is often used to explore the flow field structure. A typical time-varying dataset from a Computational Fluid Dynamics (CFD) simulation can easily require hundreds of gigabytes to even terabytes of storage space, which creates challenges for the consequent data-analysis tasks. Integration based geometric flow visualization is often used to explore the flow field structure when scientists attempt to visualize and understand the data generated from simulations, the huge size of the data is one of the major challenges. To address this challenge, a lot of research has been pursued [10], [5], [11] focusing on large scale data visualization. However, most of the techniques were developed for the visualization of scalar field data.

Visualization of vector data has also been an active area of research [12], [13]. Regarding large scale time varying 3D vector fields, fewer studies have been conducted [14], [15] for several reasons. First, the size of the vector data sets is three times or more that of the corresponding scalar field. Therefore, traditional workstations generally do not have the memory capacity or the processing power needed to visualize such huge data sets. Second, when directly applied to 3D vector data, most of the effective 2D vector field visualization methods face the “visual clutter” problem. Finally, additional attention to temporal coherence is required for visualizing time varying vector data. Consequently, previous work [12], [16] for vector field visualization focused primarily on 2D data sets, steady flow field, and the topological aspect of the vector fields, such as, the associated seed/glyph placement problem.

1.03 THESIS OBJECTIVES

This research presents new techniques for vector data visualization of extremely large time varying vector data using high performance computing. The high level requirements that guided the formulation of the new techniques are (a) support for large dataset sizes, (b) support for temporal coherence of the vector data, (c) support for distributed memory high performance computing and (d) optimum utilization of the computing nodes with multi-cores (multi-core processors). The challenge is to design and implement techniques that meet these complex requirements and balance the conflicts between them. The fundamental innovation in this work is developing efficient distributed path-lines and stream surface visualization for large time varying vector data.

1.04 ORGANIZATION OF THE THESIS

The thesis treats this topic in five chapters, in addition to a conclusion chapter, an appendix and a list of references.

Chapter 1 gives an overview for visualization of time varying data, a quick review on related work, motivation, objective and thesis's organization.

Chapter 2 gives the basic concepts and background for visualization of time varying data.

Chapter 3 gives a detailed survey for the current approaches and methods for visualization. Also, it discusses the advantages and disadvantages of each approach as a comparison among them.

Chapter 4 describes the system architecture and the proposed visualization techniques. It shows the detailed operations and scenarios for path-lines. Then it presents enhancement on the architecture to work on multi-cores machines. Finally, it compares the proposed technique with the other approaches.

Chapter 5 explains the use of the proposed technique in and stream surfaces visualization. It then shows the experiments used to test the proposed technique and results extracted from it.

CHAPTER 2. INTRODUCATION TO VECTOR FILED VISUALIZATION

In this chapter, the basic algorithms for scientific visualization are presented. In practice, a typical algorithm can be thought of as a transformation from one data form into another [17]. These operations may also change the dimensionality of the data. For example, generating a streamline from a specification of a starting point in an input 3D dataset produces a 1D curve. The input may be represented as a simple finite difference mesh associated with its solution fields, while the output may be represented as a poly line. Such operations are typical of scientific visualization Systems that repeatedly transform data into different forms and ultimately transform it into a representation that can be rendered by the computer system.

The algorithms that transform data are the core of data visualization. To describe the various transformations available, we need to categorize algorithms according to the structure and type of transformation. By structure, we mean the effects that transformation has on the topology and geometry of the dataset. By type, we mean the type of dataset that the algorithm operates on.

Structural transformations can be classified in four ways [4], depending on how they affect the geometry, topology, and attributes of a dataset. Here, we consider the topology of the dataset as the relationship of discrete data samples (one to another) that are invariant with respect to geometric transformation. For example, a regular, axis aligned sampling of data in three dimensions is referred to as a volume, and its topology is a rectangular (structured) lattice with clearly defined neighborhood voxels and samples. On the other hand, the topology of a finite element mesh is represented by an (unstructured) list of elements, each defined by an ordered list of points. Geometry is a specification of the topology in space (typically 3D), including point coordinates and interpolation functions. Attributes are data associated with the topology and/or geometry of the dataset, such as temperature, pressure, or velocity. Attributes are typically categorized as being scalars (single value per sample), vectors (n-vector of values), tensor matrix), surface normal's, texture coordinates, or general field data. Given these terms, the following transformations are typical of scientific visualization systems:

- Geometric transformations alter input geometry but do not change the topology of the dataset. For example, if we translate, rotate, and/or scale the points of a polygonal dataset, the topology does not change, but the point coordinates, and the geometry do change.
- Topological transformations alter input topology but do not change geometry and attribute data. Converting a dataset type from polygonal to unstructured grid, or from image to unstructured grid, changes the topology but not the geometry. However, the geometry changes whenever the topology does, so topological transformation is uncommon.
- Attribute transformations convert data attributes from one form to another, or create new attributes from the input data. The structure of the dataset remains unaffected. Computing vector magnitude and creating scalars based on elevation are data attribute transformations
- Combined transformations change both dataset structure and attribute data. For example, computing contour lines or surfaces is a combined transformation.

We also may classify algorithms according to the type of data they operate on. The meaning of the word ‘‘type’’ is often somewhat vague. Typically, ‘‘type’’ means the type of attribute data, such as scalars or vectors. These categories include the following [18]:

- Scalar algorithms:

  Scalars are single data values associated with each point and/or cell of a dataset. Scalar algorithms operate on scalar data. An example: the generation of contour lines of temperature on a weather map.

- Vector algorithms:

  Vector data is a 3D representation of direction and magnitude. Vector data often results from the study of fluid flow or data derivatives, Vector algorithms operate on vector data. Showing oriented arrows of airflow (direction and magnitude) is an example of vector visualization.

- Tensor algorithms:

Operate on tensor matrices. An example of a tensor algorithm is to show the components of stress or strain in a material using oriented icons.

- Modeling algorithms:

Generate dataset topology, dataset geometry, surface's normal or texture data. ''Modeling algorithms'' tends to be the catch-all category for algorithms that do not fit neatly into any single category mentioned above. For example, generating glyphs oriented according to the vector direction and then scaled according to the scalar value is a combined scalar/vector algorithm. For convenience, we classify such algorithm as a modeling algorithm because it does not fit squarely into any other category.

In this research we are concerned more with the 3D vector data visualization techniques which face more challenges for example vector data set contains three or more times the data to its corresponding scalar field, most of the effective 2D vector field visualization methods face visual clutter when directly applied to 3D vector data Finally, additional attention to temporal coherence is required for visualizing time-varying vector data [19].

In the following sections we are going to go through the common vector visualization techniques to understand their nature and their challenges in details. Then we will move on large scale time varying 3D vector data visualization and the latest researches which are concerned about them

2.01 VECTOR DATA VISUALIZATION

Flow visualization is an important topic in scientific visualization and has been the subject of active research for many years. Typically, data originates from numerical simulations, such as those of computational fluid dynamics, and needs to be analyzed by means of visualization to gain an understanding of the flow. With the rapid increase of computational power for simulations, the demand for more advanced visualization methods has grown. This section presents an overview of important and widely used approaches to flow visualization.

We start with a rather abstract definition of a vector field by making use of concepts from differential geometry and the theory of differential equations. For more detailed background information on these topics, we refer to textbooks on differential topology and geometry [20] [21] [22] [23]. Although this mathematical approach might seem quite abstract for many applications, it has the advantage of being a flexible and generic description that is applicable to a wide range of problems. We first give the definition of a vector field.

2.01.1 MATHEMATICAL DESCRIPTION

Let M be a smooth m-manifold with boundary, let N be an n-dimensional sub manifold with boundary ($N \in M$), and let $I \in IR$ be an open interval of real numbers. A map

$$U: N \times I \to TM \tag{2.1}$$

is a time-dependent vector field provided that $u\ x, t\ \in TxM$

An element $t \in I$ serves as a description for time; $x \in N$ is a position in space. TM Is a tangent bundle—the collection of all tangent vectors, along with the information of the point of tangency. Finally, TxM is the tangent space associated with$x$. The vector field maps a position in space and time,(x, t) to a tangent vector located at the same reference point$x$. For a tangential time-dependent vector field, the mapping remains in the tangent bundle TN and therefore does not contain a normal component. That is,

$$U: N \times I \to TN \tag{2.2}$$

For a non tangential vector field, a related tangential vector field can be computed by projection from TxM to TxN, i.e., by removing the normal parts from the vectors.

Integral curves are directly related to vector fields. Let $u : N \times I \to TN$ be a continuous (tangential) vector field. Let x_0 be a point in N and $J \subset I$ be an open interval that contains t_0. The C^1 map $\varepsilon_{x_o,t_0}: J \to N$ with

$$\varepsilon_{x_o,t_0}\ t_0\ = x_0\ and\ \frac{d\varepsilon_{x_o,t_0}(t)}{dt} = u(\varepsilon_{x_o,t_0}\ t\ , t) \tag{2.3}$$

Is an integral curve for the vector field with initial condition $x = x_0 \; at \; t = t_0$. The subscripts in the notation of ε_{x_o,t_0} denote this initial condition. These integral curves are usually referred to as path-lines, especially in the context of flow visualization. If u satisfies the Lipschitz condition, the differential equation for ε_{x_o,t_0} has a unique solution.

In all practical applications of flow visualization, the data is given on a manifold of two or three dimensions. To investigate a vector field on an arbitrary curved surface, the above formalism is necessary and, for example, the issue of describing the surface by charts has to be addressed. Very often, however, N is just a Euclidean space. This allows us to use a simpler form of the tangential vector field given by

$$u: \Omega \times I \rightarrow IR, \;\; x,t \;\; \rightarrow u \;\; x,t \qquad (2.4)$$

The vector field is defined on the n-dimensional Euclidean space $\Omega \subset IR^n$ and depends on time $t \in I$. We use boldface lowercase letters to denote vectors in n dimensions. The reference point x is no longer explicitly attached to the tangent vector. In this modified notation, the integral curve is determined by the ordinary differential equation

$$\frac{dx_{path} \;\; t; x_o, t_0}{dt} = u \;\; x_{path} \;\; t; x_o, t_0 \;\; , t \qquad (2.5)$$

We assume that the initial condition $x_{path} \;\; t; x_o, t_0 \;\; = x_0 \; is \; given \; at \; time \; t_0$ t_0, i.e., all integral curves are labeled by their initial conditions (x_0, t_0) and parameterized by t. By construction, the tangent to the path line at position x and time t is precisely the velocity $u \;\; x,t \;\;$. In more general terms, the notation $x(t; x_0, t_0)$ is used to describe any curve parameterized by t that contains the point x_0 for $t = t_0$.