The Visual Computer manuscript No.
(will be inserted by the editor)

Hanli Zhao · Charlie C. L. Wang (corresponding author) · Yong Chen · Xiaogang Jin

Parallel and Efficient Boolean on Polygonal Solids

Submitted February 6, 2011; Accepted March 11, 2011

Abstract We present a novel framework which can efficiently evaluate approximate Boolean set operations for B-rep models by highly parallel algorithms. This is achieved by taking axis-aligned surfels of Layered Depth Images (LDI) as a bridge and performing Boolean operations on the structured points. As compared with prior surfel-based approaches, this paper has much improvement. Firstly, we adopt key-data pairs to store LDI more compactly. Secondly, robust depth peeling is investigated to overcome the bottleneck of layer-complexity. Thirdly, an out-of-core tiling technique is presented to overcome the limitation of memory. Real-time feedback is provided by streaming the proposed pipeline on the many-core graphics hardware.

Keywords Boolean operations · Layered Depth Images · Depth peeling · Out-of-core · CUDA

1 Introduction

Boolean operations, such as union ($\cup$), subtraction ($-$) and intersection ($\cap$), are useful for combining simple models to create complex solid objects on a *Constructive Solid Geometry* (CSG) tree, which has a variety of applications in *computer-aided design and manufacturing* (CAD/CAM), virtual reality and computer graphics (e.g., [7, 17, 18]).

Polygonal meshes are the most popular boundary representation (B-rep) for 3D models. Although many commercial CAD systems, such as Rhinoceros [3] and ACIS [1], are able to compute Boolean on polygons, they have difficulty in modeling very complex models (e.g., the scaffold of Bone as shown in Fig. 1). Polygons provide the simplest way to approximate freeform solids and the computation of Boolean is not necessary to be exact in many areas, such as biomedical engineering, jewelry industry and game industry. This gives us an opportunity to overcome the robustness problem [14] and to speed up the computation by using an approximate method. Recent researches have shown that densely oriented point sets are competent to fulfill such a task. Several algorithms [4, 23, 32] have been investigated to perform Boolean on oriented points but they do not operate on B-rep models directly. Some methods [6, 30] can process B-rep models using well-structured points in Layered Depth Images (LDI). However, both the efficiency and the memory-usage still have much room for improvement.

In this paper we aim to develop a framework that fully exploits the advantages of LDI [27] and the power of *Graphics Processing Unit* (GPU) for evaluating approximate Boolean operations on polygonal meshes. Our algorithm is efficient, parallel, and scalable compared with prior approaches. These good properties are achieved by introducing the following work. First of all, we adopt a compact representation for LDI which contains axis-aligned surfels sampled from the boundary of input solids. By utilizing the atomic operation in graphics global memory, we develop a programmable rasterizer to capture all layers of LDI in a single pass. Moreover, the problem of limited sampling resolution is avoided by splitting the whole working envelope into multiple smaller tiles and processing them independently using the out-of-core strategy. Note that the computation in the whole pipeline is highly parallel, thus allowing for a GPU-based

H. Zhao
College of Physics & Electronic Information Engineering, Wenzhou University, Wenzhou, 325035, China
E-mail: hanlizhao@gmail.com

C. C. L. Wang
Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, China
E-mail: cwang@mae.cuhk.edu.hk

Y. Chen
Epstein Department of Industrial and Systems Engineering, University of Southern California, CA, USA
E-mail: yongchen@usc.edu

X. Jin
State Key Lab of CAD & CG, Zhejiang University, Hangzhou, 310058, China
E-mail: jin@cad.zju.edu.cn

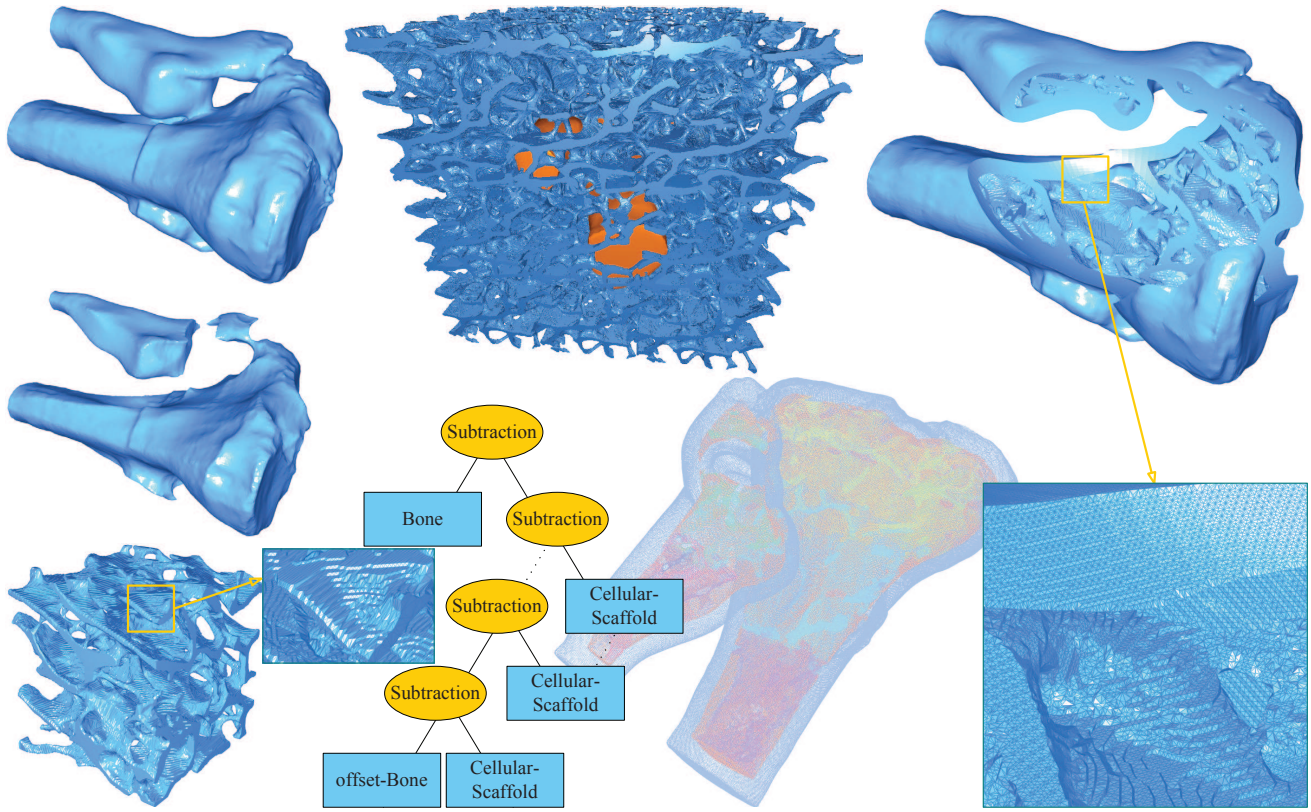

Fig. 1 A high-quality osseous scaffold is modeled with our system: (left) input models (Bone, offset of Bone, and cellular-scaffold), (middle) the layout of solids where $3 \times 3 \times 3$ of cellular-scaffold models are tiled together, resulting in a CSG tree with a depth of 28, (mid-bottom) surfel-represented Boolean result, where surfels with different colors are from different input solids, and (right) the cross-section view of the output osseous tissue. The approximation resolution is 512×512 in this example.

implementation. To the best of our knowledge, this is the first algorithm that is capable of computing Boolean combinations of freeform polygonal shapes in real-time (see the examples shown in Fig. 11). Even for a very complex CSG tree such as the one shown in Fig. 1, we are able to obtain the result of the high-quality osseous tissue in 1.7 sec. on an NVIDIA Geforce GTX 480 GPU while consuming only 600MB of the graphics memory.

In summary, this paper addresses the following important questions in geometric modeling:

- How to support real-time approximate geometric feedback for a CSG tree with polygonal models?
- How to support highly accurate approximation that may be beyond the memory capacity of graphics hardware?

The solution to these problems is the major contribution of our approach.

The rest of our paper is organized as follows. After briefly reviewing some related previous work in the next section, Section 3 introduces an overview of our new Boolean evaluation framework. Section 4 presents the parallelization techniques on the GPU in detail. Experimental results and related discussions are given in Section 5. Finally, Section 6 concludes the advantage of our approach and suggests some future work.

2 Related Work

Boolean operations of solids have been investigated in computer graphics for many years. For a comprehensive survey, we refer readers to [26]. Here, we only review the Boolean algorithms based on surfels [24] and oriented points.

Many approaches [9–12, 25] employ the image-based depth-layering or depth-peeling technique to achieve the interactive rendering of CSG models. They gradually peel candidate points from the boundaries of CSG primitives and only display the nearest boundary points of the Boolean result. The points can be fast sampled using the graphics hardware [8] and encoded as pixels in LDI. However, these approaches only solve the fast rendering problem but do not provide the ability to obtain the geometric shape of resultant solids.

Surfel is a representation that can approximate the shape of a solid model on some oriented sample points. Boolean operations on freeform solids represented by sur-

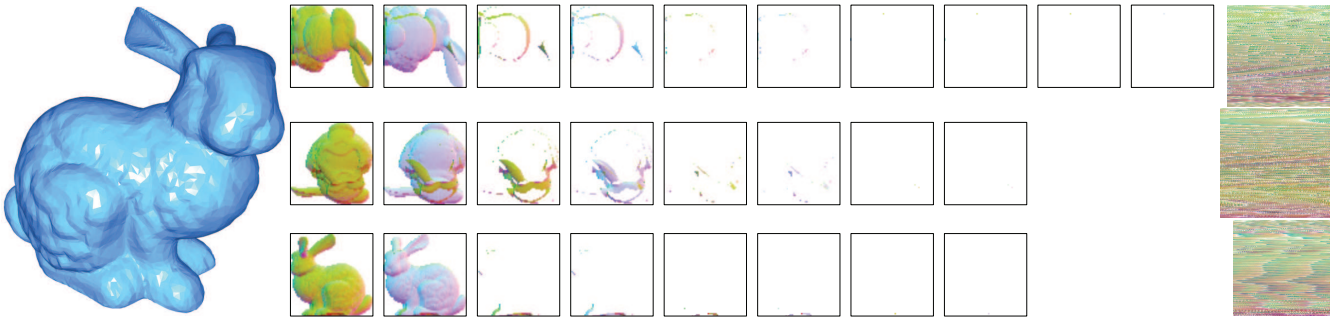

Fig. 2 A comparison between LDI and CLDI sampled from (left) Bunny with the resolution of 150 × 150 pixels: (middle) three axis-aligned LDI texture arrays with 10, 8, and 8 layers respectively, and (right) the corresponding CLDI with only 34.7k, 40.1k, and 30.9k elements respectively. We can see that only 18% pixels of LDI need to be stored in CLDI for the representation of the same model.

fels have been developed in [4,23]. Since most CAD/CAM applications, such as CAE analysis, CNC machining and rapid prototyping, need the B-rep of solid models, additional surface reconstruction from the surfel-represented solid is required.

Zhou et al. [32] introduced a parallel surface reconstruction algorithm by extending the marching cubes algorithm [20] with a GPU-based octree. They also presented an interactive Boolean tool for oriented point set. However, their adaptive marching cubes algorithm tends to smooth the sharp features in the underlying geometry. Moreover, their in-core reconstruction can only handle octrees with a limited number of depths.

Chen and Wang [6] presented an approximate Boolean approach that can preserve sharp features in the resultant solids. They made use of a variant of LDI, named as *Layered Depth-Normal Images* (LDNI), for the efficient volumetric test [29], and employed the quadratic error function (QEF) in dual contouring [15] for surface generation. However, only the sampling of LDNI is performed using GPU-based depth peeling whereas other algorithms are performed on the CPU. Although multi-threaded schemes are designed, the parallelism is limited. Wang et al. [30] further transferred the LDNI-based algorithm to the GPU to improve the efficiency. Unfortunately, as the workload on each thread is not well balanced, the primary algorithm still cannot provide real-time feedback. In addition, their algorithm cannot process models which contain more than 256 LDI layers because the stencil buffer used in their sampling method only has 8 bits in the current graphics hardware architecture. Moreover, redundancy in graphics memory consumption seriously limits its applications. All these problems are solved by our approach below.

3 Framework Overview

An expression of Boolean operations is usually described as a binary tree (i.e., CSG tree), on which leaves represent primitive shapes and inner nodes are Boolean operators. First of all, if the out-of-core computation is enabled, the bounding volume is divided into multiple volumetric tiles with the same size. We then perform Boolean on each tile along three axes separately.

Algorithm 1 Streaming Boolean Operations

```
Input Boolean tree accompanied with polygonal meshes
for each volumetric tile in bounding volume do
    for each axis in {x, y, z} do
        for each node n by depth-first tree traversal do
            if n is a leaf then
                Sample axis-aligned surfels by rasterizing n
                Shell sort per-pixel surfels
            else
                Merge sort candidate surfels from children
                Boolean on the sorted candidate surfels
            end if
        end for
        Obtain boundary surfels from the root node
        Accumulate the QEF matrices for boundary cells
        Mark interior/exterior cells along the axis
    end for
    Output vertices by solving the QEF matrices
    Output indices by connecting boundary cells
end for
return merge all tiles of polygons into a closed manifold
```

The final Boolean result is obtained by recursively traversing the CSG tree using the depth-first scheme. For each leaf node, we convert the B-rep model into a solid bounded by surfels using axis-aligned ray-casting. For each inner node, we first collect all candidate surfels from its children nodes and sort the merged samples according to their depth values. Next, valid boundary surfels are extracted by performing Boolean on the candidate surfels in an efficient way. After traversing the CSG tree, the resultant surfel-represented solid is obtained at the root node.

Efficient surface reconstruction is conducted to output B-rep models as the result of Boolean. We develop a modified version of dual contouring in this paper. The optimal position of each boundary cell is determined by the QEF matrix, which can be accumulated by all the boundary surfels in the cell.

Note that the whole pipeline design is highly parallel, which enables a real-time response by GPU-based implementation. We adopt key-data pairs to store the points more compactly as compared with prior LDI-based algorithms. As a result, in addition to memory reduction, the parallelism is also improved as no threads are launched for void operations. Algorithm 1 provides the pseudocode of our approximate Boolean framework.

4 Parallelization on the GPU

As shown in Algorithm 1, except for a few control settings, every processing step can be streamed on the many-core graphics hardware, including conversion of polygons into surfel-represented solids, Boolean on surfels, B-rep reconstruction and volumetric tiling. We choose NVIDIA's CUDA [2] computing environment in our GPU implementation. CUDA exposes new hardware features which are very useful for data-parallel computation. One important feature is that it allows arbitrary gathering and scattering of graphics memory in GPU threads. In this section, we introduce our parallel and efficient Boolean algorithm in detail.

4.1 Compact LDI Data Structure

In the prior approaches, LDI is usually stored as an array of two-dimensional textures. Its size grows linearly with the layer-complexity of a model. However, as shown in Fig. 2 (left), the actual number of sampled points in each texture decreases significantly with the increase of the depth layer. Such a primary way to store LDI consumes too much graphics memory and limits the application of using LDI to represent complex models (e.g., the scaffold models in Fig. 1).In this paper, we develop an efficient data structure to store such sparsely distributed data.

Inspired by the solver developed in [28] for large sparse linear systems which only store non-zero values with corresponding indices, we adopt a compact representation here, called CLDI, for the sparse data points in LDI. On each pixel in a model represented by LDI, there are multiple surfels with different depth values. We store their depth values as well as normals sequentially in a long data array. The access to each surfel in the data array is achieved by using a key array where each key stores the number of surfels per pixel as well as the index of its first surfel in the long data array. As illustrated in Fig. 2 (right), all empty data points (about 82% pixels in this example) are not stored, thus a significant memory reduction is achieved.

In order to improve the performance of CLDI, our algorithm uses GPU-based parallel primitives [13] heavily. The *scan* primitive is employed to count the required data size before allocating a GPU array and to obtain indices of sample points to the array. The *compact* primitive is used to discard empty sample points, while the *sort* primitive efficiently clusters elements with a sorting identifier. In addition to memory reduction, we also explain how CLDI could improve the performance below.

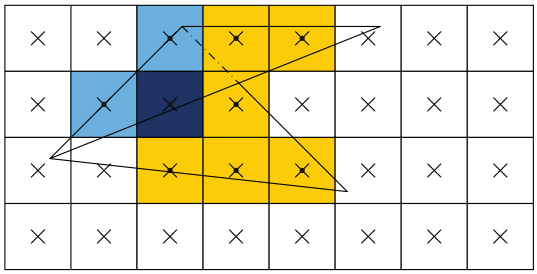

Fig. 3 The colored pixels are inside the two adjacent triangles according to rasterization rules. The yellow pixels are covered once while the light blue and dark blue ones are covered twice. The light blue pixels just lie on the silhouette edge and should be rasterized twice in order to guarantee the closure of solid represented by surfels.

4.2 Robust Sampling

Rasterizer on the graphics hardware is designed according to rasterization rules about how to map geometric data into raster data. The raster data are snapped to integer locations and per-pixel attributes are linearly interpolated from per-vertex attributes before being passed to a pixel shader. Any pixel center which falls inside a triangle is drawn. A pixel is also assumed to be inside if it lies on the top edge or the left edge of the triangle. A top edge is an edge that is exactly horizontal and above the other two edges of the triangle, and a left edge is an edge that is not exactly horizontal and on the left side of the triangle.

An edge is called silhouette if two triangles that share this edge are facing opposite directions with respect to the viewpoint. There is a special case that a silhouette is the top/left edge of its two adjacent triangles and there are some pixels which just lie on this silhouette. According to rasterization rules as well as the depth less comparison function, these pixels are rasterized only once in traditional depth-peeling techniques, as illustrated in Fig. 3. However, such a case will break the boundary of solids as described in [6, 29]. To avoid this problem, Wang et al. [30] used the stencil buffer instead of the depth buffer. They peel each layer of surfels by gradually increasing the stencil value. Unfortunately, such a sampling technique introduces a new issue. The stencil buffer only has 8 bits in the current graphics hardware architecture and 8-bit data can only represent maximum 256 values. If a complex B-rep model contains more than 256 LDI layers, missampling will occur because of the stencil buffer. Moreover, modern hardware rasterizers can only capture a rather limited number of fragment data in one render pass [5, 22], thus for complex models, multiple rendering passes are needed. As a result, sampling B-rep models

into correct surfel-represented solids becomes the main bottleneck of Boolean techniques presented in [6, 30].

In this paper, we devise a robust and fast sampling method which is inspired by the main idea of FreePipe in [19]. First of all, rasterization resolution R_s, axis-aligned bounding box (AABB) and appropriate orthogonal model-view-projection matrix are set as input parameters. Back-face culling is disabled and frustum clipping is enabled. We then rasterize the B-rep model according to the scan-line scheme and rasterization rules. For each scanned pixel, we obtain an interpolated depth. In order to avoid read-modify-write (RMW) hazards in graphics global memory, the *atomicInc* operation [2] of CUDA is employed. We allocate a counter per pixel in global memory and initialize it to 0. With the *atomicInc* operation, the k-th incoming fragment of a certain pixel increases the counter to $k+1$ and return the source value k. Finally, the depth and normal are stored into the k-th entry of CLDI without RMW hazards. Note that k is a 32-bit unsigned integer and can be used to sample a model with up to 2^{32} layers.

The proposed sampling algorithm performs two passes of the programmable rasterization. In the first pass, we only increase the counter array to obtain the per-pixel numbers of surfels. Then the *scan* primitive is utilized to count the size of CLDI and to obtain per-pixel indices to CLDI. In the second pass, all surfel data are truly filled in CLDI. As can be seen in Fig. 2, all layers of surfels are compactly captured in the second pass. Moreover, two surfels are correctly sampled on silhouette edges, which ensures the correctness of a solid represented by the sampled surfels.

4.3 Boolean on Surfels

To evaluate the result of each inner node on the CSG tree, Boolean operations are performed on the models represented by the children nodes. The output surfels of children nodes are collected as candidate surfels. Before applying Boolean operations, surfels on a ray must be sorted by their depth values. CUDA does not support recursive functions, so we choose the *Shell* sort to obtain surfels sorted on each pixel from the leaf nodes. For inner nodes, since the surfels in each of their children nodes have been sorted during prior depth-first traversal, one-pass merge sort, which has linear time complexity, can be used to generate the candidate list of sorted surfels.

Let $\{p_i | i = 1, 2, ..., m\}$ be the sorted surfels on a pixel, d_i be the depth of p_i, and f_i be the entry/exit flag of the solid interior represented by children surfels. For each surfel-represented intermediate solid, every per-pixel surfel with even index represents *entering* the solid interior and we set $f_i = 1$, while the one with odd index represents *leaving* and we set $f_i = -1$. For subtraction, the entry/exit flags of surfels from the right child are interchanged to obtain the complement. By going through $\{p_i | i = 1, 2, ..., m\}$ sequentially, we identify a pair of points p_i and p_j that satisfy $|d_i - d_j| \leq \epsilon$ and $f_i \cdot f_j < 0$, where ϵ is slightly greater than 0. If a pair of such points are identified, both will be removed from CLDI to avoid the ambiguity introduced by overlapped surfaces. Then we go through each p_i sequentially again and accumulate each f_i iteratively. As a result, the entry of the solid interior increases f_i by 1 while the leave decreases f_i by 1. Now, the interior/exterior configuration on samples can be easily determined by checking f_i sequentially: $f_i = 1$ denotes the entry of resultant solid for union and subtraction whereas $f_i = 2$ denotes the entry for intersection. After applying Boolean on CLDI, invalid surfels are effectively discarded whereas boundary surfels are retained. An illustration has been given in Fig. 4.

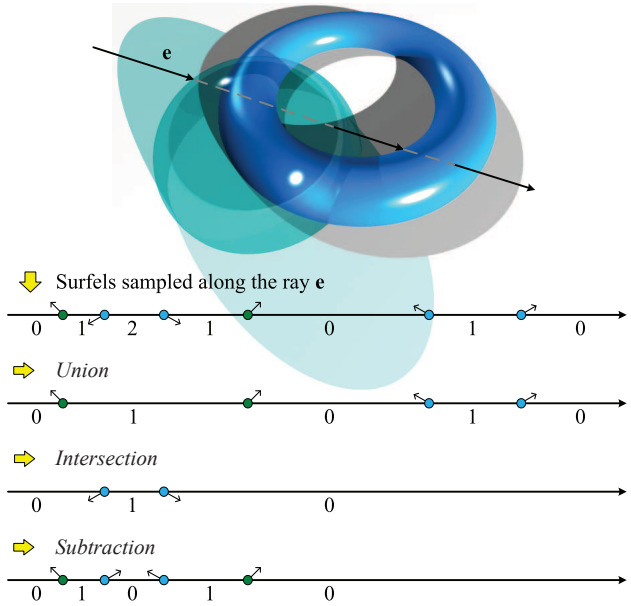

Fig. 4 By processing each pixel independently, the elimination of interior surfels on Sphere and Torus could be highly parallel. See how the entry/exit flags are changed accordingly.

Compared with the sparsely distributed textures for storing LDI or LDNI in prior methods [6, 30], although threads for empty pixels stop immediately, the resource preparation and recycling, and thread synchronization waste a lot of time. The parallelism is therefore reduced. On the contrary, our method shows better parallelism since we launch threads only for non-zero pixels when using the compact data structure, CLDI.

4.4 B-rep Reconstruction

After the whole CSG tree is traversed using the depth-first strategy, the resultant surfel-bounded solid can be obtained at the root node. An efficient surface reconstruction algorithm is then required to output a B-rep model. Kobbelt et al. [16] proposed a feature-preserved

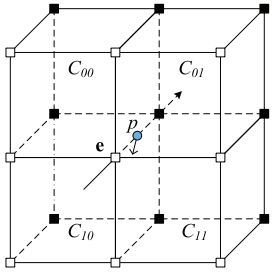

Fig. 5 A boundary surfel p intersected with the axis-aligned ray **e** corresponds to four adjacent boundary cells C_{00}, C_{01}, C_{10}, and C_{11}.

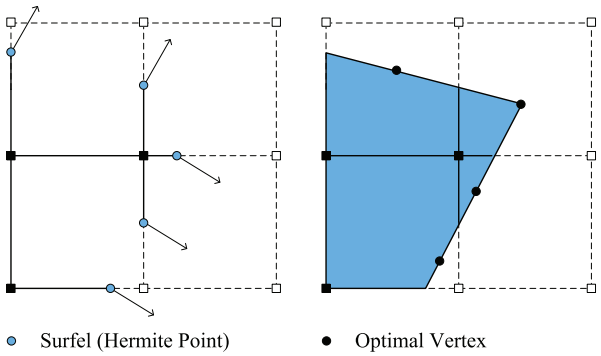

Fig. 6 2D illustration of (left) Hermite data and (right) the dual contouring.

Extended Marching Cubes method and Ju et al. [15] improved their method by using dual contouring to avoid the explicit test for sharp features. In this paper, we modify their dual contouring to fit for our compact representation and implement it on the GPU.

Since orthogonal projection is used in the sampling step, each pixel width is $\delta = \|AABB\|/R_s$. The AABB is divided into multiple equal-sized cubic cells with an edge length δ. As shown in Fig. 5, each cell is positioned to intersect the axis-aligned sampling rays; thus, the resolution is $R_g = R_s + 1$. Each boundary surfel corresponds to four adjacent boundary cells and serves as a Hermite sample in these cells. The construction of QEF matrix from these Hermite samples is actually an accumulation procedure. At the beginning, the elements of QEF matrices are all assigned as zero. We then scatter each boundary surfel to its four neighboring cells and append the related QEF parameters into an array. We associate each element of the array with a cell identifier and an index. After scattering, the *sort* primitive [13] is used to cluster elements with the same identifier. Next, we sum up per-cell QEF parameters and eliminate duplicated cell elements by using the *compact* primitive [13]. The resultant QEF matrix is obtained after conducting accumulation along all the three axes of sampling.

After the QEF matrices of all boundary cells are established, we can compute optimal positions following Ju et al. [15]. We can see the vertices are moved to shape features in Fig. 6. The topology of boundary polygons is reconstructed simply by processing each axis independently. For each boundary surfel p, we connect the four adjacent boundary cells to form a quadrangle. The face of the quadrangle must conform to the orientation of p.

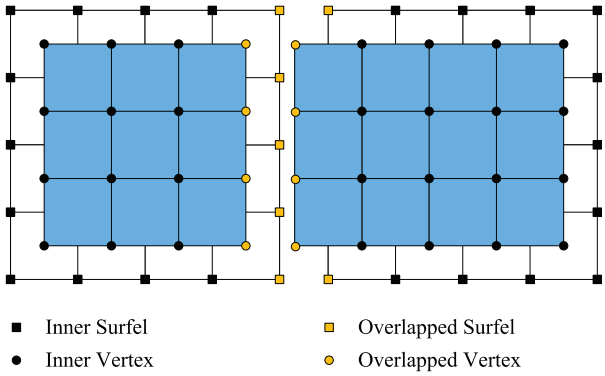

Fig. 7 The overlapped parts between adjacent tiles are processed asymmetrically to guarantee a correct topology.

4.5 Out-of-Core Boolean

The above Boolean evaluation algorithms are computed in-core. However, for very large sampling resolutions, the size of CLDI can grow quickly beyond the memory capacity provided by the graphics hardware even if we employ such a compact representation. An effective out-of-core scheme is needed to meet the requirement of high approximation accuracies.

In order to fit the graphics memory constraints, we split the AABB into smaller volumetric tiles and apply Algorithm 1 to them independently. The critical part then is to guarantee that these independently generated meshes can be merged into a proper two-manifold surface. In this paper, we define the tiles to be overlapped with their adjacent tiles by one pixel width δ. By taking advantage of frustum culling, we only sample surfels within current volumetric tiles. During dual contouring, for each cell, point scattering and quadrangle output should be avoided to be performed more than once (see Fig. 7). Specifically, the surfels in the overlapped parts are not scattered to adjacent tiles. In addition, for each quadrangle, if any of its four adjacent cells lies in the right, front or top overlapped area, it is discarded. This asymmetric contouring technique guarantees that point scattering and quadrangle output are performed exactly once.

The proposed out-of-core technique enables virtually unlimited rasterization resolutions since we can split the bounding volume into tiles and process them sequentially to save memory or in parallel to reduce computation time.

Test	Boolean Expression	Model	Triangles	Model	Triangles
1	CGI2011 ∩ Hexigon	CGI2011	5K	Hexigon	33K
2	Bunny − (offset-Bunny − cubic-Truss) − Sphere	Bunny cubic-Truss	60K 701K	offset-Bunny Sphere	381K 30K
3	(Dragon ∪ Dragon ∪ Dragon) − Buddha	Dragon	49K	Buddha	378K
4	Dinosaur − Fandisk − Helix	Dinosaur Helix	112K 74K	Fandisk	3K
5	Hub − Flower	Hub	162K	Flower	15K

Table 1 Test Boolean expressions with input models.

Test	Resolution	LDI Layers for Each Model	Memory	Triangles
1	$(512 \times 2)^2$ $(512 \times 4)^2$	(26, 6, 2) + (828, 2, 2) (26, 6, 2) + (828, 2, 2)	223MB 609MB	2,380K 9,866K
2	128^2 512^2	(10, 8, 10) + (10, 8, 8) + (54, 54, 56) + (2, 2, 2) (12, 10, 10) + (14, 10, 12) + (54, 54, 56) + (2, 2, 2)	41MB 489MB	227K 3,729K
3	128^2 512^2	(12, 14, 8) + (12, 14, 8) + (12, 14, 8) + (12, 16, 10) (14, 16, 10) + (14, 16, 10) + (14, 16, 10) + (16, 22, 10)	18MB 274MB	87K 1,441K
4	128^2 $(512 \times 2)^2$	(6, 12, 12) + (4, 6, 4) + (6, 8, 8) (8, 12, 12) + (6, 6, 4) + (8, 8, 8)	7MB 194MB	25K 1,676K
5	128^2 512^2	(16, 18, 4) + (8, 6, 6) (16, 18, 4) + (8, 6, 6)	17MB 351MB	163K 2,503K

Table 2 Statistics of our tests. The sampling resolutions, peeled LDI layers, graphics memory peaks, and numbers of output triangles are reported.

Test	Ours		Wang et al.'s Method		Rhinoceros	ACIS
	Coarser Mesh	Finer Mesh	Coarser Mesh	Finer Mesh		
1	3.68s	12.59s	FAILED	FAILED	FAILED	FAILED
2	121ms	944ms	717ms	5.32s	FAILED	FAILED
3	84ms	585ms	328ms	2.59s	19s	FAILED
4	51ms	2.46s	183ms	FAILED	10s	633s
5	53ms	643ms	202ms	2.09s	FAILED	FAILED

Table 3 Performance comparisons. In addition to its high efficiency, our system can deal with various complex Boolean trees.

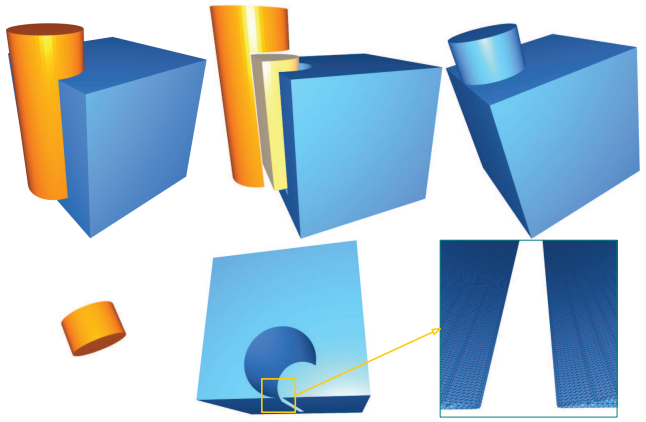

Fig. 8 Validation tests: (top left) input Cube and Cylinder primitives, (top middle) three component models by − and ∩; and for the case of Cylinder tangentially contacting Cube, results of (top right) ∪, (bottom left) Cylinder − Cube and (bottom right) Cube − Cylinder.

5 Results

We have tested our novel Boolean system on a PC with a 2.80GHz Intel Core i5 760 CPU (4GB main memory) and an NVIDIA GeForce GTX 480 GPU (480 streaming processors and 1.5GB graphics memory). Our algorithm uses floating point arithmetic and approximate computation, and involves no explicit intersection calculation. For each cubic cell with edge length δ, the maximum error is the diagonal length $\sqrt{3}\delta$ (ref. [31]). Fig. 8 shows the results of validation tests using basic Cube and Cylinder primitives. Our approach is able to produce Boolean solids with sharp features preserved and robustly deal with overlapped cases. The subtraction of Cylinder results in a cracked Cube because of the degeneration of thin shells and the approximation error which is still within our controllable maximum error $\sqrt{3}\delta$. We can get a better accuracy with a finer grid resolution R_g as more polygons will be generated. We use low-resolution intermediate models for real-time preview while high-resolution models for final output.

We further test our algorithm using complex models and Boolean expressions listed in Table 1. The corresponding statistics are given in Table 2. The maximum resolution of each tile is set as 512^2, while the numbers of LDI layers are counted in **x**, **y** and **z** axes, respectively. Moreover, we record the peaks of graphics memory consumed in our algorithm. The output Boolean models are shown in Fig. 9 and Fig. 11, where the intersection parts are zoomed in for better illustration.

With the data compaction and out-of-core techniques, the memory peaks vary from 7MB to 609MB. All the test

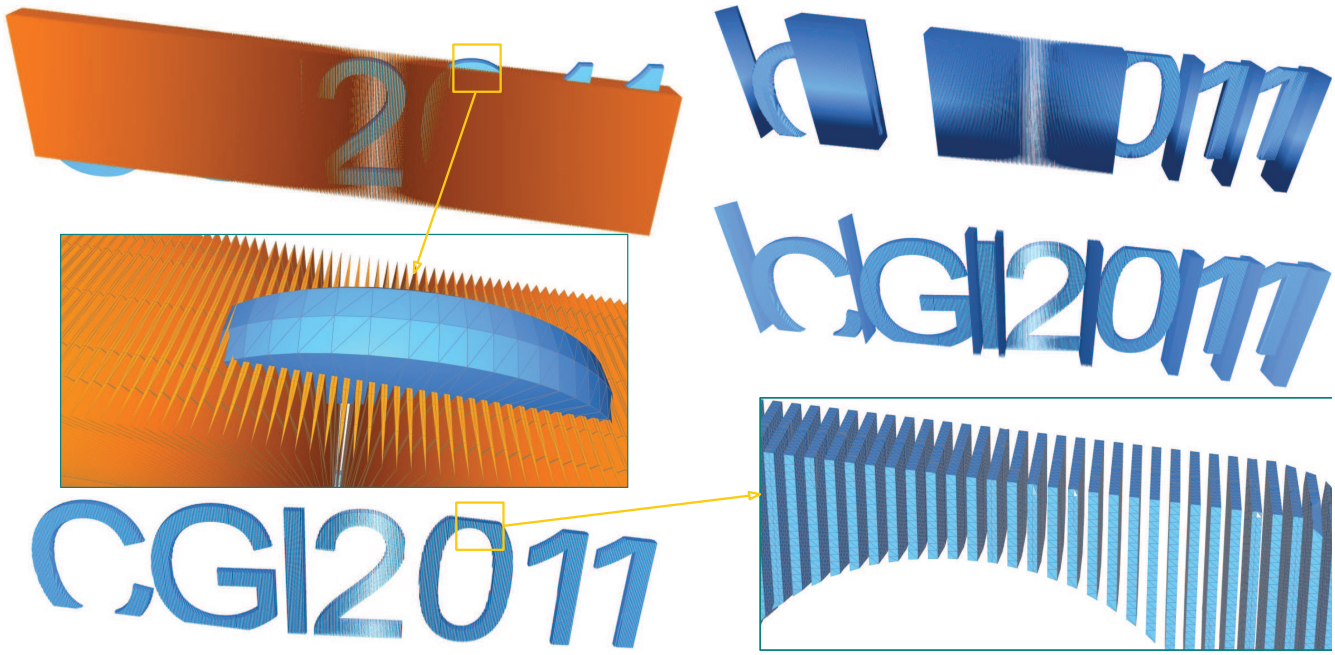

Fig. 9 An example of CGI2011 ∩ Hexagon: (top left) the input CGI2011 model and hundreds of Hexagons, (top right) the result of Rhinoceros, (mid-right) the result of ACIS, and (bottom) our result.

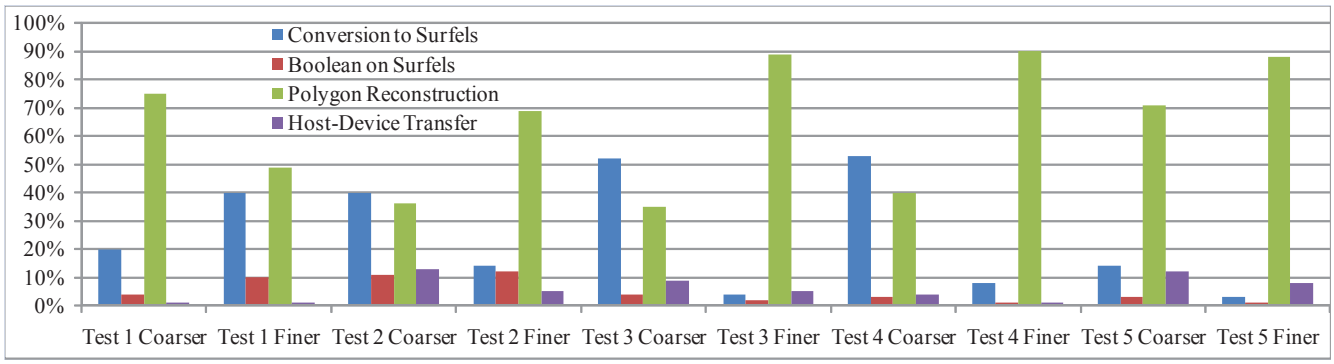

Fig. 10 Statistics of ratio of running time in each step.

data are small enough to be fit in the graphics memory entirely. In the prior algorithm of Wang et al. in [30], each LDNI pixel requires 8 bytes of graphics memory, so 476MB are consumed to store all pixels of LDNI of the finer hollowed-Bunny and their dual contouring without compaction requires even more memory. Their in-core algorithm actually cannot model high-resolution solids. We also test to split the CGI2011 model by hundreds of Hexagons and Wang et al.'s system crashes for this test because the Hexagon model has more than 256 LDI layers. To our surprise, even Rhinoceros [3] and ACIS [1] fail to produce a correct solid for the decent-topology models due to numerical errors, whereas our approach is able to output a satisfying result (see Fig. 9).

Table 3 shows the performance comparisons among our system, Wang et al.'s system [30], Rhinoceros and ACIS. Multi-pass depth peeling and sparse data structure seriously decrease the parallelism of Wang et al.'s GPU algorithm, their performance still cannot meet the requirement of real-time interaction. On the contrary, more than three times of speedup is gained with our new approach. The commercial softwares Rhinoceros and ACIS can only process simple models and both CPU-based algorithms are naturally much slower than the GPU-based ones. With our novel system, real-time visual feedback is achieved with the resolution of 128^2 and thus complex models can be designed on the fly by interactive editing of CSG trees, as demonstrated in our accompanying screen-captured video. Even for models with very high resolutions, we are able to compute the final models within a few seconds. With the increase of capacity of graphics memory, larger tiles can be used to further improve the performance. Since our framework is highly parallel not only inside each tile but also among tiles, we can resort to a multi-GPU system or GPU cluster [21] for better approximation and even higher efficiency. Note

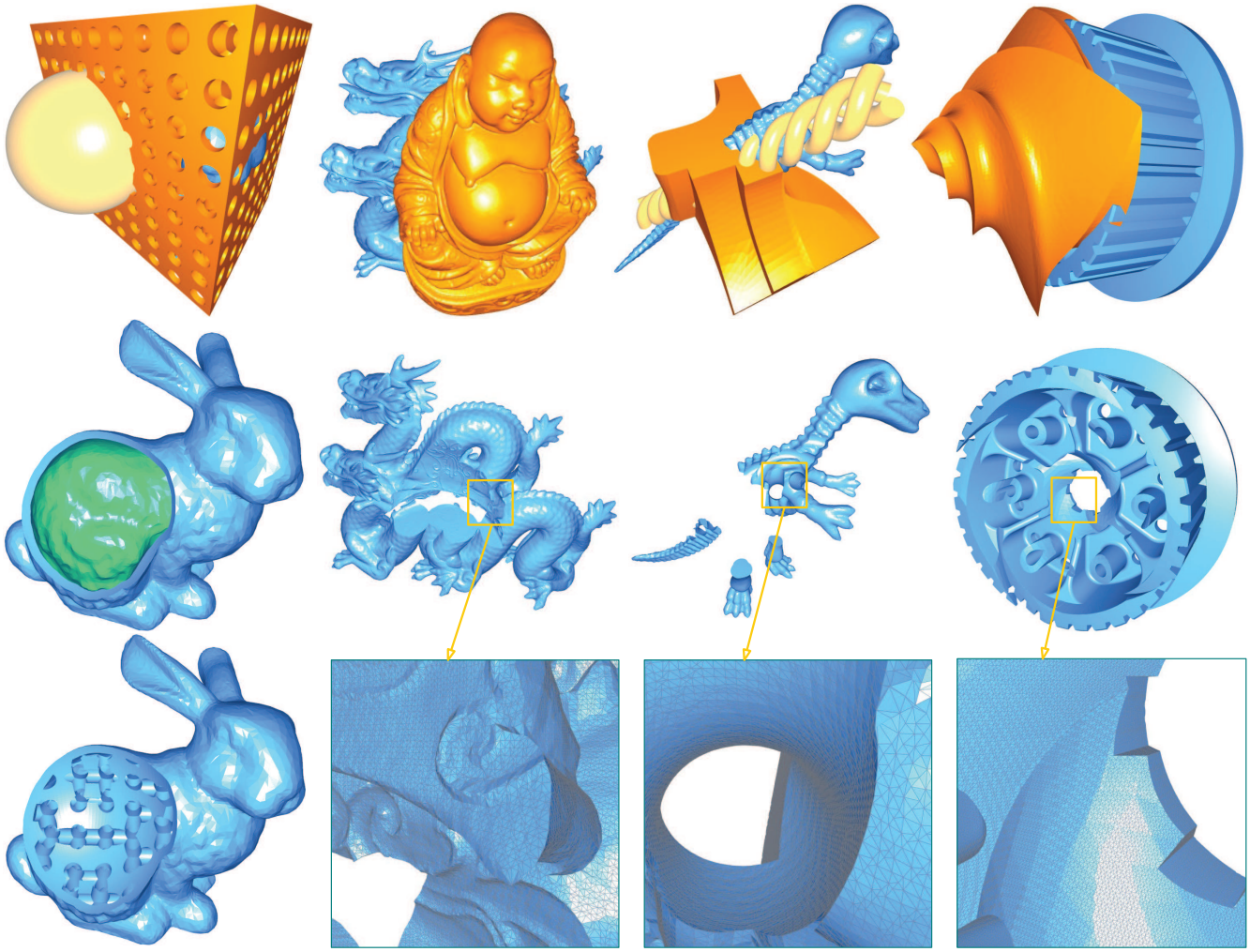

Fig. 11 Example Boolean results with zoomed in (from left to right): Bunny − (offset-Bunny − cubic-Truss) − Sphere, (Dragon ∪ Dragon ∪ Dragon) − Buddha, Dinosaur − Fandisk − Helix, and Hub − Flower.

that prior systems are prone to fail for complex CSG trees or high resolutions in approximation, whereas our system is much more robust.

We give the ratio of running time in each step in Fig. 10. Boolean on surfels is rather fast and few data are transferred between the host and the device. The efficiency of sampling B-rep models to sorted surfels is determined by the layer-complexity of input solids. Polygon reconstruction is highly dependent on the sampling resolution and consumes a great portion of time.

6 Conclusions

In this paper, we have presented an efficient solid modeling framework which computes approximate Boolean operations on polygonal solids using highly parallel algorithms. The main idea is to convert B-rep models into models represented by well structured surfels, discard invalid interior surfels according to the CSG trees, and contour them back into B-rep models as output. Boolean on the axis-aligned surfels is rather simple and fast. By streaming boolean operations, an out-of-core volumetric tiling technique is introduced to generate highly accurate results with high sampling resolution.

The number of reconstructed polygons in current implementation depends on the sampling resolution. In the future we will explore an adaptive technique to further improve the performance in the B-rep reconstruction step. As our CLDI structure is general, we also plan to incorporate CLDI into various modeling applications.

Acknowledgements

The work was partially supported by HKSAR Research Grants Council (RGC) General Research Fund (GRF) on project CUHK/417508. The research of H. Zhao was partially supported by the Zhejiang Provincial Natural Science Foundation of China (Grant No. Y1110004),

the Scientific Research Fund of Zhejiang Provincial Education Department, China (Grant No. Y201010070), the Science and Technology Plan Program of Wenzhou, China (Grant No. H20100088) and the Open Project Program of State Key Lab of CAD&CG, Zhejiang University, China (Grant No. A1117). The research of X. Jin was supported by the Zhejiang Provincial Natural Science Foundation of China (Grant No. Z1110154) and the National Natural Science Foundation of China (Grant No. 60833007).

References

1. 3d acis modeling. In *http://www.spatial.com*. Spatial Corporation, 2010.
2. Cuda programming guide for cuda toolkit 3.0. In *http://developer.nvidia.com/object/gpucomputing.html*. NVIDIA Corporation, 2010.
3. Rhinoceros. In *http://www.rhino3d.com*. Robert McNeel & Associates, 2010.
4. Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. *ACM Transactions on Graphics*, 22(3):651–656, 2003.
5. L. Bavoil and K. Myers. Order independent transparency with dual depth peeling. In *Technical Report*. NVIDIA Corporation, 2008.
6. Yong Chen and C. L. Charlie Wang. Layered depth-normal images for complex geometries - part one: Accurate modeling and adaptive sampling. In *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Brooklyn, NY, 2008. ASME.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computional Geometry*. Springer, 1997.
8. C. Everitt. Interactive order-independent transparency. In *Technical Report*. NVIDIA Corporation, 2001.
9. J. Goldfeather, J.P.M. Hultquist, and H. Fuchs. Fast constructive-solid geometry display in the pixel-powers graphics system. In *Proceedings of SIGGRAPH '86*, pages 107–116, New York, NY, 1986. ACM.
10. J. Goldfeather, S. Molnar, G. Turk, and H. Fuchs. Near real-time csg rendering using tree normalization and geometric pruning. *IEEE Computer Graphics and Applications*, 9(3):20–28, 1989.
11. John Hable and Jarek Rossignac. Blister: Gpu-based rendering of boolean combinations of free-form triangulated shapes. In *Proceedings of SIGGRAPH '05*, pages 1024–1031, New York, NY, 2005. ACM.
12. John Hable and Jarek Rossignac. Cst: Constructive solid trimming for rendering breps and csg. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1004–1014, 2007.
13. Mark Harris, Shubhabrata Sengupta, and John D. Owens. Parallel prefix sum (scan) with cuda. In Hubert Nguyen, editor, *GPU Gems 3*. Addison Wesley, 2007.
14. C. Hoffmann. Robustness in geometric computations. *ASME Journal of Computing and Information Science in Engineering*, 1(2):143–155, 2001.
15. Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Transactions on Graphics*, 21(3):339–346, 2002.
16. Leif P. Kobbelt, Mario Botsch, Ulrich Schwanecke, and Hans-Peter Seidel. Feature sensitive surface extraction from volume data. In *Proceedings of SIGGRAPH '01*, pages 57–66, New York, NY, 2001. ACM.
17. Shankar Krishnan, M. Gopi, Dinesh Manocha, and Mark R. Mine. Interactive boundary computation of boolean combinations of sculptured solids. *Computer Graphics Forum*, 16(3):67–78, 1997.
18. Shankar Krishnan, Dinesh Manocha, M. Gopi, Tim Culver, and John Keyser. Boole: A boundary evaluation system for boolean combinations of sculptured solids. *International Journal on Computational Geometry and Applications*, 11(1):105–144, 2001.
19. Fang Liu, Meng-Cheng Huang, Xue-Hui Liu, and En-Hua Wu. Freepipe: a programmable parallel rendering architecture for efficient multi-fragment effects. In *Proceedings of the ACM Symposium on Interactive 3D Graphics and Games*, pages 75–82, New York, NY, 2010. ACM.
20. William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of SIGGRAPH '87*, pages 163–169, New York, NY, 1987. ACM.
21. C. Müller, S. Frey, M. Strengert, C. Dachsbacher, and T. Ertl. A compute unified system architecture for graphics clusters incorporating data locality. *IEEE Transactions on Visualization and Computer Graphics*, 15(4):605–617, 2009.
22. K. Myers and L. Bavoil. Stencil routed a-buffer. In *ACM SIGGRAPH '07 Technical Sketch Program*. ACM Press, 2007.
23. M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *Proceedings of SIGGRAPH '03*, pages 641–650, New York, NY, 2003. ACM.
24. H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH '00: ACM SIGGRAPH 2000 Papers*, pages 335–342, New York, NY, USA, 2000. ACM.
25. A. Rappoport and S. Spitz. Interactive boolean operations for conceptual design of 3-d solids. In *SIGGRAPH '97: ACM SIGGRAPH 1997 Papers*, pages 269–278, New York, NY, USA, 1997. ACM.
26. J. Rossignac and A. Requicha. Solid modeling. *Encyclopedia of Electrical and Electronics Engineering*, 1999.
27. Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of SIGGRAPH '98*, pages 231–242, New York, NY, 1998. ACM.
28. S. Toledo, D. Chen, and V. Rotkin. Taucs, a library of sparse linear solvers (version 2.2). In *http://www.tau.ac.il/~stoledo/taucs/*. 2003.
29. Matthias Trapp and Jürgen Döllner. Real-time volumetric tests using layered depth images. In *Proceedings of EUROGRAPHICS '08*, pages 235–238, 2008.
30. Charlie C. L. Wang, Yuen-Shan Leung, and Yong Chen. Solid modeling of polyhedral objects by layered depth-normal images on the gpu. *Computer Aided Design*, 42(6):535–544, 2010.
31. Charlie C.L. Wang. Approximate boolean operations on large polyhedral solids with partial mesh reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, Published Online, 2010.
32. Kun Zhou, Minmin Gong, Xin Huang, and Baining Guo. Data-parallel octrees for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 16(6), 2010.