University of West Bohemia
Department of Computer Science and Engineering
Univerzitni 8
30614 Plzeň
Czech Republic

Iso-surface extraction from time-varying data

State of the Art and Future Research

Slavomír Petrík

Supervisor:Václav Skala

Technical Report No. DCSE/TR-2007-06
June, 2007

Distribution: public

Supported by the projects: *VIRTUAL* - Virtual Research-Educational Center of Computer Graphics and Visualization, MSMT Czech Rep. No: 2C 06002, http://virtual.zcu.cz, *3DTV* - Integrated Three-Dimensional Television - Capture, Transmission and Display, FP6-2003-IST-2, Network of Excellence, No:511568, http://3DTV.zcu.cz, *INTUITION* - Network of Excellence on VIrtual Reality aNd VirTUal Environments AppIIcaTIONs for Future Workspaces, FP6-2003-IST-2, Network of Excellence No:507248-2, http://intuition.zcu.cz.

Technical Report No. DCSE/TR-2007-06
June 2007

Iso-surface extraction from time-varying data

Slavomír Petrík

Abstract

This work is focused on the iso-surfaces extraction from scalar data sets with dynamic simulation mesh. Such data sets usually originate from Computational Fluid Dynamic (CFD) simulations, where moving boundaries of a simulation domain force a simulation mesh to change itself with each discrete time step. Up to now, each time step has been treated as a stand-alone entity, because of lack of the methods capturing spatial and temporal coherency in data sets of such nature.

The work covers the existing techniques for iso-surfaces extraction from time-varying scalar data sets with static mesh and provides an introduction to the problematic of dynamic meshes and an overview of the initial research done. Finally our concept of handling data sets with time-varying meshes is described.

keywords: iso-surface, extraction, scalar field, time-varying mesh

Abstrakt

Táto práce se zabývá extrakci iso-ploch ze skalárních datových množin obsahujících časově proměnnou síť. Taková data pocházejí například z numerických simulací proudění kapalin. Nedostatek zobrazovacích metod umožňujících práci s časově proměnnými sítěmi způsobil, že v existujících aplikacích bylo s kažým časovým krokem zacházeno jako se samostatnou datovou množinou.

Táto práce popisuje existujíci metody extrakce iso-ploch z časově proměnných skalárnich dat se statickou síti a poskytuje úvod do problematiky časově proměnných sítí. Rovněž je popsán úvodní výzkum v této oblasti a náš koncept dopočítavaní konektivy mezi vrcholy sítí v sousedních časových hladinách.

klíčová slova: iso-povrch, extrakce, skalárni pole, časově proměnná sít̆

Acknowledgements

The author would like to thank to his supervisor Prof. Václav Skala, and his colleagues Martin Janda, Libor Váša, Petr Vaněček, Ivana Kolingerová and the others for numerous consultations and comments. The author would also like to thank Randy P. Hessel from Engine Research Center, University of Wisconsin-Madison, USA and to Jindřich Kňourek from CIV, University of West Bohemia, Czech Republic for providing data sets with dynamic simulation mesh. This work was supported by the projects VIRTUAL MSMT Czech Rep. No: 2C 06002, 3DTV FP6-2003-IST-2 Network of Excellence, No:511568, INTUITION FP6-2003-IST-2 Network of Excellence, No:507248-2.

Contents

Chapter 1

Introduction

This work focuses on visualization of time-varying data sets. In particular the problem of iso-surface extraction from the data sets with dynamic simulation mesh is addressed. Since introduction of the famous *Marching Cubes* algorithm [28] in 1987 a lot of research has been done in iso-surface extraction towards time-varying data sets. This work presents both the previous work and our contribution to the field.

1.1 Visualization, scientific visualization

'Scientific visualization': an application of computer graphics which is concerned with the presentation of potentially huge quantities of laboratory, simulation or abstract data to aid cognition, hypotheses building and reasoning.

http://www.wikipedia.org, March, 2007

Visualization in any form has always been a powerful tool for understanding information, be it an inventor's idea, explanation of solution to some problem or more recently the data from measurements or simulations. Ancient inventors used visualization to store and communicate their ideas. Recent advances in science and technology require new forms of visualization.

In the past decades *scientific visualization* was born and evolving into a valuable tool for visual exploration and understanding of scientific data. A proper visualization is often the fastest and the most intuitive way for scientists to gain insight into the large numeric data sets, which have to be explored and evaluated in order to accomplish their research.

1.2 Definitions

In order to avoid ambiguities in terminology a list of terms and meanings is presented in this sub-chapter.

Simulation mesh is a structure witch discretizes continuous simulation domain. Simulation mesh consists of interconnected vertices (point samples within a simulation domain), creating the bounded sub-spaces - *cells*. The terms *mesh* and *grid* are used interchangeably with the same meaning. In *static simulation mesh* geometry the mesh cells and their correspondence between adjacent time steps remains fixed during a simulation, while in the *dynamic mesh* the number of mesh cells and their geometry may vary along the time dimension.

Active cell is a mesh cell which satisfies condition: $min \le q \le max$, where min/max are minimum and maximum of all values associated with cell's vertices and q is a user-defined scalar value.

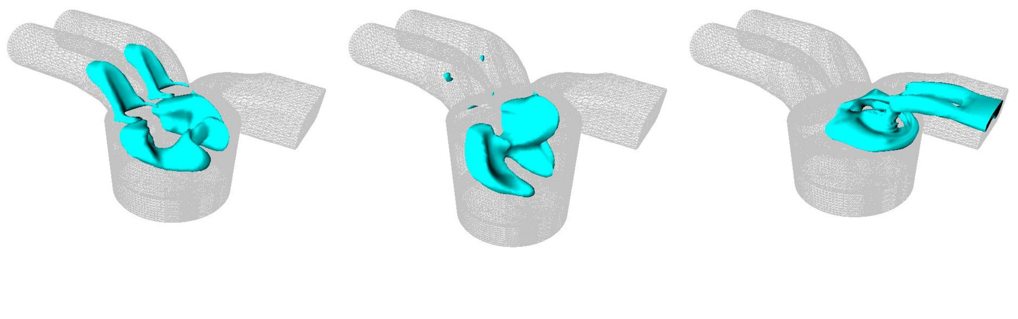
Iso-surface is a surface represented by the points of constant value (e.g. pressure, temperature, velocity), constructed within a volume of data. Mathematically, the iso-surface S for iso-value q is a surface:

$$S(q) = \{\mathbf{x} \mid F(\mathbf{x}) = q\}$$

where $\mathbf{x} \in S$ is a 3D point and $F(\mathbf{x})$ is a scalar function F: $R^3 \to R$ defined over 3D points. 2D version of iso-surface is iso-contour.

1.3 Iso-surface extraction from dynamic simulation meshes

Data sets consisting of multiple time steps are common in many scientific and engineering fields, including medicine, computational fluid dynamic or geology. Iso-surfaces proved to be a valuable tool for visual exploration of the static and time-varying data. Figure 1.1 shows the example of evolving iso-surface of a constant viscosity 0.017 inside an engine valve.



(a) time step 21 (b) time step 30 (c) time step 136

Figure 1.1: Iso-surface of total viscosity (iso-value=0.017) at the time steps 21, 30 and 136 from a time-varying data set of combustion process in an engine.

Even if the dynamic simulation meshes have many scientific and industrial applications [2, 17, 15] and the techniques for their generation are being rapidly developed [16], there is a lack of the suitable visualization techniques. This work focuses on the iso-surface extraction from the data sets with dynamic simulation mesh. This task includes many challenges:

- space efficient representation of dynamic simulation mesh,
- fast structures for active cell identification,
- large data sets handling.

Today the data sets with dynamic simulation mesh are treated as a set of stand-alone static data volumes defined at each discrete time step. Therefore, our aim is to develop data structures and methods able to exploit spatial and temporal coherence in data sets with dynamic mesh to reduce space and time requirements of iso-surface extraction process.

1.4 Organization

This work is divided into three parts. First part (chapter 2) provides an overview of techniques for visualization of scalar fields and their evolution over time. In the second part (chapter 3), the techniques for iso-surface extraction from data sets with static simulation mesh and time-varying scalar values are described.

Chapter 4 covers the main topic of this work - visualization techniques for data sets with dynamic simulation mesh. Introduction to the problematic of dynamic meshing is sketched together with description of the initial research done in this field. Finally our newly developed techniques contributing to this field are described followed by the proposal of future research.

Chapter 2

Visualization of scalar field structure and evolution

Often demanded task in scientific visualization is to show structure of a scalar field. Moreover, when data set covers state of the scalar field over certain period of time, it is often useful to visualize its evolution. An overview of the techniques dedicated to these two purposes is sketched in this chapter, starting with experimental visualization which served as an inspiration for computer aided visualization described in the second part of the chapter. Sections 2.2.1 to 2.2.3 focus on visualization of static scalar fields, followed by the *Chronovolumes* and *Illustration-inspired techniques* in the sections 2.2.4 and 2.2.5, which describe visualization of scalar field evolution.

2.1 Experimental visualization

Prior to the age of computers, the experimental techniques for visualization of fluid flow were widely used. Experimental fluid flow visualization techniques, categorized according to [35, 21] are described in this chapter.

1. *Addition of Foreign Material.* To visualize a fluid flow behavior over time, the small particles of foreign material can be injected into it. Their motion ruled by a fluid flow is then analyzed and evaluated. Techniques differ according to the type of foreign material and style in which it is added into flow. *Time lines* are formed by the row of small particles (such as hydrogen bubbles), released perpendicularly to the flow. *Streak line* arises when colored dye is injected into the flow from a fixed position over a period of time.

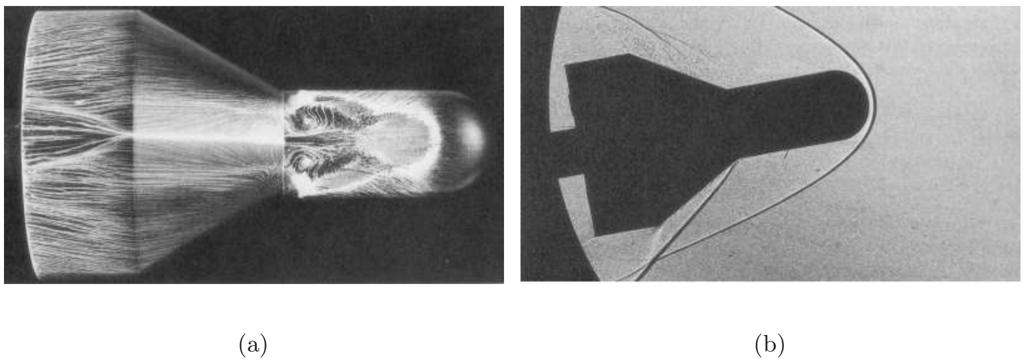


Figure 2.1: Experimental visualization of supersonic flow. (a) Oil flow patterns on the surface. (b) Shadowgraph of the density distribution. Both pictures are from [35].

2. *Optical Techniques.* This branch of experimental visualization techniques is based on the fact that the light reflection index is changing with varying density. Technique is applicable only in the flows with varying density.

3. *Addition of heat and energy.* The effect of adding heat into the flow field to artificially change local density is utilized. Then a structure of flow field can be visualized using optical techniques.

Final picture made by the experimental visualization techniques shows structure of the flow in the single moment, enhancing places with higher and lower value of an observed quantity. Computer aided techniques, described in the next chapter 2.2, are more-or-less trying to find the ways to construct and visualize structures over the scalar field to approximate the pictures produced by the experimental techniques.

2.2 Computer aided visualization

Inspired by the experimental visualization and supported by the recent advances in graphics hardware, a computer aided visualization of scalar fields evolution is quickly developing in the past decades. This section gives and overview of the commonly used graphics techniques to visualize a structure and evolution of scalar field. Different branch of techniques such as *particle tracing*, *stream lines*, or *streamlets* are used for vector and tensor fields visualization [22, 37, 25], however they are out of scope and interest of this work.

2.2.1 Direct volume rendering

Probably the simplest technique to visualize a scalar field is to render a scalar field directly by applying *transfer function.* Transfer function maps the optical properties like color and opacity to the values in data set. During the visualization no geometric information need to be calculated, because 3D data are directly used to render the resulting image [26] of a scalar field.

The first step in Direct volume rendering is the *reconstruction process.* Given a set of original discrete samples in the dataset, a ray is cast from eye position through the each pixel of the resulting image and further through a data volume. A ray is sampled by a constant rate that has to be high enough to visualize structure of a scalar field properly. Value of each single sample along the ray is computed by the *reconstruction filter* [29, 30].

After reconstruction a transfer function is designed and applied onto computed values of sampled rays. This is called *classification step.* Finally the casted rays are integrated along themselves using volume rendering integral:

$$C = \int_0^D c(s(\vec{x}(t)))e^{-\int_0^t \tau(s(\vec{x}(t')))dt'} dt \tag{2.1}$$

Integral 2.1 incorporates the parts of *emission-absorption optical model*, in which C is the pixel output color, $\vec{x}(t)$ denotes a ray casted into a volume (t is the distance to eye), scalar value corresponding to the position is denoted by $s(\vec{x}(t))$, $\tau(s(\vec{x}(t')))$ is the *light absorption coefficient* and $c(s(\vec{x}(t)))$ is the light emitted by the particles.

Research on volume rendering is now focused on interactivity and utilization of advanced graphics hardware to increase the efficiency of the rendering process [19, 18]. Volume rendering of large unstructured meshes is discussed in [7]. Volume rendering of time-varying data is discussed in [5, 61].

2.2.2 2D slicing and Orthogonal slicing

2D slicing [50] investigates the structure of scalar field by cutting it in the desired position by the plane, usually aligned with some of the basis axis. Information about the scalar values along the cutting plane may be visualized by the direct their direct mapping of the scalar data on the cutting plane or illustrated by the extraction of iso-contours.

Orthogonal slicing is the technique applicable to 3D scalar fields. It has been developed as an extension of the single 2D slicing of 3D scalar field. A set of mutually orthogonal slices is constructed, each of which shows the structure of a scalar field in the desired plane and position. Such set of slices gives better insight into structure of a scalar field than a single parallel 2D slice.

2.2.3 Iso-contours and iso-surfaces

This section provides overview of the methods for extraction of iso-surface geometry from static data sets (data specified only at the single moment in time). Extraction of iso-surface geometry is not the main focus of this work. However, it is an integral part of the iso-surface extraction and there we discuss the topic in this chapter. Note that this is not a comprehensive list of all methods. The techniques are sorted in categories according to the principle they are based on. The example methods are mentioned by each category.

Marching methods. The initial work on extraction of iso-surface geometry was presented by Lorensen and Cline in 1987. Their *Marching Cubes* (MC) algorithm [28] works over regular grids and is easy to implement. Its limitations lie mainly in the high time complexity, since all the cells in a mesh have to be traversed. Another drawback of the MC method is the possible ambiguity in triangulation when reconstructing part of an iso-surface within a single active cell. These limitations were the main boost for the next research in this field.

Marching Tetrahedra (MT) [31, 51] approach works over more general case of tetrahedral meshes. Since any mesh can be decomposed into tetrahedral one this in generally not considered as a limitation. As in the case of MC algorithm, an iso-surface is approximated by triangles extracted by the linear interpolation of the values in the vertices of a simulation mesh. MT solves the triangulation ambiguity problem of the MC method. The two possible cases of triangulation within a single tetrahedron are depicted by the figure 2.2.

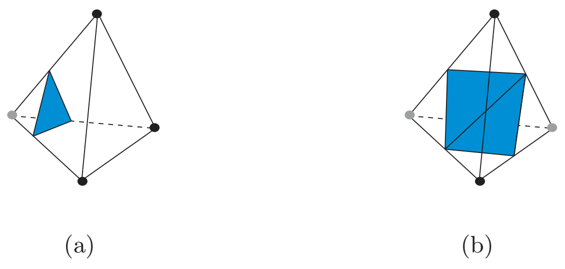

(a) (b)

Figure 2.2: **Marching tetrahedra.** The two cases where triangulation is required, and the resulting triangulations. Case (b) can be rendered as a quadrilateral rather than two triangles, since the surface is guaranteed to be planar [51].

Dual contouring method [24] address the *crack problem* of MC method and capture the sharp features of the iso-surfaces. Hermite data along the edges (exact intersection points and normals) are assumed to be known prior to iso-surface extraction process.

In 2005 Anderson et. al introduced *Marching Diamonds* [3] (MD) technique for unstructured meshes. Rather that linearly interpolate along the edges of a cell, the pieces of iso-surface are computed from diamond composed of two neighboring tetrahedra (or triangles in 2D). Diamond provides more information about the surrounding data (more vertices) than a single tetrahedral cell. Result are visually smoother iso-surfaces with significantly lower amount of generated triangles when compared to MT method.

The methods for effectively identifying only those cells which are intersected by the iso-surface (active cells) have been in focus of many research. Fast identification of the active cells prior to the iso-surface extraction itself accelerates the whole process, because non-active cells are not investigated. An overview of the techniques for active cells extraction is given in the chapter 3.1.

Ray-traced iso-surfaces. Ray-tracing can be used for iso-surface visualization, when proper *transfer function* is used [54]. Sramek [47] defined ray-iso-surface intersection as the solution of system of equations:

$$X = A + t\vec{u} \tag{2.2}$$

$$F(x, y, z, \rho_{ijk}) = t_1, \tag{2.3}$$

where A is the eye position, $\vec{u}$ is the ray direction vector, t_1 is a threshold value, F is an interpolation function and ρ_{ijk} is a function value in some neighborhood of voxel v_{ijk}. Sramek discusses various interpolation functions F.

Parker et al. [32] optimized ray-tracing for iso-surface extraction by grouping cells into macro-cells, marked with minimum and maximum sample value within them. Only the macro-cells and their child cells that contains the desired iso-value are recursively traversed and considered for ray-iso-surface intersection computation. If more than one ray-iso-surface intersection is computed for one ray, the one closest to the eye is considered for visualization.

Point-based iso-surfacing. In 2004 Co et al. [39] proposed meshless method for construction of crack-free iso-surfaces from *multiblock data*. Such data sets involve multiple grids of various layout and position, often overlapping each other. Their method disregards connectivity information in the overlapping grids. Next the single samples are binded to a new grid G. A local RBF interpolant $H_k(x, y, z) = \sum_{j=1}^{N} a_j B_j(x, y, z)$, is associated to each RBF cell's center c_k of grid G, computed from N samples within sphere of radius $R_w = 2\sqrt{s_x{}^2 + s_y{}^2 + s_z{}^2}$ from a cell's center, where $(s_x,\ s_y,\ s_z)$ are dimensions of single cell in RBF grid G. a_j is the *blending coefficient* and B_j is a multi-quadric function. Iso-surface is then generated from triangles constructed within the cells of original grids, using MC algorithm. These triangles are then sampled and function value *F(x,y,z)* and gradient are computed for each sample using interpolants H_k of the RBF cell a sample belongs to, and from its 26 neighbors by the *partition of unity* function ϕ_k $(\sum_k \phi_k(x, y, z) \equiv 1)$: $F(x, y, z) = \sum_{k=1}^{M} \phi_k(x, y, z) H_k(x, y, z)$

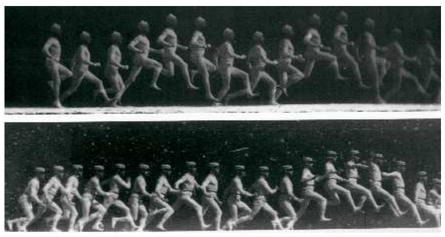
Finally, function values of the samples and associated gradients are used as the input parameters of the Newton-Raphson root-finding scheme to move the samples onto the real iso-surface of interest. So the resulting iso-surface is not just approximated by the triangles.

Co and Joy [14] used the similar RBF interpolation scheme for iso-surface generation from large scattered data set. The method constructs a small local tetrahedralization around the place at which the iso-surface is constructed. Inside each local tetrahedralization a the products of RBF interpolation and point-based rendering method is used to extract geometry of iso-surface. This allows to process large-scale data sets because once the resulting iso-surface is extracted, local tetrahedralization can be discarded.

2.2.4 Chronovolumes

The challenge of capturing the objects that vary in time on a single 2D picture, is well known among photographers for almost 100 years. It is known as *Chronophotography*. Woodring and Shen [60] combined chronophotography and direct volume rendering techniques (chapter 2.2.1) to visualize evolution of the scalar data.

The resulting visualized objects are constructed by first integrating the four dimensional voxel data (x, y, z, t) along the time axe. After this, each voxel in the processed data has value $\mathbf{v}(x, y, z, t)$ from integrating along the time. Such a preprocessed volume is called *chronovolume*. The regular volume rendering methods are then used to obtain final two dimensional image of a scalar field evolution. Special care is taken to the construction of integration function (chapter 2.2.1), which has to reflect the goal of visualization. Woodring and Shen proposed [60] five different integration function (*Alpha compositing*, *First Temporal Hit*, *Additive Colors*, *Minimum/Maximum Intensity* and *Edge Enhancements*), each of which enhances different temporal feature.


(a)

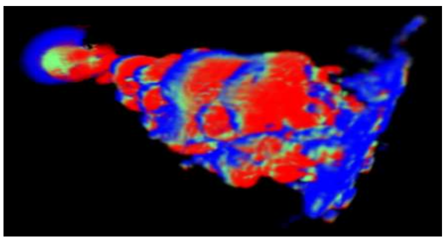

(b)

Figure 2.3: (a) Example of chronophotography. (b) Alpha compositing used over chronovolume for 3 consecutive time steps (time is moving from red to blue) in the visualization of Jet dataset [60].

2.2.5 Illustration-inspired techniques

Joshi and Rheingans introduced illustration-inspired approach [23], which shows the temporal changes in the scalar data over a period of time. This is done by first extracting the features of interest from the investigated scalar field and track them over a sequence of n consecutive time steps. Finally the *illustration features* are computed and added into the image, to indicate the temporal changes of a feature (i.e. positional, rotational). Figure 2.4 shows the *illustration features* giving an impression of object movement during a period of time.

Four illustration features (*Speed lines, Flow ribbons, Opacity modulation and Strobe silhouettes*) are described in [23] (Fig. 2.4). Their suitability for description of a specific temporal change of observed feature is also discussed.

Usage and suitability of the illustration-inspired techniques depends highly on the visualized data and on a nature and intensity of the involved temporal changes. Authors also discuss usability of this approach for flow visualization.

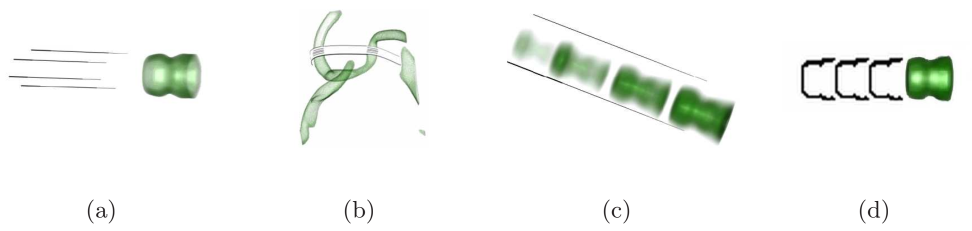


Figure 2.4: Illustration features added to feature extracted from scalar time-varying field to indicate its temporal changes [23]. (a) Speed lines, (b) Flow ribbons, (c) Opacity modulation, (d) Strobe silhouettes.

Having described the most used techniques for visualization of scalar field structure and evolution the cornerstone for scalar field visualization is put down. The following chapter is focused on extraction of the active cells from time-varying data sets with static simulation mesh.

Chapter 3

Iso-surface extraction from static simulation mesh

In 1987 Lorensen and Cline introduced popular *Marching Cubes* algorithm [28] for iso-surface extraction from static volumetric data sets. Many researchers inspired by this seminal work tried to improve and extend it in many ways.

This chapter provides an overview of the methods for:

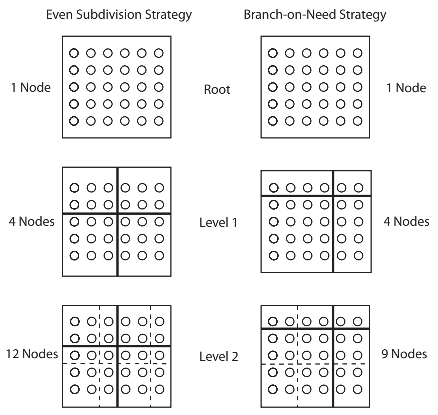
1. iso-surface extraction from static data sets,
2. iso-surface extraction from time-varying data sets with static simulation mesh.

Even if the this family of methods and data sets is not a core topic of this work, their overview is provided because they serve as a start point for the research on iso-surface extraction from the data sets with dynamic simulation mesh.

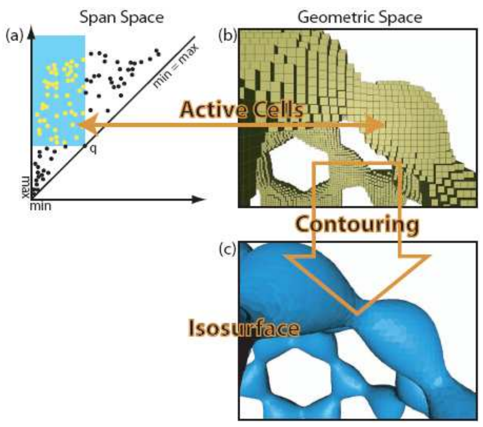
3.1 Efficient iso-surface extraction from static data

The iso-surface extraction has been greatly popularized by the *Marching Cubes* [28] algorithm. Despite its simplicity, the Marching Cubes algorithm suffers from computational inefficiency, since all the cells in a data set have to be traversed in order to extract desired iso-surface. Rather than traversing all the cells in a dataset, a family of techniques for fast active cells identification was developed.

Wilhelm and van Gelder introduced a ***Branch-On-Need Octree*** (BONO) [59]. The technique is inspired by the traditional octree, but spatial subdivision is more efficient than in the standard octree. done is a different manner. Space of the BONO node is divided by assigning the-largest-possible-power-of-two rows and columns to a lower branch and the rest of the subspace is divided among the



(a)



(b)

Figure 3.1: (a) 2D example of BONO subdivision. The largest possible power of two number of rows and columns is assigned to a lower left branch, and the rest of the space is divided among the other quadrants (picture and description from [59]). (b) Principle of span space usage for fast identification of the active cells (picture from [55]).

other three "quadrants" (in 2D). Figure 3.1a shows the 2D example of BONO subdivision of regularly placed cells. *Min/max* values of the cells are stored in the nodes of a BONO tree, so only those subtrees satisfying condition min < iso-value < max are traversed during active cells extraction. The time complexity of this method is *O(K + log N/K)*, where *K* is the number of iso-surface cells, and *N* is the total number of cells.

Shen and Johnson introduced ***Sweeping Simplices*** technique [43]. The technique uses a cell list division scheme, which assigns the cells into the groups at various level according to the *min/max* values. For each subgroup the minimum list (cells sorted by the *min* values) and the corresponding sweeping list (cells sorted by the *max* values) is constructed. During the active cells extraction, the minimum lists and sweeping lists are restricted by the selected iso-value. Then the efficient comparison scheme is used to quickly extract only the cells in both restricted lists, which are in fact the active cells.

Livnat et al. introduced the notion of ***Span Space*** [27]. In the Span Space the cells are represented by their minimum (x-coordinate) and maximum values (y-coordinate). Active cells are identified by the simple restriction of Span Space along min and max axis. Figure 3.1b illustrates the process of iso-surface extraction using Span Space (the area of active cells for iso-value q is enhanced).

Livnat et al. originally proposed kd-tree for spatial subdivision of Span Space in the ***NOISE*** algorithm [27]. The points, representing the active cells can be extracted by simply traversing kd-tree with worst case time-complexity of $O(\sqrt{N} + K)$.

Shen et al. proposed lattice subdivision of Span Space in the ***ISSUE*** technique [42]. ISSUE lowers the time complexity of the active cells search to $O(log(\frac{N}{L}) + \frac{\sqrt{N}}{L} + K))$, where L is the user specified parameter defining the number of lattice elements along each axis.

The ***Interval Tree*** technique [12] guarantees worst-case optimal efficiency. Cells, represented by the min/max intervals are grouped at the nodes of balanced binary tree. For any iso-value query, at most one path from a root node down to a leaf node is traversed.

I/O efficient technique for visualization of iso-surfaces from time-varying data sets which exceed limits of physical memory has been presented by Chiang [11]. The I/O efficient decomposition of interval tree is used and combined with an out-of-core approach to load active cells' geometry and scalar values.

Another solution has been found in work on ***Contour trees*** [9], and construction of *Seed Set* [8, 52] containing cells intersected by the iso-surface components (one seed per component). The iso-surface construction begins at a seed and is traced through neighboring cells using adjacency information.

Span-triangle data structure was introduced by von Rymon-Lipinski et al. [53]. Span-triangle is focused on iso-surface extraction from medical volume data sets, sampled as either 8 bits or 16 bits integers, moreover cropped to an exploration range $[e_{min},\ e_{max}]$. Skeleton of Span-triangle structure is base array. Each element in the base array corresponds to the cells x with min value $b = x_{min}\ -\ e_{min}$ and contains pointer to its span array and cell info array. An element of span array at position s holds pointer to the first item with value span $s = x_{max}\ -\ x_{min}$ in a cell info array. Figure 3.2 shows example of Span-triangle structure for exploration range $[e_{min},\ e_{max}] = [0,\ 3]$ and iso-value $v\ =\ 2$. Using fast Radix sort algorithm, the structure can be constructed in $O(\mathrm{N})$ time and takes $O(\mathrm{N})$ space.

Bordoloi and Shen [6] introduced the space efficient technique for iso-surface extraction from large scientific data sets. The original mesh cells are first represented as the points in 2D ***UV-space***. U coordinate of the cell C is determined as $u = C_{maximum} + C_{minimum}$ and V coordinate as $v = C_{maximum}$ - $C_{minimum}$. Then the U and V axis are quantized, which results into a set of rectangular regions within UV-space. Given an iso-value, the decision boundaries for UV-space are computed. The cells stored in the rectangular regions within such decision boundaries are selected for extraction of iso-surface geometry. Due to the quantization of the U and V axis the min/max values of the cells within each rectangular region of UV-space do not need to be saved. Only the space for cells *ID*s and the space for representing rectangular regions in UV-space are required.

Waters et al. [56] use ***fixed-sized buckets*** to divide the list of cells in a scalar field. A record for one cell includes a cell's *min/max* and *ID* values. List of the cells is first sorted according to the minimum values. Such sorted list is then split into intervals of the same size B - buckets. Figure 3.3 illustrates organization of the cells in the *min/max* space and their assignment to the buckets. Similarly to Span Space the active cells are identified by restriction of the 2D space along the *min* (x) and *max* (y) axis. Buckets are traversed along the *min* (x) axe and tested for $y_{cell} = max_{cell} < max$ condition. The time complexity for creating the list is $O(N\ logN + \frac{N}{B}(B\ logB))$ and for identifying the active cells $O(K + B)$. Space complexity is $O(N)$.

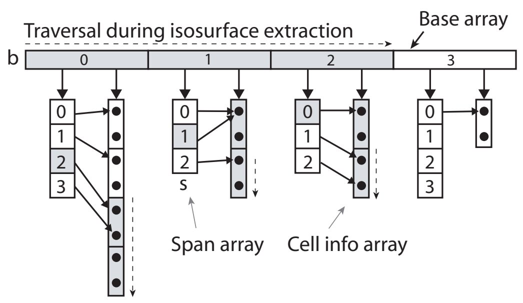


Figure 3.2: Sample of the span-triangle data structure for exploration range $[e_{min},\ e_{max}] = [0,\ 3]$ and isovalue $v\ =\ 2$. Active element of the data structure for isovalue $v\ =\ 2$ are emphasized by a gray background.

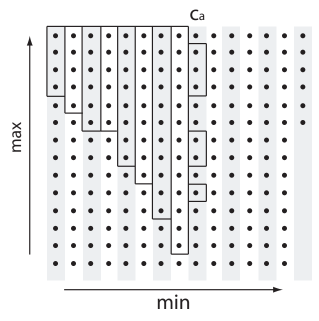


Figure 3.3: Iso-surface extraction using fixed-sized buckets. The cells of scalar field are organized in a space similar to Span space and divided into the buckets of equal size. Parts of the buckets containing the active cells are emphasized by the black boxes. The cells in the last bucket c_a must be checked for both the minimum and maximum value, therefore creating non-continuous block of cells.

3.2 Temporal Hierarchical Index Tree

Shen et al. introduced *Temporal Hierarchical Index Tree* (THIT) [40] for iso-surface extraction from time-varying scalar fields with static simulation mesh. First the cells are assessed by their temporal variation. For a time-varying field, a cell may has multiple corresponding points in the Span space for the different time steps. To characterize a cell's scalar variation over time, the area over which the corresponding points spread in the Span space is measured.

Then a Temporal Hierarchical Index Tree is built, placing the cells with low temporal variation over time closer to the root of the tree. Criterion for low temporal variation is that the points corresponding to the same cell are located with 2x2 elements of lattice subdivision of the Span space. For other cells that do not satisfy the criterion of low temporal variation the root time interval is divided in half. Process continues recursively into each of two subtrees. The leaf nodes contain cells with the highest scalar variation over time, so that the cells' time-specific extreme values are used. Search index for each node of the THI tree is created by the ISSUE [42] algorithm. The min/max values in the nodes closer to the root node are used to refer to a cell for more than one time step, which contributes to the lower overall size of the tree structure.

Given an iso-surface query at time step t, THI tree is traversed and the nodes that contain the active cells are visited. In the visited nodes the ISSUE search for the active cells is performed.

This method accelerates iso-surface extraction from time-varying data. However at each time step the entire data domain (time step) is loaded into the main memory. The iso-surface extraction process potentially needs to access all of the time steps in the time-varying dataset, which may cause memory overhead.

3.3 Temporal Branch-on-Need Tree (T-BON)

The *Temporal Branch-on-Need Tree (T-BON)* [48] extends the three dimensional Branch-On-Need Octree (BONO) [59] (figure 3.1) for time-varying iso-surface extraction. The method focuses on minimizing the number of I/O operations, by reading from disk only those portion of search structure and data necessary to construct the current iso-surface.

First a BONO tree [59] is computed for each time step. Information about general infrastructure of a BONO tree is saved to the disk once for an entire dataset. Then the extreme values for nodes are computed and stored separately for each time step.

During iso-surface extraction a query of the form (*iso-value, time*) is processed by first testing if the iso-range of the root node of BONO tree for desired *time* cover the desired iso-value. If so, root node's children are tested for iso-range as well. Once the leaf node is hit and covers the desired iso-value then the disk block containing the data for the cells in that leaf node is added to the list. Once this process is over, all block in the list are read from the disk. This clearly minimizes the I/O access. However, T-BON does not exploit any temporal coherence in the data.

3.4 Time-space partitioning tree

Time-space partitioning tree (TSP) [41] is the octree-based data structure, efficiently encoding coefficients of spatial and temporal variation of the data to speed up visualization of time-varying data. TSP tree is a standard full octree. Each node of TSP tree stores a binary tree, recursively bisecting time-span [0, t] of the entire data set until the unit time step is reached. For each node of binary time-tree associated with each TSP tree node the coefficients of *spatial* and *temporal variations* are computed from the original time-varying data.

During the volume-rendering-based visualization the partial images from sub-volumes are cached for the current time step. When another time step is selected a TSP tree is traversed. The test whether particular sub-volume's spatial and temporal variations exceed the user selected thresholds is performed, in case of which this sub-volume is rendered again. Otherwise a cached rendered partial image is used, speeding up the visualization process.

TSP enables user to manage the trade-off between the visualization speed and accuracy through the thresholds of spatial and temporal coherence. Similar error-based iso-surfacing method [55] is described later in the chapter 3.7.

3.5 4D approach

In 1996 Weigle and Banks introduced *recursive contour meshing* method [57] of decomposing n-dimensional simplex into a set of (n-*1*)-dimensional simplices, including the 4D case. Based on this work, two years later, they published the method [58] for iso-surface extraction from time-varying scalar fields.

Basic idea of their 4D method [58] is to look at the time-varying data as to be the one static 4D dataset. Each sample of the original dataset is identified by three spatial coordinates x, y, z and one time coordinate t. This interpretation of 3D time-varying data together with the usual way of extracting iso-surfaces from the mesh composed of n-dimensional simplices (over n-dimensional data) lead them to construction of a mesh of 4-simplices. To find an iso-surface $f(x,y,z,t) = 0$, they apply two constraints to the 4-simplicial mesh.

In the first pass a set of *iso-volumes* $f_1(x,y,z,t) = 0$ are extracted by imposing *iso-value constraint* over the 4-simplices. This results in a set of 3-simplices intersected by the iso-surfaces of given iso-value.

To produce iso-surface for the particular time step c_t the second pass is done, imposing the *time constraint* $f_2(x,y,z,t) = t - c_t = 0$. This produces the resulting iso-surface, composed of 2-simplices, for desired iso-value and time. Although this method elegantly captures temporal coherence in a dataset, its high computational demands makes it impractical for the large data sets.

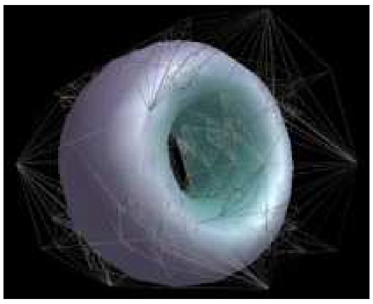


Figure 3.4: Iso-surface satisfying 2 constraints in 4-dimensional space. 3-simplexes (wire frames) are the result of imposing first (*iso-value*) constraint. Extracted iso-surface satisfies the second (*time*) constraint (picture from [58]).

3.6 Out-of-core visualization

Recent advances in computational performance enable scientists to perform large-scale simulations, producing data sets which are usually a magnitude larger than is a size of physical memory of workstation-class computers. Thus, various *out-of-core* visualization approaches have been proposed. Principle of the out-of-core approaches is to split original data set into a range of files and using only those of them needed for visual analysis. One of the main issues solved when using out-of-core approach is to minimize the number of disk I/O operations necessary to access required data. This concept is usually used as the extension of the existing methods, enabling them to work with large-scale data set.

As an example of the out-of-core method the work of Reinhard et al. [36] is briefly described. Their method is focused on fast out-of-core iso-surface visualization. The method first partitions the whole data set into a range of small files. Each file contains the data for one time step and a certain range of iso-values. Efficient 28 bytes-per-voxel data format is used to store time-varying data in the files, reducing the necessary disk-memory traffic. For visualization of the iso-surfaces they use ray-tracing based method of Parker et al. [32]. The technique has been implemented and tested on 32 processors SGI Origin 2000 computer, with 12 GB memory, which together with out-of-core nature of the technique allows visualization of the iso-surfaces at the interactive frame rates.

3.7 Adaptive extraction of time-varying iso-surfaces

Solution to the problem of fast iso-surface extraction from large time-varying data sets proposed by Gregorski et al. [20], utilizes adaptive mesh refinement scheme and an out-of-core approach (chapter 3.6). The key of the adaptive extraction method is the recursive mesh refinement scheme which splits initial volume into a set of tetrahedra. These are further grouped into the diamonds organized in a hierarchical structure.

During the preprocessing phase an iso-surface approximation error, minimum and maximum data values enclosed by each diamond and normalized gradient vector at each data point are computed. At runtime, the refinement process creates a set of tetrahedra, describing domain around the iso-surface, based on user supplied iso-surface approximation error, iso-value and time step.

If another time step is selected the visible diamonds' *min*, *max* and *error* tolerance values are checked. If these fit the user defined threshold values, then all the tetrahedra within such diamonds are used for iso-surface extraction. Otherwise, the refinement process (sequence of *split* and *merge* operations) is initiated.

When the iso-value is changed then the refinement process can start either from the root diamond of a diamond hierarchy or from a current refinement of a mesh, checking the min, max and error tolerance values of the diamonds. The way in which the refinement process is initiated depends on the data difference between time steps t_i and t_{i+1}.

3.8 Difference intervals

Waters et al. introduced the *Difference intervals* technique [55] for better utilization of spatial and temporal coherence in a data set. Static mesh with time-varying scalar values is assumed. Input of the method are desired iso-value and time step. Iso-surface evolution is visualized by first computing the active and inactive cells from an entire data set (or only from a selected time span) and then visualized using this information.

In a preprocessing time a set of active cells is first computed for the first time step, using an arbitrary existing technique. Active / inactive cells are then computed from this initial information for the next time step. Movement of cell in the *span space* [27] is observed and classified into 11 cases (figure 3.5), which results into designation of each cell with either *add* (cell became active) or *remove* (cell became inactive) operation. Eventually, a cell can holds its status in a next time step. The changes between t_i and t_{i+1} are encoded into an operation set D_i. In a preprocessing step the operation sets are computed for all the adjacent pairs of time steps. The technique for temporal compression of a sequence of operation sets is also discussed.

Visualization utilizes the encoded operation sets to update only the necessary portions of the iso-surfaces according to the *add* or *remove* operation assigned to each single cell. In this way an interactive playback of iso-surface evolution of constant iso-value q is possible in both forward and backward direction.

A point-based rendering technique is utilized for visualization of the iso-surfaces itself. Once the desired iso-value is changed the preprocessing steps has to be done with this new parameter again.

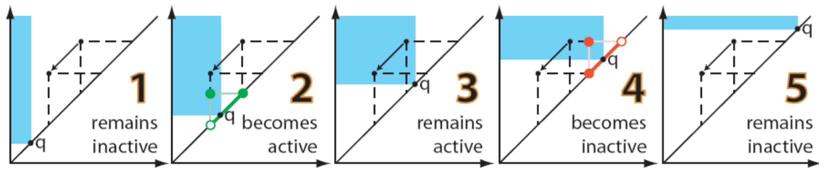


Figure 3.5: Examples of the *add* and *remove* operations assigned to a cell according to its movement in Span space. These operations are then encoded in the operation sets and used for interactive playback of the evolving iso-surfaces for a given iso-value (pictures from [55]).

3.9 Persistent hyperoctree (PHOT)

Persistent hyperoctree (PHOT) [44] introduced by Shi and allows to extract a set of the active cells and simultaneously only those of them that are visible from the current point of view (relevant cells). By saving the time to extract occluded cells and more efficient tree construction, significant speed up in iso-surface extraction is achieved.

By deleting the nodes which contain only the inactive cells in their sub-volume and collapse the nodes that contain only one children in an octree a *compact octree* is made (figure 3.6). Each PHOT's node has 8+k *jumpers* (k is a small constant) with associated version number. Root node of hyperoctree represents an entire hyper volume and has 16 children, each representing one hyperoctant. To make hyperoctree persistent all the cells are collected and two copies are held for each of them. One copy holds a cell's *min* value as its key and the other one holds the *max* value. The cells are sorted according to their key values in ascending order and the list is traversed. If a cell has its *min* value as its key it is stored in the current version of the tree, otherwise the new version is created and the cell is stored into it (figure 3.7). When traversing a PHOT, the root node with the largest version number smaller or equal to iso-value is identified and its subtree is traversed by following the latest jumpers no later than iso-values.

For 3D iso-surface extraction by 4D slicing along the time axis a 4D cells are constructed and indexed by their value ranges using PHOT. Given an iso-value and time parameters the active and relevant cells are extracted, and then sliced along the time axis to determine the iso-surface at given time.

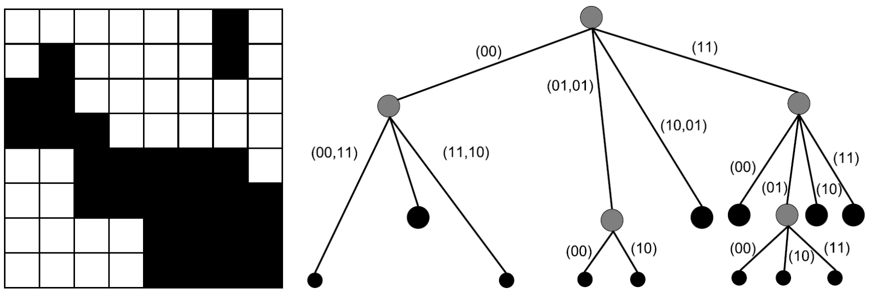


Figure 3.6: Compact hyperoctree for the set of active cells (the black ones on the left sub-figure). As can be seen from the right sub-figure the nodes which contain only one children are collapsed (picture from [44]).

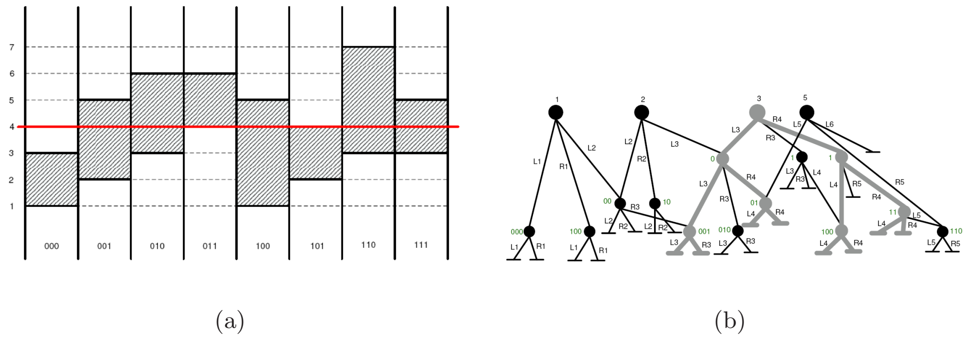
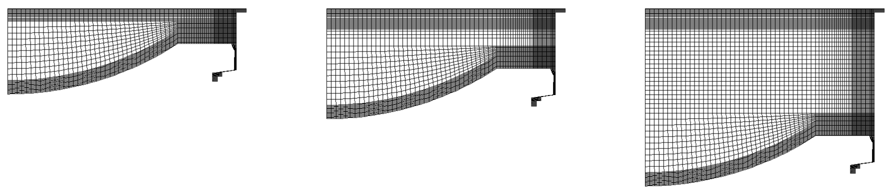


Figure 3.7: Example of PHOT for static 1D data (picture from [44]).

Chapter 4

Iso-surface extraction from dynamic simulation mesh

The methods presented in the chapter 3 assume, that a simulation mesh remains static during the course of simulation. So, geometry of mesh cells and their correspondence between time steps remains constant. However, simulation domain may change from time step to time step, because of the moving parts or dynamic boundaries. In such a case the *dynamic meshing* techniques are often employed to maintain quality of a simulation mesh under deforming domain boundaries. Figure 4.1 shows dynamic mesh from a simulation of combustion process in an engine. The mesh changes its layout (number of cells and their geometry) according to the vertical position of a moving piston. Dynamic mesh provides better discretization of changing simulation domain and positively influences overall accuracy of the solution.



(a) time step 400 (b) time step 420 (c) time step 440

Figure 4.1: Example of dynamic mesh from a simulation of combustion process in an engine. The simulation mesh is reconstructed according to the vertical position of a moving piston. Layout of the simulation mesh is depicted for the time steps 400, 420 and 440.

While the techniques for generating dynamic meshes are being rapidly developed, there is a lack of suitable visualization methods. Treating each time step as a separate static subset of data is common in visual exploration of the data sets with dynamic mesh. Therefore, this work is focused on development of the visualization techniques capable of interactive visual exploration of data sets with dynamic simulation mesh.

The rest of this chapter splits into two logical parts. The first part (chapters 4.1 to 4.2) provides brief introduction to the theory and application of dynamic meshes. The second part (chapter 4.3) describes existing visualization techniques for dynamic simulation mesh, followed by our concept of iso-surface extraction from this specific kind of data sets.

4.1 Introduction to dynamic meshing

Principle of dynamic meshing lies in the conditional update of generated initial simulation mesh at each time step. Criteria and conditions of mesh quality and mesh update depend strongly on particular application.

There are many kinds of conditional mesh update covered by the term *dynamic meshing*. In the simulations where the zones of frequent changes in data values are moving, the fixed uniform grids are computationally inefficient. Therefore, solution adaptive grids are often employed, refining the mesh right at the place of the high data changes. This is called *mesh adaptation* [1].

Another kind of dynamic meshing are the *moving meshes*. Moving mesh techniques use a fixed number of grid points, and let the grid points move in space or move entire grid. Thompson et al. [49] provides a good survey of the area.

The most general case is to rebuild the whole mesh at each time step, which is usually termed as the *dynamic re-meshing*. This kind of mesh update is used in simulations with rapidly moving domain boundaries (figure 4.1). Dynamic re-meshing changes both geometry and the number of cells during mesh update.

No matter which one of the techniques described above is used to update simulation mesh, the techniques for iso-surface extraction from the data sets with static mesh (chapter 3) become inefficient as the mesh deformation grows. In the case of *dynamic re-meshing*, when the number of cells vary in time, these techniques fail completely.

The following two sub-chapters deal with methods for dynamic mesh update to better illustrate the problematic and to give an impression of how dynamic re-meshing works. The first sub-chapter describes the group of methods commonly known as *Arbitrary Lagrangian-Eulerian*. The second one describes three frequently used re-meshing techniques: *Layering*, *Spring smoothing* and *Local re-meshing*.

4.1.1 Arbitrary Lagrangian-Eulerian (ALE) methods

In the following paragraphs the term *continuum* stands for a moving fluid or other material volume.

In the CFD applications the mesh generation process begins by choosing the right kinematical description of moving and deforming continuum. This description essentially determines the relationship between a continuum and a mesh [16]. There are two classical descriptions of motion:

- *Lagrangian description*, in which each node of simulation mesh is attached to a particle in a continuum. During simulation a mesh follows the continuum in its motion because mesh nodes remains attached to the same particles. The main drawback of this method is its inability to follow large deformations of simulation domain without frequent re-meshing.

- *Eulerian description* lefts a mesh fixed during the whole simulation. The physical quantities associated with particles moving through the fixed region of space are examined. A particular value of observed quantity at a mesh node at time t corresponds to the value of observed quantity at node position in time t. Eulerian method handle the simulation domain distortions relatively easy, but at the cost of lower resolution of flow details.

There are situations that would be difficult to analyze in either the Lagrangian reference frame or the Eulerian reference frame individually. *Arbitrary Lagrangian-Eulerian* (ALE) description of continuum movement combines the best features from both Eulerian and Lagrangian description. Figure 4.2 shows example of mesh nodes and particles movement in Eulerian, Lagrangian and ALE description. In the ALE description the nodes of simulation mesh may fully or partially follow a movement of continuum or may be fixed like in the Eulerian description. The movement of nodes in ALE description offer higher ability to handle the mesh distortions caused by large deformations or movement of simulation domain boundaries.

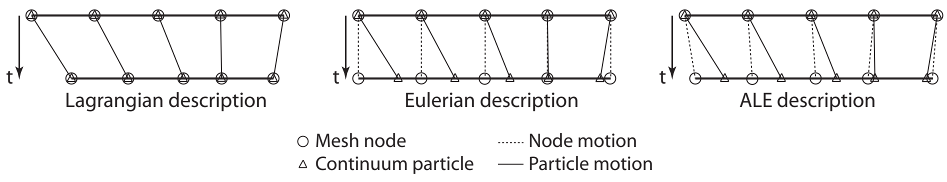


Figure 4.2: 1D example of the Lagrangian, Eulerian and ALE mesh update with respect to the actual particle motion (picture from [16]).

4.1.2 Dynamic mesh models

Layering. Hexahedral meshes can be update by dynamically adding the layers of mesh cells near the moving boundaries (figure 4.3). Layer of the cell adjacent to the simulation domain boundaries (layer j in figure 4.3) can be split or merged with its adjacent layer (layer i in figure 4.3). When the layer j grows, its height h is checked by the ideal cell height h_{ideal} condition [17]:

$$h < (1+\alpha)h_{ideal} \tag{4.1}$$

If cell height h does not meet condition 4.1, the layer j is split according to the user specified criteria and the new layer is build upon j. The mesh from figure 4.1 has been created by the layering method.

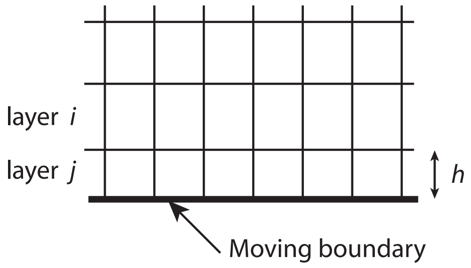


Figure 4.3: *Layering*. Layer j grows on the layer i according to the movement of domain boundaries. Height h of layer j is periodically checked until it fails to meet condition 4.1 - then the layer j is split (picture from [49]).

Spring smoothing. Spring smoothing idealizes the connections between mesh vertices as a network of interconnected springs. The initial connection of the mesh nodes is considered as a stable state of the mesh. Deformation of the mesh boundaries causes displacement of the vertices by the boundaries and generates tension on the springs. The tension on the springs connected to i-th internal mesh vertex is expressed as a force $\vec{F_i}$ in the i-th vertex:

$$\vec{F_i} = \sum_{j}^{n_i} k_{ij}(\delta\vec{x_j} - \delta\vec{x_i}) \tag{4.2}$$

where n_i is the number of vertices connected to vertex i, $\delta\vec{x_j}$ is a displacement of neighbor n_j against the vertex x_i, k_{ij} is the spring constant (Equation 4.3). So, the resulting force in the i-th vertex is proportional to the displacement along the springs connected to the node i [17]. The position of the i-th vertex is then adjusted according to the vector and magnitude of the $\vec{F_i}$. This process of *force-relaxation* on the mesh springs is applied to all internal mesh vertices.

$$k_{ij} = \frac{1}{\sqrt{\vec{x_i} - \vec{x_j}}} \tag{4.3}$$

Local re-meshing. When the displacement of the mesh boundaries is large compared to the cell size, degeneration of the mesh cells can be significant. Local re-meshing methods measure the quality of the cell shape (e.g. skewness, min/max length) and update mesh locally when it decreases under the predefined thresholds.

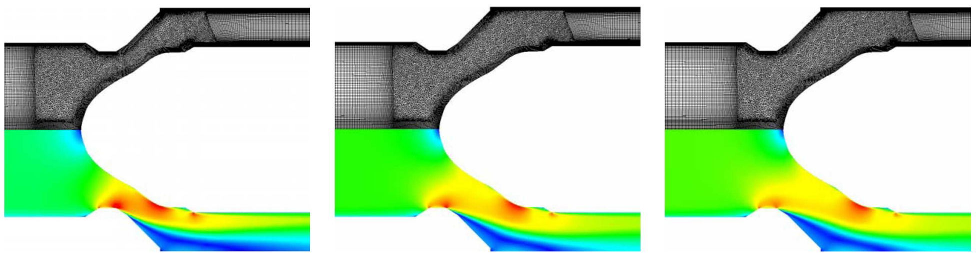
4.2 Real-world applications of dynamic meshing

This chapter discusses three examples of the industrial applications of the dynamic meshing. The fourth example is discussed in the next chapter together with applied visualization technique.

Cavallo et al. [10] used dynamic mesh in the simulation of a flow through the split-body valve. The mesh is adaptively refined and coarsened in the zones where a moving inner part of the valve creates moving mesh boundaries. Figure 4.4 shows example of the mesh layout from a split-body valve simulation [10] at various stages of the valve opening.

Amsden [2] describes dynamic meshing techniques used in KIVA-3V program for simulation of vertical and canted valves of combustion engines. Especially useful in these simulation are *Layering* and local refinement and coarsening of a simulation mesh. As stated by Amsden: *"This is not simply a matter of generating the initial grid: The grid must dynamically change during the run in response to the changing valve positions..."*.

Snyder and Sverdrup [46] discuss advantages of dynamic meshing against other methods for simulation of munition separation from under an aircraft wing. Rapid movement of falling munition requires continual modification of a simulation mesh. The ability to dynamically change a mesh between successive time steps is used to increase the speed of grid generation during a simulation.



(a) 50% open (b) 70% open (c) 80% open

Figure 4.4: Layout of the dynamically updated simulation mesh and direct rendering of the data from a split-body valve simulation (picture from [10]).

4.3 Iso-surface extraction

This chapter deals with the problem of iso-surface extraction from the data sets with dynamic mesh and is divided into three parts. The first part describes the problem itself and the related issues. The second part provides an overview of the existing techniques for iso-surface extraction designed for static simulation mesh (chapter 3), along with brief explanation why a particular technique is unable to work with data sets with dynamic mesh. This is to show that efficient extraction of the iso-surfaces from dynamic mesh requires new approach to the problem and can not be simply solved by applying existing techniques designed for static meshes. The third part, describes our approaches to the problem together with their limitations.

4.3.1 Problems

The two major problems when dealing with dynamic simulation mesh are:

1. Different number of mesh cells at each discrete time step. Geometry of the cells and possibly topology of the mesh may also vary during the course of simulation.

2. Due to the different simulation mesh at each time step the data sets with dynamic mesh are usually a magnitude larger compared to those with static mesh.

A list of particular challenges that have to be overcome by the efficient solution to the iso-surface extraction from dynamic simulation mesh can be derived directly from the two principal problems stated above:

- **Efficient representation of a dynamic mesh.** Due to the fact that geometry and topology of the cells may be different at each timestep, time and space efficient representation of dynamic mesh is necessary. At a higher level this point also includes calculation of changing mesh layout at the intermediated time steps.

- **Space and time efficient structure for identification of the active cells.** Many techniques for active cells extraction have been introduced in the past decade. Most of the existing techniques for active cells extraction assume static simulation mesh and exploit this fact to represent the mesh cell only by its *min/max* values and *id* number or pointer to the record with cell's static geometry. Therefore, new techniques have to be developed, designed specially for data sets with dynamic simulation mesh. Following chapter discuss this problem closely.

- **Suitable out-of-core strategies.** Keeping geometry of all mesh cells in the main memory could be very space-consuming (if not impossible for large data sets). Thus, only the space efficient structures holding the minimal amount of information necessary to identify the active cells, could be in the main memory. Geometry of the active cells then can be read from the disk using out-of-core techniques.

Since the mesh at each timestep is usually composed of tetrahedral or hexahedral cells, an arbitrary marching-style [3, 28, 51], ray-tracing [54] or point-based [4, 13, 14] methods can be used to extract geometry of resulting iso-surface from each single mesh cell. As stated above the main problems and challenges lie in the active cells identification and efficient out-of-core strategy that have to be done prior to extraction of an iso-surface geometry. Thus, the next sub-chapter is dedicated to the active cells identification in the dynamic simulation meshes.

4.3.2 Active cells identification

Majority of the existing techniques for active cells identification have been designed for the data sets with static simulation mesh (chapter 3). The assumption of static geometry and coherence of the mesh cells between adjacent time steps can not be in generally exploited for dynamic simulation mesh. In the following paragraphs, the existing techniques for active cells identification are listed again along with brief explanation why a particular technique can not be applied to the data sets with dynamic simulation meshes.

In THI tree *Temporal Hierarchical Index Tree* [40] a cell's temporal variation is measured as a displacement of its corresponding points (for various time steps) in Span space. Since, the correspondence of the mesh cells between adjacent time steps is in generally unknown for a dynamic simulation mesh, such temporal variation measurements in Span space can not be done, which is why the THI tree is unusable for the case of data sets with dynamic simulation mesh.

T-BON [48] use BONO tree [59] for spatial indexing of the mesh cells. BONO tree is created only once for the whole data sets, so a simulation grid has to remain static during the course of simulation. Only the min/max values of the cells are computed and stored separately for each time step. Because the T-BON technique does not exploit any temporal coherence of the data between adjacent time steps, a set of different BONO trees can be used to represent dynamic simulation mesh. This idea will be clarified later in this sub-chapter.

TSP tree [41] is a time supplement octree. Grid cells in are spatially indexed by the standard octree. So, each leaf node of TSP tree corresponds to a particular mesh cell. A binary time tree stored at each TSP node keeps an information about variation of the cells' min/max values along the time dimension. Because of the variable number of mesh cells in a dynamic simulation mesh, a TSP tree should has to be rebuild at each time step to reflect changes in mesh layout.

4D approach [58] of Weigle and Banks represents 3D time-varying data set as the 4D static data. The original 3D gridded time-varying data are represented as a mesh composed of 4D cells - the *hypercubes*. To construct a hypercube, a correspondence of the two original 3D cells, a hypercube is composed of, has to be known and one-to-one. So a simulation mesh has to remain static between any two adjacent time steps. Moreover, high computational and space requirement of the method makes it impractical for rather large data sets with dynamic simulation mesh.

In the Adaptive extraction approach [20] the original mesh cells are grouped into a hierarchy of diamonds. Since the hierarchy of the diamonds does not change during a simulation, a simulation mesh is required to remain static.

The Difference intervals [55] method encodes the changes of cell status in the successive time step as either *add* (cell become active) or *remove* (cell become inactive) operation. In order to encode status of all cells in the data set the assumption of static one-to-one correspondence between the cells at adjacent time steps has to be satisfied. This requires the simulation mesh to hold the fixed number of mesh cells during the whole simulation.

Persistent Hyperoctree (PHOT) [44] is derived from the standard hyperoctree, in which each node has 16 children (hyperoctants). Since the original 3D octants are merged in into the 4D hyperoctants a data grid in is assumed to be regular and fixed during the whole simulation.

- -

As can be seen from the list of the techniques above, the active cells identification in the dynamic meshes can not be simply solved by applying the existing techniques designed for static simulation meshes. A common approach to the this problem today is to consider data (simulation mesh and values associated with a mesh vertices) at each time step as a stand-alone static subset of data, without exploiting either the spatial or temporal coherence in the data. For such a *brute-force* approach, any of the existing technique for active cells identification for static data sets (chapter 3.1) might be used over a single time step.

Figure 4.5 shows possible usage of BONO tree [59] for data set with dynamic simulation mesh. A separate BONO tree is constructed for each time step. Resulting set of BONO trees is indexed by a two-level tree along the time dimension. This *brute-force* approach will be used as a starting point in our research and its performance will be used as a reference performance against which all the advanced techniques will be compared.

Doleisch et al. [15] introduced technique for visualization of data sets with dynamic simulation mesh. Their technique is specifically oriented onto the *layered meshes* (chapter 4.1.2). The continuous intervals are assumed in the data set

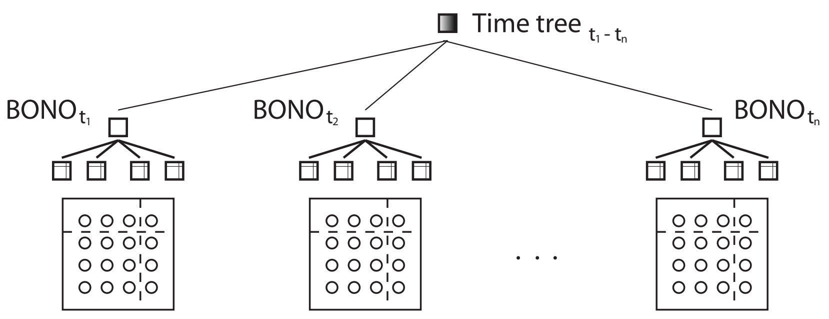


Figure 4.5: Example of BONO tree usage for data sets with dynamic simulation mesh. A separate BONO tree is constructed for each time step. Resulting set of BONO trees is indexed by a two-level tree along the time dimension.

within which a number of mesh cells and their correspondence between adjacent time steps remains static. Such intervals are called *topology zones*. However, mesh cells are not matched or tracked over the topology zones borders (*rezone points*). Figure 4.6 illustrates the principle of the method.

Within a topology zone a mesh layout is expected to change linearly, so a shape of the mesh cells as well as values associated with mesh vertices can be interpolated from the *key geometries* at the borders or inside of a topology zone. So the data value can be computer anywhere at or in between defined time steps. This allows to use most of the existing methods for iso-surface extraction (chapter 3) or any other techniques for visualization of time-varying data (chapter 2).

The assumption of the *topology zones* can be used only for a specific subset of the data sets with dynamic simulation mesh. However, in generally, each time step might represent a different topology zone. Thus, more robust techniques for iso-surface extraction from dynamic simulation meshes need to be developed.

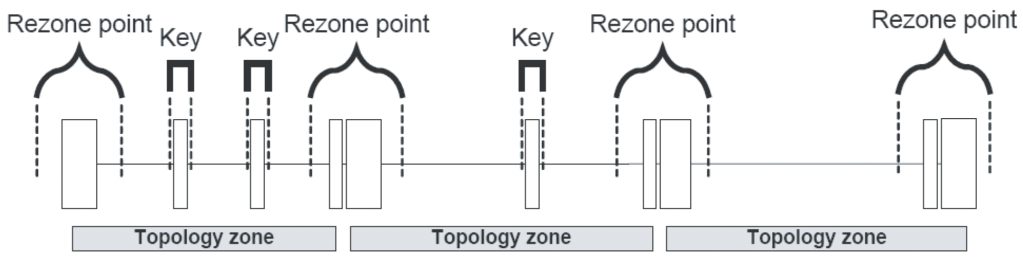


Figure 4.6: Principle of the *topology zones* technique [15]. Each topology zone consists of at least two *key geometries*. Borders of the topology zones are called *rezone points* (picture from [15].)

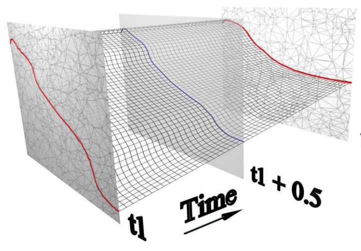
4.4 Published original contributions

4.4.1 Iso-contours extraction from time-varying meshes

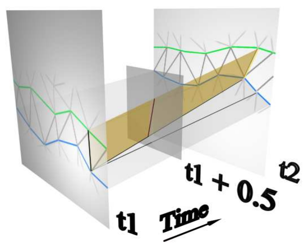
As was stated in the chapter 4.3.1 in dynamic simulation mesh the number of the mesh cells and their correspondence between adjacent time steps may not be one-to-one. Thus, prior to extraction of iso-contours for an arbitrary iso-value and time (also in between the originally defined time steps) a cell-cell correspondence has to be established between the simulation meshes at adjacent time steps. Our method [33] address the problem of establishing an edge-edge correspondence between adjacent time steps and the solution is employed for iso-contour extraction from 2D dynamic simulation meshes.

The key idea of the method is to establish an edge-edge correspondence between meshes at adjacent time steps, based on the data similarity. The method works over 2D triangular dynamic meshes.

Figure 4.7 compares the *brute-force* solution and principle of our method. In brute-force solution the iso-contours for a desired iso-value are first extracted from both adjacent time steps. Then a hyper-surface ruled at both ends by the extracted iso-contours is built and cut at the desired time.



(a)



(b)

Figure 4.7: (a) The *brute-force* approach to the problem. (b) Our method.

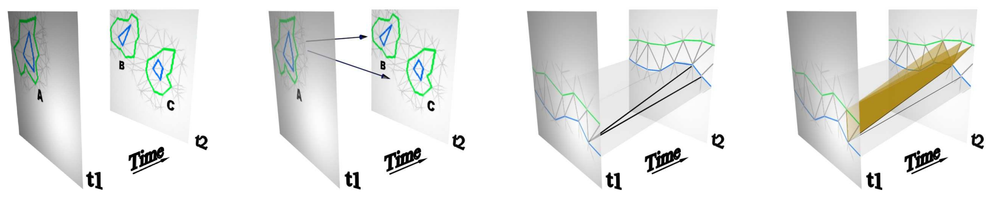
Our method establishes an edge-edge correspondence in four steps applied for each pair of consecutive time steps:

a. Iso-component extraction.

b. Tracking of iso-components between adjacent time steps.

c. Mapping of iso-components' inner envelope vertices.

d. Edge-to-edge mapping, to make out final set of edge-to-edge mappings for each iso-component extracted in step a).



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

Figure 4.8: Four stages of our method: (a) iso-components extraction, (b) feature tracking, (c) inner envelopes vertices mapping, (d) final edge-to-edge mapping.

An iso-component is a strip of neighboring triangles, covering certain range of the iso-values. Each iso-component has its with inner (IE) and outer (OE) envelope (figure 4.9).

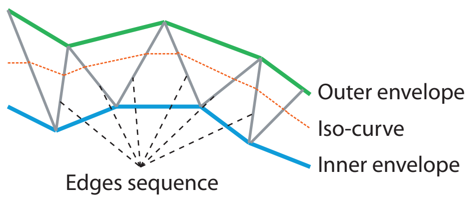


Figure 4.9: Iso-component is a strip of neighboring triangles with inner and outer envelope.

The four steps of establishing an edge-edge correspondence have to be done for each pair of the meshes at adjacent time steps.

Iso-component extraction begins by sorting a list L of all different iso-values from both adjacent time steps. Then the iso-components are extracted using continuation method and middle values of the intervals in L (figure 4.10).

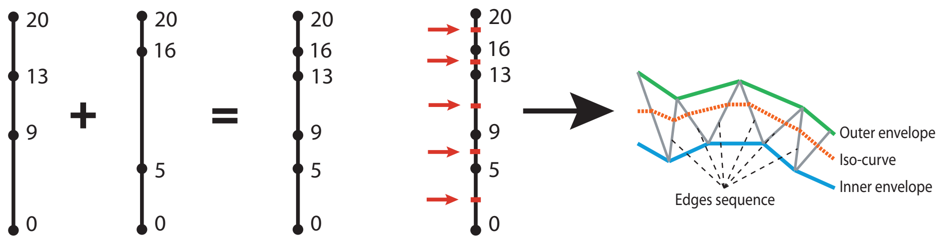


Figure 4.10: Iso-components extraction. All the different iso-values from two adjacent time steps are merged and sorted into one list L. Iso-components are then extracted using continuation method and middle values of the intervals in the list L.

Next the feature tracking techniques [38] are employed to find the best matching successors for each extracted iso-components. In our implementation, the area overlapping test is used [45]. Depending on the application, more complex feature tracking techniques can be used without disturbing the overall concept of our method.

Once the successors of iso-components are determined for each extracted iso-component, the process of mapping their IE vertices takes place. This mapping provides a guide to the final edge-to-edge mapping.

Process of IE vertices mapping is based on the notion of *candidate area* (figure 4.11a), which is defined for each mapped IE vertex. Candidate successor vertex has to lie within this area. Employing this principle, the closest from such restricted set of candidate vertices is chosen as the successful one. During the mapping process the projections (Fig. 4.11b) of vertices from successive time slices are considered rather than computing with the real space-time elements. More detailed description of the set of rules can be found in [33].

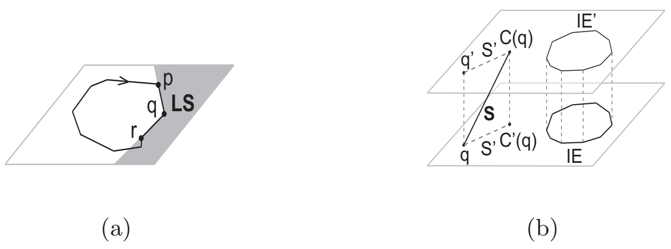


Figure 4.11: (a) Candidate area of the vertex q (shadow region). (b) Projection of slicechord S.

Connecting the IE vertices of successive iso-components by the slicechords enables us to map the edges of related iso-components. Edges from connected vertices are mapped. By this way a list of each extracted iso-component edges mapping onto the edges of its successive iso-components are build up. Extracted lists of iso-intervals, iso-components and edge-to-edge mapping are saved and can be used anytime for the final visualization.

Visualization

Having determined the set of iso-components for each iso-interval and a set of edge-to-edge mapping for each iso-component, resulting iso-curve evolution can be visualized.

First the iso-components are selected from the interval covering a user-selected iso-value and time. Then the vertices C and E (figure 4.12) are interpolated out at the edges of successive iso-components based on selected iso-value. User-selected time rules the interpolation of point R (on the connection of C and E). Resulting iso-curve is composed of line segments connecting a set of R points.

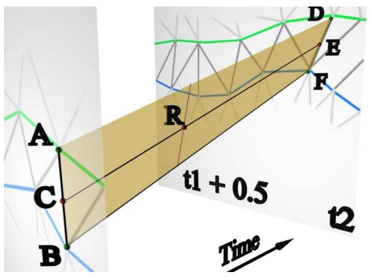
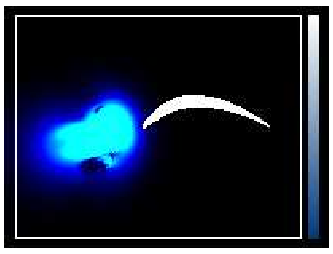


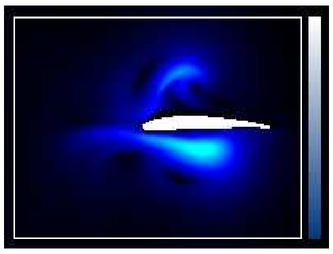
Figure 4.12: Visualization. First the vertices C and E are interpolated out at the edges of successive iso-components based on selected iso-value. User-selected time rules the interpolation of point R (on the connection of C and E).

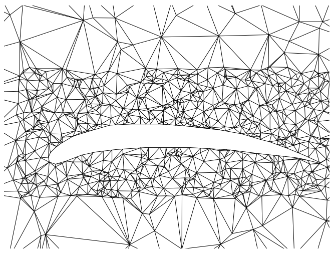
All the tests have been performed on two processors Intel Pentium workstation with 2GB of RAM. Method has been implemented in C# language.

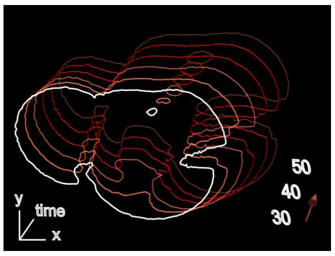
Airfoil data set is the result of simulation of low-speed air wave hitting the leading edge of the flexible airfoil. Scalar values associated with the mesh vertices are the velocity magnitudes of advancing air wave. The data set consists of 200 time steps each represented by the triangular adaptive mesh with 16 000 to 17 000 vertices per time step. Simulation mesh adapts to the changing shape of flexible airfoil at each simulation step. For the test we took every 10th time slice and computed the iso-contour evolution in between them by our method. Figure 4.13 shows original data from the Airfoil data set, dynamic mesh and evolving iso-contours.

Finally, the figure 4.14 shows the GUI of the application, implementing our iso-contour extraction method. Iso-contours rendered in the visualization window in figure 4.14 are from the data set originating from the simulation payload release from under an aircraft wing.

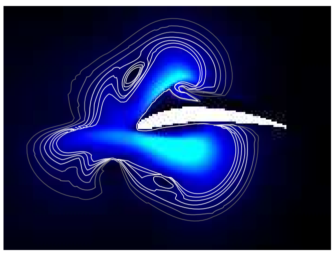

(a)


(b)


(c)



(d)


(e)

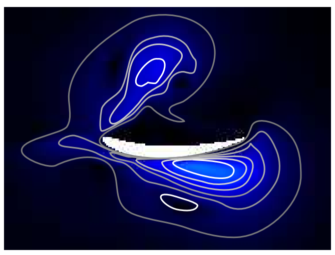
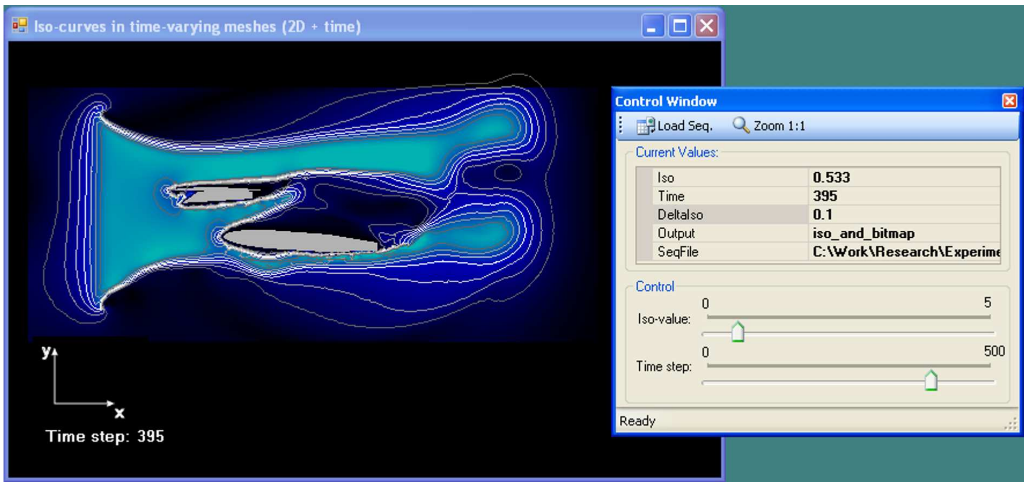

(f)

Figure 4.13: Airfoil dataset. (a)-(b) input data (speed magnitude) at the time steps 10 and 100, (c) dynamic adaptive mesh (time step 80), (d) iso-contour evolution computed by the proposed method between the time steps 30 and 50, (e)-(f) extracted iso-contours at the time steps 74 and 186.



Iso	0.533
Time	395
DeltaIso	0.1
Output	iso_and_bitmap
SeqFile	C:\Work\Research\Experime

Figure 4.14: Iso-contours extracted from payload-release simulation (iso-value=0.533, time step: 395). Blue rendered original data at the background (speed magnitude values) are just for illustration and comparison purposes.

4.4.2 Z-Diamonds: Iso-surface extraction from dynamic meshes

Z-Diamonds [34] is simple and efficient algorithm for extraction of the iso-surfaces from the data sets with dynamic mesh. Z-Diamonds method works over triangular or tetrahedral meshes. No assumption is made about the way a simulation mesh changes between adjacent time steps. We also do not attempt to reconstruct the evolving iso-surfaces in between time steps defined in a data set. Figure 4.15 outlines the principle of the Z-Diamonds method.

Since a simulation mesh dynamically changes, we do not try to match the original mesh cells between adjacent time steps. Instead the mesh at each time step is preprocessed into a list of diamonds. Each diamond is composed of two neighboring simplicial cells, sharing a common face (i.e. two triangles in 2D or two tetrahedra in 3D, figure 4.16). Diamonds are not matched or tracked between adjacent time steps.

Pairing of the tetrahedral mesh cells into the diamonds has several advantages over the simple tetrahedral representation of a mesh. Firstly, a single diamond is represented by five vertices and five scalar values, in comparison to the eight vertices with associated scalar values necessary to represent two separate tetrahedra. Since the active diamonds are dynamically loaded for each iso-surface query, such "five-vertex" representation of the diamonds reduces the I/O traffic during interactive visualization. Secondly, it allows use of the Marching Diamonds algorithm [3] to produce smoother resulting iso-surfaces.

Once we have a list of diamonds for each time step, we need a data structure to support fast extraction of the active diamonds. Because of the large amount of data needed for representation of dynamic simulation mesh, we keep only the *min/max* values of the diamonds in the main memory. The *min/max* values of all diamonds from all time steps are indexed by a common TSP tree [41].

During the visualization phase the queries in the form (*iso-value, time step*) are accepted and processed. IDs of the active diamonds are extracted by travers-

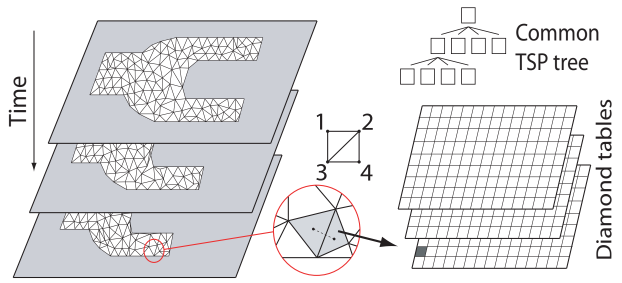


Figure 4.15: Outline of the principle of the Z-Diamonds method. Original mesh cells are paired into diamonds. *Min/max* and ID values of the diamonds are then organized in the diamond tables (one for each time step). A common TSP tree is used to index the diamonds stored in the diamond tables for all time steps.

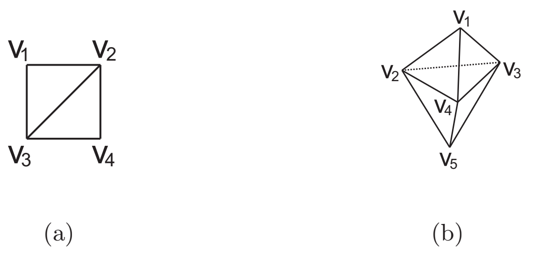


Figure 4.16: Each diamond is composed of two neighboring simplicial mesh cells, sharing a common face. Reference diamonds are depicted for 2D (a) and 3D (b) simulation mesh.

ing a common TSP tree using given *iso-value* and *time step*. Geometry of the active diamonds is then dynamically loaded from a disk in an out-of-core fashion. Iso-surface geometry can be extracted from the loaded active diamonds using techniques like Marching tetrahedra [51] or Marching Diamonds [3].

Since we extract a queried iso-surface only at the discrete time steps, specified in a data set, we do not deal with the topological changes of the iso-surfaces in between adjacent time steps. We assume that the data sets processed by our method are sampled sufficiently along the time dimension.

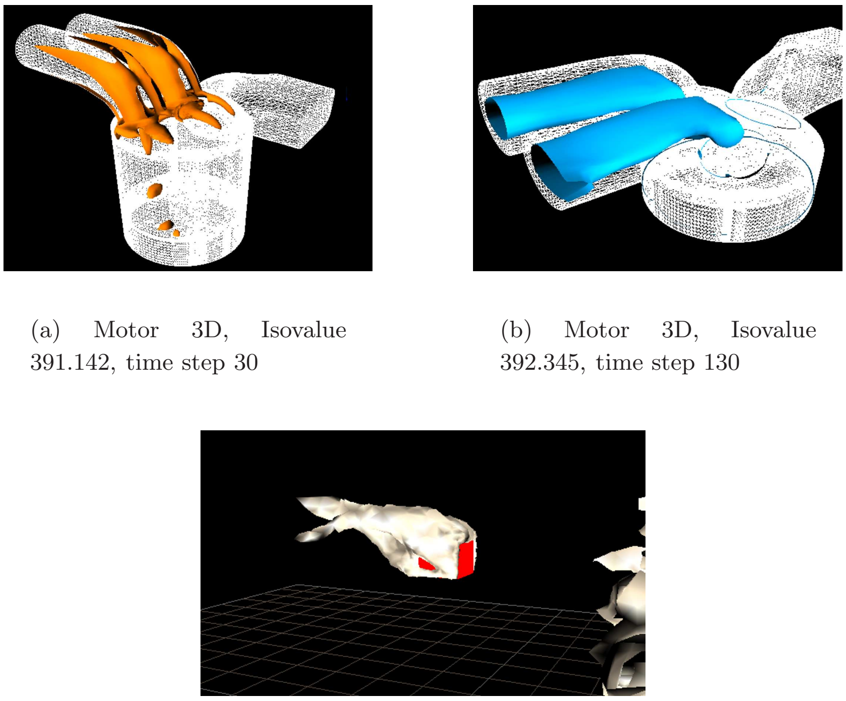
Table 4.1 provides overview of the data sets used for tests as well as the preprocessing times. Table 4.2 draws the iso-surfaces extraction times during interactive visualization. Query execution times stated in the table 4.2 include time for active cells extraction by traversing a TSP tree, loading of active diamonds geometry from disk and extraction of iso-surface geometry from loaded active diamonds. Finally the figure 4.17 shows iso-surface for selected iso-values and time steps extracted by the Z-Diamonds method during interactive visualization.

Data set	# of time steps	# of cells per time step	Data set size	Preprocessing time	Size of preprocessed data
Motor	148	40k to 115k	3.2GB	86mins 32s	1.2GB
Wind tunnel	700	400k to 430k	7.5GB	120mins 8s	3.4GB

Table 4.1: Data sets used for testing Z-Diamonds method with the preprocessing times and sizes of the preprocessed data sets.

Data set / iso-value / time step	Extraction time	# of active diamonds	# of triangles on the iso-surface
Motor / 391.124 / 30	612ms	15,900	36,768
Motor / 392.345 / 130	514ms	6,741	15,398
Wind tunnel / 8.612 / 451	289ms	3,133	4,932

Table 4.2: Extraction times, numbers of active diamonds and numbers of triangles on the resulting iso-surfaces for selected iso-values and time steps for *Motor* and *Wind tunnel* data sets.


(a) Motor 3D, Isovalue 391.142, time step 30

(b) Motor 3D, Isovalue 392.345, time step 130

(c) Wind tunnel, Iso-value 8.612, time step 451

Figure 4.17: Iso-surfaces for selected iso-values and time steps extracted by the Z-Diamonds method.

Chapter 5

Summary and future research

5.1 Summary

This state-of-the-art report summarizes the research work done in iso-surface extraction from time-varying data sets and provides description of our contribution to the field. The main focus of this work is the iso-surface extraction from the data sets with dynamic simulation mesh. Only a few approaches addressing this problem have been published yet (chapter 4.3.2). The natural first step when staring new research is the investigation of the existing techniques, which is why this report covers all three fields of iso-surface extraction from time-varying data:

1. Iso-surface extraction from static data (chapter 3.1).
2. Iso-surface extraction from data sets with static simulation mesh (chapters 3.2 to 3.9).
3. Iso-surface extraction from data sets with dynamic simulation mesh (chapter 4).

Chapter 3.1 provides an overview of the methods for iso-surface extraction from static data. Even if this part of the research area deals with the data defined for a single moment in time, it provides a starting point for iso-surface extraction from time-varying data.

In the recent years a range of methods for time-varying scalar fields have been introduced (chapters 3.2 to 3.9). These methods work with the assumption that a simulation mesh remains fixed during the course of simulation. Known correspondence of the mesh cells between adjacent time steps allows to build space efficient data structures, utilizing temporal coherence among corresponding cells.

The third part (chapter 4) deals with iso-surface extraction from data sets with dynamic simulation mesh. The chapter begins with a short introduction into the problematic of dynamic meshing of simulation domains (chapter 4.1), supported by the three real-world examples (chapter 4.2).

Sub-chapter 4.3 deals with iso-surface extraction from data sets with dynamic simulation meshes - the main focus of our research. First part of the chapter discuss three main challenges of this visualization problem: efficient representation of a dynamic mesh, space and time efficient structure for active cells identification and out-of-core strategies.

Sub-chapter 4.3.2 presents a list of methods for fast iso-surface extraction from data sets with static meshes once again, along with brief explanation of why a particular method can not be applied onto the data sets with dynamic simulation mesh. This is to show that each of three challenges described in sub-chapter 4.3.1 *Problems*, requires different approach than in the case of static simulation mesh. Two possible solutions to the problem of iso-surface extraction from the data sets with dynamic simulation meshes are listed in the second part of sub-chapter 4.3.2:

1. *The brute-force solution.* Each time step is treated as a stand-alone static subset of data and one common tree or list is created, connecting all the time steps along the time dimension (figure 4.5). With this approach, the temporal coherency of the data remains unexploited.

2. *Solution of Doleisch et al. [15].* The continuous intervals along the time dimension are assumed, within which a correspondence between the cells at adjacent time steps remains fixed (*topology zones*). This allows to compute the data at any time within topology zone, even in between the originally defined time steps. So, the smoothly evolving iso-surfaces can be visualized.

From the research of the existing methods for iso-surface extraction from data sets with static simulation mesh and initial research done in the area of dynamic simulation, we made the three following conclusions:

1. Since the techniques for generation of dynamic simulation meshes are being developed rapidly, there is a lack of suitable visualization techniques. Iso-surface extraction from the data sets with dynamic meshes is under-research area with only two known approaches (chapter 4.3.2).

2. Existing techniques for fast identification of the active cells used for data sets with static simulation mesh cannot be applied onto the data sets with dynamic mesh.

3. The three main problems (chapter 4.3.1): *efficient representation of a dynamic mesh, space and time efficient structure for active cells identification and out-of-core strategies* require new methods able to handle dynamic simulation mesh.

5.2 Future research

The goal of our research is a space and time efficient method for iso-surface extraction from data sets with dynamic simulation mesh. In particular, we focus on interactive iso-surface extraction (along both spatial and temporal dimension) on a workstation class computers.

Fixed simulation mesh is one of the basic assumption of all existing techniques for iso-surface extraction from time-varying data sets (chapter 3). This assumption prevents their usage with the data sets with dynamic simulation mesh. This claim is supported by the analysis of these methods in the chapter 4.3.2 of this work. Therefore, we propose not to continue in our research by extending the existing methods towards the data sets with dynamic simulation mesh.

The data sets with dynamic simulation mesh are usually a magnitude larger compared to those with static mesh. Therefore a space requirements play an important role in the future techniques for iso-surface extraction from this kind of data.

Coming out of the low space requirements conclusion from the previous paragraph, we propose to continue our research by first develop the space and time efficient techniques for iso-surface extraction from the static data sets and then extend them towards the time-varying data sets with dynamic simulation mesh. This is in fact our key idea of how to achieve our main research goal stated at the beginning of this chapter.

5.2.1 Iso-surface extraction from static data

The lowest space requirements of the existing methods for iso-surface extraction from static data are around $3N$, where N is the number of mesh cells, plus some overhead for a data structure. Table 5.1 summarizes the space requirements of the existing methods for iso-surface extraction from static data (chapter 3.1).

Method	Space required	Reported space complexity
NOISE (kd-tree)	$3N$ + kd-tree overhead	$O(N)$
ISSUE (lattice)	$4N$ + kd-trees at *min*=*max* line	$O(N)$
Interval tree	$H + M$	$O(H + M)$
Span-Triangle$^{\#}$	N + base array + span arrays	$O(N)$
Fixed-sized buckets	$3N$ + *min* dictionary	$O(N)$
Quantized search	$3N^{+}$ N + U and V tables^{++}	$O(N)$

$^{+}$ = Temporary structure $^{\#}$ = Only applies to quantized data
$^{++}$ = Quantized data H = # interval tree nodes
M = # of distinct intervals in data

Table 5.1: Space requirement comparison.

Usually the *min*, *max* and *ID* attributes are somehow indexed by the data structure (therefore $3N$). Note that even if the reported space complexity of all methods is $O(N)$ the real space requirements of the methods differ.

The following example illustrates the impact of the real space requirements. Lets assume that most of the todays workstation class computers have 2GByte of main memory. A 3D volume data set (regularly sampled) of the size 512^3 (4 bytes per sample) takes 512MB of memory. As stated in the table 5.1 the requirements of the *ISSUE* method are $4N$. If the cells IDs are stored also as 4-byte integers, then the $N = 512^3$ data set occupy 2036 MB, which clearly exceeds the size of the allocable main memory.

Therefore, we propose to focus the first phase of our research at the development of space efficient method for iso-surface extraction from static data sets. Since the space needed for storage of cells IDs will always take N numbers (usually 4-bytes integers), our goal is to achieve $2N$ and lower space requirements. The three main challenges of this goal are:

1. Efficient representation of the cells' *min*/*max* values.

2. Usability of a structure resulting from data dimensionality reduction for a fast identification of the active cells.

3. Resulting data representation has to allow investigation of the temporal coherency between the cells at adjacent time steps.

ad 1) We believe that $2N$ (or lower) space requirements are achievable via the methods for *dimensional reduction*, like: Principal component analysis (PCA), Multidimensional scaling (MDS) or Clustering. The dimensional reduction techniques can be applied onto the pairs of cells' *min*/*max* values, representing a group of such pairs by one number or vector. As an inspiration the Quantized search method is discussed in the next paragraph.

Bordoloi and Shen introduced the Quantized search [6] method, which exploits quantization of the cells *min*/*max* values. However, their method requires temporal structure of the size $3N$ and the resulting search error is around 2%. Moreover, the data needs to be sorted three times according to various criteria during the construction of the search structure.

ad 2) The second challenge includes two aspects. First the resulting data structure should provide the optimal or near optimal search time of the active cells $O(K)$, where K is the number of the active cells.

The second aspect is the additional memory overhead caused by the search structure, which should also be minimal, compared to the $2N$ (or lower) requirements for representing the *min*/*max* pairs. For example: in the case of Quantized search method, the space for N cells IDs is required and moreover an extra space is required to store the *U and V* tables, defining the partitioning (quantization) of the *UV-space*.

ad 3) The third challenge provides connection point with the second phase of our research, which is detailed in the next chapter.

5.2.2 Iso-surface extraction from dynamic simulation mesh

The purpose of the second phase of our research is to extend the space efficient methods for iso-surface extraction from static data sets (previous chapter) toward time-varying data sets with dynamic simulation mesh. The primary challenge of this phase is an effective strategy of exploiting temporal coherency in the data.

In a dynamic simulation mesh the number of mesh cells and geometry change over time. Therefore, the implicit correspondence of the mesh cells can not be in generally exploited as in the case of fixed simulation mesh.

The results of our previous research [33] show that spatial geometric correlation of the mesh cells seem to be the hard task. This is because one vertex of the simulation mesh may be interpolated to many in the following time step. This may results in a set of possible cell-cell mappings without any tools to decide which solution is the right one.

Considering the one-to-many-interpolation problem discussed in the previous paragraph, *we propose to focus our research at the extraction of the iso-surfaces only from the data defined at the discrete time steps*. Development of the methods for computation of the data evolution in between defined time steps remains the research task which require a lot of time and goes far beyond the horizon of this doctoral work.

Coming out of the goal of the first phase of our research, which was: data structure size less than $2N$, where N is the number of the mesh cells at one particular moment in time, we intend to achieve the same goal in the second phase. The resulting size of the data structure for T time steps should be $2M$ or less, where M is the total number of all cells from all time steps T.

We propose to investigate the temporal coherency of the group of cells in the alternative spaces like *Span space* [27]. Correlating the mesh cells in such alternative spaces provides more flexibility, since the main problem - changing geometric position and of the mesh cells, losses a lot of its significance.

As stated in the previous chapter the data structure for iso-surface extraction developed in the first phase of our research should allow investigation of the cell temporal coherency. Because we intend to build our structures for iso-surface extraction from static data over mentioned the alternative spaces, we propose to encode only the difference between the data structures at adjacent time steps. This should lead to the significant space savings, while preserving the originally declared near-optimal extraction time of the active cells.

Bibliography

[1] F. Alauzet, P. J. Frey, P. L. George, and B. Mohammadi. 3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations. *Journal of Computational Physics*, 222(2):592–623, 2007. [cited at p. 28]

[2] A. A. Amsden. KIVA-3V: A block-structured KIVA program for engines with vertical or canted valves. Technical Report LA-13313-MS, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA, May 1997. [cited at p. 7, 31]

[3] J. C. Anderson, J. Bennett, and K. I. Joy. Marching Diamonds for unstructured meshes. In *IEEE Visualization 2005*, 2005. [cited at p. 12, 33, 41, 42]

[4] J. A. Bærentzen and N. J. Christensen. Hardware accelerated point rendering of isosurfaces. *Journal of WSCG*, 11(1):41–48, feb 2003. [cited at p. 33]

[5] F. Bernardon, S. Callahan, J. Comba, and C. Silva. Interactive volume rendering of unstructured grids with time-varying scalar fields. In *Eurographics Symposium on Parallel Graphics and Visualization*, pages 51–58, 2006. [cited at p. 11]

[6] U. Bordoloi and H.-W. Shen. Space efficient fast isosurface extraction for large data sets. In *IEEE Visualization'03*, pages 201–208, 2003. [cited at p. 19, 48]

[7] S. P. Callahan, L. Bavoil, V. Pascucci, and C. T. Silva. Progressive volume rendering of large unstructured grids. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2006)*, 12(5), September-October 2006. [cited at p. 11]

[8] H. Carr and J. Snoeyink. Path seeds and flexible isosurfaces using topology for exploratory visualization. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 49–58, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. [cited at p. 19]

[9] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Comput. Geom. Theory Appl.*, 24(2):75–94, 2003. [cited at p. 19]

[10] P. Cavallo, A. Hosangadi, and V. Ahuja. Transient simulations of valve motion in cryogenic systems. In *AIAA Fluid Dynamics Conference and Exhibit*, 2005. [cited at p. 31]

[11] Y.-J. Chiang, C. T. Silva, and W. J. Schroeder. Interactive out-of-core isosurface extraction. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 167–174, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. [cited at p. 19]

[12] P. Cignoni, P. Marino, C. Montani, E. Puppo, and R. Scopigno. Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170, 1997. [cited at p. 19]

[13] C. S. Co, B. Hamann, and K. I. Joy. Iso-splatting: A Point-based Alternative to Isosurface Visualization. In *Proceedings of Pacific Graphics 2003*, pages 325–334, 2003. [cited at p. 33]

[14] Ch. S. Co and K. I. Joy. Isosurface generation for large-scale scattered data visualization. In *Proceedings of VMV 2005*, pages 233–240, 2005. [cited at p. 14, 33]

[15] H. Doleisch, M. Mayer, M. Gasser, P. Priesching, and H. Hauser. Interactive feature specification for simulation data on time-varying grids. In *SimVis'05*, pages 291–304, 2005. [cited at p. 7, 34, 35, 46]

[16] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodriguez-Ferran. *Arbitrary Lagrangian-Eulerian methods, Encyclopedia of Computational Mechanics.* John Wiley and Sons, 2004. [cited at p. 7, 29]

[17] J. Donea, A. Huerta, J.-Ph. Ponthot, and A. Rodriguez-Ferran. *Modeling Flows in Moving and Deforming Zones, Fluent Manual.* Fluent Inc., 2005. [cited at p. 7, 30]

[18] K. Engel and T. Ertl. Interactive high-quality volume rendering with flexible consumer graphics hardware. In *Eurographics '02*, 2002. [cited at p. 11]

[19] K. Engel, M. Kraus, and T. Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, New York, NY, USA, 2001. ACM Press. [cited at p. 11]

[20] B. Gregorski. Adaptive extraction of time-varying isosurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):683–694, 2004. Student Member-J. Senecal and Member-M. A. Duchaineau and Member-K. I. Joy. [cited at p. 24, 34]

[21] H. Hagen, H. Müller, and G. M. Nielson, editors. Springer, 1993. [cited at p. 9]

[22] C. R. Johnson, Y. Livnat, L. Zhukov, D. Hart, and G. Kindlmann. Computational field visualization. In Björn Engquist and Wilfried Schmid, editors, *Mathematics Unlimited – 2001 and Beyond*, volume 2, pages 605–630. Springer-Verlag, 2001. [cited at p. 10]

[23] A. Joshi and P. Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *IEEE Visualization'05*, page 86, 2005. [cited at p. 15]

[24] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. *ACM Transactions on Computer Graphics*, 21(3):339–346, 2002. [cited at p. 12]

[25] D. A. Lane. Visualizing time-varying phenomena in numerical simulations of unsteady flows. Technical Report NAS-96-001, Moffett Field, CA 94035, 1996. [cited at p. 10]

[26] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Computer Graphics*, 9(3):245–261, 1990. [cited at p. 11]

[27] Y. Livnat, H.-W. Shen, and Ch. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, 1996. [cited at p. 18, 24, 49]

[28] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, pages 163–169, New York, NY, USA, 1987. ACM Press. [cited at p. 5, 12, 17, 33]

[29] S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In R. Daniel Bergeron and Arie E. Kaufman, editors, *Proceedings of IEEE Visualization '94*, pages 100–107, 1994. [cited at p. 11]

[30] D. P. Mitchell and A. N. Netravali. Reconstruction filters in computer-graphics. In *SIGGRAPH '88*, pages 221–228, New York, NY, USA, 1988. ACM Press. [cited at p. 11]

[31] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, 1993. [cited at p. 12]

[32] S. Parker, P. Shirley, Y. Livnat, Ch. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *IEEE Visualization '98*, pages 233–238, 1998. [cited at p. 13, 23]

[33] S. Petrik and V. Skala. Iso-contouring in time-varying meshes. In *Proceedings of the SCCG 2007*, pages 216–223, 2007. [cited at p. 36, 38, 49]

[34] S. Petrik and V. Skala. Z-Diamonds: A fast iso-surface extraction algorithm for dynamic meshes. In *To appear in proceedings of the IADIS Computer Graphics and Visualization 2007*, 2007. [cited at p. 41]

[35] F. H. Post and T. van Walsum. Fluid flow visualization. In *Focus on Scientific Visualization*, pages 1–40, 1991. [cited at p. 9, 10]

[36] E. Reinhard, Ch. Hansen, and S. Parker. Interactive ray tracing of time varying data. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 77–82, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. [cited at p. 23]

[37] A. Sadarjoen, T. van Walsum, A. J. S. Hin, and F. H. Post. Particle tracing algorithms for 3D curvilinear grids. Technical Report DUT-TWI-94-80, Delft, The Netherlands, 1994. [cited at p. 10]

[38] R. Samtaney, D. Silver, N. Zabusky, and J. Cao. Visualizing features and tracking their evolution. *Computer*, 27(7):20–27, 1994. [cited at p. 38]

[39] Ch. S.Co, S. D. Porumbescu, and K. I. Joy. Meshless isosurface generation from multiblock data. In *IEEE TCVG Symposium on Visualization*, 2004. [cited at p. 13]

[40] H.-W. Shen. Isosurface extraction in time-varying fields using a temporal hierarchical index tree. In *VIS '98: Proceedings of the conference on Visualization '98*, pages 159–166, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press. [cited at p. 21, 33]

[41] H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (tsp) tree. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 371–377, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press. [cited at p. 22, 33, 41]

[42] H.-W. Shen, Ch. D. Hansen, Y. Livnat, and Ch. R. Johnson. Isosurfacing in span space with utmost efficiency (ISSUE). In Roni Yagel and Gregory M. Nielson, editors, *IEEE Visualization '96*, pages 287–294, 1996. [cited at p. 19, 21]

[43] H.-W. Shen and Ch. R. Johnson. Sweeping simplices: A fast iso-surface extraction algorithm for unstructured grids. In *IEEE Visualization'95*, pages 143–150, 1995. [cited at p. 18]

[44] Q. Shi and J. JaJa. Isosurface extraction and spatial filtering using persistent octree. In *IEEE Visualization 2006*, Los Alamitos, CA, USA, 2006. IEEE Computer Society Press. [cited at p. 25, 26, 34]

[45] D. Silver and X. Wang. Tracking and visualizing turbulent 3d features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, 1997. [cited at p. 38]

[46] D. O. Snyder and J. Sverdrup. Dynamic meshing models store separation from transonic aircraft. In *Journal articles by fluent software users*. Fluent Inc., 2005. [cited at p. 31]

[47] M. Sramek. Fast surface rendering from raster data by voxel traversal using chessboard distance. In *VIS '94: Proceedings of the conference on Visualization '94*, pages 188–195, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press. [cited at p. 13]

[48] P. Sutton and Ch. D. Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon). In *VIS '99: Proceedings of the conference on Visualization '99*, pages 147–153, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press. [cited at p. 21, 33]

[49] J. F. Thompson, B. K. Soni, and N. P. Weatherill. *Handbook of Grid Generation*. CRC Press, New York, 1999. [cited at p. 28, 30]

[50] Ch. Tietjen, B. Meyer, S. Schlechtweg, B. Preim, I. Hertel, and G. Strau. Enhancing slice-based visualizations of medical volume data. In *IEEE/Eurographics Symposium on Visualization (EUROVIS'06)*, pages 123–130. IEEE, 2006. [cited at p. 11]

[51] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999. [cited at p. 12, 33, 42]

[52] M. van Kreveld, R. van Oostrum, Ch. Bajaj, V. Pascucci, and D. Schikore. Contour trees and small seed sets for isosurface traversal. In *SCG '97: Proceedings of the thirteenth annual symposium on Computational geometry*, pages 212–220, New York, NY, USA, 1997. ACM Press. [cited at p. 19]

[53] B. von Rymon-Lipinski, N. Hanssen, T. Jansen, L. Ritter, and E. Keeve. Efficient point-based isosurface exploration using the span-triangle. In *IEEE Visualization'04*, pages 441–448, Washington, DC, USA, 2004. IEEE Computer Society. [cited at p. 19]

[54] I. Wald, H. Friedrich, A. Knoll, and Ch. D Hansen. Interactive isosurface ray tracing of time-varying tetrahedral volumes. Technical Report UUSCI-2007-003, 2007. Conditionally accepted at IEEE Visualization 2007. [cited at p. 13, 33]

[55] K. W. Waters and Ch. S. Co. Using difference intervals for time-varying isosurface visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1275–1282, 2006. [cited at p. 18, 22, 24, 25, 34]

[56] K. W. Waters, Ch. S. Co, and K. I. Joy. Isosurface extraction using fixed-sized buckets. In *IEEE VGTC Symposium on Visualization*, pages 207–214, 2005. [cited at p. 19]

[57] Ch. Weigle and D. C. Banks. Complex-valued contour meshing. In Roni Yagel and Gregory M. Nielso, editors, *IEEE Visualization '96*, pages 173–180, 1996. [cited at p. 22]

[58] Ch. Weigle and D. C. Banks. Extracting iso-valued features in 4-dimensional scalar fields. In *VVS '98: Proceedings of the 1998 IEEE symposium on Volume visualization*, pages 103–110, New York, NY, USA, 1998. ACM Press. [cited at p. 22, 23, 34]

[59] J. Wilhelms and A. van Gelder. Octrees for faster isosurface generation. *ACM Trans. Graph.*, 11(3):201–227, 1992. [cited at p. 17, 18, 21, 33, 34]

[60] J. Woodring and H.-W. Shen. Chronovolumes: a direct rendering technique for visualizing time-varying data. In *VG '03: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume graphics*, pages 27–34, New York, NY, USA, 2003. ACM Press. [cited at p. 14]

[61] J. Woodring, C. Wang, and H. Shen. High dimensional direct rendering of time-varying volumetric data, 2003. [cited at p. 11]

Appendix A - Publications, project assignments

Related publications

Petrik S., Skala V.: Z-Diamonds: A Fast Iso-surface Extraction Algorithm for Dynamic Meshes. *Proceedings of the IADIS Computer Graphics and Visualization 2007*, Lisbon, Portugal, pp. 67-74, 2007, ISBN 978-972-8924-39-3.

Petrik S., Skala V.: Iso-contouring in Time-varying Meshes. *Proceedings of SCCG 2007*, Budmerice, Slovakia, pp. 216-223, 2007, ISBN 978-80-223-2292-8.

Technical Reports

Petrik S., Skala V.: Report on Videoconferencing Systems, *Technical Report No. DCSE/TR-2006-04*, University of West Bohemia in Plzen, Czech Republic, June 2006. Updated on May 18, 2007.

Other publications

Vokorokos L., Petrik S.: Proposed Parallel and Distributed Architectures for Behavioral Animation. *Proceedings of 9th IEEE International Conference on Intelligent Engineering Systems*, Cruising on Mediterranean Sea, 2005.

Vokorokos L., Blišťan Peter, Petrik S., Ádam N.: Utilization of Parallel Computer System for Modeling of Geological Phenomena in GIS. *Metalurgy Journal*, vol. 43, no. 4, pp. 287-291, 2004.

Vokorokos L., Petrik S., Ádam N.: Aplikácia 3D informačných systémov v oblasti prevádzky výrobných technológií. *Proceedings of the 6th International Scientific Conference: New Trends in the Operation of Production Technology 2003*, Prešov, Slovensko, pp. 426-431, 2003.

Project assignment

- 2006 - present: *VIRTUAL* - Virtual Research-Educational Center of Computer Graphics and Visualization, MSMT Czech Rep. No: 2C 06002, http://virtual.zcu.cz,
- 2005 / 2006: *3DTV* - Integrated Three-Dimensional Television - Capture, Transmission and Display, FP6-2003-IST-2, Network of Excellence, No:511568, http://3DTV.zcu.cz,
- 2005 / 2006: *INTUITION* - Network of Excellence on VIrtual Reality aNd VirTUal Environments ApplIcaTIONs for Future Workspaces, FP6-2003-IST-2, No:507248-2, http://intuition.zcu.cz.

Teaching activities

2005/2006:

- KIV/WIN Introduction to programming in MS Windows
- KIV/ZIT Introduction to information technologies