Realtime view-dependent isosurface visualization for regular volume data

Maxim V. Kazakov
Moscow Institute of Physics and Technology
Dolgoprudny, Russia

Abstract

The problem of the interactive visualization of isosurfaces in volume data arises in medical applications, visualization of mathematical simulation results and many other areas. We propose an adaptive approach for interactive extraction and visualization of isosurfaces that employs hierarchical representation for the volume data and view-dependent isosurface reconstruction at the different levels of detail. To speedup extraction process an isosurface is constructed only in the visible part of the dataset and its updates are performed incrementally as observer moves. Possible cracks between isosurface parts constructed at different hierarchy levels are eliminated by proposed stitching procedure. Due to low dataset preprocessing costs our approach has been proven to handle dynamical updates in volume data.

Keywords: *interactive isosurface reconstruction, volume datasets, Marching Cubes, level of detail.*

1. INTRODUCTION

Isosurface construction is one of widely adopted approach for scalar volume datasets visualization. Methods constructing polygonal approximation of isosurface are appealing in the applications where rendering rates are need to be near real ones because of powerful 3D hardware accelerators boosting rendering of polygonal meshes. Existing polygonal isosurface construction methods are based on Marching Cubes algorithm [9], which performs triangulation for every cell of a scalar volume dataset thus reconstructing the whole isosurface. Several methods exist to disambiguate cell triangulation and eliminate “holes” that are possible in a mesh produced by original MC algorithm [2,5,10]. However simple and robust, Marching Cubes algorithm produces unacceptably huge amount of polygons for datasets that are typical for CT and numerical simulations. Besides, mesh construction times also make a real-time visualization for such datasets almost impossible.

To handle such a mesh complexity one can employ various mesh simplification methods on a post – process step [3,4,5,7,8,11,14,16] but it does not help with mesh construction times that are generally determined by the dataset’s dimensions. The latter issue is of a concern for methods that construct hierarchies of datasets with successively smaller dimensions from source dataset that are employed for a coarser isosurface representation [13,15]. Some of them are based on the octree approach where each of the eight neighboring cells corresponding to octree nodes can be recursively combined into a larger cell that corresponds to parent octree node.

Selecting a hierarchy level for cells to be processed during isosurface construction one controls a level-of-detail for resulting surface. This idea had its further refinement in [15] where isosurface is constructed with dataset’s hierarchy level varying over dataset parts thus producing varying level-of-detail in resulting surface’s parts. A distance to observer and local surface curvature determines hierarchy level selected at a dataset part. Another selection criteria can be applied as well.

Construction of isosurface patches at different dataset hierarchy levels imposes problems with isosurface continuity [5]. This issue was addressed in [5, 15] where special stitching procedure for isosurface patches extracted from different hierarchy levels was proposed.

Implementation of this adaptive hierarchical approach in isosurface construction was shown to be a basis for real-time isosurface visualization [15]. Although the method for interactive isosurface visualization this paper describes is based on similar ideas it proposes certain optimizations on isosurface extraction and visualization processes. Another issue concerned by method proposed is a visualization of dataset modifications. The paper layout is as follows: in the Section 2 we overview hierarchical construction approaches for faster extraction and visualization of isosurfaces. Section 3 addresses stitching isosurface patches generated at different level of hierarchy. Section 4 describes how we employ view dependency to minimize processing and amount of polygons produced during isosurface construction. Section 5 addresses visualization of dynamically updated datasets. Paper is finalized with results described in Section 6. Conclusion and plans for future are in Section 7.

2. HIERARCHICAL ISOSURFACE CONSTRUCTION

Hierarchical approach in volume visualization implies usage of datasets originated from the source dataset that can be used to construct successively coarse representations. Some methods construct such a dataset hierarchy applying low - pass filter to the source data several times and then adaptively process smoothed dataset regions extracting isosurface in it [5,15]. Although creating appealing visualization results, those methods require complex and lengthy filtering of the dataset.

Another approach used to construct such a dataset hierarchy employs subsampling. In this case the even samples along each direction are employed to create a subsampled dataset representing another level of hierarchy.

If datasets from neighboring hierarchy levels differ for two times in size for each direction, it is easy to construct an octree over their cells. As one cell contains eight smaller ones from neighbor hierarchy level, the corresponding octree node refers eight other nodes corresponding to smaller cells. Octree nodes may carry some information used during isosurface extraction/construction. In our case the main goal is checking an existence of an isosurface patch in cell. When using an octree the node existence can represent a surface patch existence for a corresponding cell so an isosurface can be constructed by simply traversing existing

nodes in octree and pruning traversal as required level-of-detail is reached.

As shown in [15], usage of more than 2 or three detalization levels and corresponding dataset hierarchy levels results in distracting visual artifacts during isosurface construction so a support for all $\log_2 N$ hierarchy levels (where N is a number of samples along direction for original dataset) is excessive. Moreover, usual octree organization where each node requires up to eight (possibly 4-byte) pointers to its children might be memory consuming. We propose another hierarchical data structure that holds information about surface patch presence in dataset cell. For each dataset hierarchy level a bitmask is created and supported. Each cell containing isosurface patch in it or one of its children cells has corresponded "one" in bitmask while others have "zeroes" set in it. Such a bitmask adds a memory overhead no more than 12,5 % for each dataset hierarchy level if byte is used for scalar value and even less if four-byte floats or eight-byte doubles are used. The memory overhead imposed by bitmask allows checking an isosurface patch existence in a cell from any hierarchy level of interest using a single bitmask lookup instead of traversing of several octree levels. Moreover, the amount of operations required for bitmask construction does not exceed corresponding amount for an octree construction.

Summarizing above it could be said that our approach while being based on hierarchical isosurface extraction / construction ideas as illustrated in [15] uses its own hierarchical data structures that speed up checking an isosurface patch existence in any cell from any dataset hierarchy level.

3. STITCHING PROCEDURE

Straightforward usage of different dataset hierarchy levels for LOD isosurface reconstruction leads to certain problems with continuity in resulting polygonal mesh [5]. As stated in [15] it is due to scalar field discontinuities introduced by subsampling or low – pass filtering used to construct levels of dataset hierarchy. Cracks in surface produced at the boundary of dataset regions with different detailing in them are illustrated at Fig. 1. That is why after applying regular MC-like method for hierarchical isosurface construction an additional stitching procedure should be performed for elimination of possible cracks.

This procedure may modify isosurface patch at finer detailing level, coarser detailing level [5] or both. Our approach employs fine-detailed patch modifications and assumes that neighboring cells are reconstructed using the same or neighbored dataset hierarchy levels (i.e. they differ for no more than two times in their dimensions).

Let us consider rectangle ABCD that is on the boundary between regions with different detailing. Linear interpolation of values at corresponding edge centers A', B', C', D' between values at vertices of the hosting edges AB, BC, CD, DA leads to patches coincidence at the edges mentioned causing points P" and Q" where isocurve intersects rect's edges to became the P' and Q'. But this does not eliminate cracks in internals of ABCD [15] because isocurve is approximated by straight line at coarser detailing level while being a broken line at the finer one. That is where our stitching procedure comes into play. It is proposed to straighten (if necessary) isocurve at the finer detailing level by moving isocurve vertices that lay inside rectangle ABCD to their neighbors on rectangle edges (moving P to P' and Q to Q'). After that modifications the finer isocurve completely coincides with coarse one.

Because result of stitching procedure is ruled only by data from the coarser level the original value at the center of ABCD is of no concern. However, it is required for initial construction of the finer isocurve. Actual value at the central vertex is of no concern but its sign choice should be ruled by a disambiguation method employed during polygonizing of isosurface. In our work the "preferred polarity" [2] approach is used so the value sign in face center is chosen to comply with "polarity" used during cell polygonizing.

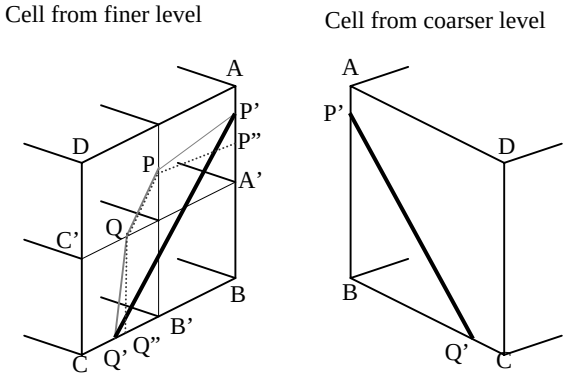

Fig. 1. P'Q' - isocurve at coarser level
P"PQQ" - initial isocurve at finer level
P'PQQ' – isocurve at finer level with interpolated values at A',B',C' and D'

Actually, only sample signs at face corners as well as interpolated samples signs in the centers of the face edges determine all vertex replacements performed during stitching. Thus all possible (valid) combinations of the signs can be easily summarized in lookup table to speedup stitching process.

4. VIEW-DEPENDENT ISOSURFACE RECONSTRUCTION

Knowledge about observer's position during visualization helps in performing certain optimizations on the visualization process. As was already mentioned in Section 2, the isosurface in different volume parts can be refined adaptively according to some criteria. Distance to a current observer' position provides us with one of such criterion that allows to leave distant surface parts at coarser levels of detalization thus reducing the number of polygons constructed. This criterion is employed in [15] where the following correspondence between distance to observer and recommended detailing level is proposed:

$$Level = \left\lfloor 1 + \log_2 \left(\frac{s+d}{3 \cdot d} \right) \right\rfloor \quad (1),$$

where $Level$ is a recommended dataset hierarchy level, $\lfloor x \rfloor$ is an integer less or equal to x, s is a distance to a cell center and $d = \sqrt{3}$ is a diagonal of the level 0 cell. This formulation assumes that cells from different hierarchy levels used for

isosurface reconstruction have their edge length depending on hierarchy level chosen as follows:

$$Length = 2^{level},\ level \in [0..N] \quad (2),$$

where $level$ is a hierarchy level used, N is a maximal hierarchy level employed, zero hierarchy level corresponds to an original dataset.

Another issue is that a camera frustum formed by clipping planes bounds a visible volume region that is only a fraction of the whole volume. One can restrict isosurface construction process to be performed only in a visible volume part. This condition should be considered for substantial processing decrease for the data to be visualized in applications where observer does not see a whole dataset at a time or this situation rather rare [6].

Both issues mentioned above can be considered along with isosurface mesh reuse and its incremental updates caused by changes in observer's position and dataset modification to reduce processing times during real-time visualization process and number of polygons isosurface mesh consists of. Mesh cache or reuse is one of the major differences of our method from [15] where it is reconstructed from the scratch for each frame.

Supporting incremental mesh updates for moving observer results in several difficulties in its implementation. First, level-of-detail changes for reasonable amount of cells after observer make its step in dataset. This causes a necessity to locate those cells and update isosurface patches in them according to a new detailing level. Next, a movement of a camera frustum requires processing the certain dataset parts for an isosurface extraction and merging with existing mesh while eliminating isosurface patches associated with cells that leave frustum.

More serious concern is that cached mesh is constructed for certain threshold value and should be dropped if threshold value changes causing difficulties in interactive adjustment of the latter. That is why our approach is appropriate when user spends more time navigating in a visualized dataset and adjusts isosurface threshold value rather occasionally.

4.1 Initial isosurface construction

Before construction of initial visible isosurface part it is necessary to initialize data structures required for hierarchical isosurface reconstruction, e.g. surface containment bitmasks for each dataset hierarchy level of interest. This initialization starts from cells for original dataset where each cell is examined for a containment of an isosurface patch in it. Checking succeeds if dataset samples at cells' vertices have different signs compared to threshold resulting in "one" set in a corresponding place in bitmask for original dataset level. Otherwise, "zero" is set.

Next, bitmasks for remaining hierarchy levels are initialized. This initialization requires only bitmask from the previous level to be available as "one" is set when one of child cells from previous level has "one" set in bitmask from their level, otherwise "zero" is set.

Initial observer's position and viewing direction along with depth of view controls a viewable volume part. For simplicity, this volume part is approximated with axis – aligned box that contains it. Isosurface construction starts with covering this box by cells from coarsest dataset hierarchy level of interest. Each cell that has isosurface patch in it (that is detected as a result of inspection of the corresponding bitmask) is checked if its level is equal to one given by (1). If it is a case, a cell gets triangulated and triangles are added to resulting isosurface mesh otherwise its processed recursively until level recommended by (1) is met. During triangulation a stitching procedure from Section 3 is applied if necessary. Information about non-empty (those having "one" bit set in a corresponding bitmask) cells that meet criterion (1) is added into hash table for a future usage. This hash table allows an easy lookup for cell information by its position and level. If a surface patch was constructed for a cell then cell information in a hash table is updated with a reference to the triangles constituting a patch.

This initialization activity results in isosurface mesh for a viewable part of a dataset with detailing level depending on a distance to observer. Another result is the prepared internal structures (such as bitmasks and hash table) that will be used for a handling of future mesh updates with observer movement and dataset modifications.

4.2 Processing incoming volume parts

As observer moves, both position and size of the closest axis-aligned bounding box containing camera frustum change. Let us denote this bounding box at previous camera position as Ω_{prev} and Ω_{curr} at current position. Volume that comes into camera frustum box at current frame is $\Omega_{curr} \setminus \Omega_{prev}$ and it can be broken into several axis-aligned boxes $B_1..B_n$. For each such box we apply procedure similar to one described in the previous subsection to construct new isosurface part. The only difference comes from the following consideration. When previous isosurface part was constructed the cells intersecting its box boundary were possibly added. Isosurface construction for the new volume regions should account for those cells and omit from processing those ones that intersect a common boundary of

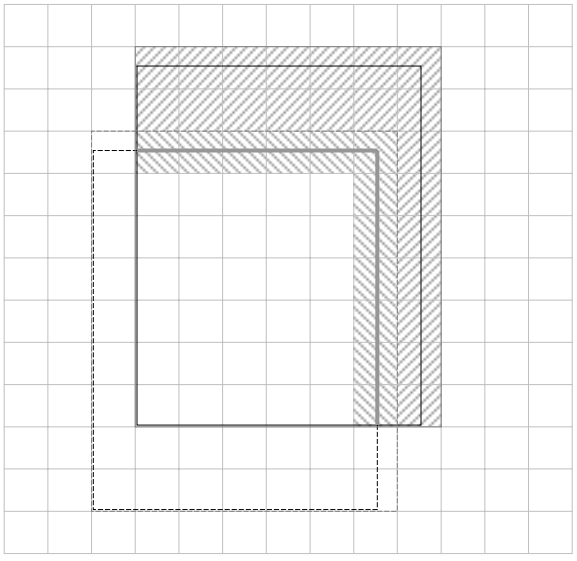

Fig. 2. bounds Ω_{prev}, bounds cells that cover Ω_{prev}, bounds Ω_{curr}, bounds cells that cover Ω_{curr}, is a common part of Ω_{curr} and Ω_{prev} boundary, marks actually processed cells, marks cells omitted from processing.

Ω_{prev} and Ω_{curr}. The result of applying this procedure is illustrated in Fig. 2.

4.3 Updating isosurface within camera frustum

Besides adding isosurface parts for volume regions incoming into camera frustum box the camera movement forces modifications in the isosurface part that is already constructed.

First, for cells leaving the bounding box for camera frustum corresponding triangular patches should be removed from isosurface mesh. Second, for cells that remain in the bounding box detailing level recommended by (1) may change so they (and isosurface patches within them) should be replaced with ones satisfying (1). Third, changed detailing level in cells mentioned above requires updates in surface stitching with their neighbors.

Solution for all those tasks is based primarily on content of the hash table described in Section 4.1. Effectively this hash table contains information concerning all non-empty cells that met criterion (1) and are within the bounding box for a camera frustum for previous observer position. Traversing a cell list for this hash table one may perform all update activities mentioned above.

All cells in the list that are not contained by the current bounding box are erased from the hash table along with their corresponding triangular patches (if any) removed from isosurface mesh.

Cells that do not satisfy (1) (patches implied) are replaced for their children or parent cells depending on increase or decrease of recommended detailing level by (1).

Boundary conditions determined by a neighborhood with cells from another hierarchy level are cached in cell information in the hash table for easier detection of their changes. If such changes have place then cell's triangles are dropped and new ones are created stitched with neighbor cells' patches.

Accomplishing update tasks means that both isosurface mesh and hash table is consistent with observer's position and direction of sight, hash table is ready to handle future updates in observer parameters.

5. VISUALIZATION OF DYNAMICALLY UPDATED VOLUME DATA

Real-time visualization of dynamically changing scalar data may be employed to emphasize the dynamics of some monitoring or scientific simulation results. Another application area concerns usage of geometric environment constructed as isosurface for some time-dependent scalar function in a virtual reality system.

Modifications of the source dataset have several follow-ups to be considered. Besides dataset modification itself they may require updates in simplified datasets for other levels of dataset hierarchy. It is not the case if those datasets are just a result of subsampling of the source dataset, but when they are obtained from original dataset by a low-pass filtering (like in [15]) then this filtering procedure must be applied once again. That filtering requires reasonable computational resources and results in noticeable time spent on it. That is why in our implementation filtering is not supported (yet).

Another issue is related to necessity for updates in internal structures employed during isosurface construction and visualization. In our case those structures are bitmasks for each dataset hierarchy level and cells hash table if changes occur in a visible dataset part. Finally, updates in the dataset possibly cause modifications in an isosurface mesh. Once again it is a case only if visible portion of the dataset is modified.

All those issues are considered by the following update procedure: completed dataset modifications make possible updating bitmasks for each dataset hierarchy level. During bitmask update the checking if a sample update in a dataset node requires corresponding update in a constructed isosurface mesh is eased by examining updates in the masks for each cell that contain this node. If mesh update is necessary and node is within visible dataset part then the hash table is used to lookup all non-empty cells that contain this node and related triangular patches. All patches (and, correspondingly, the isosurface mesh) are updated according to changes in isosurface caused by sample modification in node. As this update procedure completes for each update in a dataset then both internal structures and the isosurface mesh reflects the actual dataset contents and may be employed for handling of the future updates in a dataset.

Some inefficiency in this implementation comes from the consideration that update of a one dataset sample may require as many as eight cell updates. If neighbor samples are modified then cells that share them will be processed twice during update. To reduce this overhead it is proposed to cache dataset updates based on the spatial sample proximity and perform cell update for a bunch of dataset sample updates.

6. RESULTS

An implementation was employed as part of a hybrid volume modeling system [12,1] for fine-tuning of a model prepared on previous modeling stages. In our implementation user can interactively navigate and carve model when it seems fit. Fig. 3 illustrates high-resolution synthetic model with several detailing presets – with no level-of-detail enabled and two intermediate settings. Model is reconstructed and visualized from 256x256x256 dataset. Frame rates achieved were ranging from 3-5 fps with LOD disabled and viewable volume dimensions of 100x100x100 (64700 triangles per frame, Fig. 3a) to 40-45 fps with three levels of LOD enabled and same viewable volume (5900 triangles per frame, Fig. 3c) on Intel Celeron 300A machine with nVidia TNT graphics accelerator. This model was carved using a sphere carver with its radius and position controlled during carving. Modifications cover from 1% to 10% of the model's viewable volume and do not cause substantial drop in frame rate making interactive carving possible. Carving results are illustrated on Fig. 3e.

7. CONCLUSION AND FUTURE WORK

This work employs multiresolution dataset hierarchy to make interactive level-of-detail isosurface visualization of scalar dataset possible. Usage of frame-to-frame coherence along with clipping the processing to a camera viewing frustum makes a difference of our approach from analogs. Reuse of existent isosurface geometry minimizes dataset processing during interactive navigation in it. Adaptive level-of-detail isosurface reconstruction based on multiresolution dataset hierarchy keeps polygon number in an isosurface mesh moderate for interactive visualization with sufficient detailing in surface areas close to observer. The

incremental mesh detailing updates performed as observer moves allow keeping mesh up-to-date at a little processing cost.

Processing times for modifications in source dataset allow interactive visualization of the changes affecting up to 10% of the visible dataset volume. This renders interactive carving on the visualized isosurface possible thus making for our approach usage in an interactive volume modeling system.

Issues that remain to be addressed are the high cost of isosurface threshold changes that is useful during interactive exploration of scalar datasets and employment of filtered datasets for coarser levels of dataset hierarchy that improves visual appearance of an isosurface extracted. Both are subjects of our future work on improvement of the approach.

8. ACKNOWLEDGEMENTS

Author would like to thank Prof. Alexander Pasko and Benjamin Schmitt for supplied datasets and helpful comments on this paper. Author would also like to thank for a support the chair of computational mathematics of MIPT and Samsung Software Membership.

9. REFERENCES

[1] V. Adzhiev, M. Kazakov, A. Pasko and V. Savchenko “Hybrid system architecture for volume modeling”, Computers & Graphics vol. 24, issue 1, Feb. 2000, pp.67-78

[2] J. Bloomenthal. "Polygonalization of Implicit Surfaces", Computer Aided Geometric Design, vol. 5, 1988, pp. 341-355

[3] J. Cohen, A. Varshney, D.Manocha, G. Turk, H. Weber, P. Agarwal, F. P. Brooks Jr. and W. V. Wright. "Simplification Envelopes", Computer Graphics, Proc. Ann. Conf. Series (Proc. Siggraph'96), 1996, pp. 119-128

[4] M. Ech, T. DeRose, T. Durchamp, H. Hoppe, M. Lounsberry, and W. Stuetzle. "Multiresolutional Analysis of Arbitrary Meshes", Computer Graphics Proc. Ann. Conf. Series (Proc. Siggraph'95), 1995, pp. 173-182

[5] T. He, L. Hong, A. Varshney, S. Wang. “Controlled Topology Simplification”, IEEE Trans. on Visualization & Computer Graphics, vol. 2, no. 2, 1996, pp. 171-184

[6] L. Hong, S. Muraki, A. Kaufman, D. Bartz, T. He, “Virtual Voyage: interactive voyage in human colon”, SIGGRAPH’97, Computer Graphics Proceedings, 1997, pp.27-34

[7] H. Hoppe. “Progressive Meshes”, Computer Graphics, Vol. 30.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. "Mesh Optimization", Computer Graphics Proc. Ann. Conf. Series (Proc. Siggraph'93), 1993, pp. 19-26

[9] W. Lorensen, H. Cline. “Marching Cubes: A High Resolution 3D Surface Construction Algorithm”, Computer Graphics, vol. 21, no. 4, 1987, pp. 163-169

[10] G. Nielson, B. Hamann. “The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes”, Proc. Visualization’ 91, pp. 83-91, Oct. 91

[11] J. Rossignac, P. Borrel. “Multi-Resolution 3D Approximations for Rendering Complex Scenes“, Modeling in Computer Graphics, Springer-Verlag, June-July 1993, pp. 455-465

[12] B. Schmitt, M. Kazakov, A. Pasko, V. Savchenko "Volume sculpting with 4D spline volumes", The 2000 International Conference on Imaging Science, Systems, and Technology, (CISST'2000: Las Vegas, June 26-29, 2000) (to appear)

[13] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. "Decimation of Triangle Meshes", Computer Graphics, vol. 26, no. 2, 1992, pp. 65-70 (Proc. Siggraph'92)

[14] G. Turk. "Re-Tiling Polygonal Surfaces", Computer Graphics, vol. 26, no. 2, pp. 55-64, 1992 (Proc. Siggraph'92)

[15] R. Westermann, L. Kobbelt, T. Ertl. “Real-time Exploration of Regular Volume Data by Adaptive Reconstruction of Iso-Surfaces”, The Visual Computer, (1999) 15, pp.100-111

[16] J. C. Xia, J. El-Sana, A. Varshney. “Adaptive Real-Time Level-of-Detail-Based Rendering for Polygonal Models”, IEEE Trans. on Visualization & Computer Graphics, vol. 3, no. 2, 1997, pp. 171-183

Author:

Maxim V. Kazakov, a postgraduate of Moscow Institute of Physics and Technology.
Postage Address: MIPT-7, Dolgoprudny, Moscow region, 141700, Russia
E-mail: max@crec.mipt.ru

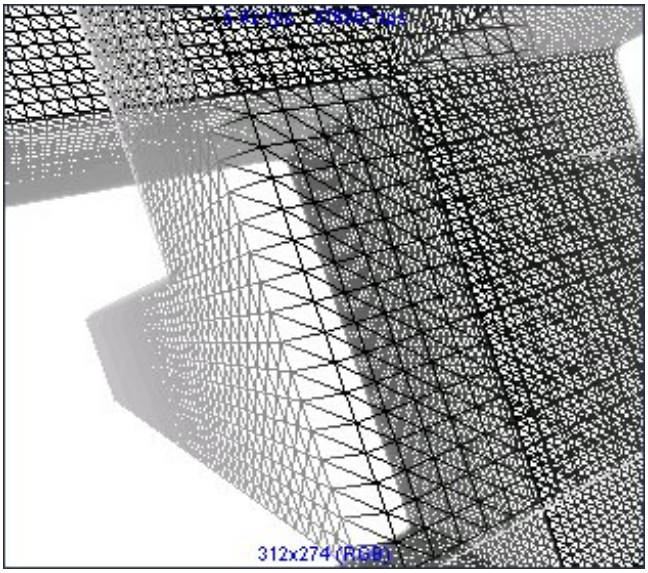

Fig. 3a. Viewable dataset of dimensions 113x92x103 with 64706 triangles in reconstructed isosurface with LOD disabled. Frame rate is about 6 fps.

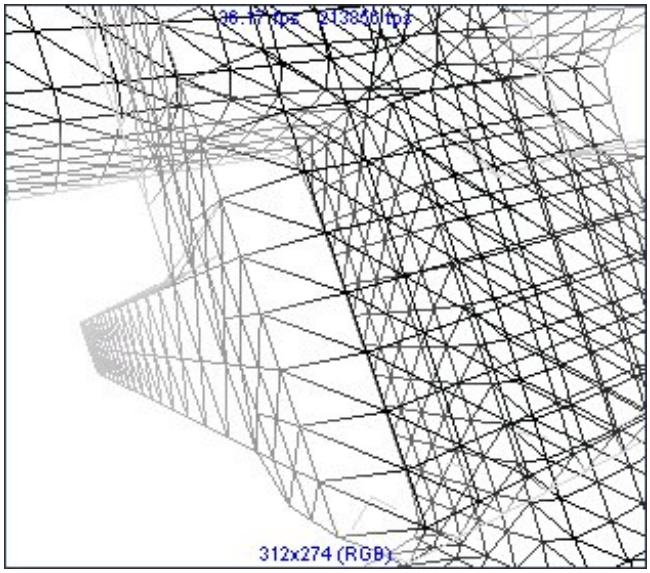

Fig. 3c. Viewable dataset of dimensions 113x92x103 with 5911 triangles in reconstructed isosurface with 3 different levels of detail in it. Frame rate is about 45 fps.

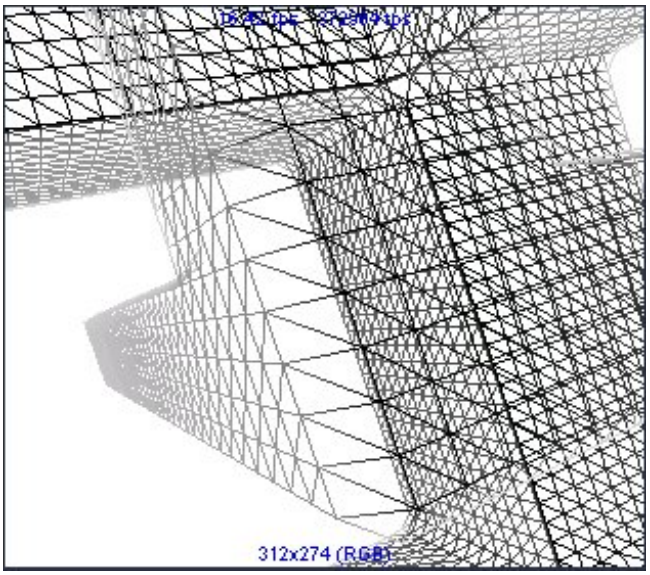

Fig. 3b. Viewable dataset of dimensions 113x92x103 with 21067 triangles in reconstructed isosurface with 2 different levels of detail in it. Frame rate is about 16 fps.

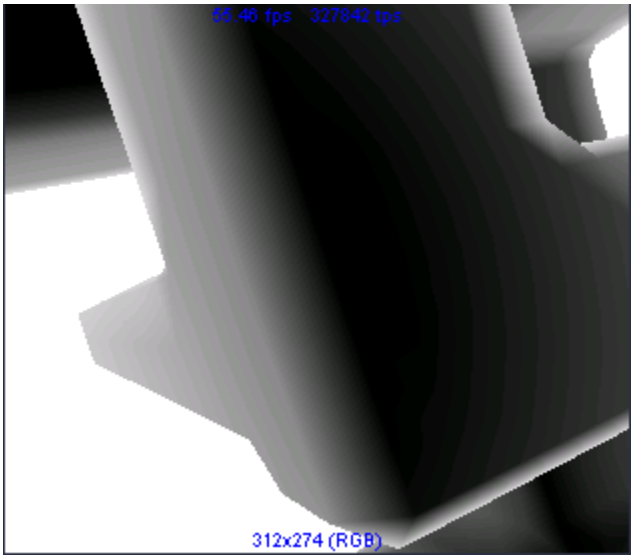

Fig. 3d. Solid shading view for the isosurface with 3 different levels of detail in it.

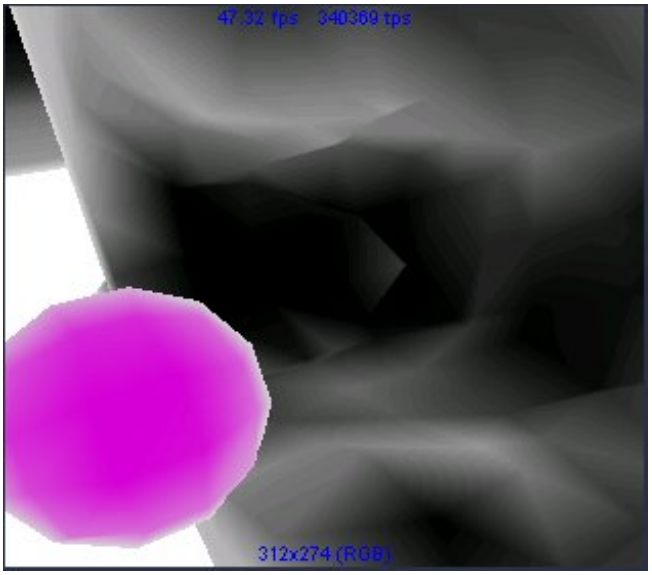

Fig. 3e. Interactive carving in the isosurface with 3 different levels of detail in it.

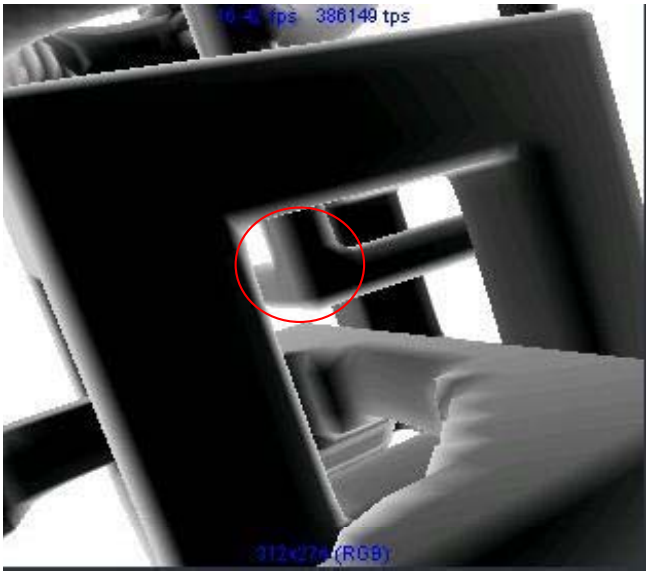

Fig. 3f. Overall view of the model visualized. Red line bounds fragment shown on previous figures.