Dynamic View Dependent Isosurface Extraction

Yarden Livnat, Charles Hansen

UUSCI-2003-004

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

Enter date here: 2002

Abstract:

We present a dynamic view-dependent isosurface extraction method. Our approach is based on the 3-stage approach we first suggested in WISE [15], i.e., front-to-back traversal, software pruning based on visibility and final rendering using the graphics hardware. In this work we re-examine the WISE method and similar view-dependent extraction methods, and propose modifications that accelerate the extraction time by a factor of 5 to 10. In particular we suggest using a bottom-up approach for rendering occluding triangles, replacing the traditional marching cubes triangulation with triangle fans and using point-based rendering for sub-pixel sections of the isosurface.

Dynamic View Dependent Isosurface Extraction

Yarden Livnat Charles Hansen

Scientific Computing and Imaging Institute
School of Computing
University of Utah

Abstract

We present a dynamic view-dependent isosurface extraction method. Our approach is based on the 3-stage approach we first suggested in WISE [15], *i.e.*, front-to-back traversal, software pruning based on visibility and final rendering using the graphics hardware. In this work we re-examine the WISE method and similar view-dependent extraction methods, and propose modifications that accelerate the extraction time by a factor of 5 to 10. In particular we suggest using a bottom-up approach for rendering occluding triangles, replacing the traditional marching cubes triangulation with triangle fans and using point-based rendering for sub-pixel sections of the isosurface.

1 Introduction

The availability of inexpensive yet powerful desktop computers and parallel supercomputers in few location leads to the development of remote visualization techniques. The fundamental drive for this paradigm enables the scientist to perform very large simulations on a remote supercomputer while visualizing and investigating the results on the local desktop. Remote isosurface visualization of large datasets poses a special challenge due to the limited bandwidth of the intermediate network.

In respond to this challenge, current research efforts aim to simplify the geometry of the isosurface *after* extracting the isosurface and *before* it renders or transmits over a network [2, 3]. In effect, the aim of such methods is to reduce the complexity of rendering an isosurface to a sub-linear complexity with respect to the size of the original isosurface. However, such methods do not address the initial challenge of extracting and constructing the isosurface.

In this paper, we present a dynamic view-dependent isosurface extraction method that is based on 3-stage approach we suggested in WISE [15],*i.e.*, front-to-back traversal, software pruning based on visibility and final rendering using the graphics hardware. We begin with a review of earlier work on isosurface extraction in Section 2. In Section 3 we identify the bottlenecks in the WISE approach and propose several modification and additions. We present experimental results and conclude with future work in Section 4.

2 Previous work

Early work [17, 9, 23, 7, 24, 11, 22, 12] on isosurface extraction focused mainly on a complete isosurface extraction and thus exhibit a worst case time complexity of $O(n)$, where n is the size of the data. In 1996, Cignoni *et al.* [6] presented an optimal isosurface extraction method based on Livnat *et al.* [16] span-space. However, as the size of the datasets grow new challenges arise. First, the size of the datasets often is larger than the available memory. Second, the size of the extracted isosurfaces can overwhelm even the most advance graphics systems. The first challenge, the size of the data, is addressed by out of core isosurface extraction methods [4, 5, 1] which aim at maintaining the data on disk and loading only relevant sections of it at a time. Another research effort, view dependent isosurface extraction, targets the size of the extracted isosurface by extracting only the visible portion of the isosurface.

View-dependent isosurface extraction was first introduced by Livnat and Hansen [15]. Their approach is based on a front-to-back traversal of the data while maintaining a virtual framebuffer of all the extracted triangles. The virtual framebuffer is used during the traversal to cull sections of the dataset, which are hidden from the given view point by closer parts of the isosurface. Livnat and Hansen also proposed a particular implementation for the virtual framebuffer and for performing visibility queries against this framebuffer. Their method, termed WISE, is based on shear-wrap factorization [13] and Greene's coverage maps [10]. Jinzhu and Shen [8] presented a distributed parallel view-dependent approach that uses multi-pass occlusion culling with an emphasis on load balancing.

A ray casting approach was presented by Parker *et al.* [20, 19]. This approach lends itself very well to large shared memory machines due to the parallel nature of ray casting. An additional benefit of the ray casting approach is the ability to generate global illumination effects such as shadows. Liu *et al.* [14] also used ray casting but instead of calculating the intersection of each ray with the isosurface, their rays are used to identify active cells. These active cells are then used as seeds to the more traditional isosurface propagation method.

Zhang *et al.* [25] presented a parallel out of core view dependent extraction method. In this approach, the sections of the data are distributed to the various processors. For a given isovalue, each processor uses ray casting to generate an occlusion map. The maps are then merged and redistributed to all the processors. Using this global occlusion map, each processor extracts its visible portion of the isosurface. They noted that updating the occlusion map during the traversal is expensive even when using hierarchical occlusion map, and thus opt not to update the occlusion maps and rely on the first occlusion map approximation.

3 SAGE

The WISE algorithm provides a particular implementation of the view dependent approach. The performance of the WISE algorithm demonstrated the potential benefits of such an approach. The two most prominent weaknesses of current view-dependent methods, which are based on the WISE approach, are the ratio of triangle intersections per screen cell and the fill rate of the screen tiles hierarchy.

In the following we present a new approach to view dependent isosurface extraction that aims at addressing these weaknesses. This approach is based on the WISE method and lessons learned from it and thus is termed SAGE. Our new approach present four new features:

1. Bottom-up updates to the framebuffer hierarchy while maintaining a top-down queries.

2. Scan conversion of multiple triangles at once, *i.e.*, the use of concave polygons.

3. Replacing sub-pixel [meta-] cells with points and normals.

4. Fast estimation of the screen bounding box of a given [meta-] cell.

3.1 A Bottom Up Approach

Current view-dependent isosurface extraction methods [15, 8] that use a virtual framebuffer, employ hierarchical structure in order to accelerate visibility queries. However, if the triangles are small then each update (rendering of a triangle) of the hierarchy requires a deep traversal of the hierarchy. Such traversals are expensive on one hand and generally add only a small incremental change.

To alleviate the problem of projecting many small triangles down the hierarchical tile structure, we employ a bottom-up approach, as shown in Figure 1. Using this approach the contribution of a small triangle is limited to only a small neighborhood in the hierarchy, *i.e.*, few tiles at the lowest level.

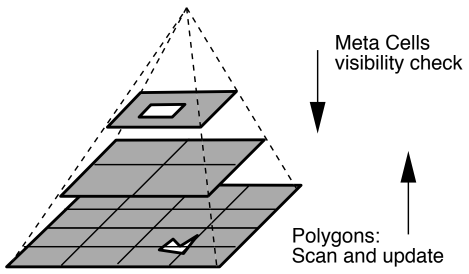

Figure 1: Bottom-up and top-down usage in SAGE.

The bottom-up approach is realized by projecting the triangles directly on to the bottom level, which is at the screen resolution. Only the tiles that are actually changed by the projection of the triangle will be further checked to see if they cause changes up the hierarchy. Since the contribution of the triangle is assumed to be small, its effect up the hierarchy will also be minimal.

3.2 Scan Conversion of Concave Polygons

One of the disadvantages of a top-down approach based on the hierarchical tiles is that this algorithm is restricted to convex polygons. In the WISE algorithm, this restriction forced the projection of triangles by only one triangle at a time.

To alleviate this restriction, the SAGE algorithm employs a scan conversion algorithm, which simultaneously projects a collection of triangles and concave polygons. The use of the scan conversion algorithm is made particularly simple in SAGE due to the bottom-up update approach. The projected triangles and polygons are scan-converted at screen resolution at the bottom level of the tile hierarchy before the changes are propagated up the hierarchy. Applying the scan conversion in a top-down fashion would have made the algorithm unnecessarily complex.

Additional acceleration can be achieved by eliminating redundant edges, projecting each vertex only once per cell and using triangle strips or fans. To achieve these goals, the marching cubes lookup table is first converted into a triangles fans format. The usual marching cubes lookup table contains a list of the triangles (three vertices) per case.

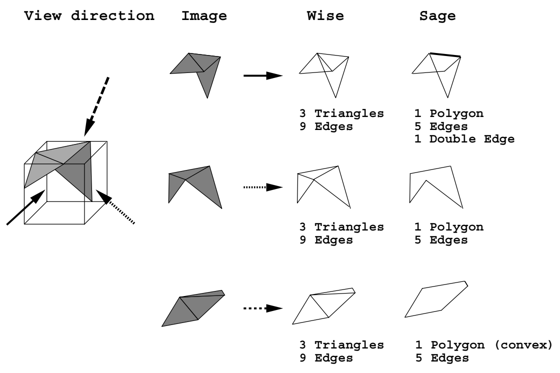

Figure 2: Comparison between WISE and SAGE.

A comparison of the WISE and the SAGE algorithms with respect to the number of polygons and edges that are projected onto the hierarchical tiles is shown in Figure 2.

3.3 Rendering Points

Another potential savings is achieved by using points with normals to represent triangles or [meta-] cells that are smaller than a single pixel. The use of points in isosurface visualization was first proposed as the *Dividing Cubes Method* by Lorensen and Cline. Phister *et al.* [21] used points to represent surface elements (surfels) for efficient rendering of complex geometry. The surfels method employs a pre-process sampling stage to create an octree-based surfel representation of a given geometry. During rendering the surfels are projected onto the z-buffer and shaded based on their attributes such as normal and texture. Special attention is given to the removal of holes and correct visibility.

The use of point rendering is an improvement over the WISE algorithm as the exact location of each screen pixel center is known during the scan line and the visibility tests. In contrast, WISE operated on a warped image of the final image and thus the exact location of each pixel was not easily obtained. Whenever a non-empty [meta-] cell is determined to have a size less then two pixels and its projection covers the center of a pixel, it is represented by a single point. Note that the size of the bounding box can be almost two pixels wide (high) and still cover only a single pixel. Referring to Figure 4, we require,

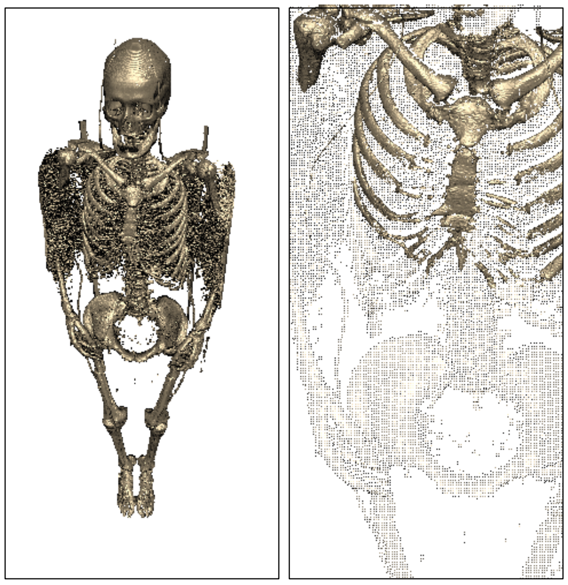

Figure 3: Rendering points. The left image was extracted based on the current view point. The right image shows a closeup of the same extracted geometry.

```
if ( right - left < 2 ) {
    int l = trunc(left);
    if ( left - l >= 0.5 ) l++;
    if (l+0.5 < right && right < l+1.5)
        // create a point
    else
        // ignore [meta-] cell
}
```

and similarity for the bounding box height.

Figure 3 shows an example in which some of the projected cells are small enough such that they can be rendered as points. On the left is the image as seen by the user while on the right is a closeup view of the same extracted geometry (*i.e.*, the user zoomed in but did not re-extract the geometry based on the new view point). Notice that much of the image on the left is represented as points. Points are not only useful in accelerating the rendering of a large isosurface but also assists in remote visualization since less geometry needs to be transferred over the network.

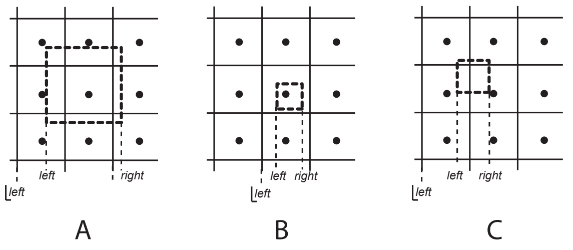

Figure 4: Pixel center and the projected bounding box. A point is created for cases A and B but not for C.

3.4 Fast Estimates of a Bounding Box of a Projected Cell

The use of the visibility tests adds an overhead to the extraction process that should be minimized. Approximating the screen area covered by a meta-cell rather than computing it exactly can accelerate the meta-cell visibility tests. In general the projection of a meta-cell on the screen has a hexagon shape with non-axis aligned edges. We reduce the complexity of the visibility test by using the axis aligned bounding box of the cell projection on the screen as seen in Figure 5. This bounding box is an overestimate of the actual coverage and thus will not misclassify a visible meta-cell, though the opposite is possible.

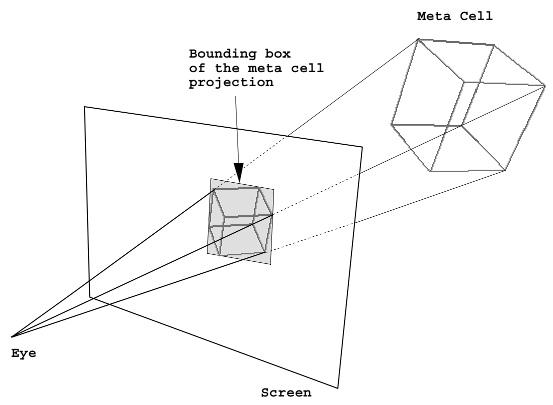

Figure 5: Perspective projection of a meta-cell, the covered area and its bounding box.

The problem is how to find this bounding box quickly. The simplest approach is to project each of the eight vertices of each cell on to the screen and compare them. This process involves eight perspective projections and either two sorts (x and y) or 16 to 32 comparisons.

The solution in SAGE is to approximate the bounding box as follows. Let P be the center of the current meta-cell in object space. Assuming the size of the meta-cell is (dx, dy, dz), we define the eight vectors

$$D = (\pm\frac{dx}{2}, \pm\frac{dy}{2}, \pm\frac{dz}{2}, 0)$$

The eight corner vertices of the cell can be represented as

$$V = P + D = P + (\pm D_x, \pm D_y, \pm D_z)]$$

Applying the viewing matrix M to a vertex V amounts to:

$$VM = (P + D)M = PM + DM$$

After the perspective projection the x screen coordinate of the vertex is:

$$\frac{[VM]_x}{[VM]_w} = \frac{[PM]_x + [DM]_x}{[PM]_w + [DM]_w}$$

To find the bounding box of the projected meta-cell we need to find the minimum and maximum of these projections over the eight vertices in both x and y. Alternatively, we can overestimate these extrema values such that we may classify a non-visible cell as visible but not the opposite. Overestimating can thus lead to more work but will not introduce errors.

The maximum x screen coordinate can be estimated as follows,

$$\begin{aligned}\max(\frac{[VM]_x}{[VM]_w}) &\le \frac{\max([PM]_x + [DM]_x)}{\min([PM]_w + [DM]_w)} \\ &\le \frac{[PM]_x + \max([DM]_x)}{\min([PM]_w + [DM]_w)} \\ &\le \frac{[PM]_x + [D^+M^+]_x}{\min([PM]_w + [DM]_w)}\end{aligned}$$

where we define the $^+$ operator to mean to use the absolute value of the vector or matrix elements.

Assuming that the meta-cells are always in front of the screen we have

$$V_z > 0 \Rightarrow P_z - D_z^+ > 0 \Rightarrow [PM]_z - [D^+M^+]_z > 0$$

thus,

$$\max \frac{[VM]_x}{[VM]_w} = \begin{cases} \frac{[PM]_x + [D^+M^+]_x}{[PM]_w - [D^+M^+]_w} & \text{if numerator} \ge 0 \\ \frac{[PM]_x + [D^+M^+]_x}{[PM]_w + [D^+M^+]_w} & \text{otherwise} \end{cases}$$

Similarly, the minimum x screen coordinate can be overestimated as,

$$\min \frac{[VM]_x}{[VM]_w} \le \begin{cases} \frac{[PM]_x - [D^+M^+]_x}{[PM]_w + [D^+M^+]_w} & \text{if numerator} \ge 0 \\ \frac{[PM]_x - [D^+M^+]_x}{[PM]_w - [D^+M^+]_w} & \text{otherwise} \end{cases}$$

The top and bottom of the bounding box are computed similarly.

The complete procedure for estimating the bounding box (see Figure 6) requires only two vector matrix multiplications, two divisions, four multiplications, four comparisons, and six additions.

4 Results

To evaluate the performance of the SAGE method we compare it to the performance of the Bon Octree [24], NOISE [16] and WISE [15] methods. We also compare the performance of the three acceleration modifications we propose, namely: scan conversion of convex polygons, bounding box estimation and point-base rendering. The various experiments were done using the head and legs sections from the visible female dataset [18]. The head dataset has dimensions of 512x512x208 for a total of 104MB, and the legs dataset is 512x512x617 for a total of 308MB.

4.1 SAGE vs. Other Extraction Methods

The tests cases included (see Figure 7) a normal view of each dataset, a closeup of a small section, and a distant view. Each test was performed twice using the isovalues corresponding to the skin (600.5) and bone (1224.5). The tests were selected such that each will reflect different characteristics. For the visible woman dataset, the skin isosurface had

```
**Precompute**$M^+$:
  $M = ViewMatrix$
  $M^+_{i,j} = \|M_{i,j}\|$

**Compute**( $t,\ f_1,\ f_2$ )
  return $t > 0\ ?\ t * f_1\ :\ t * f_2$

**FindBouningBox**( **cell** )
  $P =$ center of cell
  $D = (\frac{dx}{2}, \frac{dy}{2}, \frac{dz}{2}, 1)$
  $PM = p * M$
  $DM = d * M^+$

  $f_{far} = \frac{1}{PM_w + DM_w}$

  $f_{near} = \frac{1}{PM_w - DM_w}$

  $right\ \ = \text{Compute}(PM_x + DM_x,\ f_{near},\ f_{far})$
  $left\ \ \ = \text{Compute}(PM_x - DM_x,\ f_{far},\ f_{near})$
  $top\ \ \ \ = \text{Compute}(PM_y + DM_y,\ f_{near},\ f_{far})$
  $bottom = \text{Compute}(PM_y - DM_y,\ f_{far},\ f_{near})$
```

Figure 6: Procedure for (over)estimating the bounding box of a projected cell.

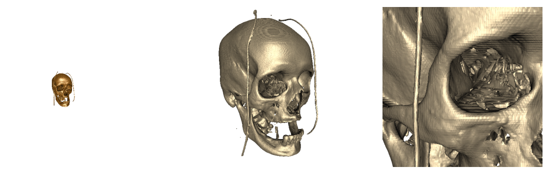
(a) Distant (b) Normal (c) Closeup

Figure 7: Example of the three views.

large contiguous areas of coverage whereas the isosurfaces for the bone exhibit complex structures with many holes and cavities. The closeup test demonstrates one of the benefits of view dependent isosurface extraction when only a small section of the isosurface in needed. In contrast, the distant tests shows examples where not even the *visible* isosurface should be extracted, rather only the visible portion with regard to the resolution of the screen. The size of the objects on the screen, in the case of the distance view, also corresponds to their size when the full dataset (see Figure 3) is used.

The experiments were done on an SGI *Onyx*2 (using a single CPU). Table 1 and Table 2 illustrate the results for these tests.

In all the tests the view dependent approaches, WISE and SAGE, consistently reduced the number of extracted polygons. The SAGE algorithm generated far fewer triangles then the WISE method (except for one case) by up to factors 70 (1.5%) for the distance cases, factor of 1.5 for closeup, and up to a factor of 7 on the normal view cases. With respect to

Table 1: Visible Woman Head Dataset

view	method	extraction time (sec)	number of polygons	view (sec)
Skin				
any	Octree	10.9	1,430,824	2.6
	NOISE	10.2		
distant	WISE	35.1	292,242	0.6
	SAGE	3.4	18,645	<0.1
normal	WISE	35.8	344,628	0.6
	SAGE	4.4	195,408	0.3
closeup	WISE	4.6	43,222	0.1
	SAGE	0.6	36,939	<0.1
Bone				
any	Octree	17.0	2,207,592	4.6
	NOISE	14.6		
distant	WISE	13.9	271,075	0.5
	SAGE	4.1	12,747	<0.1
normal	WISE	32.7	278,735	0.7
	SAGE	4.5	153,617	0.2
closeup	WISE	10.6	84,599	0.2
	SAGE	1.5	67,808	0.2

Table 2: Visible Woman Legs Dataset

view	method	extraction time (sec)	number of polygons	view (sec)
Skin				
any	Octree	33.4	3,264,755	6.2
	NOISE	27.1		
distant	WISE	37.5	968,073	2.0
	SAGE	0.7	14,917	<0.1
normal	WISE	70.4	935,784	1.8
	SAGE	12.1	122,229	0.2
closeup	WISE	16.6	364,394	0.80
	SAGE	5.7	234,607	0.51
Bone				
any	Octree	18.9	2,328,940	4.7
	NOISE	16.3		
distant	WISE	18.0	412,530	0.9
	SAGE	0.5	6,099	<0.1
normal	WISE	33.3	406,262	0.9
	SAGE	6.8	503,545	0.1
closeup	WISE	9.6	180,270	0.5
	SAGE	3.8	122,262	0.3

the extraction time, the SAGE method outperformed WISE consistently by factors of 3 to 7, and in one case up to a factor of 50.

These results depend on the depth complexity of the full isosurface as well as the visible portion of the isosurface and its footprint on the screen.

4.2 SAGE Acceleration Modification

The key feature of the SAGE algorithm is the use of the bottom-up rendering approach. In addition, SAGE utilizes three acceleration modifications different than its predecessor algorithm WISE: scan-conversion of concave polygons, bounding box approximation and point-based rendering. In Table 3 and Table 4 we show the advantage these three modifications exhibit. The experiments were done using the same datasets, on a 1.7 MHz Pentium 4 Linux machine with 768MB and GeForce4 graphics card.

The results show that using point-based rendering, when appropriate, consistently accelerate the rendering time by a factor of 10. This result is especially appealing for isosurface extraction from very large datasets where many meta cells project onto the same pixel on the screen. Using scan-conversion of concave polygons shows a consistent reduction of about 30% in the extraction time. The bounding box algorithm had the least impact and in general reduces the extraction time by 7-9%. It is interesting to note that for the distance test cases the bounding box estimation performed worst then the full projection by about 1% if point-based rendering was not used. However, when combined with point-based rendering, the bounding box estimation consistently reduced the rendering time by more then 30%.

Another interesting point is the fact that SAGE performed the best when only a small section of the isosurface is visible (closeup) or if the isosurface is far and point-based rendering can be used. This is especially intriguing when one considers that in the first case the isosurface covers most if not all of the image (*i.e.*, the scan conversion has to touch each and every pixel) while in the later case the isosurface covers only a very small faction of the final image.

SAGE performs the worst when the isosurface is large enough such that the proposed point-base rendering is not used but the triangles it generates are very small. This issue should be further investigated in future research.

5 Acknowledgments

This work was supported in part by grants from the DOE ASCI, the DOE AVTC, the NIH NCRR, and by the National Science Foundation under Grants: 9977218 and 9978099. The authors would like to thank Blythe Nobleman and Chris Johnson for their contribution.

References

[1] C. L. Bajaj, V. Pascucci, D. Thompson, and X. Y. Zhang. Parallel accelerated isocontouring for out-of-core visualization. In *Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics*, pages 97–104. ACM Press, 1999.

[2] Martin Bertram, Mark A. Duchaineau, Bernd Hamann, and Kenneth I. Joy. Bicubic subdivision-surface wavelets for large-scale isosurface representation and visualization. In *Proceedings of Visualization 2000*, pages 389–396, October 2000.

[3] Martin Bertram, Daniel E. Laney, Mark A. Duchaineau, Charles D. Hansen, Bernd Hamann, and Kenneth I. Joy. Wavelet representation of contour sets. In *Proceedings of Visualization 2001*, pages 303–310, October 2001.

[4] Yi-Jen Chiang and Cláudio T. Silva. I/O optimal isosurface extraction. In Roni Yagel and Hans Hagen, editors, *IEEE Visualization '97*, pages 293–300, 1997.

[5] Yi-Jen Chiang, Cludio T. Silva, and William J. Schroeder. Interactive out-of-core isosurface extraction. In *Proceedings of the conference on Visualization '98*, pages 167–174. IEEE Computer Society Press, 1998.

Table 3: Visible Woman Head Dataset

view	Scan.	BBox Est.	Points	Extract (msec)	Draw (msec)	Tri. (points)
Skin						
octree				1,493	576	1.430M
distant	reg	no	no	1,590	2	13,610
	new	no	no	1,084	2	13,610
	new	yes	no	1,100	2	13,610
	-	no	yes	159	0.5	(5,751)
	-	yes	yes	112	0.5	(5,751)
normal	reg	no	-	1,857	80	191,483
	new	no	-	1,294	80	191,483
	new	yes	-	1,205	80	191,483
closeup	reg	no	-	155	3	22,628
	new	no	-	102	3	22,628
	new	yes	-	99	3	22,628
Bone						
octree				2667	894	2.207M
distant	reg	no	no	1,820	1.4	9,736
	new	no	no	1,326	1.4	9,736
	new	yes	no	1,279	1.4	9,736
	-	no	yes	169	0.3	(4,289)
	-	yes	yes	115	0.3	(4,289)
normal	reg	no	-	2,062	70	167,949
	new	no	-	1,512	70	167,949
	new	yes	-	1,380	70	167,949
closeup	reg	no	-	615	25	57,669
	new	no	-	432	25	57,669
	new	yes	-	437	25	57,669

Table 4: Visible Woman Legs Dataset

view	MC.	BBox Est.	Pts	Extract (msec)	Draw (msec)	Tri. (points)
Skin						
octree				11,407	1,832	3.874M
distant	reg	no	no	4,688	8	32,618
	new	no	no	3,306	8	32,618
	new	yes	no	3,376	8	32,618
	-	no	yes	475	2	(14,900)
	-	yes	yes	300	2	(14,900)
normal	reg	no	-	4,826	63	150,818
	new	no	-	3,331	63	150,818
	new	yes	-	3,058	63	150,818
closeup	reg	no	-	976	42	102,103
	new	no	-	683	42	102,103
	new	yes	-	638	42	102,103
Bone						
octree				2,381	1,044	2.2329M
distant	reg	no	no	2,593	3	12,976
	new	no	no	1,841	3	12,976
	new	yes	no	1,705	3	12,976
	-	no	yes	274	0.3	(6333)
	-	yes	yes	180	0.3	(6333)
normal	reg	no	-	2,593	30	76,931
	new	no	-	1,818	30	76,931
	new	yes	-	1,608	30	76,931
closeup	reg	no	-	1,101	39	95,710
	new	no	-	775	39	95,710
	new	yes	-	742	39	95,710

[6] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Optimal isosurface extraction from irregular volume data. In *Proceedings of IEEE 1996 Symposium on Volume Visualization*. ACM Press, 1996.

[7] R. S. Gallagher. Span filter: An optimization scheme for volume visualization of large finite element models. In *Proceedings of Visualization '91*, pages 68–75. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[8] Jinzhu Gao and Han-Wei Shen. Parallel view-dependent isosurface extraction using multi-pass occlusion culling. In *Parallel and Large Data Visualization and Graphics*, pages 67–74. IEEE Computer Society Press, Oct 2001.

[9] M. Giles and R. Haimes. Advanced interactive visualization for CFD. *Computing Systems in Engineering*, 1(1):51–62, 1990.

[10] Ned Greene. Hierarchical polygon tiling with coverage masks. In *Computer Graphics*, Annual Conference Series, pages 65–74, August 1996.

[11] T. Itoh and K. Koyamada. Isosurface generation by using extrema graphs. In *Visualization '94*, pages 77–83. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[12] T. Itoh, Y. Yamaguchi, and K. Koyyamada. Volume thining for automatic isosurface propagation. In *Visualization '96*, pages 303–310. IEEE Computer Society Press, Los Alamitos, CA, 1996.

[13] Philippe G. Lacroute. Fast volume rendering using shear-warp factorization of the viewing transformation. Technical report, Stanford University, September 1995.

[14] Zhiyan Liu, Adam Finkelstein, and Kai Li. Progressive view-dependent isosurface propagation. In *Proceedings of Vissym'2001*, 2001.

[15] Y. Livnat and C. Hansen. View dependent isosurface extraction. In *Visualization '98*, pages 175–180. ACM Press, October 1998.

[16] Y Livnat, H. Shen, and C. R. Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Trans. Vis. Comp. Graphics*, 2(1):73–84, 1996.

[17] W.E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[18] National Library of Medicine. The visible human project, 1986. http://www.nlm.nih.gov/research/visible/visible_human.html.

[19] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 1999.

[20] S. Parker, P. Shirley, Y. Livnat, C. Hansen, and P.-P. Sloan. Interactive ray tracing for isosurface rendering. In *Visualization 98*, pages 233–238. IEEE Computer Society Press, October 1998.

[21] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: Surface elements as rendering primitives. In Kurt Akeley, editor, *Siggraph 2000,*

Computer Graphics Proceedings, pages 335–342. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[22] H. Shen and C. R. Johnson. Sweeping simplicies: A fast iso-surface extraction algorithm for unstructured grids. *Proceedings of Visualization '95*, pages 143–150, 1995.

[23] J. Wilhelms and A. Van Gelder. Octrees for faster iso-surface generation. *Computer Graphics*, 24(5):57–62, November 1990.

[24] J. Wilhelms and A. Van Gelder. Octrees for faster iso-surface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992.

[25] Xiaoyu Zhang, Chandrajit Bajaj, and Vijaya Ramachandran. Parallel and out-of-core view-dependent isocontour visualization using random data distribution. In D. Ebert, P. Brunet, and I. Navazo, editors, *Joint Eurographics — IEEE TCVG Symposium on Visualization (VisSym-02)*, pages 1–10, 2002.