

JACOBS UNIVERSITY BREMEN

# Topology - Based Isosurface Representation

by

Maja Grintal

Guided Research: Final Report  
supervised by: Prof. Dr. Lars Linsen

in the  
School of Engineering and Science  
Computer Science

May 2009

JACOBS UNIVERSITY BREMEN

# *Abstract*

School of Engineering and Science  
Computer Science

Guided Research Thesis

by Maja Grintal

A lot of visualization of scalar fields can come down to the extraction and rendering of isosurface. Isosurfaces may be used for representing volumetric data for visualizing fluid flow in computational fluid dynamics. They are widely used in medical imaging, allowing the visualization of internal organs, bones or other structures. Many large data sets have to be visualized interactively as required by many applications such as computer aided surgery or scientific visualization.

Marching cubes algorithm plays an important role in isosurface extraction and is used most widely due to simpleness and efficiency. However when the isovalue is changed slightly, the entire algorithm is executed again, although only minor changes occurred to the isosurface. The isosurface extraction may be simply too slow in order to be able to change the isovalue interactively. This project concentrates on investigating effective way to do this interactively reducing the computational cost.

# *Acknowledgements*

I would like to thank [Prof. Dr. Lars Linsen](#) for the great guidance throughout the whole guided research. His suggestions and evaluations helped me very much during the research.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>List of Figures</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 About . . . . .	1
1.2 Related Work . . . . .	1
<b>2 Work Documentation</b>	<b>3</b>
2.1 Modeling . . . . .	3
2.2 Isosurface Correspondance . . . . .	4
2.3 Investigation . . . . .	5
2.4 Determining Points . . . . .	5
2.5 Methods Investigated . . . . .	6
2.5.1 Method . . . . .	6
2.5.1.1 Description . . . . .	6
2.5.1.2 Problem . . . . .	6
2.5.2 Method . . . . .	7
2.5.2.1 Description . . . . .	7
2.5.2.2 Problem . . . . .	7
2.5.3 Method . . . . .	7
2.5.4 Description . . . . .	7
2.6 Connection of Triangles . . . . .	8
2.6.1 Isosurface Expanding . . . . .	8
2.6.2 Isosurface Shrinking . . . . .	9
2.7 Datastructure . . . . .	10
<b>3 Evaluation</b>	<b>11</b>
3.1 Testing and Discussion of Results . . . . .	11
<b>4 Future Work</b>	<b>14</b>
4.1 Further Improvements . . . . .	14
4.1.1 Implementation Improvements . . . . .	14

---

4.1.2 Future Research . . . . .	14
<b>5 Conclusion</b>	<b>16</b>

# List of Figures

2.1	Data . . . . .	3
2.2	maximum isovalue . . . . .	4
2.3	minimum isovalue . . . . .	4
2.4	3 triangle division approach: solid Isovvalue 500 . . . . .	6
2.5	3 triangle division approach: wire Isovvalue: 500 . . . . .	6
2.6	Connecting Triangles: Neighbouring Triangle Approach . . . . .	7
2.7	Neighbouring Triangle Problem . . . . .	8
2.8	Connecting Triangles: Final Approach . . . . .	9
3.1	Isovvalue 324 MC Approach . . . . .	12
3.2	Isovvalue 324 Our Approach . . . . .	12
3.3	Isovvalue 100 hill in the structure . . . . .	12
3.4	Isovvalue 970 and Isovvalue 420 . . . . .	13
3.5	Isovvalue 230 and Isovvalue 64 . . . . .	13

# 1

## Introduction

### 1.1 About

Volumetric data is used in many disciplines such as biomedical science, computer graphics, and visualization. Visualization of such data is important to understand their geometrical properties. There are many ways of visualizing volumetric data. Isosurface representation is the most common one. Isosurface (3-D contour surface) is a surface that represents points of a constant value (e.g. pressure, temperature, velocity, density) within a volume of space. In other words is a level set of a continuous function whose domain is 3D-space. Many methods for generating isosurface have been reported. Most of these generate an isosurface as an approximated polyhedron composed of small triangles. One of the well-known methods for generating isosurfaces is the marching cubes (MC) method. The MC method tessellates the space into small cubic cells and generates triangle patches in the cells that intersect boundary surfaces.

This guided research is focused on finding an appropriate solution for representing the isosurface, when the isovalue is changed only slightly without executing the MC algorithm again. Furthermore exploring different ways for appropriate representation of the isosurface. Also the coherence of isosurfaces is being exploited to store a whole set of isosurfaces in one representation.

### 1.2 Related Work

For visualization of volume data one of the most common and useful tool is isosurface extraction. In the last years, the resolution of volumetric data sets has been increasing dramatically. This applies to typical measurement data arising e.g. in medical imaging

as well as to output of large-scale numerical simulations. Very, often, an isosurface, represented as a triangle mesh, is extracted in a preprocessing step. This can be very time-consuming when standard marching algorithms are used. Therefore, a variety of methods have been designed to speed up the extraction step [7,8], or to limit it to visible triangles [10, 11]. A lot of research is done in optimizing these algorithms. The original MC method was introduced by Lorensen and Cline[9]. [2] Several authors have proposed extensions with the aim of generating consistent and topologically correct isosurfaces.

Isosurface topology provides insight into the fundamental structure of isosurface behavior across isovalues. Topological changes occur if critical points exist. Gunter H. Weber, Geric Scheuermann and Bernd Hamann give insight into detecting critical points and regions in scalar fields.[1] Also a method for detecting critical regions has been proposed by Weber[3] et al. They consider piecewise trilinear data sets and detect critical points at grid vertices, faces, and the interior of cells.[1] Van Kreveld[5] used contour trees to speed up isosurface extraction. A contour tree is a graph that describes how contours in a level set appear, join, split or disappear. Fujishiro [6] used a hyper-Reeb graph for exploration of scalar fields. A hyper-Reeb graph encodes changes of topology in an extracted isosurface.

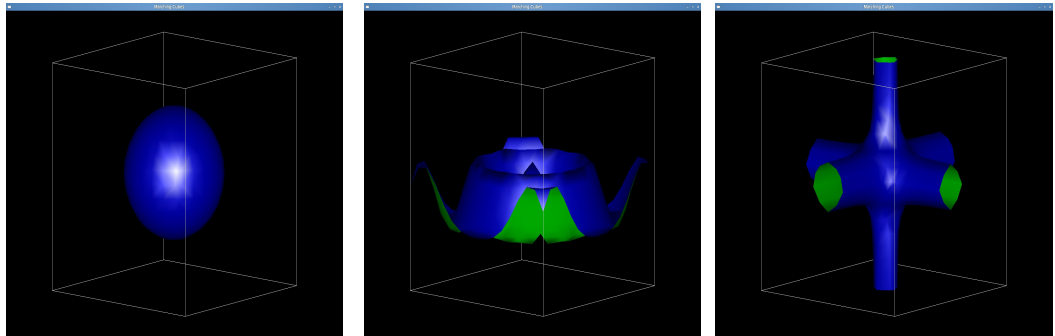
However, there is not much research done in the field how to model the changes in topology if the isovalue is slightly change. We want to find a way how to model the new isosurface without executing MC algorithm again, but rather by finding a correspondence between isosurfaces.

## 2

# Work Documentation

## 2.1 Modeling

The initial Isosurface is rendered using Marching Cubes(MC) Algorithm. The MC method tessellates the space into small cubic cells and generates triangle patches in the cells that intersect boundary surfaces. For implementation of the Marching cube algorithm an open source code by Cory Boyd is used.[4]. For the isosurface generation three different datasets can be used, which are visualized as a sphere, a wave function and two intersecting cylinders. See figure .



---

FIGURE 2.1: The 3 Different Data Sets used

OpenGL is used to visualize the isosurface and to respond to user interactions at any time. Both the wireframe and the solid mode of the structure can be used to help us investigating the new structure.

## 2.2 Isosurface Correspondence

When the isovalue is changed for a small amount the isosurface will change only slightly or not at all. It may happen that when changing the isovalue the isosurface is only expanding or shrinking.

After executing the Marching Cubes algorithm we store the whole structure all triangles that are connected in a global structure. When changing the isovalue for each point of the triangles a corresponding point from a second isosurface is found. This is done so that for each point of the first isosurface we measure the smallest distance to a point in the second configuration. Since the triangle connection is saved we use the same configuration but with the new corresponding points to connect the new isosurface.

In the figure 2.2 we can see the initial isosurface for maximal isovalue (points concentrated in the middle) and in figure 2.3 the isosurface that we get by corresponding triangles for almost minimal isovalue. As we see the topology is not changed. The number of triangles used in the first isosurface is the same as the number of triangles in the last. This is used directly if the number of points found from first generation of the isosurface and the number of points found in the second generation does not change. If there are more points for the second isosurface generation, we continue with further investigation.

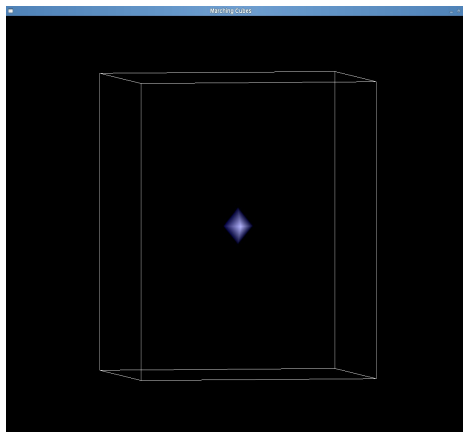


FIGURE 2.2: maximum isovalue

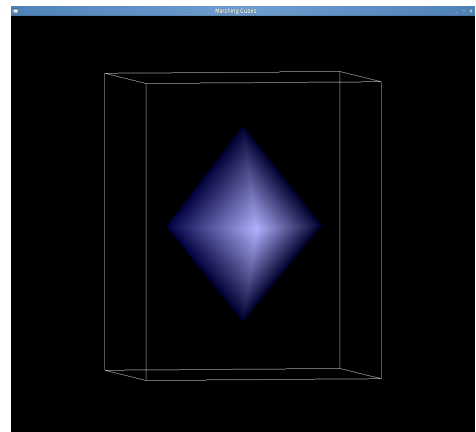


FIGURE 2.3: minimum isovalue

## 2.3 Investigation

As mentioned before as the isovalue changes the topology of the isosurface may change also. Because we are interested in only small changes of the isovalue we do not immediately observe very drastic changes in topology. We want to represent the new isosurface from the previous configuration. We need to investigate two different ways. Whether there are more points in the new configuration than the previous or whether there are less. In the first case points and triangles should be inserted in order to get the correct structure, in the second case two or more points correspond to one point in the second configuration which means that some of the existing triangles should be deleted. For the first part we disregard critical points and/or regions. The methods investigated for finding the correct most appropriate connection between the new triangles are discussed in the [2.5 Methods Investigated](#) section.

## 2.4 Determining Points

In section [2.2 Isosurface Correspondance](#) we discussed how to find the corresponding triangles for the second isosurface from the previous configuration. To determine wheather and where the new points should be inserted we look in the triangles from the second isosurface. One approach that we looked into was finding the longest edge of the new triangle, taking the middle point of it and finding the closest point to the middle one. However this is not good approach since the longest edge may actually return a vertex of the triangle instead of determining the new point which exists and is closer to other edge from outer side.

Better way which gives us correct results is using the centroid of the triangle. Namely for each of the triangles of the second isosurface we search for the closest point from second configuration to the centroid of that triangle.

If the found point is same as any of the vertices of the triangle, we can conclude that the same connection between points can be used, so we keep the same triangle. If the found point is different from any vertex of the triangle that we are looking into, we will need to include that point in the new configuration, making new triangles and deleting the old one. In section [2.5 Methods Investigated](#) we discuss different possibilities problems and the main challenge of finding the new connection of the triangles.

## 2.5 Methods Investigated

### 2.5.1 Method

#### 2.5.1.1 Description

When we find the closest point of the second isosurface to our triangle, we need to insert new triangles. One method is dividing the triangle that we already have in 3 different triangles. These 3 triangles we get so that we connect each two vertices of the triangle together with the new point.

#### 2.5.1.2 Problem

The main problem in this approach is that the initial edges of the triangle we are working on never get divided. So when the structure is growing we can observe the long edges that are preserved all the time when changing the isovalue. Basically those edges correspond to the very first triangle that was generated by MC algorithm and then propagated by the corresponding points every time the isovalue is changed.

From the figure 2.4 we can observe that all points were correctly determined, however looking into the wireframe of the same object, figure 2.5, constructed according to this method we notice the long edges that are preserved. This method does not give us very smooth structure.

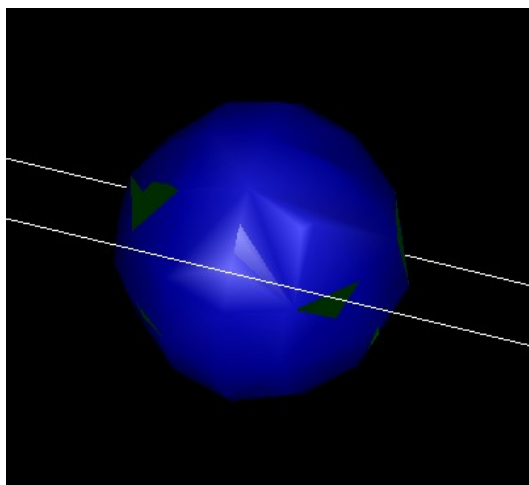


FIGURE 2.4: 3 triangle division approach: solid Isovalue 500

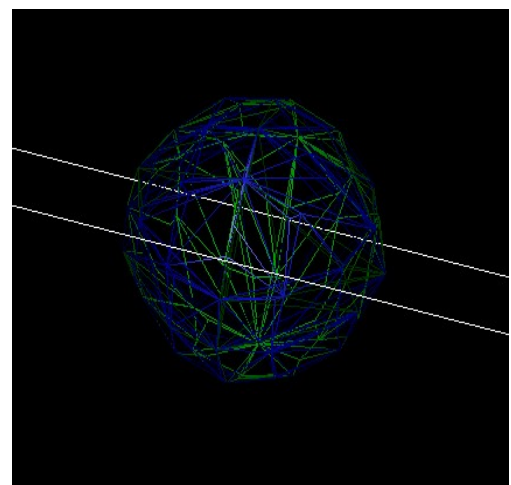


FIGURE 2.5: 3 triangle division approach: wire Isovalue: 500

## 2.5.2 Method

### 2.5.2.1 Description

The second method investigated was dividing each triangle in two other triangles. First of all when finding the closest point to the triangle we look which edge of that triangle is closest to the new point. This is done so that we compare distances from the middle points of the edge to the new point. After determining the edge we divide the current triangle in 2 new triangles, so that we connect the opposite vertex to the edge, to the new point and to one vertex of the edge. Because we actually deleted one edge, we need to search for a neighboring triangle who shares that edge to divide it also into two triangles. This is visualized in figure 2.6

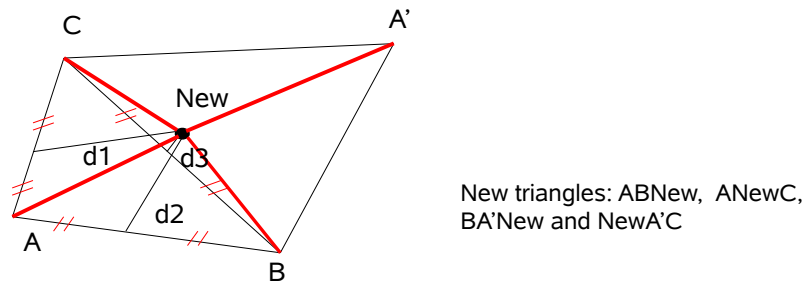


FIGURE 2.6: Connecting Triangles: Neighboring Triangle Approach

### 2.5.2.2 Problem

The problem with this method is that it is possible to get some kind of hills that appear if two triangles are neighbors, but both have different closest point to a different closest edge. They are connected with the closest points and with other neighboring triangles. In figure 2.7 the region that may occur in this case is highlighted with the red line.

## 2.5.3 Method

### 2.5.4 Description

Combination of these two methods produces the most correct results. This final method is discussed in section 2.6 Connection of Triangles

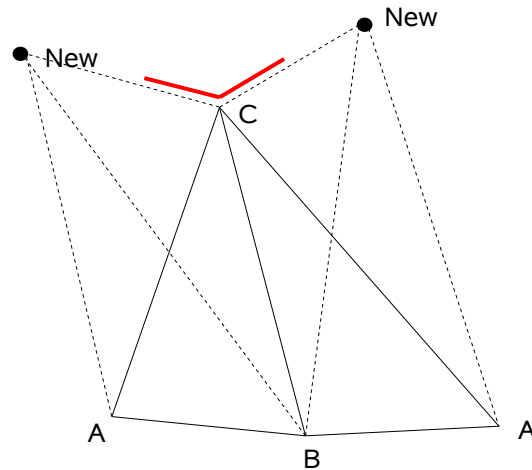


FIGURE 2.7: Neighboring Triangle Approach Problem. Two neighboring triangles connected with 2 different new points, forming kind of hill between them.

## 2.6 Connection of Triangles

The algorithm that we came up with for determining good connection between triangles is based on the previous methods discussed above in section [2.5 Methods Investigated](#).

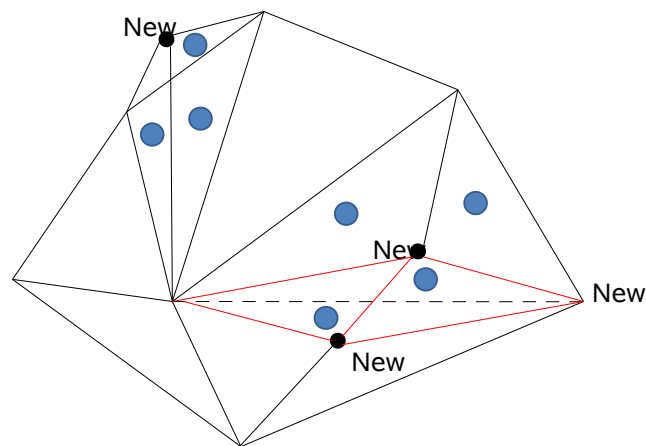
### 2.6.1 Isosurface Expanding

By Isosurface expansion we mean that the isosurface that needs to be modeled has more points than the previous one. For each triangle that we have from the second isosurface we check whether there is a point that needs to be inserted. We determine new points as discussed in section [2.4 Determining Points](#). We start looking into all triangles from the second isosurface found by the correspondence of points. For the closest point to a triangle we check whether it is the same as any of the vertices of the triangles. If the point is a vertex we keep the same triangle, if it is different we need to include that point in our configuration. A triangle that needs to be divided I am gonna call it critical. First we divide all critical triangles in 3 parts as described in [2.5.1.1 Description](#). All new constructed triangles we save it in other structure.

Then when finishing with the first division we continue with the newly constructed triangles. Basically for all those triangles we search whether the neighboring triangle was divided or not, i.e. it is saved in the same structure or not. If there is not neighboring

triangle being divided we put the old triangle, namely the triangle from the first division, in our structure.

If a neighbor was found first we check whether the distance of the edge that the two triangles share is longer than the distance of the opposite vertices of both triangles. If it is longer we connect the two vertices opposite of the found edge forming 2 new triangles. The new triangles consist of vertex of first triangle opposite the edge, first vertex of edge and second vertex of the other triangle. Same for the other triangle with the second vertex of the edge that they are sharing. In figure 2.8 the triangles with red are the new triangles that are connected in other way. And the rest are preserved.




---

FIGURE 2.8: Connecting Triangles: Final Approach

So instead of putting the old two triangles in our structure we put the new two. We do not have more triangles, just the connection changes. This is done to prevent in a way forming of very long edges. The way that we are gonna change the connection is more intuitive since the isosurface tends to expand.

### 2.6.2 Isosurface Shrinking

The shrinking of isosurface basically represents movement from larger isosurface to smaller one. Mainly there are more points in the first isosurface than in the second one. That would suggest that two or more points of the first isosurface correspond to same point in the second one. Some of the triangles from the first isosurface will be lost in the representation of the second.

First for all the triangles we find the corresponding points in the second isosurface. For the triangles in the second isosurface we check whether the values of at least two vertices are the same. If we find that two vertices from a triangle collapse to the same point we can conclude that that triangle will not exist in the second isosurface so we neglect that one.

## 2.7 Datastructure

The main idea behind exploring the coherence of the isosurfaces was to help us store the whole set of isosurfaces in one representation. Everything will be precomputed and then directly an isosurface for specific isovalue will be displayed.

When running the algorithm on large set of isovalues, for example 900, we decide which triangle belongs to which isovalue. We have saved the structure as triangle, range, where the range consist of two integers. The first integer is the value of the isovalue when a triangle appeared for the first time in some of the isosurfaces, and the second one is the value of the last isovalue when the triangle appeared for the last time in an isosurface.

In this way we save the connection of triangles with the beginning points when the triangle was seen for the first time. Later when an isovalue is inserted we can directly display the triangles for which the isovalue belongs in their range. We need to do just a correspondence of triangles to find the new values of the isosurface that we need to display. This is quite efficient because instead of storing 1000ands of triangles for many isosurfaces we actually store very small number. The correct values of the points of the triangles for each different isovalue are not stored. The values that are stored in our structure are the values of the points from the first isosurface where the triangle appeared for the first time.

To find the correct values of the vertices for each triangle one needs to find correspondence of points. One needs to be careful when searching for the correspondence of the triangles since when many points are inserted it may happen that we don't get the correct point that we wanted but another point with a smaller distance. This can be resolved in different ways. Whether searching in some intervals or going step by step changing the values of the triangle.

# 3

## Evaluation

### 3.1 Testing and Discussion of Results

The primary testing was done on the first data set, the sphere. This was used because it is quite intuitive, meaning the isosurface expands for which we needed the testing in first place. The initial structure has very few triangles and as the isovalue changes the structure becomes more alike a sphere adding new points and constructing new triangles all the time.

First the correspondence of the points was tested which gave us correct results all the time. In the figures ?? and ?? we can see the result from starting with a very few triangles for a large isovalue to a very minimal isovalue.

Second part of testing included testing of determining new points. Our method of doing so, gave us very correct results. The points that lie on the second isosurface and need to be inserted were always found. We have done a comparison between the MC algorithm and our method and we can notice that our algorithm gives correct results for finding new points. The Marching Cubes ?? algorithm was directly executed for an isovalue of 324. And our algorithm started with isovalue of 970 going to 324 3.2 with a change of the value of -2. We can see the different connection and the points being inserted. One can notice that all the points are correctly determined.

Starting with a maximal isovalue of 970 going to the the minimal 24 we can notice that we get quite good results. However sometimes happen that we get “hills” in our structure. As more and more points are being inserted probably we get some small errors in our structure. For example for isovalue of 100 we can get the hill shown in figure 3.3

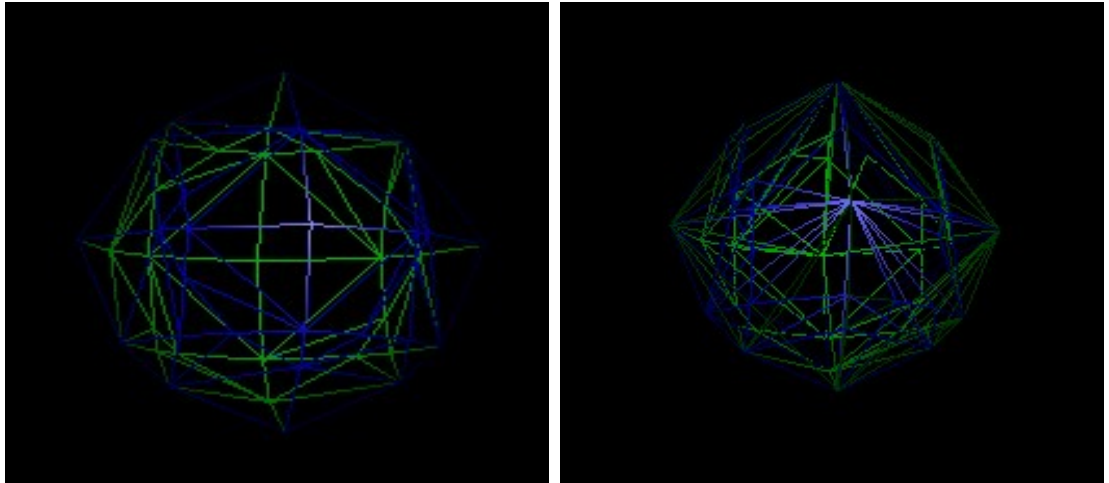


FIGURE 3.1: Isovalue 324 MC Approach

FIGURE 3.2: Isovalue 324 Our Approach

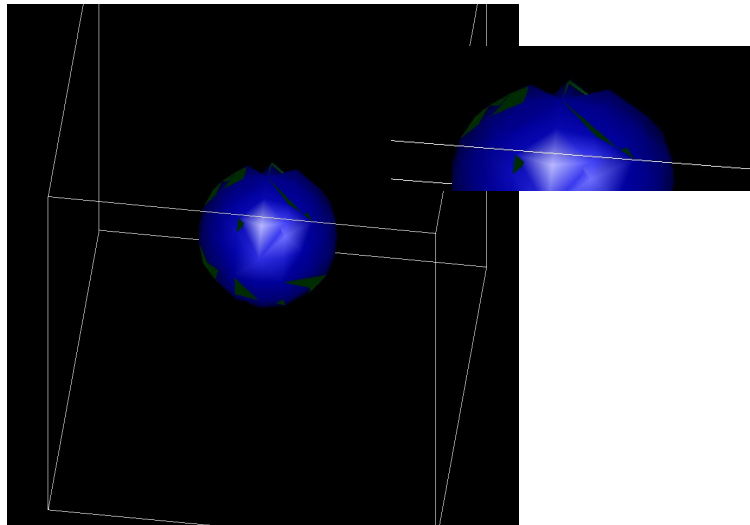
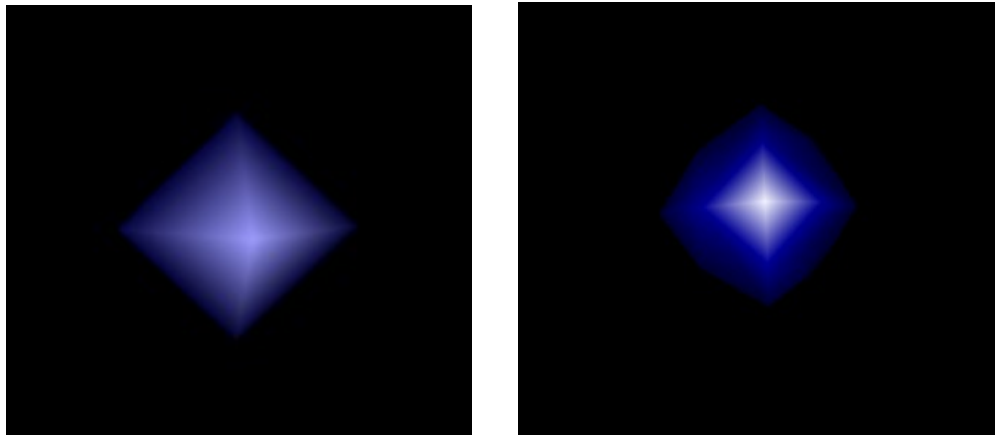


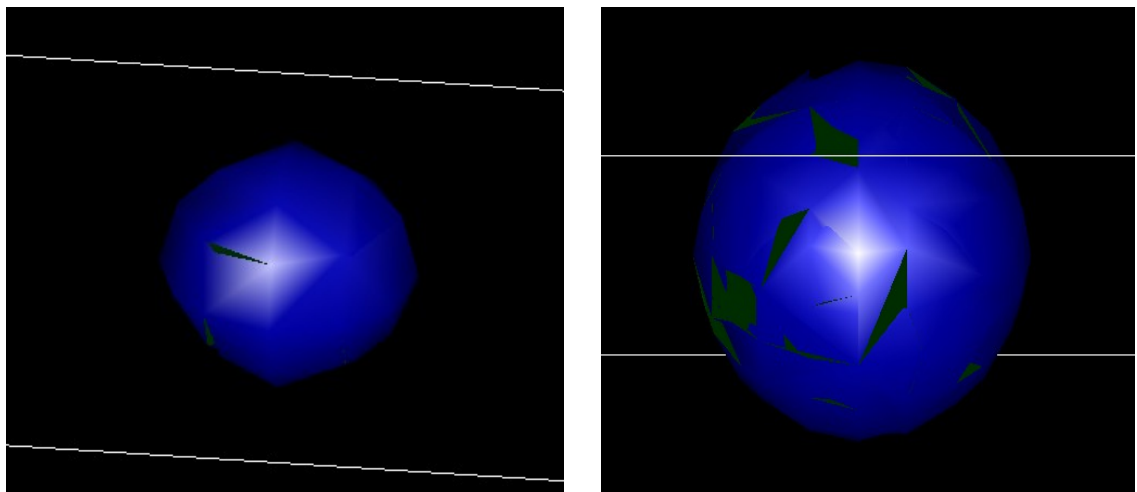
FIGURE 3.3: Isovalue 100 hill appears in our structure

In the next figures 3.4 and 3.5 some of the snapshots for different isovalues can be seen. We can observe the newly formed structure for a range of 900 different isovalues. We can conclude that the structure for the new approach is quite accurate for a great range of isovalues, still when inserting more and more points there can be some small errors that need to be resolved in other way.



---

FIGURE 3.4: Isovalue 970 and Isovalue 420



---

FIGURE 3.5: Isovalue 230 and Isovalue 64

## 4

# Future Work

## 4.1 Further Improvements

### 4.1.1 Implementation Improvements

The implementation for storing the whole structure can be improved in different ways.. Right now after storing the whole structure only the values of the vertices when the triangle appears for the first time are stored. However when finding correspondence for points it may happen that a triangle's connection appeared for the first time for very large isovalue and also appeared in very small isovalue. That would mean that we have great distance between the values of the vertices. If we do only correspondance of points in one step then it is quite obvious that we can get wrong points as the closest one. This issue can be solved in a way wheather we can keep instead of the initial values when the triangle appears, a value somewhere in the middle. This would give more accurate results. Also we can have an approach of finding the new values in 2 or 3 steps, such that we take intervals between the start isovalue of the triangle. We compute the new values of the vertices for an isovalue approximately one third in the range and then once again for two thirds. For example if the triangle is in range of 900 - 300, and we want to display isosurface for isovalue at 350, we find correspondence of points from 900 to 700, 700 to 500 and then to 350.

### 4.1.2 Future Research

This implementation focuses on very small change in the isosurface. However it may happen that there exist critical points and regions. In this case the topology of the isosurface changes. This research can be continued in examining the behavior at these

regions. In a  $C^2$ -continuous function  $\nabla f$ , critical points occur where the gradient  $f$  vanishes, i.e., where  $\nabla f = 0$ . By considering the definitions from calculus, it is possible to classify a point by considering its neighborhood. [1] Any given position of data set can be determined by considering a small neighbor around it. There are three types of critical points that are needed to be consider:

- at a local minimum a closed component is created
- at a saddle the *genus* of an isosurface changes, or components separate/merge
- at a local maximum a closed surface component vanishes

For localized minimum/maximum the minimum/maximum is surrounded by larger/smaller values. At a saddle either holes will appear/disappear or surface components separate or merge at a point or along a region.

## 5

# Conclusion

With our research we tried to develop a completely new way of representing isosurfaces. The algorithm that we came up with is quite efficient and at the end only small number of triangles for a large dataset of isosurfaces can be stored. This can find great use in practice especially in the field of biomedical visualization as well as graphics and visualization. With further improvements this can be very efficient way of saving isosurfaces in precomputation step, and in representing them without doing all computation done with the marching cube algorithm.

# Bibliography

- [1] Gunter H. Weber, Gerik Scheuermann, Bernd Hamann. Detecting Critical Regions in Scalar Fields, Joint Eurographics - IEEE TCVG Symposium on Visualization, 2003
- [2] Thomas Gerstner, Renato Pajarola. Topology Preserving and Controlled Topology Simplifying Multiresolution Isosurface Extraction.
- [3] Gunther H. Weber, Gerik Scheuermann, Hans Hagen, and Bernd Hamann. Exploring scalar elds using critical isovalues. In: Robert J. Moorhead, Markus Gross, and Kenneth I. Joy, editors, IEEE Visualization 2000, pages 171178, IEEE, IEEE Computer Society Press, Los Alamitos, California, 2002.
- [4] Marching Cubes Example Program by Cory Bloyd
- [5] Marc van Kreveld, René van Oostrum, Chandrajit Bajaj, Valerio Pascucci, and Daniel Osvaldo. Contour trees and small seed sets for isosurface traversal. In: Jean-Daniel Boissonnat, editor, Proceedings of the Thirteenth ACM Symposium on Computational Geometry, pages 212219, ACM Press, New York, New York, June 46 1997.
- [6] Issei Fujishiro, Taeko Azuma, and Yuriko Takeshima. Automating transfer function design for comprehensible volume rendering based on 3D eld topology analysis. In: David S. Ebert, Markus Gross, and Bernd Hamann, editors, IEEE Visualization 99, pages 467470, IEEE, IEEE Computer Society Press, Los Alamitos, California, October 2529, 1999.
- [7] T. Itoh and Y. Yamaguchi. Isosurface Generation using Extrema Graphs. In Proceedings IEEE Visualization 94, pages 7783. IEEE Computer Society Press, 1994.
- [8] Y. Livnat, H. Shen, and C. Johnson. A Near Optimal Isosurface Extraction Algorithm using the Span Space. IEEE Transactions on Visualization and Computer Graphics, 2(1):7383, 1996.
- [9] W. Lorensen and H. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Computer Graphics, 21(4):163169, 1987.
- [10] T. Gerstner, M. Rumpf, and U. Weikard. Error Indicators for Multilevel Visualization and Computing on Nested Grids. Computers Graphics, 24(3):363373, 2000.

- [11] Y. Livnat and C. Hansen. View Dependent Isosurface Extraction. In Proceedings IEEE Visualization 98, pages 175180. IEEE Computer Society Press, 1998.