University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitní 8
306 14 Plzeň
Czech Republic

Iso-Surface Extraction and Approximation Error

State of the Art and Concept of Doctoral Thesis

Jan Patera

Technical Report No. DCSE/TR-2004-15
October, 2004

Distribution: Public

Technical Report No. DCSE/TR-2004-15
October, 2004

Iso-Surface Extraction and Approximation Error

Jan Patera

Abstract

At present, many devices produce the volume data as their output. The well-known data acquisition tools are e.g. MRI, CT or PET scanners, which are used mainly in the medical field. The volume data can represent density, velocity, humidity etc. in the form of structured or unstructured grid. Scientists need to explore such data to study their inner properties and relations and to make decisions. The volume data visualization is a strong tool for such explorations. There are many methods for the volume data visualization. The volume rendering and the surface rendering approaches are governing two main visualization branches in this field. The volume rendering methods visualize the data as a whole. In the apart of that, the surface rendering methods are trying to find surfaces in the volume data and visualize them.

The group of methods for the surface rendering contains also methods for the iso-surface extraction from the volume data. This thesis provides a reasonable overview of the main techniques that serve for the volume data acquisition as well as the state of the art in the field of the iso-surface extraction methods. The important methods are sensibly described together with appropriate references given and their difficulties, pros and cons are discussed as well. The offered work also concerns a research that was recently done and the outline of possible directions in which the future research can head towards.

This work was supported by the Ministry of Education of the Czech Republic – project MSM 235200005.

Copies of this report are available on
http://www.kiv.zcu.cz/publications/
or by surface mail on request sent to the following address:

University of West Bohemia in Pilsen
Department of Computer Science and Engineering
Univerzitní 8
306 14 Plzeň
Czech Republic

Index

1 Introduction

1.1 Overview

In recent period of time the volume data play a significant role in many scientific areas. The volume data are spread across many professions we can imagine. In the medical field, various devices such as CT scanners, MRI scanners, etc. produce the volume data. The volume data are also produced as a result of mathematical or physical simulations. Various devices use ultrasonic tester that can produce the volume data as an output as well.

Each year new technologies are investigated and the existing devices are improved. Also devices that produce the volume data are improved due to acquisition speed, accuracy and other parameters. Higher accuracy implies larger resolution to be used during the data acquisition process. The higher resolution is used the bigger volume data are generated. At present (year 2004) the acquisition devices provide thousands samples in each direction in a 3D space and a resulting volume data file can reach terabytes in size.

As there are many various ways of how to obtain the volume data, there are also various kinds of volume data sets. Two main groups of the volume data are roughly identified as the structured volume data and the unstructured volume data.

The structured volume data have easier data structures and adjacency among samples is easy to find due to regularity. Hence from a sample index we can easily find its coordinates. The passing is a simple matter and the structured volume data requires less memory space per data sample. These data are at most produced by the medical scanners. On the other hand, the unstructured data have more complex data structures, higher memory requirements but provide more freedom for scientists to represent complex environments and perform complicated simulations. Our interest is devoted to the structured volume data.

One pretty good thing on the volume data is that its samples are taken not only on a surface of an object but also from its interior. Thus volume data characterizes the whole object. Samples can represent e.g. temperature, density and flow speed or air humidity at an appropriate position. To understand the underlying relations, structure and properties of an object we need to visualize the volume data. The visualization must be reasonable fast, because slow visualization decreases our perception and approximately 75% of information we get is accepted visually.

There are two main techniques for the volume data visualization. The first approach is based on volume rendering (the ray tracing like methods) and the second one on surface rendering (the iso-surface extraction like methods). As known, the volume rendering methods are complex and works with the whole volume data. These methods are out of scope of this work. The surface rendering methods visualizes surfaces that are stored in the volume data, the iso-surface extraction methods are used to find such surfaces and their overview is given later.

The field of the iso-surface extraction is quite large. There are various methods used for the extraction such as view dependent techniques, parallel or distributed

approaches, external memory (or sometimes called I/O) techniques, multiresolution (LOD) based extraction, etc. We are interested in standard (single processor) iso-surface extraction methods which find active cells and extract an iso-surface from them. The state of the art of such methods is presented in the first part of this work. In general, we can describe the iso-surface generation and visualization with the following steps:

- Search for all active cells (such cells that are intersected by the iso-surface)
- The iso-surface and normal vectors approximation within these cells (using a set of triangles)
- Iso-surfaces visualization (visualization of a set of triangles; different iso-surfaces can be visualized with different colors depending on a selected threshold value, alpha blending, etc.)

Let us make a remark about Shannon's sampling theorem. A band (frequency) limited function is a function whose Fourier frequency spectrum is limited by some frequency f_{MAX}. The functions with higher frequencies do not contribute to the total function value (they have zero amplitude). The Nyquist's criterion equals to $2*f_{MAX}$. Shannon's sampling theorem says that each band limited function that is sampled with higher frequency then the Nyquist's criterion is exactly represented with such samples (and can be fully reconstructed – theoretically). When a function is sampled under the Nyquist's criterion it comes to aliasing.

The volume data are in most cases sampled under the Nyquist's criterion (aliasing). Of course that the volume data can be in a special case (depends on a scanned object) sampled above Nyquist's criterion frequency, thus without aliasing, but the resulting data set would have the extremely large size. Note that a sampled object is represented in most cases with some error. The scan error size depends both on an object scanned and on a resolution used.

Also the extracted iso-surface is not in computer graphics represented exactly but with an error. As everybody can imagine an iso-surface can be extracted with several different methods, all methods approximate the iso-surface with some geometrical primitives such as triangles, quadrilaterals or patches and use different interpolation of the data values inside of a cell such as linear or trilinear interpolation. An error of the iso-surface depends on all mentioned factors that are used method, used interpolation and used geometrical primitives. The error behaviour is discussed in the second part of this work. This error is the area of our interest.

The total approximation error consists of the sampling error and the iso-surface approximation error. In our work we discuss only the total approximation error, which is the sum of both previously mentioned errors.

1.2 Problem Definition

The problem of the iso-surface extraction is clearly defined in [25] by Pasko et al. Let the continuous real trivariate function $\xi = f(x,y,z)$ be defined for the domain $x \hat{I} X = [x_1, x_2]$, $y \hat{I} Y = [y_1, y_2]$ and $z \hat{I} Z = [z_1, z_2]$. The following problem of analysis of this function will be considered: for any given value $\xi = c$ the properties of the preimage $f^{-1}(c) = \{x, y, z: x \hat{I} X, y \hat{I} Y, z \hat{I} Z, f(x,y,z) = c\}$ need to be investigated. The geometric model of the domain is parallelepiped in the space *xyz*. The function $\xi = f(x,y,z)$ together with the domain *xyz* is geometrically interpreted as a hypersurface

(for $\xi = c$) in the $xyz\xi$ space. The geometric model of the preimage $f^{-1}(c)$ is the surface S, in xyz domain, which is usually called an **iso-surface**.

Furthermore, in our case, we do not know exactly the function $\xi = f(x,y,z)$. We have only its samples in e.g. regular grid. The domain xyz can be then subdivided with respect to the regular grid into a set of parallelepiped cells. The task is to find a surface S and visualize it.

1.3 Organization

At first, a brief description of basic volume data acquisition techniques, concretely CT, MRI and PET, is given. The ultrasound technique is omitted because it is not used as often as mentioned three techniques. Afterwards, the overview of existing methods in the field of the iso-surface extraction is made. In the next chapter the comparison of a set of methods is made. The comparison concerns different aspects such as an error of the iso-surface approximation, the iso-surface area, etc. And finally some conclusions and future work are presented.

1.4 Basic Terms

Volume data – is a set of samples. Samples are placed in arbitrary grid vertices which are called nodes (see Section 3.2).

Threshold (iso-value) – is in most cases a user specified value, but this process can be also automated in special cases. A threshold determines the value that we want to visualize in volume data. This value is visualized e.g. in a form of iso-surface. The threshold can represent e.g. pressure, density or temperature, etc.

Iso-surface – is a surface in the volume data such that the data values on it are constant. The data values on the iso-surface and hence also the iso-surface itself are determined by a threshold value (threshold in short). The process for finding an iso-surface is called the iso-surface extraction and the iso-surface is approximated with a set of primitives, in most cases triangles.

Positive node – a grid node is said to be positive if its data value after a threshold subtraction remains positive.

Negative node – is a node whose data value is negative after a threshold is subtracted from it.

Local coherence – is some kind of a relation among adjacent nodes or cells.

Global coherence – is a relation between the two different iso-surfaces that are defined with two close but distinct threshold values.

Here we would like to mention used symbols in mathematical expressions and equations:

c	scalar value
$\mathbf{c} = (c_1, c_2, c_3)$	vector or point
$\mathbf{C}$	matrix or point
$f(x,y,z)$, $F(x,y,z)$	function of three variables

log(*x*), **ln**(*x*)	mathematical functions
$\lfloor c \rfloor$	integer value of *c* (same as **Floor**(*c*); Examples: **Floor**(2.4) = 2, **Floor**(-2.1) = -3, **Floor**(2.6) = 2)
(*x,y,z*)	point or vector coordinates
[*a;b*]	closed interval
(*a;b*)	open interval
% or **mod**	modulo division
F, A, B, …	constant numbers

Table 1 – Used mathematical symbols overview

The introduction to volume data and to the problem of the iso-surface extraction was made as well as the overview of this work and basic terms description. The next chapter is focused on the overview of basic methods for volume data acquisition.

2 Hardware for Volume Data Acquisition

In this chapter, basic volume data acquisition methods (Computed Tomography, Magnetic Resonance Imaging and Positron Emission Tomography) are overviewed. Of course, there are other methods or modifications of mentioned ones. These are not included in this chapter (e.g. ultrasound). The purpose of this chapter is to give the brief and basic overview in this area.

2.1 Computed Tomography – CT

History

In 1972 Computed Tomography (also known as Computed Axial Tomography) was used in the medical area for the first time by Hounsfield and Cormac. It is based upon X-ray attenuation measurement at object's cross-sections (slices). The volume data itself are afterwards reconstructed and filtered from these measurements. The word tomography comes from the Greek words "tomeo" meaning "cut" and "graphia" meaning "describing". For more information, see [W5], [W7] or [22].

Basic Principle

Let's consider X-ray tube, X-ray detector and an object between them. We know the intensity of the X-ray in the source tube and, we can measure, intensity in the sensor after X-ray passes the object. From these two values we can determine μ as an average linear attenuation coefficient, which describes the reduction of X-ray by object's material (therefore it is the property of material). The X-ray intensity decrease depends on object's proton number, density and thickness as well, Figure 1.

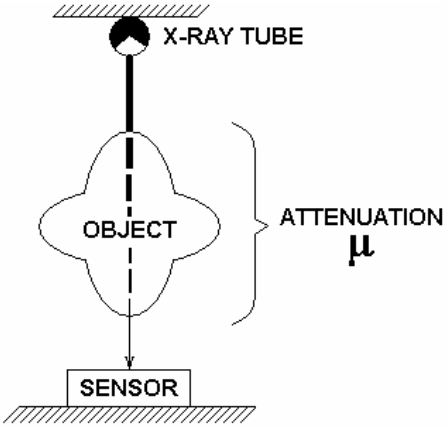


Figure 1 – Attenuation μ

Slice Acquisition

Imagine we have an X-ray tube and many X-ray detectors (thousands), which are situated in the arc of a circle, see Figure 2. Fan beam of X-rays, aimed onto detectors, is produced by the X-ray tube. The radiation is registered by detectors after object passing and this information is furthermore processed by a computer. During slice exposure, both the X-ray tube and detectors are rotating around object 360^{o} degrees (thousands measurements). Rotation time is usually up to 10 seconds. From gained attenuation coefficients, which pass an object's slice in different directions, we reconstruct an object by a reverse process to a performed acquisition – filtered back projection (Radon transform and Fourier transform are used).

Samples

The acquired samples in volume data grid are called *CT numbers*. The coefficient μ itself describes the linear attenuation between the X-ray tube and an X-ray detector and it is dependent on the object density (atomic number). *CT numbers* are related to the linear attenuation of the water, as given by Equation (1).

$$CT\_number = \frac{m_{TISSUE} - m_{WATER}}{m_{WATER}} * 1\,000 \qquad (1)$$

For example the water has *CT number* equal to 0 ($m = 0.181\ cm^{-1}$), air = -1 000 ($m = 0.0003\ cm^{-1}$), bone ≈ +1 000 ($m = 0.46\ cm^{-1}$). *CT number*s are also known as Hounsfield units. Artifacts in resulting set of samples can arise because of patient movement or metal objects (tooth filling) inside of the CT scanner.

CT Examination

The CT examination itself begins with the object overview – the object is scanned in a gantry (Figure 2), but without the X-ray tube rotation. Afterwards the proper area of interest is selected and a complete CT scan begins. The contrasting substance can be applied before examination.

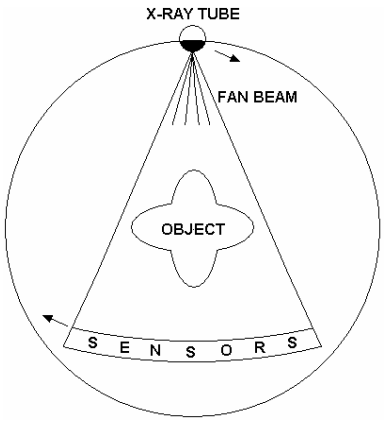
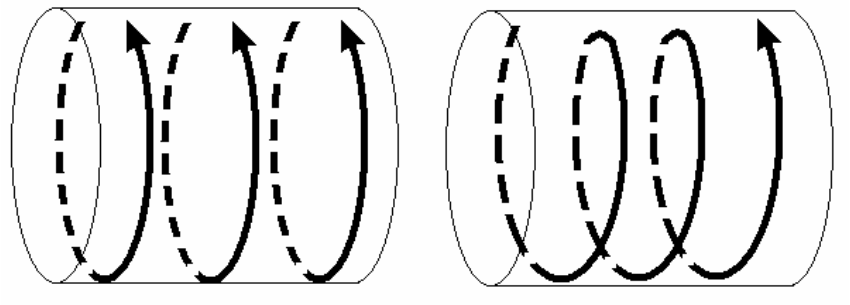


Figure 2 – The gantry

There are two basic ways how can the X-ray tube with detectors rotate in 3D, the first is conventional CT and the second is spiral CT, see Figure 3. Both these techniques produce slices in axial plane. When conventional CT is performed, one slice of an

object is scanned and table with object then moves about constant distance, after that another scan is done, etc. So we gain equidistant slices of whole examined object.

In the other hand spiral CT is based on the continuous table movement and scanning, so the trajectory of X-ray tube is spiral. Complete set of slices is then used for arbitrary plane or 3D reconstruction. The main advantages are fast examination of whole object due to continuous scanning and possibility to create overlapping slices for 3D reconstruction.



a) b)

Figure 3 – Conventional a) and spiral b) scans

2.2 Magnetic Resonance Imaging – MRI

History

Magnetic resonance was discovered by Bloch and Purcell independently in 1946. Both were awarded Nobel Prize in 1952. After the introduction of Computed Tomography in 1972, hospitals were able to buy such expensive medical imaging hardware. This fact started a fast investigation of MRI, which is based on (nuclear) magnetic resonance. Word nuclear was omitted, because of its unpopularity in 1970s. Lauterbur, in 1973, demonstrated magnetic resonance plus back projection (CT) to create images. Current MRI data acquisition method was used in 1975 by Ernst. More on MRI is in [W9] and [W6].

Basic Principle

MRI is based on protons magnetic field modification by radio pulses. Atomic nucleus rotates around its rotary axis (spin), and this produces a small magnetic field for nucleuses with odd proton number. The field intensity depends on a rotation speed and a nucleus charge size. Hydrogen H^1 has one proton in nucleus, odd proton number and many objects contain it. Many protons in human body are stored in water. Up to 60% of body weight is the water, depending on age and sex. Proton does have a small mass, but apart of that it rotates very fast, so it produces noticeable magnetic field we are able to measure. Therefore, hydrogen is widely used in MRI.

Normally, the proton rotary axes are pointing in random directions. At first the examined tissue is inserted into static magnetic field (up to 2 Tesla). All protons inside of it are getting aligned with this field; some of them in the direction, the others against it. The number of protons aligned with static field is always slightly greater than the number of protons aligned against it. The stronger the field is the greater amount of protons is aligned in the field direction, see brilliant example in [W9]. The resulting

protons magnetic field direction is then pointing in the same direction as the static field does, because of excess protons aligned in its direction.

When the protons are aligned, they perform two kinds of movement, see Figure 4. First, the proton rotates around its rotary axis (spin). Second, the proton rotates around imaginary cone's axis (precess). Precess frequency depends on magnetic field strength.

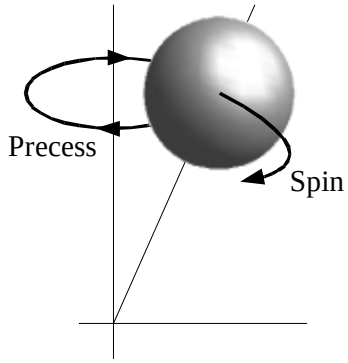


Figure 4 – Spin and precess of the proton

When the radio frequency (RF) pulses are applied at the frequency of precesses (resonance frequency) proton magnetic moment changes the angle and precesses of all protons are synchronized. After RF pulses are finished, all protons are getting to their original state and the needed time is called relaxation time. The time needed to restore original magnetic moment position is called relaxation time T1 and the time needed to unsynchronize previously synchronized precesses is called relaxation time T2. The T1 and T2 are both dependent on a material, where the examined protons are placed. These times can not be measured directly, instead of that, times are measured relatively by their comparisons.

During T1 and T2, when RF pulses are switched off, the protons retransmit RF pulses at the resonance frequency and we can measure this energy with special induction coils. Moreover, the resonance frequency depends on the magnetic field and it can be locally modified by gradient coils. Therefore, we can locate the source of retransmitted RF pulses on the basis of resonance frequency.

2.3 Positron Emission Tomography – PET

History

The first PET camera for medical purposes was built by Hoffman, Ter-Pogossian and Phelps in 1973. Four years later (1977) the full-body PET scanner was constructed. During the following years PET resolution and sensitivity was improved, see example images in [W3]. CT or MRI scanners, both acquire the object structure information only, PET images describe chemical functioning of an object as shown in [W1].

Basic Principle

The PET is used for life objects only, because PET images describe not a structure, but chemical functioning (metabolism) of an object. A special substance, which is called a

tracer, is introduced to a patient. Tracer is a substance, which has an excess number of protons; therefore it emits positrons (positive antiparticle to an electron). The tracer exists in a body for few hours and it still emits positrons. When a positron is emitted, it travels 1-3 mm and then it annihilates with an electron. After the annihilation two collinear γ-rays are produced; each of them aims at opposite direction (both of them lay at the same line). Highly precise sensors are capable to detect these γ-rays and estimate the distance, which both of them traveled.

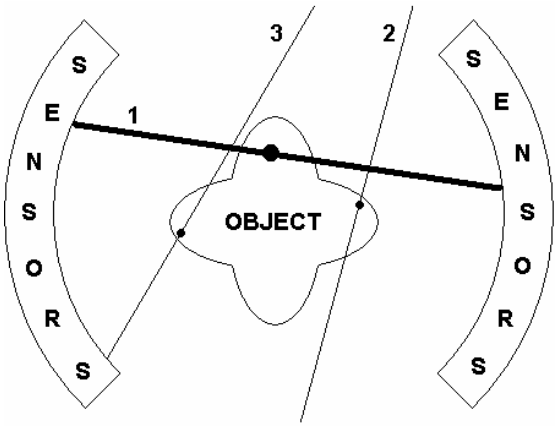


Figure 5 – Annihilation and γ-rays (1 … accepted event, 2 and 3 ... rejected events)

Few situations can occur as evident from Figure 5:

- both γ-rays are recorded by sensors (accepted event)
- only one of γ-rays is recorded (rejected event)
- none of γ-rays is recorded (rejected event)
- more than one annihilation occurred within the same time window (rejected event)

As mentioned at the beginning of this chapter, there are also other methods for volume data acquisition such as ultrasound [22], etc. These methods are out of scope of this text. The next chapter deals with volume data.

3 Volume Data

In following chapter a voxel and a cell are described, volume data lattices types are shown and also the curvature computation in any voxel is described together with the appropriate reference given.

3.1 Voxel and Cell

The voxel (Figure 6) is a 3D representation of a pixel and is based on the nearest neighbor interpolation of spatial samples. Therefore, the interior of whole voxel has the same value. We can imagine voxel as a cube. In the apart of that, the cell is composed of 8 adjacent spatial samples and an arbitrary interpolation can be used to gain interior values of the cell. The cell can be drawn as a cube as well.

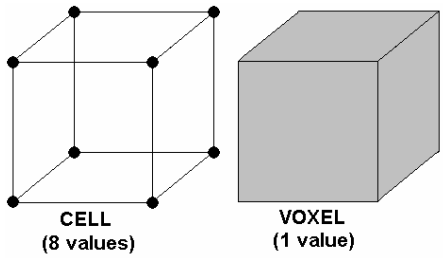


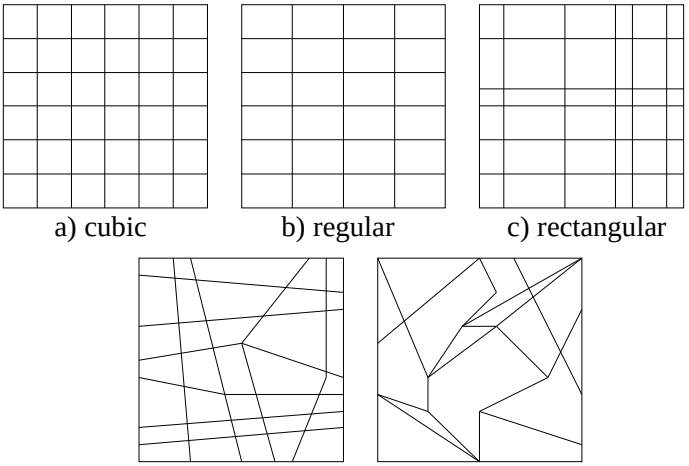
Figure 6 – The voxel and the cell

3.2 Grids Types

There are several categories [38] of grids, which are used to store volume data samples. The main branch is stated bellow (Figure 7):

- Cubic Grid – (Cartesian, equidistant) all the cells have the same cube shape.
- Regular Grid – (anisotropic rectilinear, uniform) all the cells have the same block shape.
- Rectangular Grid – (rectilinear, orthogonal) all the cells in a row or a column share faces but they can have different block shape.
- Structured Grid – (non-rectilinear, curvilinear) all the cells can have different shape, but topologically it is cubic grid. It is necessary to store vertex coordinates for all vertices.
- Unstructured Grid – all the cells can have an arbitrary shape (polyhedral, hexagonal, pyramidal, etc). It is necessary to store vertex coordinates for all vertices.
- Block Structured – more types of different structured grids in the same volume data set.

- Hybrid Grid – more types of structured and unstructured grids mixed together in the same volume data set.



a) cubic b) regular c) rectangular

d) structured e) unstructured

Figure 7 – 2D outline of grid types

3.3 Volume Data Curvature

Another thing that we would like to mention in this chapter is curvature computation in each voxel. We would like to utilize this to develop a new iso-surface extraction algorithm as stated in the chapter 6. That is the main reason why we mention the voxel curvature computation in this work.

Kindlmann et al. [13] uses the volume data curvature to construct a transfer function in volume rendering and the authors also propose the use of curvature for uncertainty visualization when extracting the iso-surface.

The process that is needed to compute the curvature at any point in a 3D scalar field can be summarized as follows:

1. Measure the first partial derivatives to obtain the gradient **g**. Gradient **g** is computed as $\mathbf{g} = \nabla f(x, y, z) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)$ and the normal vector **n** as $\mathbf{n} = -\frac{\mathbf{g}}{\| \mathbf{g} \|}$. Compute the matrix **P** as $\mathbf{P} = \mathbf{I} - \mathbf{n} \otimes \mathbf{n}^T$, where ⊗ represents vector direct product (the result is matrix).
2. Measure the second partial derivatives to construct the Hessian matrix **H** and compute a matrix **G**. Hessian matrix has the form $\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial x \partial z} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} & \frac{\partial^2 f}{\partial y \partial z} \\ \frac{\partial^2 f}{\partial z \partial x} & \frac{\partial^2 f}{\partial z \partial y} & \frac{\partial^2 f}{\partial z^2} \end{bmatrix}$ and then compute the matrix **G** as $\mathbf{G} = -\frac{\mathbf{PHP}}{\| \mathbf{g} \|}$.

3. Compute the trace of **G** [W11] as $Tr(\mathbf{G}) = \sum_{i=1}^{n} \mathbf{G}_{ii}$ and Frobenius norm of **G** [W11] as $\|\mathbf{G}\|_F = \sqrt{Tr(\mathbf{G}\mathbf{G}^T)}$. Using trace and Frobenius norm, compute the requested curvatures κ_1 and κ_2 as $k_{1,2} = \frac{T \pm \sqrt{2F^2 - T^2}}{2}$.

The matrix **P** projects into the tangent plane that is defined by **n**, Hessian matrix **H** characterizes how the **g** changes with small changes in position.

The authors also perform a three dimensional data value reconstruction at an arbitrary position as a multiplication of three one dimensional convolutions in *x, y* and *z* directions. The convolution in x direction is defined as

$$k(t) = f(x) * g(x) = \int_{-\infty}^{+\infty} f(t-t)g(t), \tag{2}$$

where $f(x)$ represents the object in volume data and $g(x)$ is convolution kernel.

Convolutions in other directions are defined similarly. Authors of [13] claim that the first order partial derivatives in *x, y* and *z* directions can be constructed using the first order derivative of the convolution filter, as outlined in Figure 8. The convolution formula is then defined as

$$k'(t) = f(x) * g'(x) = \int_{-\infty}^{+\infty} f(t-t)g'(t), \tag{3}$$

where $f(x)$ represents the object in volume data and $g(x)$ is derivative of the convolution kernel.

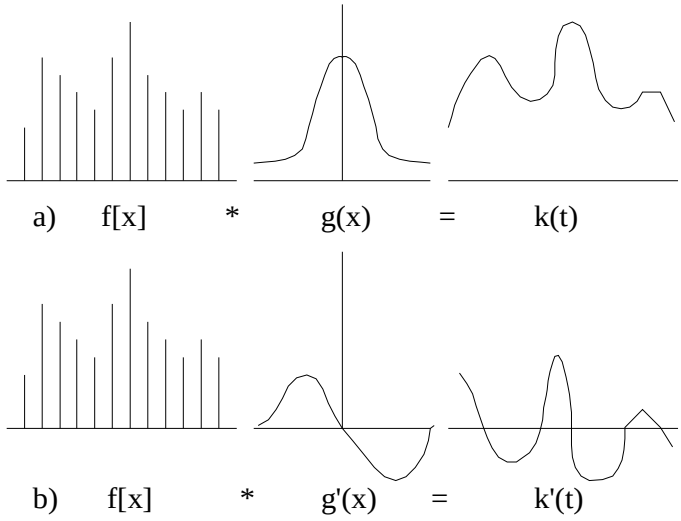


Figure 8 – Derivative calculation via derivative of the convolution filter outline (f[x] … discrete samples in x direction, g(x) … convolution filter and k(t) … reconstructed function)

As mentioned the data values interpolation is made using multiplication of convolution filters in *x, y* and *z* directions. The combination of convolution filters in multiplication determines the result. The authors compute the second order derivative of the function $f(x,y,z)$, $\partial^2 f / \partial x \partial y$ as a multiplication of the first order derivative filters of f in x and y

directions and zero order filter in *z* direction, etc. The accuracy of the interpolation depends mainly on the quality of chosen filters and their combination.

The authors of [13] use the curvature computation also in the field of iso-surface extraction to visualize the flowline curvature κ_f. Flowline curvature characterizes the degree to which an iso-surface changes its shape as a function of small changes in threshold. When the $\kappa_f = 0$, the change between adjacent iso-surfaces shapes, is zero and they are parallel. It describes the change between iso-surfaces rather than a change of one iso-surface only. In special orthonormal basis (tangent plane basis and its normal **n**) [13] it holds that

$$\nabla \mathbf{n} = -\frac{1}{\| \mathbf{g} \|} \mathbf{PH} = \begin{bmatrix} k_1 & 0 & s_1 \\ 0 & k_2 & s_2 \\ 0 & 0 & 0 \end{bmatrix} \tag{4}$$

Generally the curvatures are not isolated as in (4). The flowline curvature can be computed using Equation (4) again with the use of a Frobenius norm as ([13])

$$k_f = \left\| \nabla \mathbf{n} (\mathbf{n} \otimes \mathbf{n}^T \right\|_F = \sqrt{s_1 + s_2} \tag{5}$$

If we move along the normal, off of the surface or deeper into it, the normal tilts according to σ_1 and σ_2.

Kindlmann et al. claim, in their paper, that the flowline curvature can be used for uncertainty visualization within the iso-surface (e.g. by a color mapping). The term uncertainty means that the iso-surface is in some places good approximation of the material boundary and in some places the iso-surface approximates the boundary of the material with less trustworthiness (due to significant changes in its shape with tiny changes in the threshold). These places are indicated with high flowline curvature.

Main facts concerning volume data were mentioned. The volume data are further processed with extraction methods. Most of methods work with the set of cells instead of volume data. These methods are reasonably described in the following part of this work.

4 Iso-Surface Extraction Methods Survey

This chapter is the main part of this work. It covers methods for the iso-surface extraction from regular volume data and their speedup techniques. In previous chapters we discussed volume data acquisition methods and volume data lattices. The next step for volume data visualization is, in the case of iso-surfaces, the iso-surface extraction. Mentioned methods generate an iso-surface in a form of a set of triangles, that is further visualized using conventional methods.

Word iso comes from Greek word isos meaning equal. Iso-surface connects points of equal value (temperature, tense, altitude, pressure, etc.).

4.1 Marching Cubes

This method was originally published by Lorensen and Cline in 1987, see [17]. It can be thought as an extension of the 2D Marching Squares method (iso-lines extraction) to 3D. Marching Cubes (MC abbreviation will be used) has O(N) time complexity, where N is the number of all cells which can be created from given volume data. Inputs are the threshold value and the structured scalar volume data. Output is a set of triangles which is an approximation of the iso-surface. MC sequentially processes all cells which can be constructed from the volume data. Main steps of MC algorithm are (for one cell):

- Cell construction from volume data samples
- Comparison of 8 cell vertices values with a threshold and an index computation (8-bit)
- Normal vectors approximation in the cell vertices
- The use of index to find all intersected edges
- Triangles vertices and normal vectors approximation at all intersected edges

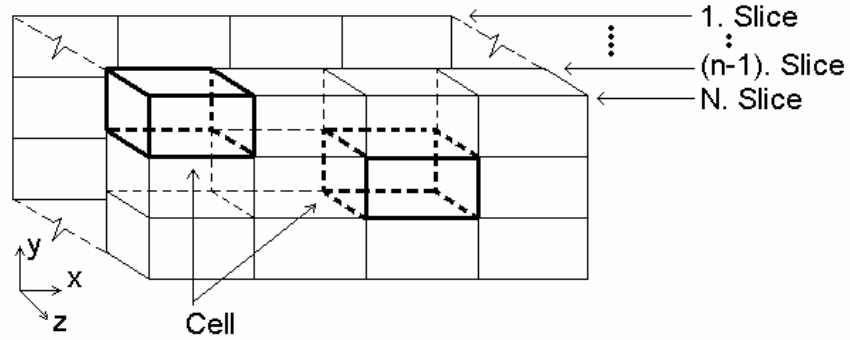


Figure 9 – The cell in structured volume data

A cell has a cube shape and is easily created from the structured volume data. Appropriate four samples from two adjacent slices form the cell see Figure 9. The index is then created as an 8-bit binary number. Each cell has 8 vertices and each vertex

corresponds to one bit in index. Bit is set to 0 if the sample value in corresponding vertex is lower than threshold value (negative node) and vice versa (positive node). If the index (Figure 10) is 0 or 255 the cell is not intersected by the iso-surface. In total, we have 256 possibilities how an iso-surface can intersect our cell. All 256 possibilities can be reduced into 15 basic cases because of negation of index, rotation of a cell or symmetry, see Zara et al. [38] or Figure 11. The computed index points into a triangle table, where triangle configurations are stored. The triangle configurations table stores intersected cell edges for the given index and appropriate iso-surface approximation (inside the cell) by triangles. Two adjacent cells share 4 edges (one cell wall), so a generated set of triangles is linked properly between these cells. MC generates up to 4 triangles per cell, which means that for higher data resolutions the number of generated triangles grows quickly. A cube cell has 12 edges which can be intersected by an iso-surface.

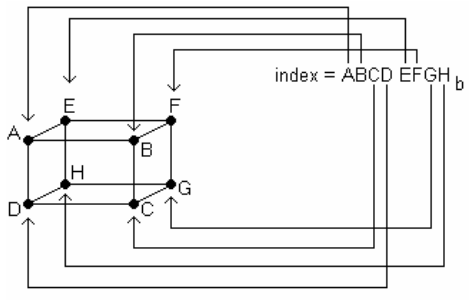


Figure 10 – The 8-bit index creation

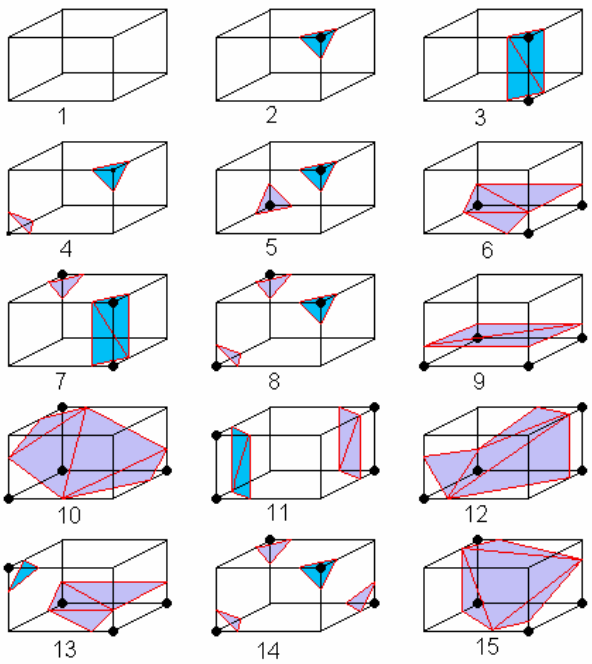


Figure 11 – All 15 basic cases in MC

An iso-surface lies between two different volumes. Therefore, normal vectors can be approximated with data gradients $\nabla f = \mathbf{g} = (g_0, g_1, g_2)$ and calculated using symmetric difference. For an inner sample (does not lie on the border of volume data) with coordinates $\mathbf{x}_i = (x_i, y_i, z_i)$ the normal vector $\mathbf{g}$ is calculated as:

$$g_0 = \frac{f(x_i + a, y_i, z_i) - f(x_i - a, y_i, z_i)}{2a}$$
$$g_1 = \frac{f(x_i, y_i + b, z_i) - f(x_i, y_i - b, z_i)}{2b} \quad (6)$$
$$g_2 = \frac{f(x_i, y_i, z_i + c) - f(x_i, y_i, z_i - c)}{2c}$$

Where *a*, *b* and *c* are distances of adjacent samples in *x*, *y*, and *z* directions; *f(x,y,z)* is the value of the sample with coordinates *x*, *y* and *z*. When calculating normal vectors in samples that are stored at the volume data boundary, the one side difference is used.

In the triangle table there are stored intersected edges corresponding to a computed cell index. Linear interpolation is used to find accurate coordinates of triangles vertices at the intersected edges:

$$\mathbf{x}_P = \mathbf{x}_A + \frac{\mathbf{x}_B - \mathbf{x}_A}{f(\mathbf{x}_B) - f(\mathbf{x}_A)}(threshold - f(\mathbf{x}_A)) \quad (7)$$

Where $\mathbf{x}_A$ and $\mathbf{x}_B$ are coordinates of both edge ends; $f(\mathbf{x}_A)$ and $f(\mathbf{x}_B)$ are samples values in these ends; $\mathbf{x}_P$ is the interpolated position of the triangle vertex. The division by zero can not arise, because such an edge is not intersected by any iso-surface.

Normal vectors in the cell vertices are already calculated see Equation (6). To calculate normal vectors in triangle vertices that lie at the intersected edges, linear interpolation is used again:

$$\mathbf{q} = \mathbf{a} + \frac{(\mathbf{b} - \mathbf{a})}{f(\mathbf{b}) - f(\mathbf{a})} * (threshold - f(\mathbf{a})) \quad (8)$$

Where **a** and **b** are normal vectors at the edge ends; $f(\mathbf{x}_A)$ and $f(\mathbf{x}_B)$ are data values that are stored in the edge ends; **q** is the resulting normal vector in the triangle vertex.

MC is a simple method for the iso-surface extraction with minimal memory requirements for computation. It generates relatively low number of triangles when compared to Marching Tetrahedra (Section 4.3), Centered Cubic Lattice (Section 4.4) and others. Marching Cubes method sometimes produces holes in resulting set of triangles due to ambiguity (Figure 15), in Schoeder et al. [29], there are shown some approaches how to solve this problem. Overview of possible approaches is also given in [24] by Ning and Bloomenthal.

4.1.1 Local Coherence Speedup

A local coherence speedup (3D cell buffer) can be used for Marching Cubes, Marching Tetrahedra 5 and Marching Tetrahedra 6 methods. For Centered Cubic Lattice it can be used as well, but implementation is rather complicated.

The local coherence speedup takes advantage of already computed results from adjacent cells, it skips redundant computations. This technique can be implemented via a 2D cell buffer as shown in Figure 12. The buffer stores one slice of cells (cell indexes, interpolated normal vectors, interpolated triangles vertices on edges, etc.) Therefore, its memory requirements depend on the volume data resolution. The cell buffer decreases total number of interpolations. The speedup is more noticeable for the inner cells then for the border ones. The inner cells can undertake results computed previously from left, bottom and previously stored cells.

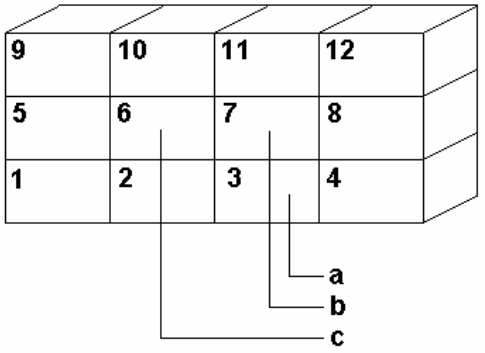


Figure 12 – Cell buffer speedup overview (a...a cell situated UNDER the actually processed cell, b...actually processed cell will be stored here, c...a cell situated on the left side of the actually processed cell)

Furthermore, the computation time can be reduced by placing triangle vertices to the edge center and edge end points instead of time consuming interpolations. However, the quality (mainly approximation accuracy and visual effect) of the output set of triangles will be lower. Such approach is suitable for the fast iso-surface preview. Such algorithm is called Discretized MC and was published by Montani et al. in [20].

4.1.2 More Accurate Normal Vector Estimation

The typical approximation of the normal vector inside the volume data is shown in Equation (6). To get better normal vector approximation we can include other adjacent samples to the normal vector computation process, [27]. In the following equation, the 2D diagonal differences are included. One sided difference can be used as well, for simplicity we consider unit cube cells:

$$\begin{aligned}
\mathbf{g} = \nabla f(x_i, y_i, z_i) = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right] \approx \frac{1}{2}\left(f(x_i+1, y_i, z_i) - f(x_i-1, y_i, z_i)\right)\mathbf{i} + \\
+\frac{1}{2}\left(f(x_i, y_i+1, z_i) - f(x_i, y_i-1, z_i)\right)\mathbf{j} + \frac{1}{2}\left(f(x_i, y_i, z_i+1) - f(x_i, y_i, z_i-1)\right)\mathbf{k} + \\
+\frac{1}{2\sqrt{2}}\left(f(x_i+1, y_i+1, z_i) - f(x_i-1, y_i-1, z_i)\right)(\mathbf{i}+\mathbf{j}) + \\
+\frac{1}{2\sqrt{2}}\left(f(x_i+1, y_i-1, z_i) - f(x_i-1, y_i+1, z_i)\right)(\mathbf{i}-\mathbf{j}) + \\
+\frac{1}{2\sqrt{2}}\left(f(x_i+1, y_i, z_i+1) - f(x_i-1, y_i, z_i-1)\right)(\mathbf{i}+\mathbf{k}) + \\
+\frac{1}{2\sqrt{2}}\left(f(x_i+1, y_i, z_i-1) - f(x_i-1, y_i, z_i+1)\right)(\mathbf{i}-\mathbf{k})
\end{aligned} \tag{9}$$

Where $\mathbf{x}_i = (x_i, y_i, z_i)$ is *i*-th sample; $f(\mathbf{x}_i)$ is the sample value; **i**, **j** and **k** are identity vectors in *x*, *y* and *z* directions. Also 3D diagonal differences can be used with appropriate weight, which is the length of a 3D diagonal, Figure 13. Note, that not all differences from Equation (9) are drawn.

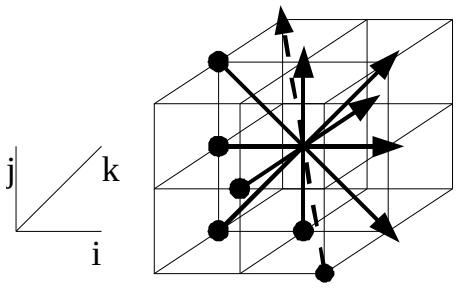


Figure 13 – Improved normal vector estimation

4.2 Improved Marching Cubes

Chernyaev [3] discusses an algorithm for the iso-surface construction from the volume data. This algorithm approximates the searched iso-surface with respect to trilinear function. The values of coefficients of trilinear function are derived from the samples values in the cell corners, see Equation (10).

$$\begin{aligned} F(x,y,z) = {} & F_{000}\,(1-x)(1-y)(1-z) \\ & + F_{100}\,x\,(1-y)(1-z) \\ & + F_{010}\,(1-x)\,y\,(1-z) \\ & + F_{110}\,x\,y\,(1-z) \\ & + F_{001}\,(1-x)(1-y)\,z \\ & + F_{101}\,x(1-y)\,z \\ & + F_{011}\,(1-x)\,y\,z \\ & + F_{111}\,xyz \end{aligned} \tag{10}$$

When the unit cell is moved to the origin of coordinate system, then each cell corner has local coordinates and F_{iii} represents the sample value in the appropriate cell corner (i = 0 or 1). $F(x,y,z)$ is trilinear interpolation of samples values in the cell vertices (x, y, z are from [0; 1] interval), so the data value at any point inside of the cell can be evaluated.

For some kinds of cells, the iso-surface can intersect them in more than one way, but in MC look up table, there is stored only one possibility, see [3] or Figure 16 for such an ambiguous case. If all variants are taken into account, the lookup table should be increased from 15 basic cases to 33. For ambiguity cases solution Chernyaev uses the trilinear function $F(x,y,z)$. The behavior of function F in any plane, which is parallel to a cell face and intersects this cell, is bilinear. The face ambiguity (Figure 14) can be then solved in the following manner, as described below.

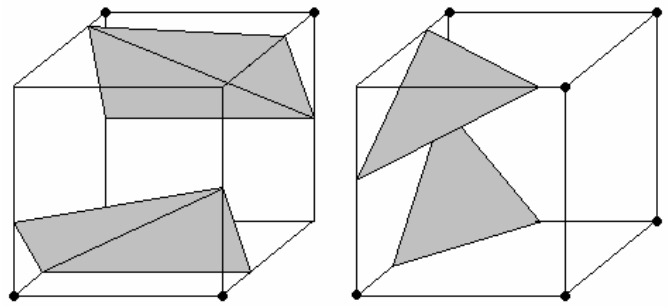


Figure 14 – Marching Cubes face ambiguity

If for example $x = x_0$ then function F has the following bilinear form:

$$F(y,z) = A(1-y)(1-z) + B\,y\,(1-z) + C(1-y)\,z + D\,y\,z \tag{11}$$

Where

$$\begin{aligned} A &= F_{000}(1-x_0) + F_{100}\ x_0 \\ B &= F_{010}(1-x_0) + F_{110}\ x_0 \\ C &= F_{011}(1-x_0) + F_{111}\ x_0 \\ D &= F_{001}(1-x_0) + F_{101}\ x_0 \end{aligned} \tag{12}$$

Four edges of the cell are intersected by the plane $x = x_0$, data values in points of intersections are *A, B, C* and *D*. When $x_0 = 0$ or $x_0 = 1$ then *A, B, C* and *D* are sample values on an appropriate cell face. Let *A, C* be positive samples and *B, D* negative, as drawn in Figure 15. If $F(y,z) = threshold$, such contour has hyperbola shape (due to bilinear interpolation). The hyperbola from Equation (11), has the form

$$ayz + by + cz + d = 0 \tag{13}$$

where

$$\begin{aligned} a &= A - B - C + D \\ b &= -A + B \\ c &= -A + C \\ d &= A \end{aligned} \tag{14}$$

and its center is situated in the point

$$\mathbf{O} = \left(-\frac{c}{a}, -\frac{b}{a} \right) \tag{15}$$

The decision, which of the two possible cases arose can be resolved by comparison of the data value in the point **O**, where hyperbola asymptotes are intersecting each other, and a *threshold*.

$$F(y_O, z_O) = \frac{AC - BD}{A - B + C - D} \tag{16}$$

To simplify the decision, the threshold is subtracted from sample values, positive and negative data values are obtained and a new threshold is zero. Therefore, new decision is based on comparison with zero and denominator in Equation (16) is always positive, so the test concerns only numerator $AC - BD > 0$ (positive samples are joined, negative separated) or $AC - BD < 0$ (positive samples are separated, negative are joined), see Figure 15.

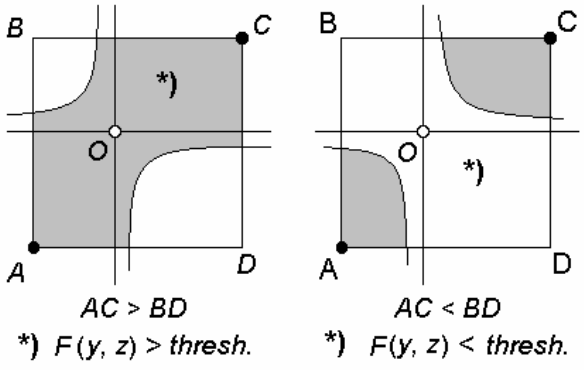
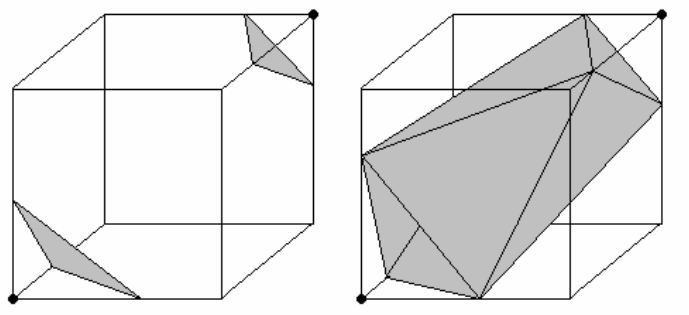


Figure 15 – The face ambiguity

There exists also internal ambiguity as apparent from Figure 16. This kind of ambiguity is solved in the manner, which is stated below.



a) b)

Figure 16 – The internal ambiguity

Let's have a case, which is stated in Figure 16 and corresponding cell samples are named as shown in Figure 17. We can cut the cell with a parallel plane to a bottom face to obtain the samples A_y, B_y, C_y and D_y, which form a new face. If there exists such a new ambiguous face and it fulfills following Equation (17):

$$A_y C_y - B_y D_y > 0 \tag{17}$$

Where

$$\begin{aligned} A_y &= A_0 + (A_1 - A_0)\, y \\ B_y &= B_0 + (B_1 - B_0)\, y \\ C_y &= C_0 + (C_1 - C_0)\, y \\ D_y &= D_0 + (D_1 - D_0)\, y \end{aligned} \tag{18}$$

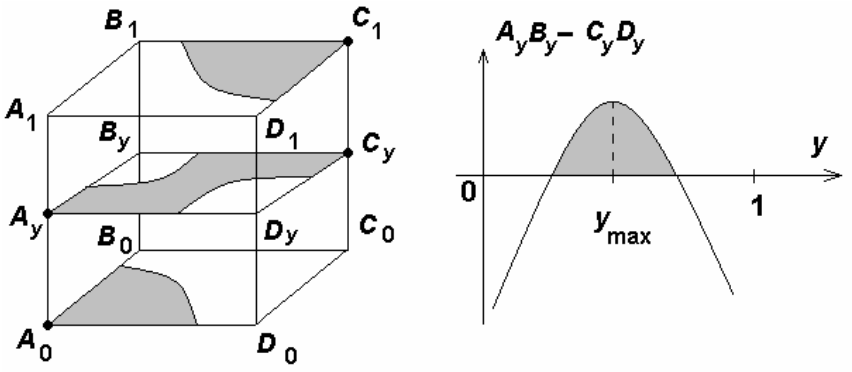
With the use of new face the decision of solving the internal ambiguity can be made. After insertion of Equation (18) into Equation (17):

$$a y^2 + b y + c > 0 \tag{19}$$

Where

$$\begin{aligned} a &= (A_1 - A_0)(C_1 - C_0) - (B_1 - B_0)(D_1 - D_0) \\ b &= A_0 (C_1 - C_0) + C_0 (A_1 - A_0) - B_0 (D_1 - D_0) - D_0 (B_1 - B_0) \\ c &= A_0 C_0 - B_0 D_0 \end{aligned} \tag{20}$$

Positive areas of the whole cell are then connected (Figure 16.b) when the quadratic function on the left side of inequality in Equation (19) has the same shape ($a < 0$) as in Figure 17.b and $y_{max} = -b/2a$ lies in the interval (0, 1), otherwise the positive areas are disconnected (Figure 16.a).



a) b)

Figure 17 – Internal ambiguity resolving

As was mentioned, the original Marching Cubes table is extended to 33 basic cases including more than one variant for ambiguity cases solution. With described decision technique the appropriate case from a table is chosen and the iso-surface is locally approximated with a set of triangles.

4.3 Marching Tetrahedra 5, 6

Marching Tetrahedra 5 (MT5) and 6 (MT6) methods are similar to MC method (Section 4.1). Difference is that the cell with a cube shape is divided into 5 or 6 sub-cells with tetrahedral shape. For MT5 there are 2 different division schemes shown in Figure 18 or follow one of these references [24], [15] and [26]. When using MT6, there is more than one possibility: [15] – 6 schemes, [32] – 2 schemes, [24] – one scheme MT5, [14] – all possible cases, or Figure 19. These sub-cells are then processed in the similar way as cells in MC. Index is, in this case, 4-bit, therefore there exists 16 possibilities how an iso-surface can intersect the tetrahedral cell. These possibilities can be due to index negation, symmetry and rotation reduced to 2 basic cases, see Figure 20.

The resulting set of triangles does not contain holes. The triangle table contains only 16 triangulations for intersected tetrahedral cell. MT6 division scheme makes all tetrahedra to have the same size, volume and shape. The amount of generated triangles is greater, when compared to MC. The more tetrahedra the more generated triangles (MT6 > MT5 > MC). The use of division of the cell into 5 tetrahedra causes that these tetrahedra does not have the same shape, etc. And moreover the need of two different divisions arises. These divisions have to be alternated regularly, because of set of triangles consistency among adjacent cells. The cube cell with 5 tetrahedra has 18 edges (12 + 6 face diagonals), with 6 tetrahedra has 19 edges (12 edges + 6 face diagonals + 1 cell diagonal) which can be intersected by an iso-surface, the maximal number of extracted triangles from one tetrahedral cell is 2.

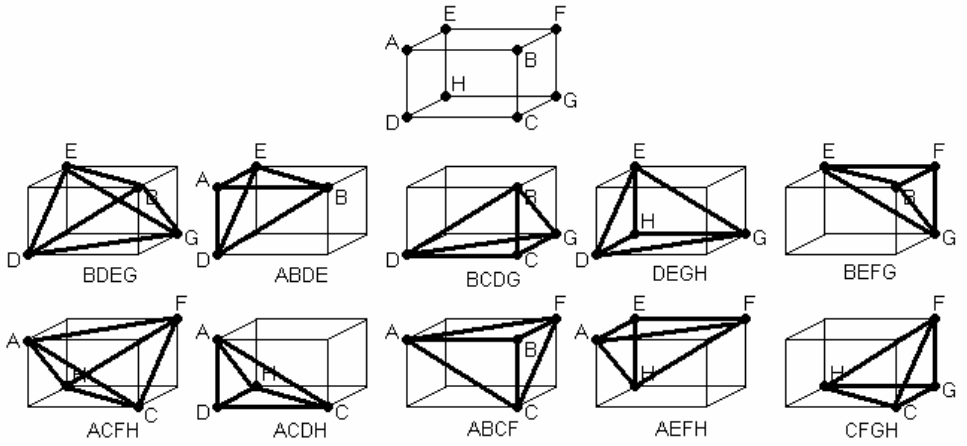


Figure 18 – Two possibilities how to divide a cell into 5 tetrahedra

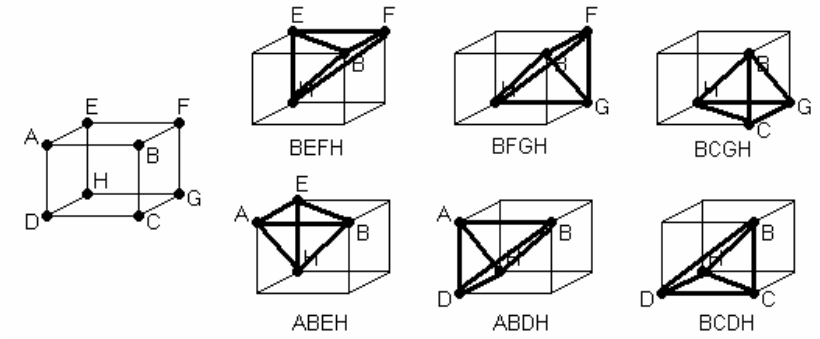


Figure 19 – One possibility of the cell division into 6 tetrahedra

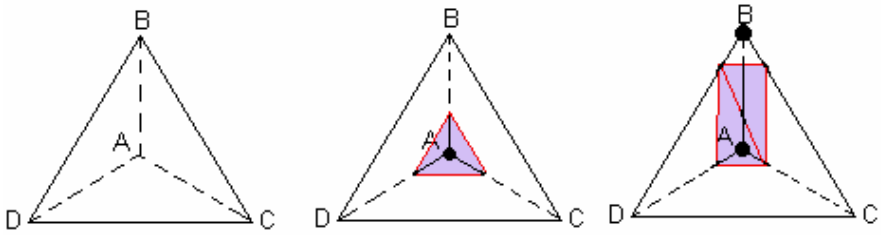


Figure 20 – Two basic cases how the iso-surface can intersect a tetrahedral cell

4.3.1 Number of Cell Tessellations

Kolcun [14] investigated the cube cell tessellations and defined surface representation of the tessellation schemes, also the part of his dissertation is devoted to grid generation and deformation. The five non-equivalent (with respect to the rotation) tetrahedra which can be created using the cube cell vertices are shown in Figure 21.

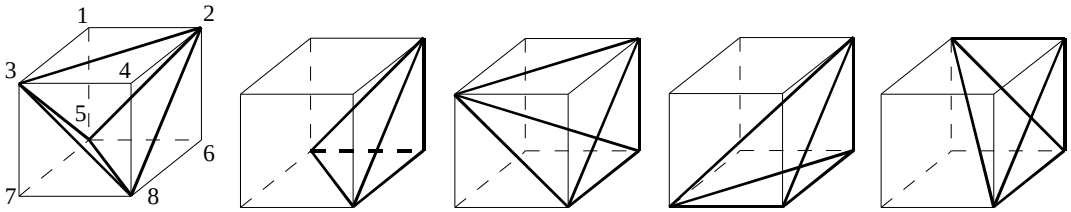


Figure 21 – Five different tetrahedra within the cell

The number of all tetrahedra which can be constructed from the cube cell vertices is following [14]:

$$\binom{8}{4} - \textit{planar quadrilaterals} = \frac{8 \cdot 7 \cdot 6 \cdot 5}{4 \cdot 3 \cdot 2 \cdot 1} - (6+6) = \frac{1680 - 288}{24} = 58 \qquad (21)$$

The planar quadrilaterals are all six cell faces and six diagonal planar cuts of the cube.

Kolcun [14] also counts all possible cube cell tessellations into six tetrahedra using an incidence matrix and a graph theory. The total number of all tetrahedral tessellations of the cube cell into six tetrahedra is 72.

The author also mentions a cube cell tessellation into three pentahedra, Figure 22 (to define the surface representation of the tessellation schemes).

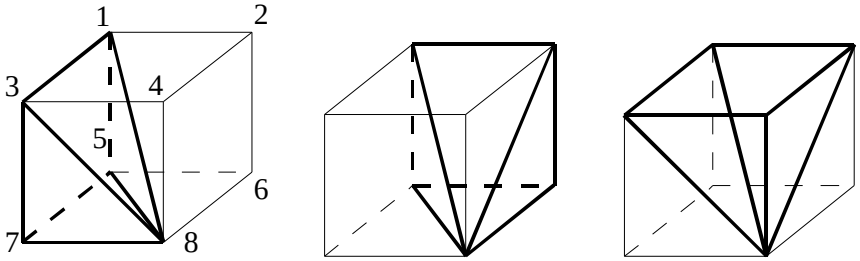


Figure 22 – Three pentahedra tessellation scheme

4.3.2 Different Interpolations on Face Diagonals

Guéziec and Hummel [9] address problem with sample value interpolation on face diagonal edges. The main problem is that a face diagonal is chosen randomly (depends on chosen division scheme) by the programmer. The value on the diagonal is interpolated using linear interpolation. That causes two different results (generated polygons), each for a different diagonal, see Figure 23.a and Figure 23.b.

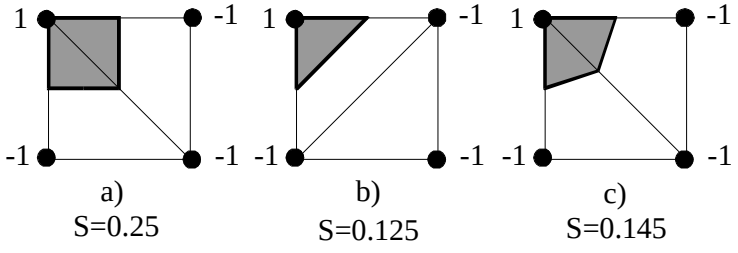


Figure 23 – The difference between linear and bilinear sample value interpolation
(a), b) … linear, c) … bilinear interpolation)

The authors solve this problem using bilinear interpolation (Section 4.13). This approach makes the difference between two face divisions smaller as obvious from Figure 23.c. We assume that the similar approach can be used with the cell internal diagonal (and the use of trilinear interpolation).

4.4 Centered Cubic Lattice

The cube cell is divided into 24 tetrahedra, Chan and Purisima [2], see Figure 24. The tetrahedron parts are shared among adjacent cube cells. The data value in the center of gravity of the cube cell is calculated via arithmetic mean. Of course, trilinear interpolation as described in [38] can be used as well as other approaches. Normal vectors in tetrahedron vertices are calculated as follows [2]:

$$
\begin{aligned}
&\mathbf{g} = (g_1, g_2, g_3) \\
&g_1 = q\left(\frac{f(x_i + a, y_i, z_i) - f(x_i - a, y_i, z_i)}{2a}\right) + \\
&\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right)}{a}\right) + \\
&\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right)}{a}\right) + \\
&\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right)}{a}\right) + \\
&\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right)}{a}\right)
\end{aligned}
\tag{22}
$$

$$g_2 = q\left(\frac{f(x_i, y_i + b, z_i) - f(x_i, y_i - b, z_i)}{2b}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right)}{b}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right) - f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right)}{b}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right)}{b}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right)}{b}\right)$$

$$g_3 = q\left(\frac{f(x_i, y_i, z_i + c) - f(x_i, y_i, z_i - c)}{2c}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i + \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right)}{c}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i + \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right)}{c}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i - \frac{b}{2}, z_i - \frac{c}{2}\right)}{c}\right) +$$
$$\frac{1-q}{4}\left(\frac{f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i + \frac{c}{2}\right) - f\left(x_i - \frac{a}{2}, y_i + \frac{b}{2}, z_i - \frac{c}{2}\right)}{c}\right)$$

Where *a*, *b*, and *c* are distances between adjacent samples in given volume data in *x*, *y* and *z* directions; *q* is weight parameter for normal vector calculation and it is from the

range [0; 1] (q determines the impact of adjacent data samples onto normal vector computation). Chan and Purisima [2] tried different values for q, their conclusion is that the value of q has a small effect on visual appearance of the resulting surface.

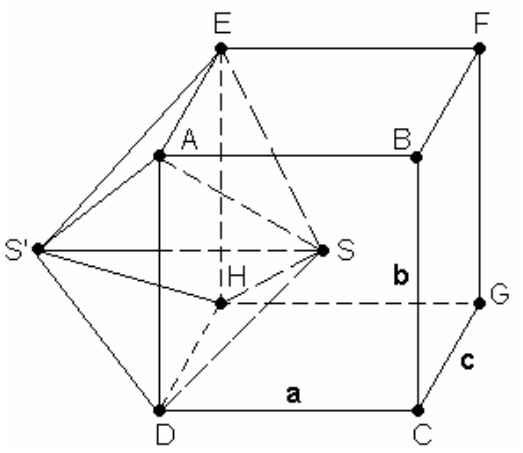


Figure 24 – A centered cube division scheme

The sharing of tetrahedra gives better link-up among adjacent cells. All tetrahedra do have the same size, volume and shape (similarly as in MT6 – Section 4.3). When the sum of volume of all 24 tetrahedra is done, we obtain the volume of two cube cells. Therefore, each cube cell is approximately divided into 12 tetrahedra (from the view of the cell volume). The number of tetrahedra is the main reason why this method generates much higher number of triangles than MC, MT5 and MT6. This method is much slower than other methods, because of the volume data approximation in the center of the cell and more complex normal vector computation.

4.4.1 Another Division Scheme

When we have the data sampled in centered cubic lattice grid and does not need to compute the central sample value, we can use following algorithm, that was proposed by Takahashi et al. [34]. They divide a parallelepiped cell into two tetrahedra and one octahedron, as in Figure 25.

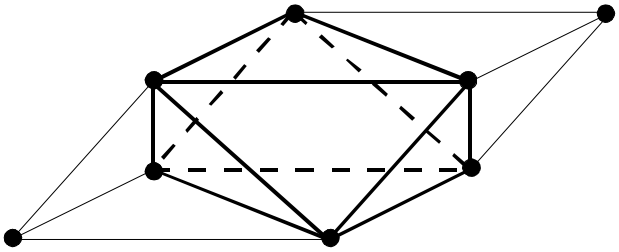


Figure 25 – The division scheme two tetrahedra and one octahedron

This approach does not solve the ambiguity. Tetrahedron has 3 basic cases of intersection, as mentioned before. The octahedron can be intersected in 64 ways, but due to symmetry and rotation these can be reduced to just 7 basic cases (two of them are ambiguous), as described in detail in [34]. The authors tested their method only using metaball implicit functions.

4.4.2 3D Chessboard Speedup

This speedup technique is often referenced as a 3D chessboard, Cignoni et al. [5], and it is based on a local coherence of adjacent cells and cell reduction together. The total number of explored cells is reduced to 25% of all cells, as it can be seen from Figure 26.

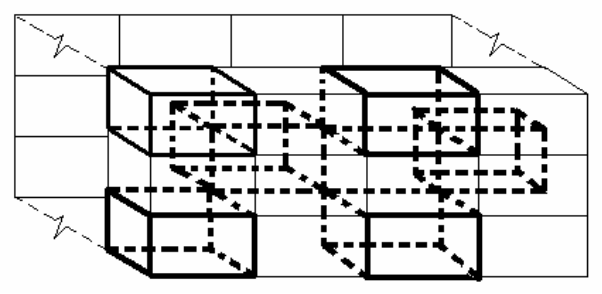


Figure 26 – The 3D chessboard speedup overview

The 3D chessboard is defined in the following way:

- Let's assume a 3D grid with resolution $I \times J \times K$ in x, y and z directions. Such volume data consist of $(I-1) \times (J-1) \times (K-1)$ cube cells. Black cells are then placed to the following positions $(2i+(k\%2), 2j+(k\%2), k)$, where $i=0...(I-2-(k\%2))/2$, $j=0...(J-2-(k\%2))/2$ and $k=0...(K-2)$. The character "*%*" stands for modulo division, e.g. $(5\%2) = (5 \textbf{ mod } 2) = 1$.

In the other words, when we have a black pane (cell) $\mathbf{C}=(i, j, k)$ inside of a volume then the adjacent black cells are such cells which shares only one corner (cell vertex) with black pane **C**.

4.5 Near Optimal Iso-Surface Extraction

More information about this speedup technique can be found in Livnat et al. [16] or Shen et al. [30]. Active cells are searched via span space (also known as interval space). Span space can be represented with a kd-tree [16] or a range (interval) tree Cignoni et al. [5]. Kd-tree is tree data structure with $O(N)$ memory complexity, where N is the total number of all cells in given volume data. Kd-tree is constructed for volume data only one time in offline preprocessing.

Span space represents a set of intervals as a set of points in 2D. The interval $I = [min; max]$ is represented as a point $\mathbf{x} = [x, y]$ where $x = min$ and $y = max$, see Figure 27. If we are looking for intervals which contain number *num* it is the same as we would be looking for points (in span space) which have greater coordinate on *max* axis (or equal to) than *num* and coordinate on *min* axis is lower than *num* (or equal to). Intervals which contain given number *num* are represented by points in hatched area in Figure 27. For each cell in the volume data we created one interval (minimal and maximal data value).

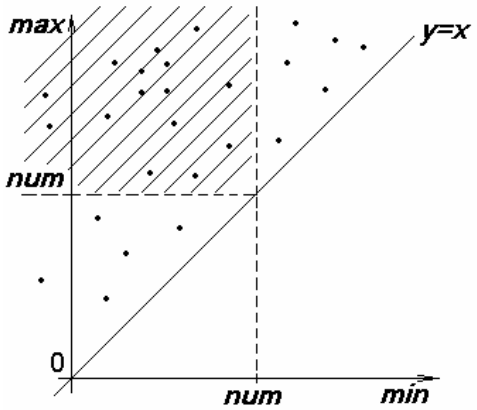


Figure 27 – A span space (intervals are represented as points in 2D)

The Kd-tree is a multidimensional binary tree. Each its node (father) contains the data value and two sub-trees (sons). In the left sub-tree there are all values lower than a value in the father node and vice versa. Binary trees are used to search among one dimensional data, in the other hand kd-trees are used to search among multidimensional data. In each kd-tree level single data dimensions are regularly alternated, e.g. points in 2D in even levels of the kd-tree there are data ordered according to *x* coordinate and in odd levels there are ordered according to *y* coordinate. In our case the node of kd-tree represents an interval of data values of the cell. For each cell there is one kd-tree node. The NOISE method takes advantage of kd-tree. Particular steps of this algorithm are as follows:

- The construction (or a load) of kd-tree for the given volume data
- The recursive search for active cells with given threshold (via kd-tree)
- The iso-surface approximation with a set of triangles within active cells

In each kd-tree node there are stored minimal and maximal data value within appropriate cell, pointers to left and right sub-trees and pointer to the full cell information. The nearly balanced kd-tree for NOISE method is used in our case. The kd-tree can be easily implemented as an one dimensional array of kd-tree nodes (relatively low memory usage but little higher computational requirements). This representation is called the pointerless kd-tree, Livnat et al. [16]. Sub-trees are then represented as intervals in this array and the *root* is placed in the middle of this interval. The balanced kd-tree construction uses Wirth's recursive algorithm [37] to search for the median (in general this algorithm can be used to search for the k^{th} smallest element). This approach is many times faster than standard quick sort algorithm and does not completely sort the whole array. The kd-tree drawn in span space is displayed in Figure 28. Steps for kd-tree construction are described bellow (principle is similar to creation of balanced binary tree):

1. Create dynamically allocated one dimensional array. The size is the same as number of cells in given volume data. For each cell there is one node of kd-tree. Fill this array with required details (minimum data value, maximum data value and pointer to original cell). Cells, where minimum equals to maximum (singular case) can be omitted to save the memory space, because these cells are not intersected by any iso-surface.

2. Sort of all elements in the array into a kd-tree. With Wirth's method find a median (*root* node) due to minimum data value among all cells. On the left side from the median there are only cells with less minimum value (left sub-tree) and vice versa (right sub-tree). The first level of kd-tree was created.
3. Recursively find *roots* in all sub-trees from previous step but now with use of maximum data value. The even (e.g. second) level of kd-tree was created. All even levels are created on the basis of maximum data value. Jump onto 4th step until the kd-tree is done (sub-trees can not be further divided).
4. Recursively find *roots* in all sub-trees from previous step but now with use of minimum data value. The odd (e.g. third) level of kd-tree was created. All odd levels are created on the basis of minimum data value. Jump onto 3rd step until kd-tree is done (sub-trees can not be further divided).

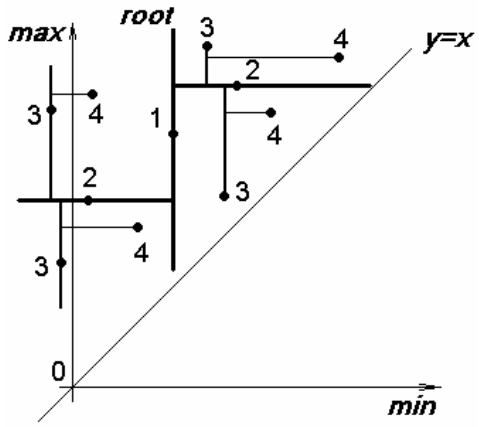


Figure 28 – A kd-tree (numbers show the kd-tree levels)

Example of the complete kd-tree, which was created from a sample set of intervals, is drawn in Figure 29. Steps for the kd-tree pass:

1. Find the kd-tree *root* in one dimensional array – *root* is at the position 0+(*number of nodes*)/2 (array is indexed from 0). The *root* divides an array into two sub-trees. The first sub-tree is from 0 to (*number of nodes*)/2–1, the second sub-tree then from (*number of nodes*)/2+1 to (number of nodes). After the *root* location, the comparison of nodes data values with given threshold is performed and decision whether this node represents an active cell or not is made. After this, the decision whether to pass both sub-trees or not follows.
2. The next step is to recursively search all sub-trees from the previous step. Note that some sub-trees need not be traversed due to decision in the previous step. Find the *roots* of these sub-trees, decide whether they represents active cells or not, etc. Repeat this step until the whole kd-tree is traversed.

As was mentioned before, odd levels of kd-tree are ordered according to minimum value and even levels according to maximum value. When the decision whether to pass both sub-trees in odd level node or not has to be done, one of the following situations can occur:

- The threshold is less than interval's minimum – we know that in the right sub-tree there are no active cells. In the right sub-tree there are all intervals'

minimum values greater than the threshold. So we are going to pass only left the sub-tree.

- The threshold is greater than interval's minimum – we know that in the left sub-tree there are all intervals' minimum values always less than the threshold value. So we can skip tests for interval's minimum values in the whole sub-tree. We are going to pass both sub-trees.

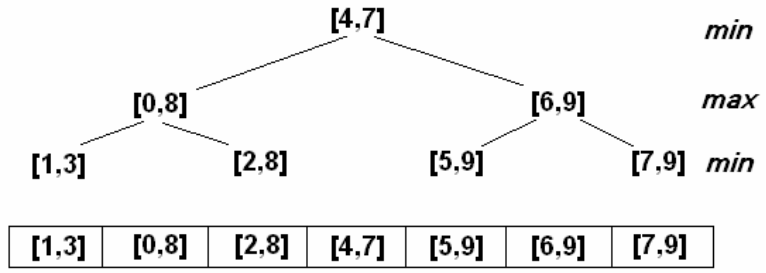


Figure 29 – A nearly balanced kd-tree for intervals: [7, 9], [2, 8], [5, 9], [1, 3], [4, 7], [6, 9] and [0, 8]

When the decision whether to pass both sub-trees in even level node or not has to be done, one of the following situations can occur:

- The threshold is greater than interval's maximum – we know that in the left sub-tree there are no active cells. In the left sub-tree there are all intervals' maximum values less than the threshold. So we are going to pass only the right sub-tree.
- The threshold is less than interval's maximum – we know that in the right sub-tree there are all intervals' maximum values always greater than the threshold value. So we can skip tests for interval's maximum values in the whole sub-tree. We are going to pass both sub-trees.

Thus, in higher kd-tree levels it can easily become a situation when both tests (interval's minimum and maximum value tests) are skipped and the sub-tree consists only from active cells. As mentioned before, the sub-tree is an interval in 1D array, thus we can pass it sequentially and extract appropriate part of the iso-surface. Described deciding mechanism skips such parts of kd-tree where are not active cells for given threshold.

A degenerate cell in the kd-tree is defined [16] as the cell which has more than one data value equal to minimum or maximum data value within the cell. When the threshold equals to maximum or minimum value, in such cell one of the following cases can occur:

- One value is equal to the extreme – the cell shares one vertex with the iso-surface
- Two values are equal to the extreme – the cell shares one edge with the iso-surface
- More than two values are equal to the extreme – the cell shares one wall or the whole cell is a part of the iso-surface

We can ignore first two cases. When we would ignore the third case, there would be a hole in the resulting set of triangles. When we would decide to draw a whole wall of the

cell, it would be drawn twice because such a wall is shared by two adjacent tetrahedra. If the whole cell is a part of iso-surface then iso-surface contains bubbles and all cell's walls are drawn twice. In Livnat et al. [16] there is proposed as a solution a small perturbation of data values and the iso-surface then intersects the cell, rather than touches it. Another solution is to utilize local coherence to decide what to draw. Note, that singular cases (cells) in iso-surface extraction are generally described as cells which do have one or more data values equal to the threshold value. This is the main difference from degenerate cell in the kd-tree.

For 10^9 cells (data resolution 1001·1001·1001) we gain kd-tree depth only 30. Therefore, the recursion depth is not too deep and we can easily use recursive algorithms to traverse such a tree. Maximum recursion depth is given by the Equation (23):

$$depth = \log_2(number\ of\ cells) + 1 = \left\lfloor \frac{\ln(number\ of\ cells)}{\ln(2)} + 1 \right\rfloor \qquad (23)$$

4.6 Isosurfacing in Span Space with Utmost Efficiency (ISSUE)

This improvement of search in span space was introduced by Shen et al. [30]. They propose equidistant span space division, which if properly used, is faster than search of active cells via kd-tree. Such division can be used for both serial and parallel iso-surface extraction.

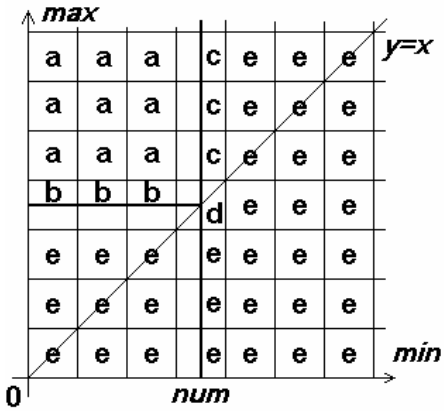


Figure 30 – The span space division and cell types

The span space is subdivided into L·L uniform squares, assuming that there exist minimum sample value and maximum sample value across the whole volume data (e.g. 8-bit unsigned samples have *min* = 0, *max* = 255).

When the threshold *t* is specified, the squares are divided into 5 categories as follows, Figure 30:

- a – cells from the volume data, which are represented by points inside of such square and they are all active cells. For threshold *t* it always holds that *t* $\hat{I}$ [*min*, *max*].

- b – for cells in these squares, the minimum condition is always fulfilled (*min* <= *t*)
- c – for cells in these squares, the maximum condition is always fulfilled (*t* <= *max*)
- d – the d square (is only one) contains all kinds of cells, hence all cells inside of it have to be checked whether they are intersected or not
- e – cells can not be in these squares, because condition *min* <= *max* would not be fulfilled

The iso-surface from all cells, which are represented by points in a-type squares, can be immediately extracted. In all b-type squares, all the cells have to be ordered according to *max*. Such a sub-set of cells can be divided by binary search in two groups (intersected and non-intersected cells). Cells in c-type squares are ordered according to min and as well divided into two groups by binary search. The last sub-set, which can contain intersected cells, is defined by d-type square. For this sub-set, the kd-tree or min-max lists are constructed and searched to find active cells. The e-type squares can not contain any cells, as mentioned before.

As perceptible, it can happen that one or more squares are empty, without cells. To quickly find non-empty squares the row data structure, see Figure 31, is created. This structure also establishes adjacency among squares.

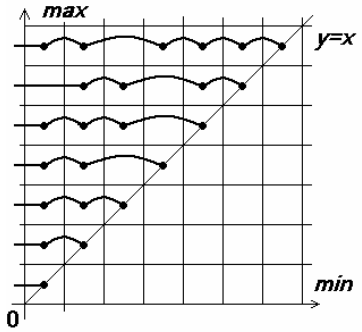


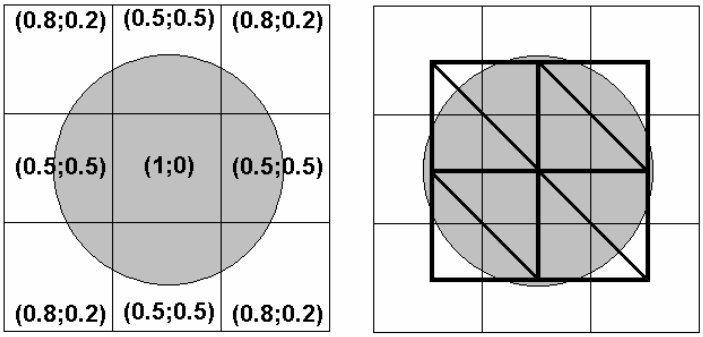
Figure 31 – Adjacent cells connection and empty cell skip

As was mentioned at the beginning, this algorithm is serial and with small modifications it can also be used in the parallel environment. Each processing element should get approximately the same work. Shen et al. [30] order all cells in span space according to e.g. *min* value. Afterwards, the array is divided into L intervals (e.g. in *min* axis), which have the same length. The boundaries of these intervals are coordinates of equidistant grid.

4.7 Material Interface Reconstruction

For different input data, where only fractional material information is given for each cell, the Material Interface Reconstruction method can be used. This approach was published by Bonnell et al. [1]. The authors do not state how to transform volume data to required input data (fractional material information is known), e.g. when the threshold is selected.

Let's focus on two dimensional cases with just two different materials. The input data are given in the structured 2D cubic grid and each square cell contains fractional material information, which is expressed by percentages (from 0 to 1). The sum of percentages of all materials contained in one square equals to 1, Figure 32.a.



a) b)

Figure 32 – Fractional material information a) and dual grid with simplices drawn in bold b)

Each square cell can be transformed to a point in a dual space and the dual grid can be then constructed, as clear from Figure 32.b. Each vertex **p** from the dual grid has appropriate coordinates and the fractional material information as well $\mathbf{p} = (x, y, m_1, m_2)$. The fractional information is represented by barycentric coordinates tuple $(m_1; m_2)$. The dual grid is furthermore divided into triangles (note, that in 3D into tetrahedra).

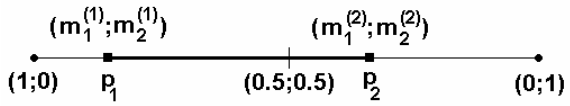


Figure 33 – Finding the border between two materials by linear interpolation

As mentioned, each end of the edge ($\mathbf{p}_1$ and $\mathbf{p}_2$) in the dual space has barycentric coordinates tuple ($\mathbf{m}^{(1)}$ and $\mathbf{m}^{(2)}$). Such edge can be drawn in a material space (m-space) as in Figure 33. Material space is the space, where only barycentric coordinates of dual grid vertices are used. The point (0.5, 0.5) in m-space represents the intuitive boundary between two materials, in such point the dominance of materials is changing. This point is important because it isolates two different materials. One of the following cases has to arise:

- The line $\mathbf{m}^{(1)}\mathbf{m}^{(2)}$ does not intersect the point (0.5; 0.5) in m-space – the corresponding edge $\mathbf{p}_1\mathbf{p}_2$ in dual space does not intersect the material boundary.
- The line $\mathbf{m}^{(1)}\mathbf{m}^{(2)}$ intersects the point (0.5; 0.5) in m-space – the corresponding edge $\mathbf{p}_1\mathbf{p}_2$ in dual space intersects the material boundary and the point of intersection can be calculated by linear interpolation:

$$(0.5;0.5) = (1-r)\,\mathbf{m}^{(1)} + r\,\mathbf{m}^{(2)} \tag{24}$$

The point of intersection $\mathbf{p}_i=(x; y)$ in the dual space is calculated with use of the scalar parameter r (that can be computed from Equation (24)) as follows:

$$\mathbf{p}_i = (1-r)\,\mathbf{p}_1 + r\,\mathbf{p}_2 \tag{25}$$

The dual grid in 2D is divided into triangles, in 3D into tetrahedra. Each edge is then checked, weather it intersects the material boundary or not. Afterwards, the computed intersection points are properly connected inside of the square cells (in 2D) in the dual space and iso-lines are drawn. When testing edges of a triangle in 2D, two basic cases can arise:

- None of three edges is intersected – whole triangle consists of one material.
- Two edges are intersected – triangle consists of two materials and two intersection points are calculated => one line is constructed.

When testing edges of tetrahedron in 3D, three cases can arise:

- None of edges is intersected – whole tetrahedron consists of one material.
- Three edges are intersected – tetrahedron consists of two materials, three intersection points are calculated => one triangle is constructed.
- Four edges are intersected – tetrahedron consists of two materials, four intersection points are calculated => two triangles are constructed.

In two material cases, this approach gives the same results and the underlying process is the same as in Marching Tetrahedra methods. When more than two materials are present in the input data, the extended algorithms have to be used, for such cases, see [1].

4.8 Extrema Graph

A graph based approach presented by Itoh and Koyamada in [10] is another alternative for fast active cells search and the iso-surface extraction. The main idea is to find extremum points in the volume data and connect them into an extrema graph. After the threshold selection the extrema graph is to be traversed and found active cells serve as a starting points for the iso-surface extraction. Afterwards, the iso-surface propagation approach is used to search for other active cells within the volume data.

The extremum point is defined as a grid node whose scalar value is higher (or lower) than the scalar values of all adjacent grid nodes that are connected to it with cell edges. In a 3D regular grid the inner nodes have always six neighbouring nodes, two in each direction (*x, y* and *z*). An extremum point defined in such a way represents a local minimum or a local maximum. An important rule that holds for extremum points is that y lie both inside and outside of a closed iso-surface. If there is an open iso-surface then it intersects the boundary of the volume data, Figure 34.

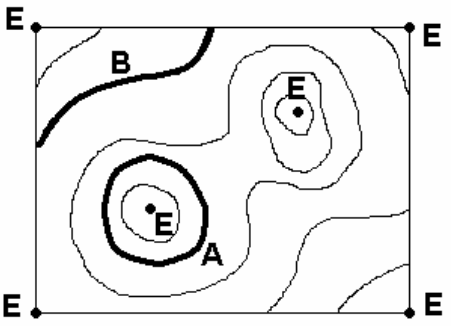


Figure 34 – The 2D case E are extremum points, A is the closed iso-line and finally B is the open iso-line

The algorithm [10] has generally these three steps:

- extremum points extraction and the extrema graph construction
- the extrema graph traverse
- the iso-surface propagation

The extraction of extremum points is made by sequential traverse of all the grid nodes. The extrema graph construction is not a simple task. The extrema graph connects all extremum points. Each arc of the extrema graph contains a set of intersected cells and their minimum and maximum data value. The aim is to get such a graph that the sum of all cells, which are intersected by arcs of a graph, is minimal. Itoh and Koyamada propose approach to find such a graph but they do not guarantee that the algorithm works well for all volume data sets (their future work should improve the algorithm). The example of an extrema graph is shown in Figure 35.

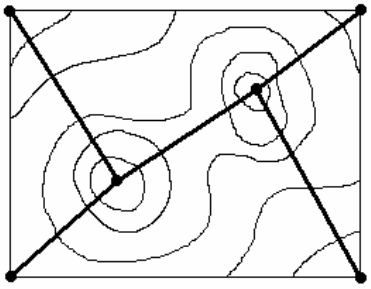


Figure 35 – The example of an extrema graph

The next step is to traverse an extrema graph and generate the starting set of all active cells (from all its arcs). For faster graph traverse the minimum and maximum values for each arc can be taken into account. It is guaranteed that the extrema graph intersects all possible iso-surfaces for all threshold values. Hence all disjoint parts of the iso-surface are to be extracted. Note that whole extrema graph has to be traversed.

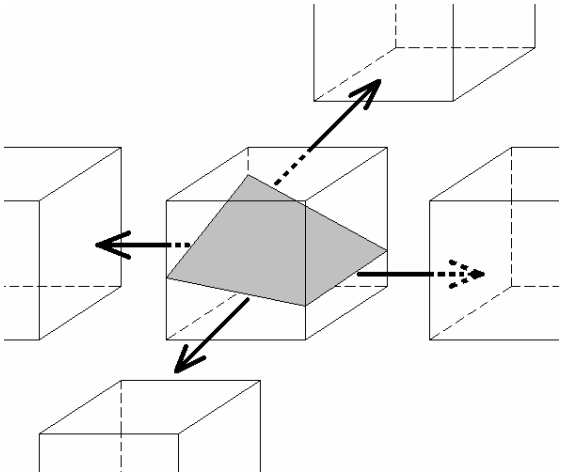


Figure 36 – The iso-surface propagation among adjacent cells

The final step uses the iso-surface propagation algorithm (Itoh and Koyamada [10] with reference to [33]) to extract all parts of the iso-surface from the starting cells. Each cell

has a flag whether it was already processed or not. All the starting cells are put into a FIFO like queue and their flags are set. The queue contains only active cells. The cells from the queue are processed until the queue is empty. The iso-surface is locally extracted from each cell. All cell faces that are intersected by the iso-surface are also shared by adjacent cells. Hence these cells are also active and are added into a queue and their flag is changed, see Figure 36. Naturally the cells with the flag set are not put into a queue. At the end the queue is empty and the iso-surface was extracted from all active cells.

The great advantage of this algorithm is that during the iso-surface extraction only active cells are visited and non-active cells in the extrema graph are simply ignored. Authors also propose an extension of mentioned algorithm that can handle special volume data with voids (missing cells) and through holes (a hole through a whole volume data).

4.9 Seed Set

Marc van Kreveld et al. [36] use a seed set for the iso-surface extraction. A cell can be thought to be a seed, so the seed set means a set of cells. For given volume data the seed set is said to be complete when any iso-surface or its disjoint part passes through one or more seeds.

An example of a seed set is e.g. an extrema graph that was mentioned in the Section 4.8. Similarly to an extrema graph with shortest total path (arcs length) a seed set with minimum cardinality is not easy to construct. Van Kreveld et al. [36] present an algorithm for a seed set construction. It is proved that their algorithm in the worst case constructs a seed set with cardinality twice as big as cardinality of a minimum seed set. The seed sets are generally smaller in size than cells contained within an extrema graph. They use a contour tree (described in [36]) as a preprocessing for a seed set construction.

An extrema graph connects local minimum and local maximum points. A contour tree connects also local minimum and local maximum points but moreover also saddle points. The sweeping algorithm is used by Van Kreveld et al. for its construction. New contours arise at local maximum points, expires at local minimum points and the contours spilt or merge in saddle points. All the cells (their data intervals) are then compared with the contour tree and the set of cells that covers scalar ranges of the whole contour tree is selected as the seed set.

This algorithm has the same advantage as a previously mentioned one (the extrema graph), non-active cells in a seed set are skipped during the iso-surface extraction process.

4.10 Volume Thinning Algorithm

The extension of image thinning algorithm, which is used for skeleton finding in 2D images, into 3D is presented by Ithoh et al. [11], also implementation details are mentioned there. Such extension is called the volume thinning algorithm and it constructs a skeleton in the volume data.

At first all extremum points are found using the algorithm of [10]. All cells are thought to serve as a starting seed set. The cells that share the extrema points are

marked. Marked cells can not be further deleted from a final seed set that will represent the extrema skeleton. All non-marked cells are repeatedly tested (by volume thinning algorithm) and the unnecessary cells are deleted from a seed set. At the end the resulting seed set forms a one cell wide extrema skeleton.

The image thinning algorithm in 2D forms a skeleton of a binary image, the skeleton is one pixel wide and describes features of figures in image processing applications. The algorithm in 3D is similar to 2D version. The 2D version is briefly described here.

It visits pixels that touch the boundary of a painted area and eliminates many of them. This process is repeated until no pixels are removed. Pixels that are not removed have to fulfill one of the following rules:

- If all the adjacent black pixels (at most eight) of the visited pixel **P** that shares an edge with **P** can not be visited by traversing the adjacent pixels through their shared edges in order (cases A and B in Figure 37).
- If the visited pixel **P** has only one adjacent black pixel that shares edge with **P**.

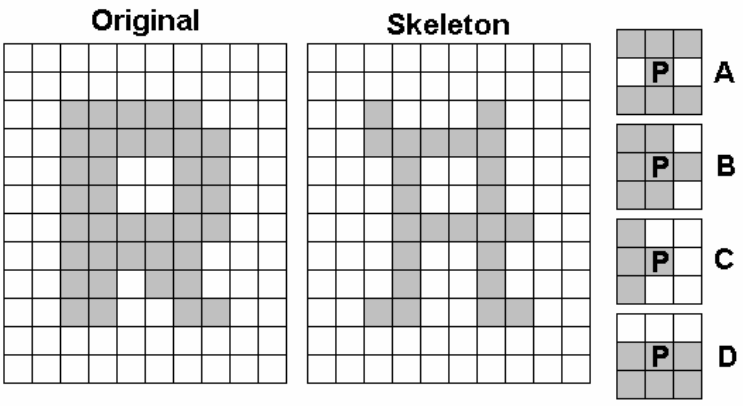


Figure 37 – The image thinning algorithm

Pixels **P** in cases C and D from Figure 37 can be removed, because they do not fulfill the rules. Original image and its skeleton are also shown in Figure 37.

The extension of the image thinning algorithm to the volume thinning algorithm is simple and intuitive. The similar rules with some extensions do apply for cells in 3D. The adjacent cells are traversed using the cell faces. The cell edges and vertices are also taken into account and each cell stores some additional information e.g. number of adjacent cells, adjacent cell pointers, boundary faces, etc.

This algorithm can be used also for the volume data that contain some voids or through holes. The seed set, which is the result of the volume thinning algorithm, can be further processed to form a kd-tree or an interval tree to accelerate its traverse.

4.11 Min-Max Lists

This method uses two sorted lists to quickly extract the searched iso-surface and the global coherence is partially used by Min-Max lists, see Giles and Haimes [8]. The cell is intersected by the iso-surface when the selected threshold is between minimum and maximum data value within such cell. The list of all probably intersected cells which

this method uses is called an action list. Such special list is updated with use of other two lists:

- Min list – this list contains for each cell its minimum data value and the pointer to cell's complete information, moreover this list is ordered according to the minimum value.
- Max list – contains maximum data value for each cell and it is also ordered but according to the maximum value.

In each cell we have minimum and maximum data value and the difference can be calculated as (maximum-minimum), the global variable Δ_Z equals to maximal difference among all cells. When the threshold T is specified for the first time, or the threshold changed about more than Δ_Z due to its previous value, all cells with minimum value in the interval $(T\text{-}\Delta_Z; T)$ are placed into the action list. The action list is then traversed and all cells which are not intersected by the iso-surface are discarded. If in the next step the threshold value increases about less than Δ_Z, all cells with minimum value in the interval $(T_{PREVIOUS}; T_{NEW})$ are added into the action list. The min list is used for fast search of such cells. If the threshold decreases about less than Δ_Z, all cells with maximum value in interval $(T_{NEW}; T_{PREVIOUS})$ are added to the action list. The max list is used for that. Afterwards, all cells which are not intersected by the iso-surface are discarded from the action list. As evident, the action list can change drastically from threshold to threshold, and its maintenance is complex.

4.12 Sweeping Simplicies

Shen and Johnson [31] described Sweeping Simplicies method for the iso-surface extraction. This method uses a global coherence and hierarchical data decomposition. Previously stated Min-Max method [8] uses two ordered lists, this method uses also two different ordered lists:

- Sweep list – for each cell, which is stored within this list, it contains a pointer to cell's complete information, maximum data value of the cell and a flag, moreover this list is ordered according to the maximum value.
- Min list – the min list contains minimum data value for each cell, and a pointer to this cell in the sweep list. Min list is ordered according to the minimum data value.

After the first threshold selection, the algorithm sets the flag for all cells in sweep list with less minimum value than the threshold. When the threshold was previously selected, the algorithm passes only cells in min list with minimum value in interval $(T_{PREVIOUS}; T_{NEW})$. If the new threshold value is greater than previous one, flags of appropriate cells are set, else they are cleared. The algorithm than passes the sweep list from the first cell with greater maximum value than the new threshold and it approximates the iso-surface, but only inside of cells whose flag is set.

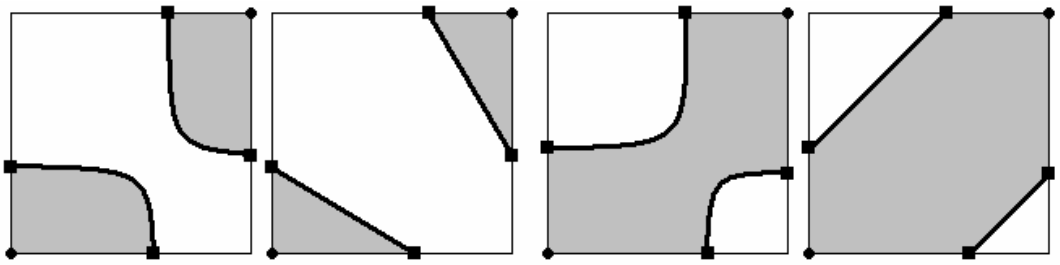
As mentioned above, the sweeping simplicies method also uses hierarchical data decomposition. There are several sub-groups and each sub-group accepts only such cells with minimum and maximum data value within its interval. Of course, it can happen that some cells do not fit into any sub-group, because of its minimum or maximum data value exceeds it. The set of such sub-groups form the first level of the tree structure. In the next step, always two adjacent sub-groups are merged into one, and cells from previous step, which did not fit into any sub-group, are tested again. Also there can be

again some cells, which do not fit into any of such sub-groups, the second level was created, etc. The last level, which will be created, will cover the data interval of whole volume data set.

Sweep and min lists are created for each sub-group. When the threshold is selected, it is easy to identify all sub-groups, which contain such number. All cells from such sub-groups are then passed using sweep and min lists and non-intersected cells are discarded. Intersected cells are then used for the iso-surface extraction.

4.13 Trilinear Interpolation

At present, trilinear interpolation is often used to represent the data value inside of a cell. The data within a cell are supposed to have a linear behaviour. As already mentioned (Section 4.2), Chernyaev [3] used two special points to solve two kinds of ambiguities. These points are called either face saddle point or body saddle point. The face saddle point is used to solve the face ambiguity and the body saddle point is used to solve the cell internal ambiguity. Cignoni et al. [4] with reference to Natarajan [21] describes Exhaustive Look Up Table (also known as ELUT) that stores all possible configurations of the iso-surface within a cell and its appropriate triangulations with respect to the topological correctness. Lopez and Brodlie furthermore extend this idea in [18]. The extension improves the representation of the iso-surface in the cell interior. Their approach also minimizes visual changes that can be caused by a small threshold change when using mentioned approaches. An example of such changes can be observed in Figure 38.



a) b) c) d)

Figure 38 – A small threshold change produces a significant visual change (in 2D)

In Figure 38.a (bilinear iso-line) and Figure 38.b (linear iso-line), the positive vertices are cut off. Afterwards, a small change was made the visual appearance of iso-line changed significantly Figure 38.c and Figure 38.d and negative vertices were cut off. These changes motivated Lopez and Brodlie to add some additional points to the final triangulation within a cell.

For simplicity let's consider a 2D case. Bilinear interpolation is used to represent the data in the 2D cell interior and an iso-line is approximated with hyperbola arcs. The more new points are added the slower visualization will follow. To minimize the number of added points and maximize the accuracy of hyperbola approximation with line segments just one special point is added for each hyperbola arc. Such point lies on hyperbolic arc and is called a shoulder point, see Figure 40.

If a function $f = f(x, y)$ is a scalar function of two variables, then the saddle point is any point **s** where the gradient of f is zero and **s** is not a local minimum or maximum.

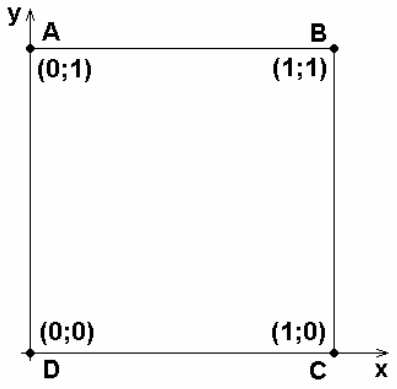


Figure 39 – A face saddle point computation

A face saddle point **s** computation example follows. At first we need to calculate coefficients of bilinear interpolation

$$F(x,y) = axy + bx + cy + d \tag{26}$$

These coefficients are obtained easily when we move the unit cell into the origin and insert cell corners coordinates and values (see Figure 39), then we obtain the system of four equations whose solution is:

$$\begin{aligned} a &= -A + B - C + D \\ b &= C - D \\ c &= A - D \\ d &= D \end{aligned} \tag{27}$$

The resulting saddle point coordinates are computed from the system of two equations (partial derivatives are equal to zero) as:

$$\mathbf{s} = \left(-\frac{c}{a}, -\frac{b}{a} \right) \tag{28}$$

And the data value in the computed saddle point is

$$F(\mathbf{s}) = d - \frac{bc}{a} \tag{29}$$

where **s** is a saddle point, $F(\mathbf{s})$ is the value of bilinear interpolation in a saddle point. Face saddle points, which are out of the unit square, are discarded.

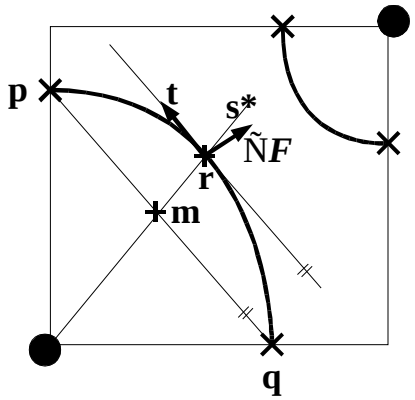


Figure 40 – Face shoulder point

The face shoulder point **r** is defined as the point of hyperbolic arc where the tangent is parallel to the chord **pq** that joins the end points of an arc on the cell borders **p** and **q**. Moreover the authors claim that a shoulder point **r** lies on a line segment **ms**, where **s** is a saddle point of bilinear interpolation. Hence, the threshold value changes the position of **r** on **ms** line segment. For the critical threshold value (equal to the data value in **s**) the **r** is at the same position as **s** and thus visual appearance is not changing significantly with small threshold changes around **s**. Note, that when a threshold value approaches **s** the hyperbolic arcs tend to appear as two perpendicular line segments and **r** approaches to **s**. The point **r** is optimal point for the area of hyperbolic arc approximation with a polynomial. It maximizes the covered area of hyperbolic arc.

The face shoulder point **r** can be found by the following procedure [18]. The bilinear interpolation at the face is defined as:

$$F(x,y) = axy + bx + cy + d \tag{30}$$

Thus partial derivatives of $F(x,y)$ that define the gradient ∇F are:

$$\frac{\partial F(x,y)}{\partial x} = ay + b \qquad \frac{\partial F(x,y)}{\partial y} = ax + c \tag{31}$$

Because for **p** it holds that x=0 and $F(x, y)$=0 in it, then from Equation (30), the tangential vector **t** we want to find is:

$$\mathbf{t} = \mathbf{q} - \mathbf{p} = \left(-\frac{d}{b}, \frac{d}{c}\right) = \frac{bc}{d}\left(-\frac{d}{b}, \frac{d}{c}\right) = (-c; b) \tag{32}$$

The following dot product has to hold at a point we are looking for (tangential and gradient vectors are perpendicular to each other):

$$\nabla F \cdot \mathbf{t} = -\frac{\partial F}{\partial x}c + \frac{\partial F}{\partial y}b = 0 \tag{33}$$

The curve at which we are looking for a shoulder point is $F(x, y)$=0 and this also holds:

$$\begin{aligned} &\frac{\partial F}{\partial x}\frac{\partial F}{\partial y} = bc + abx + acy + a^2xy = \\ &= bc + a(bx + cy + axy + d) - ad = bc - ad - aF(x,y) = bc - ad \end{aligned} \tag{34}$$

From the system of two equations Equation (33) and Equation (34):

$$\frac{\partial F}{\partial x} = \pm\sqrt{\frac{b}{c}(bc - ad)} \qquad \frac{\partial F}{\partial y} = \pm\sqrt{\frac{c}{b}(bc - ad)} \tag{35}$$

And from Equation (31) and Equation (35) we gain shoulder point coordinates **r** = (x, y) as:

$$x = \frac{1}{a}\left(\frac{\partial F}{\partial y} - c\right) \qquad y = \frac{1}{a}\left(\frac{\partial F}{\partial x} - b\right) \tag{36}$$

Note that face shoulder points which are out of the unit square (Figure 39) are discarded.

Such approach can be extended into 3D where body saddle point and bishoulder points are computed, see [18]. Note that the bishoulder point computation is not simple and its approximation has to be found using numerical methods.

4.14 Skeleton Climbing

A Skeleton Climbing method is published by Poston et al. [28]. The name of the method comes from the fact that vertices (0-skeleton, 0-cell) are examined at first, edges (1-skeleton, 1-cell) and faces (2-skeleton, 2-cell) follow. The basic principle of Skeleton Climbing method can be described with the following sequence of steps:

- Volume analysis with regards to the threshold
- Adaptive construction of simple boxes (multiresolution cells)
- Sharing the information among adjacent boxes
- The iso-surface extraction

4.14.1 1D Case

In the first step the voxels are grouped in 1D to form segments and then in 2D to form rectangles. The 1D case is described in more detail then other steps. Let's imagine a set of voxels as a set of dots in 2D. A whole line of samples is called a lign and its special subset is a dike, Figure 41.

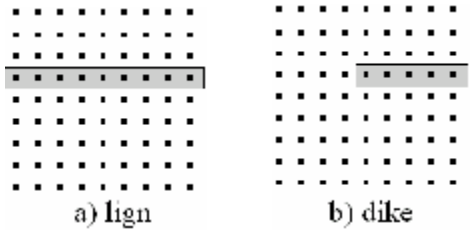


Figure 41 – The lign and the dike

The special subset means that a dike has to start at $k*2^m$ position and it ends at $(k+1)*2^m$ position, where m and k are natural numbers with respect to the data resolution. All possible dikes can be assigned IDs according to the binary tree structure Figure 42, the higher ID number the shorter dike.

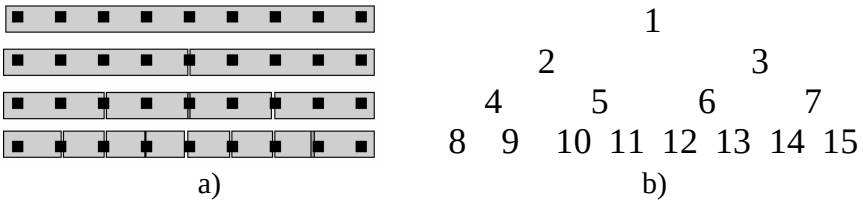


Figure 42 – All possible dikes a) and a dike binary tree b)

Every dike has its binary index in the occupancy array (00_B … not crossed, 01_B or 10_B crossed once and 11_B crossed more times by the iso-line; *1* represents a positive sample and *0* a negative one). A dike is called simple when its index is less than 11_B. All maximal length simple dikes, with respect to the binary tree dike organization (hence the dike does not need to be the longest), are stored in a simple dike array, Figure 43.

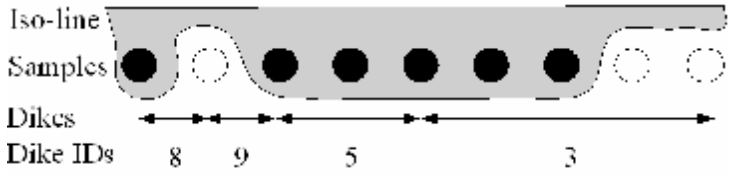


Figure 43 – The simple dike ID array (black dot means positive sample)

4.14.2 2D Case

In this step the rectangles are formed from ligns and dikes. Two dimensional elements are named a strip, a plot and a padi, Figure 44.

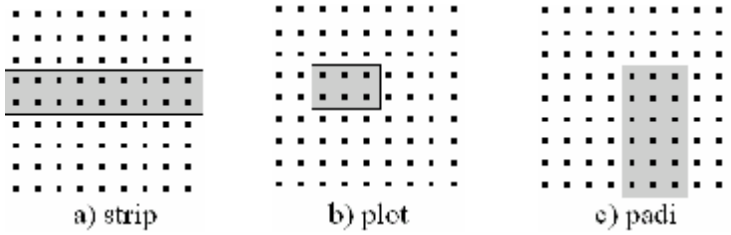
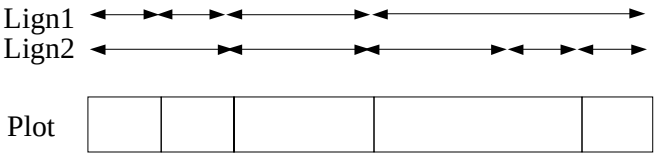


Figure 44 – The strip, the plot and the padi

The plot is simple if and only if its two dikes are also simple. Plots are also labeled with use of the binary tree. Also the simple plot array is defined in a similar manner as a simple dike array. The longest simple plots are found easily with use of the fact that the shorter the dike is the higher ID it has. Hence, when constructing the plot using two adjacent dikes (each from a different lign), the longest plot length equals to the length of a dike with the higher ID and the longer dike is subdivided, Figure 45.



a) Plot = MAX(Lign1, Lign2)

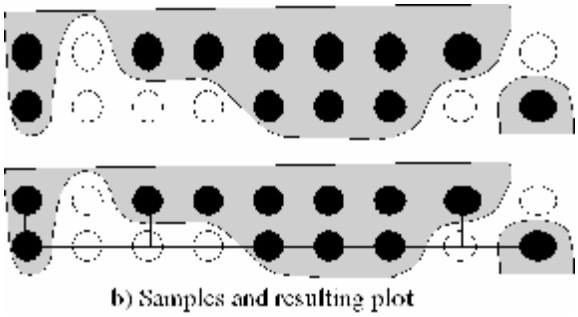


Figure 45 – The plot construction from two adjacent ligns

The main step in the volume analysis is the use of ASC2D (Adaptive Skeleton Climbing 2D) algorithm to create simple padis with maximal size. As well as dikes, plots, also

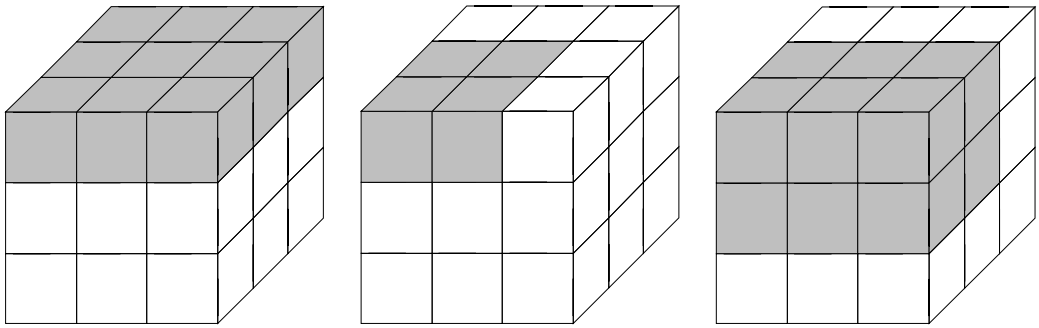
padises are created with respect to the horizontal and vertical binary tree structures. The ASC2D algorithm is described by authors in [28] in a form of a pseudocode.

The padis is simple when all plots inside of it are simple and furthermore the side dikes are simple as well. The goal is to subdivide the 2D sample grid (a farm) into as large padises as possible.

For a 2D iso-line extraction the slightly modified Marching Squares algorithm can be used at this stage.

4.14.3 3D Case

The 2D case can be analogously extended to 3D, so instead the iso-line extraction the iso-surface is extracted. A slab, a brick and a highrice are defined, Figure 46.



a) slab b) brick c) highrice

Figure 46 – The slab, the brick and the highrice

As well as in previous definitions the simple highrice (simple box) has to fulfill some conditions. It has to be composed of simple bricks and all its faces have to be simple padises. A simple brick is formed from two simple padises.

An ASC3D algorithm (pseudocode in [28]) is used for highrice construction within the volume data with respect to the binary data division in all three directions.

Sharing of the information on highrices faces among adjacent highrices is solved via face iso-lines. The algorithm subdivides adjacent highrices faces to match each other. Afterwards it extracts iso-lines on all the highrices faces. This step produces one or more edge loops and these are further triangulated (modified ear clipping algorithm) and the iso-surface is thus properly connected among adjacent highrices. Moreover, the 256 cases table is not needed. Instead 16 cases table for the iso-line extraction is used.

The authors also propose the multiresolution version of their algorithm. The multiresolution iso-surface extraction is achieved by specifying the maximum highrice size, which is set by a global parameter. This parameter is then used as a restriction in the highrices construction phase. The memory consumption requirements of their algorithm are not specified.

This method is the last mentioned one in this chapter. The overview of existing methods for the iso-surface extraction was made. All described methods can be used for the iso-surface extraction from regular volume data. Some of them, as indicated in appropriate sections, can be also used for tetrahedral networks and other volume data lattices. The following chapter will discuss our work, what we have done. The main part deals with basic methods comparison and the error analysis using mathematical objects and appropriate extracted iso-surfaces.

5 Basic Methods Comparison

The fifth chapter is about our research. It consists of tests that we have performed to compare and analyze the iso-surface approximation error of basic methods. Also our results are contained here together with our opinions and thoughts.

5.1 Data Generation

At first, we should mention the ways we use for the volume data generation. We generate synthetic samples in them. There are two basic approaches.

The first is that an object is represented by ascending (or descending) values from its centre of gravity. E.g. a sphere, the value the sample holds represents the distance between an appropriate grid vertex and a sphere center:

$$\begin{aligned} &(x-s_X)^2+(y-s_Y)^2+(z-s_Z)^2-r^2=0 \\ &f(x,y,z)=r=\sqrt{(x-s_X)^2+(y-s_Y)^2+(z-s_Z)^2} \end{aligned} \tag{37}$$

Where x, y and z are the sample coordinates in 3D; s_X, s_Y and s_Z are the sphere center of gravity coordinates; f is a computed sample value.

A set of iso-surfaces can be further extracted from such data set (Figure 47). In the case of a sphere, the threshold determines the radius of a sphere that will be extracted. The sample value can be rounded (Figure 47.a), but the new error is introduced with such an approach. When working with the integer volume data all data samples can be generated using:

$$f(x,y,z)=\left\lfloor\sqrt{(x-s_X)^2+(y-s_Y)^2+(z-s_Z)^2}\right\rfloor \tag{38}$$

The value of f is then truncated (*3.6* to *3*), but it can be also rounded (*3.6* to *4*) to an integer value.

To examine methods properties we prefer to use float numbers as samples (Figure 47.b), hence the rounding error is not introduced.

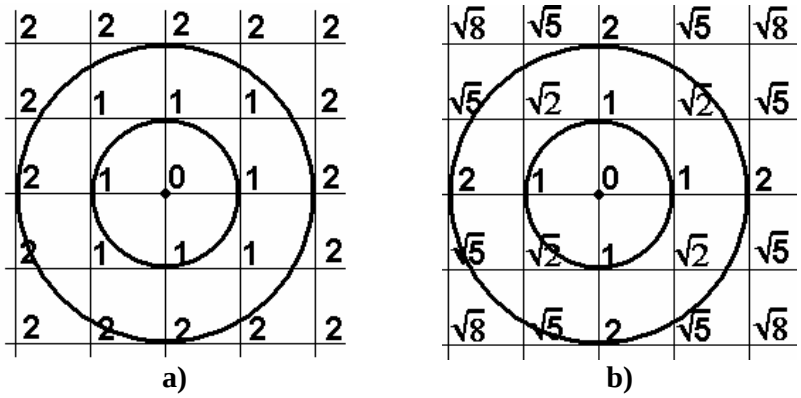


Figure 47 – From the left: a) integer data (two spheres with radius 1 and 2); b) float data

Iso-surfaces that are extracted from such data approximates the sphere surface with a set of triangles. The selected threshold determines the sphere radius. Several different iso-surfaces (spheres) can be extracted depending on the chosen threshold.

The second approach for the sample value generation we use is a distance function. The sample is assigned the value that represents its distance from the object surface. The sample has the negative value if it is inside of the object and the positive otherwise. This approach offers more flexibility in object generation than previous one. Zero threshold then represents the object surface in the volume data.

5.2 Comparison Approaches

5.2.1 Hausdorff Distance

We use Hausdorff distance [W11] for comparisons mainly for iso-surfaces that are extracted from real data sets. Firstly, we define a distance between a point p (from a surface S) and a surface S' (containing points p') as

$$d(p, S')=min||p-p'|| \tag{39}$$

for all p' from S'. Now we can define Hausdorff distance between two surfaces S and S' as

$$d_H(S,S')=max\ d(p,S') \tag{40}$$

for all p from S. Note the important thing that Hausdorff distance is not symmetrical $d(S,S')\neq d(S',S)$. When we call $d(S,S')$ the forward and $d(S',S)$ the backward distance, we can define symmetrical Hausdorff distance [1] as

$$d_{SH}(S,S')=max(d(S,S'), d(S',S)) \tag{41}$$

We utilized a METRO software tool (described in [6]) for accurate computation of Hausdorff distance of two discrete surfaces (triangle meshes). The METRO tool was mainly used to compare original triangle mesh with its simplified (e.g. decimated) version. We use it for comparison of two iso-surfaces, each generated using different method.

5.2.2 Root Mean Square Distance

We use also the Root Mean Square (RMS) of computed distances. RMS distance in discrete case is defined as [W11]

$$RMS(S,S') = \frac{\sqrt{x_1^2 + ... + x_n^2}}{n} \tag{42}$$

where n is a number of points of a mesh S', x_i (where $i=1..\ n$) represents the distance of corresponding point p_i' from S, $x_i=d(p_i', S)$. We compare S' to S.

Note that RMS is not symmetrical as well as Hausdorff distance. We do not use symmetrical RMS distance in our tests, thus there is no need to define it here. RMS is computed with METRO tool as well.

Both the Hausdorff distance and the RMS distance are calculated according to some source mesh. As such a mesh, we use a mesh generated with MC method.

5.2.3 Mathematical Data

When we know the object equation and its dimensions, we are able to compute some additional information concerning the object, such as a surface area, an object volume,

triangles positions difference from the object surface, etc. We believe that these properties are worth to compute, because they can help us to differentiate among the quality of methods from different views.

Surface area – the iso-surface is generated by an extraction method in a form of a set of triangles. We compute the total area as a sum of all triangles areas. Then we can compute the area of mathematical object and compare it with the iso-surface area obtained. For special objects such as sphere, we are able to track the error behaviour dependency on a sphere radius.

Volume enclosed with the iso-surface – for basic mathematical objects the volume is computed using appropriate formula. The volume enclosed with the iso-surface is computed in the following manner (for tetrahedra methods only). There are three cases for a tetrahedron:

1. The whole tetrahedron is outside of the iso-surface – it does not affect the total volume computation.
2. The whole tetrahedron is inside – the whole tetrahedron contributes to the total volume. The tetrahedron volume is computed easily.
3. The tetrahedron is intersected with the iso-surface – we have to compute the part of the tetrahedron which is inside of the iso-surface. As there are at most two triangles generated per tetrahedron, these triangles form two small tetrahedra with appropriate tetrahedron vertex and we are able to compute the volume of the tetrahedron part which contributes to the total volume.

Triangles position difference – we measure the difference between triangle center of gravity and the object surface. This gives us information about triangles position difference compared to the object surface.

Three mentioned properties together with Hausdorff distance and RMS distance are the main aspects that we use for extraction methods output comparison. The obtained results are discussed in the next section.

5.3 Experiments and Results

5.3.1 Used Data Sets

In this section, we will describe data sets we used for our comparisons and give the reasons why we chosen them. The main part of the used data set is a set of mathematically generated objects, Figure 48. Real data sets were used to show how the Hausdorff distance is dependent on the applied iso-surface extraction method. The brief description of used data sets follows in upcoming paragraphs.

Sphere – a sphere is an example of an object that we use to follow the error behaviour dependency on a sphere radius. The sphere equation used for data generation is a modified implicit equation

$$F(x,y,z) = \sqrt{(x-s_X)^2 + (y-s_Y)^2 + (z-s_Z)^2} - r \qquad (43)$$

where x, y and z are samples coordinates, s_x, s_y and s_z are the sphere centre coordinates, r is the sphere radius and $F(x,y,z)$ is the corresponding sample value. This equation assigns data value to all the volume data samples. The sphere is then represented with a zero threshold iso-surface. The samples that are inside of the sphere have negative values, on the sphere surface zero value and samples placed out of the sphere have

positive values. The sample value represents the distance of the sample from the sphere surface. The radius was 25 in our experiments.

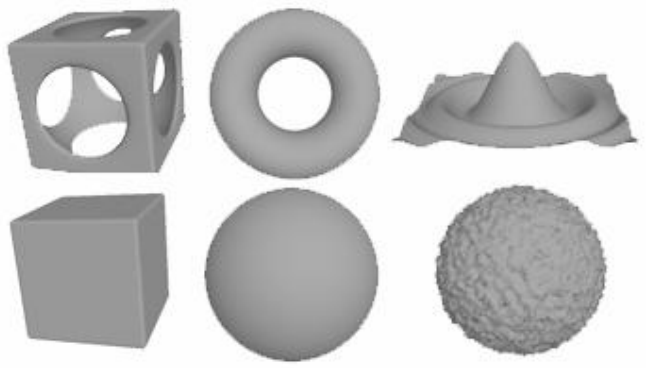


Figure 48 – Generated objects preview (csph, torus, sombrero, cube, sphere and noised sphere)

The cell edge has length 1 for our purposes. The object dimensions (e.g. radius, edge length) are then related to the cell edge length.

Noised sphere – (noisedsph) to study the influence of the noise to the shape of the output set of triangles we generate a noised sphere. The random noise is introduced (added) to all samples of the volume data. The size of the noise is given in percentage from the sphere radius size. We used radius 25 and 10% noise.

Cube – this kind of an object we use to track the behaviour and properties of the iso-surface on edges. We will show the iso-surface difference mainly visually. The inner, on surface and outer samples have the negative, zero and positive values respectively. Cube was generated using *a=b=c=42*.

Cube minus sphere – (csph) such an object was constructed to combine both properties of the sphere (*r=25*) and cube (*a=b=c=42*). The generation of it is a little bit complicated. At first, the cube is generated in the volume data. Afterwards, the values of all samples that are closer to the sphere than to the cube are modified to the new distance.

Torus – is another mathematically generated object. Torus is defined with the following equation [W11]

$$F(x,y,z) = \sqrt{(c - \sqrt{x^2 + y^2})^2 + z^2} - a \qquad (44)$$

where *x, y* and *z* are samples coordinates, *c* is a torus main radius, *a* is a torus secondary radius and *F(x,y,z)* is a corresponding sample value. The samples values are negative, zero or positive as well. Torus dimensions are *c=20* and *a=42* in our case.

Sombrero – is the last mathematically generated object we use. Its surface is defined with the mathematical equation (taken from Derive mathematical program) as:

$$F(x,y,z) = y - \frac{a\cos(b(x^2 + z^2))}{c + x^2 + z^2} \qquad (45)$$

where *x, y* and *z* are sample coordinates and *F(x,y,z)* is a corresponding sample value and *a, b* and *c* are constants modifying the shape of the function. The Sombrero parameters we used are *a=12*, *b=0.25* and *c=3*.

Real data sets – Samples of real data sets have only positive values that represent a density of the space in the sample position (we used engine.vol, ctmayo.vol and hplogo.vol sets).

5.3.2 Tests and Results

For all our mathematically generated objects, we are able to compute triangles position difference compared to the mathematical object. Firstly, a triangle center of gravity is computed. As we have routines for a point to an object distance computation, we can compute the distance of the center of gravity of a triangle from the appropriate object. The overall position difference P_{ERR} is computed as

$$P_{ERR} = \frac{\sum_{i=1}^{n} | objDist(\mathbf{O}, \mathbf{T_i}) |}{n} \tag{46}$$

where ***Ti*** (*i* goes from *1* to *n*) is the center of gravity of the *i*-th triangle, *n* is the number of triangles and *objDist(***O, X***)* is the distance of a point ***X*** from an object **O** surface.

The position difference for the sombrero object was slightly smaller and similar to results obtained for the sphere. For the cube the CCL method gives the worst results, see Figure 49. This is probably due to different interpolation of the cube edges (Figure 50). The csph object has more edges than a cube itself. The more tetrahedra we create the worse results we get. Surprisingly for the torus the MT6 method gives the greatest position difference. We think this is because of the interpolation at a cell interior edge (the longest one).

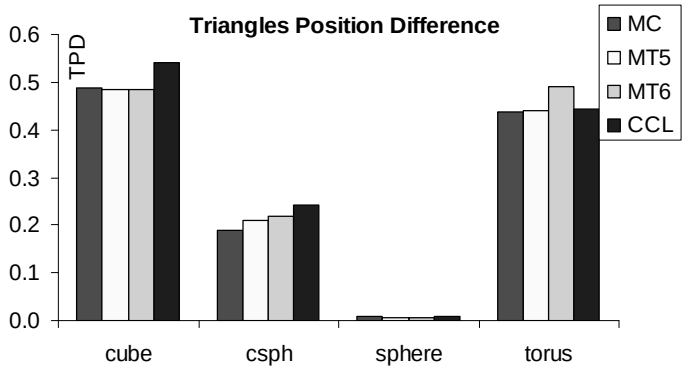


Figure 49 – The triangles position difference comparison (edge vs. smooth object)

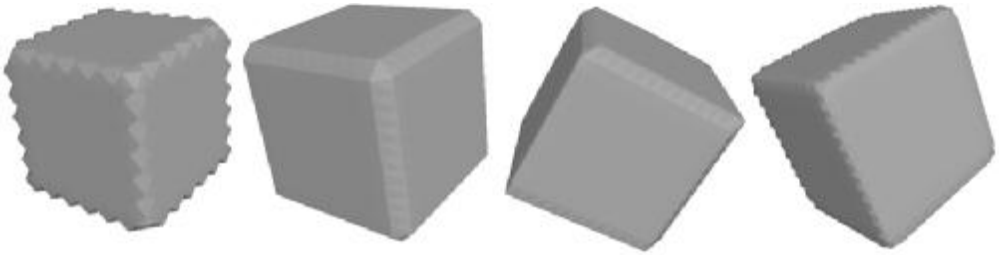


Figure 50 – The iso-surface on edges (from the left: MT5, MC, MT6 and CCL)

Note that RMS distance is related to the results of MC method. For the sphere and the torus obtained results were slightly less than results for the sombrero. Again, when the object has edges the CCL method is the worst from the view of RMS distance, see Figure 51. For the noisedsph object the CCL method gives the best results. We suppose that the central cell sample value computation (using arithmetic mean) filters data a little bit as well.

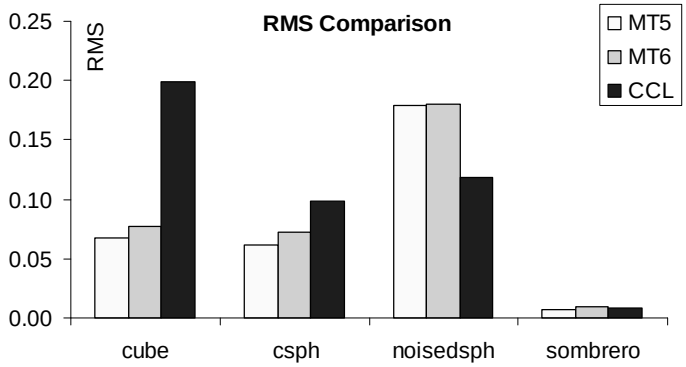


Figure 51 – The RMS distance histogram

Again, the sphere and the sombrero give approximately similar results compared to the torus. From the view of Hausdorff distance the MT6 method gives the worst results for all tested objects, see Figure 52. As you can see for the noisedsph the CCL method is the best choice. The best choice in this case is probably MT5 method because it does not generate as much triangles as CCL method.

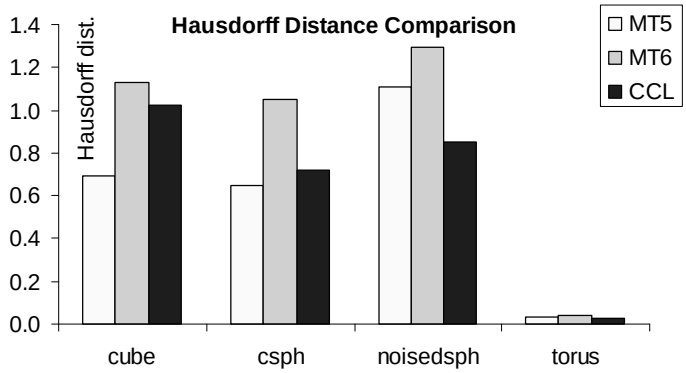


Figure 52 – The Hausdorff distance histogram

The more tetrahedra is used the larger area is extracted for all tested objects that have edges, see Figure 53. Results in Figure 53 and Figure 54 are relative due to mathematical results. For objects like the torus (does not have edges) results were approximately the same as for the sphere. We think that for the area approximation purposes the best choice is MC method.

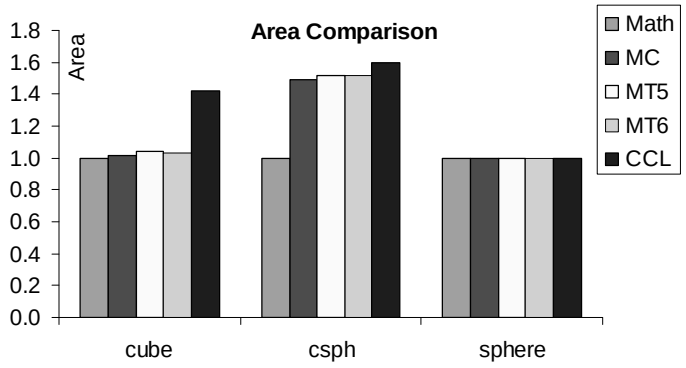


Figure 53 – The area comparison (relative to the mathematical area)

The MT5 method is in most cases slightly better than MT6 method and both methods are approaching to the original volume from below, see Figure 54. The CCL method in the other hand is in most cases approaching to the mathematically computed volume from above. The MC method is not included because it is hard to compute the volume enclosed with the iso-surface (due to 256 cases).

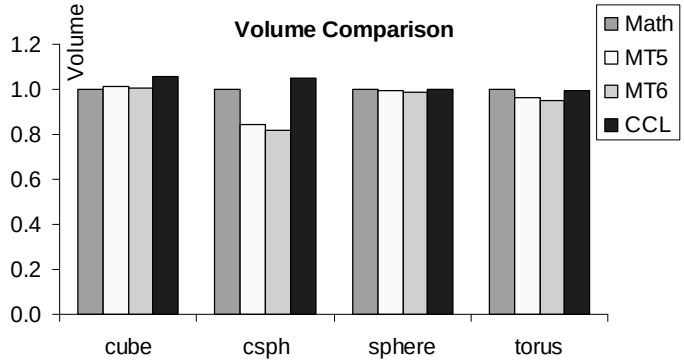


Figure 54 – The volume comparison (relative to the mathematical volume)

5.3.3 Sphere Additional Tests

The relative volume error is defined in a following way

$$Error = \frac{V_{TR} - V}{V} \tag{47}$$

where V_{TR} is a volume enclosed with the iso-surface triangles, V is the mathematically computed volume of a sphere.

The CCL method is the best choice for the volume approximation, see Figure 55. We assume that it is due to high number of tetrahedra. The CCL method error oscillates around the zero value. The MT5 gives slightly better results than MT6 method. The progress of error is similar. Both methods are approaching the zero error from below. Another thing we compare is a number of extracted triangles.

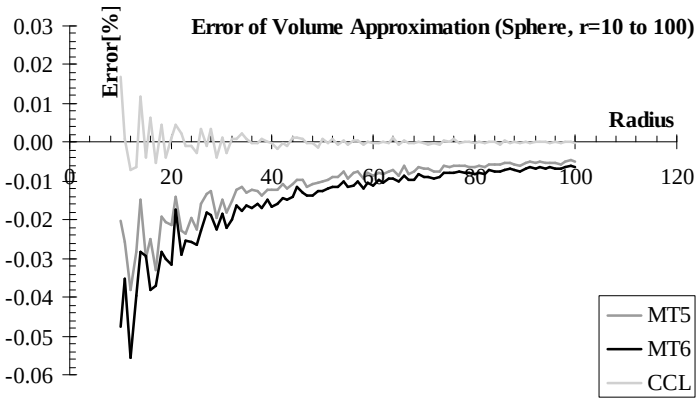


Figure 55 – The sphere volume error graph

It is well known fact that a number of generated triangles is mainly dependent on the type of the cell division, see Figure 56. The MC works with a cube cell (at most four

triangles per cell) and it does not divide it into tetrahedra (at most two triangles per tetrahedron). MT5 divides the cube cell into 5 tetrahedra, MT6 into 6 tetrahedra. In fact, CCL divides the cube cell into 24 tetrahedra, but these tetrahedra also contain parts of adjacent cube cells. When we sum the volume of all 24 tetrahedra, we obtain volume of two cube cells, so on average 12 tetrahedra per a cube cell.

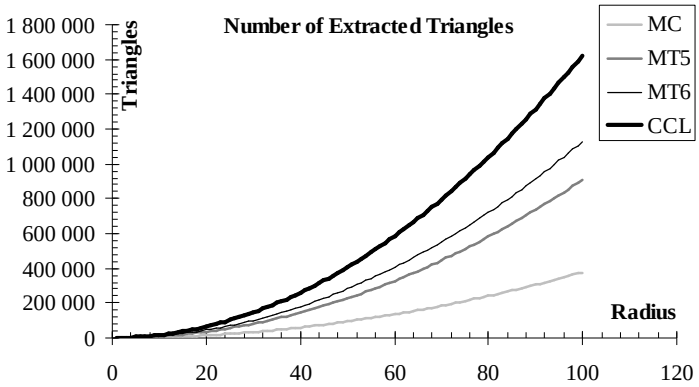


Figure 56 – The number of extracted triangles

Methods MC, MT5 and MT6 test all cells from the volume data, one by one. The CCL method checks only 25% of all cells, due to used speedup method. All mentioned methods, except NOISE, need some constant time to scan through the volume (it depends on the tests complexity and on the used method). The NOISE uses a kd-tree to find all active cells. A kd-tree parts (sub-trees), which do not contain active cells, are step by step skipped (are not completely traversed). It is the general reason why this method is so fast, assuming that a kd-tree is created in offline preprocessing. The total extraction time also depends on the size of an output set of triangles (number of interpolations). When increasing the sphere radius from zero, time consumption (and a number of interpolations) grows proportionally to the sphere surface growth ($S=4\pi r^2$), see Figure 57. Note that graphs for MC, MT5 and MT6 methods are overlapped in Figure 57. The bigger sphere the greater amount of active cells is found in the volume data.

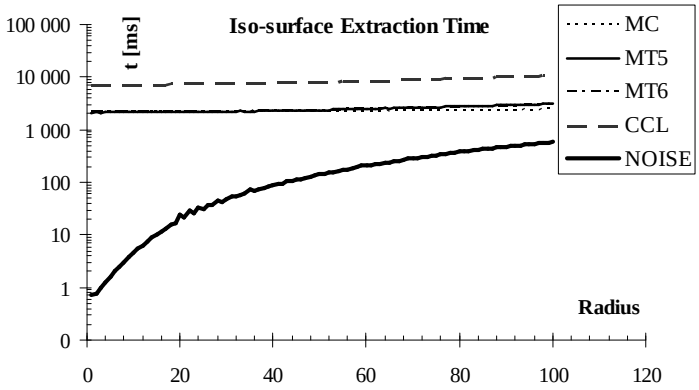


Figure 57 – The iso-surface extraction time dependency on chosen sphere radius

The average number of triangles generated per second (Figure 58) is given as a sum of all triangles divided by the time of the iso-surface extraction. The NOISE extracts most triangles per second. The reason is that NOISE traverses the lowest number of non-

active cells (it uses a kd-tree) from all described methods, therefore, redundant tests on non-active cells are not performed. In the apart of this, CCL method, which generates the largest total number of triangles, generates the smallest amount of them per second. The reason may be that the decision whether a cell is active or not is more complicated. It is necessary to check all 24 tetrahedra and these tetrahedra contain parts of adjacent cells and a computed cell center data value. Note that the number of extracted triangles for larger data sets easily overloads current graphics hardware.

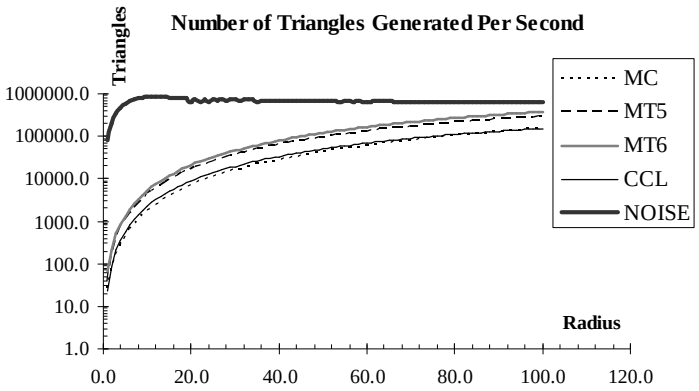


Figure 58 – The dependency of extracted triangles per second on a sphere radius

5.3.4 Real Data Results

Detailed information about the real volume data sets that were used for tests are summarized in Table 2.

Data No.	Data File Name	Data Set Resolution	Cells to be processed
1	syn_64.vol	64x64x64	262 144
2	ctmayo.vol	128x128x128	2 097 152
3	hplogo.vol	236x150x64	2 265 600
4	cthead.vol	256x256x108	7 077 888
5	engine.vol	256x256x108	7 077 888
6	bentum.vol	256x256x256	16 777 216

Table 2 – Real data sets details

A number of extracted triangles, when working with real data sets, is highly dependent on a selected threshold. Results, we gained for real data sets, were expected due to previously conducted experiments using generated objects.

For provided real data sets, it is better to use MC to extract up to four different iso-surfaces. To extract five or more iso-surfaces, it is more efficient to use NOISE method with preprocessing (kd-tree) as it is evident from Figure 59. The similar situation occurs when using MT5 or MT6, see Figure 59.

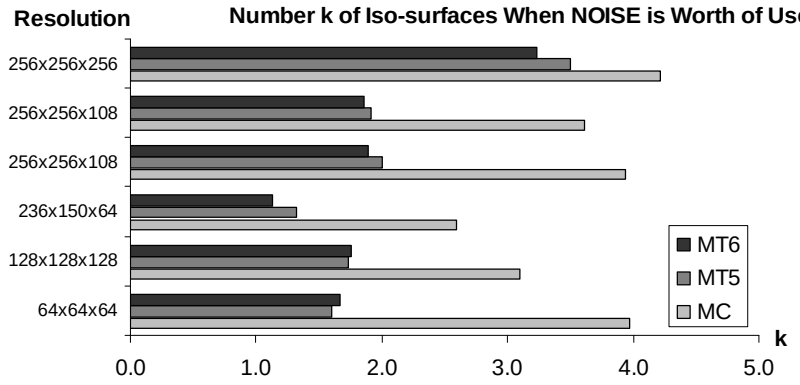


Figure 59 – The number of iso-surfaces where NOISE is worth using

From the time consumption point of view (Table 3), the best method is NOISE with a kd-tree constructed in offline preprocessing.

Data No.	MC [ms]	MT5 [ms]	MT6 [ms]	CCL [ms]	NOISE *) [ms]
1	67	100	95	270	16/240
2	585	965	880	3065	320/2431
3	675	1220	1160	3930	331/2325
4	2065	3225	3005	12170	1030/9520
5	2005	3150	2925	11730	950/9080
6	4210	4815	4745	13005	610/23640

*) iso-surface extraction/kd-tree creation

Table 3 – Extraction times from real data, threshold was selected as 40% from <min; max> data set value interval

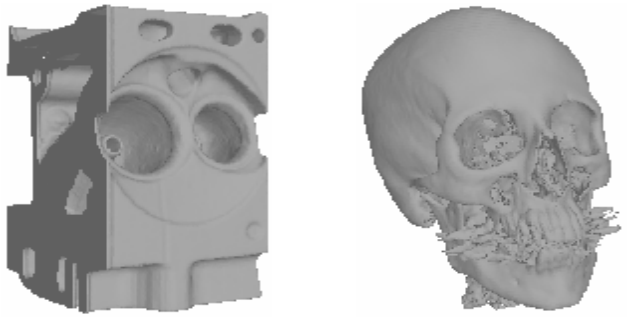


Figure 60 – Example of iso-surfaces from real data sets

All experiments were done on the PC model: HP Compaq EVO D310 (P4, 1.8GHz, 512MB RAM). Times were measured via motherboard hardware performance counter.

The performed tests and reached results were presented here. The next and last chapter will describe possible directions in future research.

6 Future Work

Iso-surfaces are still plentifully used in many areas of volume data visualization. Many methods for the iso-surface extraction take advantage of their ancestors in 2D or extend them into 3D. In 2D these methods serve for the iso-line (contour) extraction.

Methods for the iso-surface extraction that are used on personal computers are reasonably described in this work in the form of the state of the art (Chapter 4). Their properties and particular difficulties are also discussed. This description is the main goal of this work. The second part of this work (Chapter 5) is devoted to an examination of the error of the iso-surface extraction. We think that the provided information is a good starting point for the further research in this area.

Various methods for the iso-surface extraction solve several problems such as the issue of holes in the Marching Cubes algorithm or the non-active cells bottleneck. Also decimation techniques or view dependent iso-surface extraction methods can be used to decrease the output number of triangles. Extremely large data sets are processed in massively parallel or distributed environments. The extreme data sets that do not fit into system memory can be processed using external techniques for the iso-surface extraction.

However, not many methods in the field of the iso-surface extraction are dealing with the approximation of a normal vector in other way than central difference or one sided difference due to computational complexity and huge data sets. There are some approaches (linear regression [23], extended surface subdivision technique [19]) but these are still not in use. It is a well known fact that the visual information is accepted much faster than the information given in any other way. Also no one can doubt that a normal vector has a huge impact on the rendered graphical information (e.g. a triangle mesh). These are reasons for a better normal vector approximation but with respect to the computational expensiveness, this problem still remains open.

In Chapter 5 of our work, a set of methods was tested also with respect to the error of the iso-surface approximation. The authors of the appropriate methods do not present some detail analysis concerning an error estimation of their methods. We proved that different methods have different approximation errors. The methods we implemented use linear interpolation (both for normal vector and triangle vertex approximations) and hence need only two neighbouring data samples. We want to develop our own method that will consider larger neigbourhooding than current methods in order to achieve a more accurate interpolation among adjacent data samples.

In the area of interpolations, the cooperation with our colleague ing. Karel Uhlíř is possible. He uses the Radial Basis Functions (RBF) for the interpolation. We have already tested this method in 2D for the iso-line extraction. The RBF functions can interpolate the data value within a cell also with the use of local surroundings.

The synthetic sphere and mathematical objects that were used for a whole set of tests were not sufficient. More complex objects should be used to obtain results that would correspond more accurately to real data sets. As described in [14] there are many tessellations of a cube cell. We are working on the error behaviour exploration using different tessellation schemes. In addition, recent methods will be implemented and tested to gain more complete overview of all existing approaches.

The reading of interesting papers from various fields and their analysis weather they can be applied in our area or not will also follow. But the final way our research will head towards depends on a further research, results and analyses.

The following list summarizes the goals of the doctoral thesis:

1. To generate more complex objects and to explore the behaviour of tested methods on such data.
2. The extension of the 2D iso-line extraction algorithm with RBF into 3D and its implementation.
3. To test the influence of the size of local neigbourhooding to the extracted iso-surface within a cell.
4. To use the flowline curvature and develop an adaptive iso-surface extraction algorithm with use of RBF interpolation.
5. The comparison of developed algorithm with existing ones.

Acknowledgements

The thanks belong to my parents and close relatives, because they provided me with a stable and pleasant background that supports my motivation.

I would like to thank to my colleagues and friends for their valuable comments, which contributed to the research within this topic. My thanks also belong to prof. Václav Skala for his comments and financial support and for giving me an opportunity to be one of the staff members during last few WSCG conferences. In addition, I would like to thank to doc. Ivana Kolingerová for her help in administration tasks and also in my particular coordination. Karel Uhlíř should be mentioned here as well, because of his perfect cooperation with me, when working with RBF functions.

The Ministry of Education of Czech Republic project number MSM 235200005 (Information Systems and Technologies) supported this work.

7 References

[1] Bonnel,K.,S., Duchaineau,M.,A., Schikore,D.,R., Hamann,B., Joy,K.I.: Material Interface Reconstruction, IEEE Transactions on Visualization and Computer Graphics, Volume 9(4), Pages 500-511, 2003

[2] Chan,S.,L., Purisima,E.,O.: A New Tetrahedral Scheme for Iso-surface Generation, Computers & Graphics, Volume 22(1), pages 82-90, Elsevier Science Limited, 1998

[3] Chernyaev,E.V.: Marching Cubes 33: Construction of Topologically Correct Isosurfaces, Institute for High Energy Physics, Moscow, Russia, Report CN/95-17, 1995

[4] Cignoni,P., Ganovelli,F., Montani,C., Scopigno,R.: Reconstruction of Topologically Correct and Adaptive Trilinear Isosurfaces, Computers & Graphics, Volume 24(3), pages 399-418, 2000

[5] Cignoni,P., Marino,P., Montani,C., Puppo,E., Scopigno,R.: Speeding Up Isosurface Extraction Using Interval Trees, IEEE Transactions on Visualization and Computer Graphics, Volume 3(2), pages 158-170, 1997

[6] Cignoni,P., Rocchini,C., Scopigno,R.: METRO: Measuring Error on Simplified Surfaces, Computer Graphics Forum, Blackwell Publishers, Volume 17(2), pages 167-174, 1998

[7] Elvins,T.,T.: A Survey of Algorithms for Volume Visualization, Computer Graphics, Volume 26(3), pages 194-201, 1992

[8] Giles,M., Haimes,R.: Advanced interactive visualization for CFD, Computing Systems in Engineering, Volume 1(1), pages 51-62, 1990

[9] Guéziec,A., Hummel,R.: Exploiting Triangulated Surface Extraction Using Tetrahedral Decomposition, IEEE Transactions on Visualization and Computer Graphics, Volume 1(4), pages 328-342, 1995

[10] Itoh,T., Koyamada,K.: Automatic Isosurface Propagation Using an Extrema Graph and Sorted Boundary Cell Lists, IEEE Transactions on Visualization and Computer Graphics, Volume 1(4), pages 319-327, 1995

[11] Itoh,T., Yamaguchi,Y., Koyamada,K.: Fast Isosurface Generation Using the Volume Thinning Algorithm, IEEE Transactions on Visualization and Computer Graphics, Volume 7(1), pages 32-46, 2001

[12] Ju,T., Schaefer,S., Warren,J.: Convex Contouring of Volumetric Data, Department of Computer Science Technical Report, Rice University, 2002

[13] Kindlmann,G., Whitaker,R., Tasdizen,T.: Curvature-Based Transfer Functions for Direct Volume Rendering: Methods and Applications, 14th IEEE Visualization Conference, October 22-24, Seattle, Washington, 2003

[14] Kolcun,A. (Supervisor: Blahetka,R.): Perprocessing pre aplikáciu MKP v úlohách geomechaniky (in Slovak), Dissertation, Ústav geoniky Ostrava, Akademie věd ČR, 1999

[15] Krejza,M., Skala,V.: Iso-surface Extraction and Visualization in Modular Environment System, Computational Geometry Workshop Proceedings, Kocovce, STU Bratislava, Slovakia, ISBN 80-227-14587-7, pages 68-75, 2000

[16] Livnat,Y., Parker,S.,G., Johnson,C.,R.: Fast Iso-surface Extraction Methods for Large Imaging Data Sets, Center for Scientific Computing and Imaging, Department of Computer Science, University of Utah, Salt Lake City, USA, 1999

[17] Lorensen,W.,E., Cline,H.,E.: Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, Volume 21(4), July 1987

[18] Lopez,A., Brodlie,K.: Improving the Robustness and Accuracy of the Marching Cubes Algorithm for Isosurfacing, IEEE Transactions on Visualization and Computer Graphics, Volume 9(1), January-March, pages 16-29, 2003

[19] McDonnel,K.,T., Chang,Y.-S., Qin,H.: Interpolator, Solid Subdivision of Unstructured Hexahedral Meshes, The Visual Computer, Volume 20(6), August, pages 418-436, 2004

[20] Montani,C., Scateni,R., Scopigno,R.: Discretized Marching Cubes, IEEE Visualization Conference Proceedings (Washington, October 18-20), IEEE Computer Society Press, pages 281-287, 1994

[21] Natarajan,B.,K.: On Generating Topologically Consistent Isosurfaces from Uniform Samples, The Visual Computer, Volume 11, pages 52-62, 1994

[22] Nekula,J., Heřman,M., Vomáčka,J., Köcher,M.: Radiology (in Czech), Univerzita Palackého v Olomouci, ISBN 80-244-0259-9, 2001

[23] Neumann,L., Cs'ebfalvi,B., Konig,A., Groller,M.,E.: Gradient Estimation in Volume Data Using 4D Linear Regression, Computer Graphics Forum, Volume 19(3), August, pages 351-358, 2000

[24] Ning,P. and Bloomenthal,J.: An Evaluation of Implicit Surface Tilers, Computer Graphics and Applications, Volume 13(6), November, pages 33-41, 1993

[25] Pasko,A.,A., Pilyugin,V., Pokrovskiy,V. : Geometric Modeling in the Analysis of Trivariate Functions, Computers & Graphics, Volume 12(3/4), pages 457-465, 1988

[26] Patera,J. (supervisor Skala,V.): Methods for Iso-Surface Extraction and Scalar Data Visualization (in Czech), MSc. Thesis, University of West Bohemia, Faculty of Applied Sciences, 2002

[27] Pospíšil,J. (supervisor: Skala,V.): Geometrical Transformations in Discrete Environment (in Czech), MSc. Thesis, Faculty of Applied Sciences, University of West Bohemia, 1999

[28] Poston,T., Wong,T-T., Heng,P-H.: Multiresolution Isosurface Extraction with Adaptive Skeleton Climbing, In Computer Graphics Forum, Eurographics 98, Volume 17(3), pages 137-148, 1998

[29] Schroeder,W., Martin,K., Lorensen,B.: The Visualization Toolkit, 2nd Edition, Prentice Hall PTR, ISBN 0-13-954694-4, 1998

[30] Shen,H.-W., Hansen,C.,D., Livnat,Y., Johnson,C.,R.: Isosurfacing in Span Space with Utmost Efficiency (ISSUE), IEEE Visualization 96, pages 287-294, 1996

[31] Shen,H., Johnson,C.,R.: Sweeping simplicies: A Fast Iso-surface Extraction Algorithm for Unstructured Grids, Proceedings of Visualisation 95, IEEE Computer Society Press, Los Alamos, CA, 1995

[32] Skala,V., Terrades,A.,B.: Two Methods for Iso-Surface Extraction from Volumetric Data and Their Comparison, Machine Graphics & Vision, Volume 9(1/2), Poland Academy of Sciences, ISSN 1230-0535, pages 149-166, 2000

[33] Speray,D., Kennon,S.: Volume Probe: Interactive Data Exploration on Arbitrary Grid, Computer Graphics, Volume 24(5), pages 5-12, 1990

[34] Takahashi,T., Yonekura,T.: Isosurface Construction From a Data Set Sampled On a Face-Centered-Cubic Lattice, Proceedings of ICCVG 2002, Volume 2, pages 754-763, September, 2002
[35] Urbánek,J. a kolektiv: Nuclear Medicine (in Czech), Gentiana Jilemnice, ISBN 80-86527-05-0, 2002
[36] Van Kreveld,M., Van Oostrum,R., Bajaj,C., Pascucci,V., Schikore,D: Contour Trees and Small Seed Sets for Iso-surface Traversal, In Proceedings 13th Annual Symposium Computational Geometry, pages 212-220, 1997
[37] Wirth,N.: Algorithms + Data Structures = Programs, Englewood Cliffs (Prentice Hall), 1976
[38] Žára,J., Beneš,B., Felkel,P.: Modern Computer Graphics (in Czech), Computer Press, ISBN 80-7226-049-9, 1998

Internet References

[W1] Biomedical Research Foundation of Northwest Louisiana, 1998
http://www.biomed.org/pet.html
[W2] Bourke,P.: Polygonising a Scalar Field Using Tetrahedra, 1997
http://astronomy.swin.edu.au/pbourke
[W3] Brownell,L.,G.: A History of Positron Imaging, 1999
http://www.mit.edu/~glb/alb.html
[W4] General Electric, eLearning PET essentials, 2000
http://apps.gemedicalsystems.com/geCommunity/europe/nmpet/pet_education
[W5] Imaginis® Corporation, 2002
http://imaginis.com/ct-scan
[W6] Joseph P. Hornak, Ph.D.: The Basics of MRI, 2003
http://www.cis.rit.edu/htbooks/mri
[W7] Julia Carden: Basic Principles of CT Scanning, ImPACT, 2003
http://www.impactscan.org/slides/impactday/basicct
[W8] Komzák,J. (supervisor Jehlička,K.): Zpracování rušených barevných obrazů ve vhodných soustavách a transformacích (in Czech), MSc. Thesis, ČVÚT, Fakulta elektrotechniky a informatiky, Brno, 1999
http://beety.kgb.cz
[W9] Moriel NessAiver, Ph.D., 2000
http://www.simplyphysics.com/page2_1.html
[W10] Sharp,B.: Subdivision Surface Theory, Gamasutra, 2000
http://www.gamasutra.com/features/20000411/sharp_01.htm
[W11] Weisstein,E.,W.: World of Mathematics, CRC Press LLC, Wolfram Research Inc., 1999
http://mathworld.wolfram.com

Appendix A

Publications – Accepted

Uhlíř,K., Patera,J., Skala,V.: Radial Basis Function Method for Iso-line Extraction, Electronic Computers and Informatics, ECI 2004, Herľany, Slovakia, September 22-24, 2004

Publications – In Review Process

Patera,J., Skala,V.: A Comparison of Fundamental Methods for Iso-surface Extraction, Machine Graphics & Vision

Patera,J., Skala,V.: Centered Cubic Lattice Comparison, Algoritmy 2005

Bachelor Degree (Bc.) Students

Jiří Šeda (Supervisor: Jan Patera): Konverze AVI videa na volumetrická data (in Czech), Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia in Pilsen, Czech Republic, 2004. (In cooperation with MUDr. Zdeněk Chudáček from FN Bory hospital)

Study and Work Experience

2001 - 5 months stay (2nd term) as a student at the University of Bath, England, United Kingdom (http://www.bath.ac.uk)
2003 - Research worker in the area of technical science, Department of Computer Science and Engineering, University of West Bohemia, Czech Republic

Internal Seminars

Patera,J.: Metody extrakce iso-ploch a chyby aproximace (in Czech)

Teaching (Tutorials)

ZS2002 – KIV/SOJ: Low level programming in ASM (Intel 286)
LS2003 – KIV/POT: Low level programming in ASM (Zilog 80)
ZS2003 – KIV/ZPG: Basics of computer graphics (Visual C++, ANSI C)
ZS2004 – KIV/WIN: Programming under Windows (Win32, C#, .NET)