CSG Operations on Point Models with Implicit Connectivity

Nan Zhang^{*}
University of Minnesota

Huamin Qu[†]
HKUST

Arie Kaufman[‡]
Stony Brook University

ABSTRACT

We propose Point with Implicit Connectivity (PIC) as a new data structure for representing solid objects using points. In the PIC representation, an object is adaptively sampled into an octree, where each leaf cell contains at most one surface component of the object. Each surface component is represented by a vertex, together with inside/outside classification values of the cell corners. PIC objects are compact, feature-preserving, and supports easy construction of the boundary surfaces. To convert geometric objects into the PIC representation, we propose a sampling algorithm and use quadric error functions as error metrics. For CSG operations between PIC objects, we present a feature-preserving, adaptive CSG algorithm on the octrees. Our experiments show promising results for PIC objects with sharp features and large flat regions.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Graphics Data Structure and Data Types

Keywords: Point modeling, implicit connectivity, CSG operations

1 INTRODUCTION

Sampled-based representations, such as images, volumes and points, are widely used in computer graphics, due to the popularity of affordable, high resolution data acquisition devices. Effectively representing, manipulating and rendering these datasets are of great importance. According to the Nyquist sampling theorem, a continuous model should be adequately sampled to properly recover the original signal. For example, modern 3D scanning devices usually produce millions of densely-sampled points to faithfully reconstruct a model. Many of the points are unnecessary because of the existence of large flat regions. Although point simplification algorithms (e.g. [21]), have been proposed to address this problem, they can not simplify the densely sampled points near sharp features, such as corners and sharp edges.

In this paper, we propose to use additional information, the implicit connectivity, for points to adaptively represent objects containing both flat regions and sharp features. The implicit connectivity information also greatly eases the construction of a base surface for each point in an adaptively sampled point set. We use the term Point with Implicit Connectivity (PIC) as the name for our point representation. PICs are organized into an octree data structure with the cell corner signs forming the implicit connectivity. We propose a novel algorithm for implementing CSG boolean operations on PIC objects. Our main contributions are an adaptive PIC sampling algorithm on polygonal meshes and an adaptive CSG algorithm for PIC objects.

^{*}Email: nanzhang@cs.umn.edu

[†]Email: huamin@cs.ust.hk

[‡]Email: ari@cs.sunysb.edu

2 RELATED WORK

Point-Based Modeling Points are the natural representation of scanning devices output. Levoy et al. [16] have described a system of acquiring digitized models of statues using laser range scanning devices. Pfister et al. [24] have proposed the *surfel* representation for sampling and rendering complex geometric objects. Qu et al. [26] have used the O-buffer, a regular grid and quantized off-sets for organizing points in high sampling precision. Directly rendering points [24, 27] is effective since a large amount of the highly detailed surface primitives are usually projected to areas smaller than a pixel. Various methods have been proposed to construct a smooth manifold surface from point sets, including radial basis function (RBF) [7], moving least-squares (MLS) [2], and Multi-level Partition of Unity (MPU) implicit surface [20]. Using the underlying surface definition, a collection of point set processing operations are supported, such as sampling and resampling (simplification) [21], free-form deformation [22], and CSG operations [1, 22]. Pauly et al. [22] have proposed a point CSG algorithm, where MLS surface intersection is computed. Adam and Dutré [1] have proposed an interactive point CSG algorithm, where points are treated as small disks and the intersection lines are approximated by smaller disks.

Compared to the above point-based representations, our PIC representation provides an easy-to-construct, C^0 continuous underlying surface. Since we store the implicit connectivity information, the datasets can be sparsely sampled, whereas other point representations have to sample objects in adequate rates for proper surface reconstruction. After each CSG operation, PIC datasets are well-organized in sorted positions. Other point representations, however, need to be re-sorted [1] for the next CSG operation. Furthermore, rendering PIC objects only use standard graphics hardware, while other point representations need special effort in rendering.

Grid-Based Modeling The grid-based modeling techniques have been studied extensively and are often referred to as volume modeling techniques. Usually, a scalar value is stored for each grid point. The object is implicitly defined by a scalar field $f(x, y, z)$ on a discretized sampling grid. To convert geometric objects from their continuous representations, a scan-conversion process is applied. Smooth, differential objects can be well-represented by distance volumes in a regular grid [3, 11, 29]. In adaptively-sampled distance field (ADF) [9], an octree is built to sample an object according to local surface curvature. Perry and Frisken [23] have presented a volume sculpting system based on the ADF representation. To preserve sharp features of polygonal models, explicit intersection points are stored. Kobbelt et al. [14] have used directed distances in x , y , and z directions and have reconstructed the features of the original models during polygonization using the extended Marching Cubes (EMC) algorithm. Ju et al. [13] have used a similar technique, where Hermite data (i.e., exact intersection points and normals) are stored. Their method reconstructs features automatically through error minimization and uses a dual contouring algorithm to extract meshes. Varadhan et al. [30] have extended the work of Ju et al. by storing up to two intersection points per edge to reconstruct multiple surface components. However, when more intersection points are introduced, the ambiguity problem of surface reconstruction is more serious.

Our method is closely related to the feature-preserving volume modeling techniques, where a similar surface construction algorithm is used. However, our representation is initially sampled in an adaptive manner and stores no intersection points on the edges, and thus has a much lower memory footprint.

Octree-based CSG Acceleration For objects represented as boundary surfaces, octree partitioning algorithms have been proposed to efficiently evaluate CSG expressions. In the octree accelerated CSG implementation [5], the triangles of the mesh are sorted into the octree cells. The triangles in the overlapping octree cells are evaluated using a triangle splitting algorithm. In the Polygonal Map (PM) octree representation [4, 6], an octree is recursively subdivided until each cell is one of five different leaf cell types. In addition to black (full) and white (empty) cells, three new leaf cells are introduced: vertex cells, where a single vertex and its connected faces and edges are stored; edge cells, where a single edge and its faces are stored; and surface cells, a piece of a single face is stored. These three cells are special cases of gray (partially full) cells. Since the cell types are restricted to a set of simple geometries, the CSG algorithms are simplified.

The PIC representation bears resemblance to the extended octree method [4], where the vertex cells, edge cells and face cells can be represented by PICs. However, our CSG algorithm is much simpler and easier to implement. In our method, CSG operations are divided into two stages: vertex computation and connectivity computation. No intersection line is computed. The object topology is maintained by maintaining the in/out information of cell corner points.

3 THE PIC REPRESENTATION

Similar to the *surfel* representation, PIC is a discrete surface representation. However, a *surfel* is a zero-dimensional point sample without connectivity information. The PIC representation stores in a cell both a point sample and the in/out classification values of the cell corners, where each value indicates whether the corresponding cell corner point is inside or outside the object. These signs define the connection of a vertex with its neighboring vertices implicitly. We define the PIC as follows:

PIC is a cell with a representative vertex in it and the inside/outside values for the eight corner points. A vertex may contain other associated attributes, such as color, normal, texture coordinates, and a timestamp of modification.

Figure 1: The structure of a PIC has two parts: a representative vertex and eight signs for the corner points.

Figure 1 illustrates the PIC data structure. Although it is redundant to store the in/out bits of all the corner points, they just cost a few extra bits and accelerate processing speed. The PIC definition can be applied to uniform space partitioning. However, it is more desirable to store the PICs in an octree data structure. A similar representation [15] has been used for progressive isosurface compression. However, ours has a different focus of solid modeling.

To extract the contour from the PIC representation, we use the SurfaceNets algorithm [23]. Cube-based contouring algorithms, such as the Marching Cubes (MC) [18], generate points on the grid lines where the contour passes through and then connect these points according to a predefined lookup table (LUT). Since only the in/out information is stored in each cell, neither the MC algorithm nor the discrete MC algorithm [19] can be applied to extract the accurate contour. Alternatively, the SurfaceNets algorithm generates one representative vertex on the contour for each cell that the surface passes through. For each cell edge that exhibits a sign change¹, the four vertices coming from the four cells sharing that edge are connected. The PIC objects naturally support the SurfaceNets algorithm since all the representative vertices have been pre-generated. Features, such as sharp edges and corners, can be preserved as long as the representative vertices are properly positioned. We also use an extension of the SurfaceNets algorithm to extract the partial contour. The partial contour is used in CSG operations and is discussed in detail in Section 5.2.1.

4 ADAPTIVE SAMPLING

The sampling algorithm of the *surfel* [24] uses a ray-tracer to sample the objects uniformly for each layered depth cube. In our sampling algorithm, the goal is to find an optimal PIC representation for an object with guaranteed geometric accuracy and minimum redundancy. The models used in this paper are converted from manifold polygonal meshes. We propose a top-down sampling algorithm, which is adaptive and feature-preserving. Our algorithm uses an axis-aligned octree. In our top-down approach, first the signs for the root cell of the octree are computed. The cells are then recursively subdivided according to several refinement rules, such as geometric errors and surface components count. We first discuss the algorithms for computing corner signs and the quadric error metric (QEM) [10] for measuring the geometric errors and positioning the representative vertices, then give the refinement rules.

4.1 Point Classification

For each non-empty octree cell, the inside/outside tests are performed for the eight corner points. There are two algorithms used in our system for the tests:

- Point projection. A corner point is projected to the triangles contained in the cell. If the projected point is inside a triangle, the in/out classification can be determined by the signed distance. This method is similar to the projection approach of Pauly et al. [22].
- Edge/surface intersection. The signs of the two end points of an edge can be determined when there is an intersection point. It can also be inferred when no intersection point is found and the sign of one end point of the edge is known.

The first algorithm has the priority over the second. The second algorithm is performed when a corner point can't be projected to any triangle in the cell. During the adaptive sampling, space coherency is exploited to reduce computation cost. The classification signs of a cell is propagated to its neighboring cells and descendants.

¹In a signed distance field, a sign change means that the zero-isosurface passes through that edge. We use the same term here.

4.2 Quadric Error Metrics

QEM is initially developed as an error metric for mesh simplification. In our adaptive sampling algorithm, for each gray cell, a representative vertex is computed by minimizing the quadric error function (QEF) Q [10] associated with the cell:

$$Q(x) = x^T A x - 2b^T x + c \quad (1)$$

where x is the representative vertex, A is a 3×3 matrix, b is a column vector, and c is a constant. For detailed explanation on the quadric error function, we refer the readers to [10]. $Q(x)$ represents the sum of square distance of x to a set of planes passing through the cell. The error δ^2 is defined as the residue after minimization. Normally, the position of x can be solved by solving a linear equation $Ax = b$ to minimize the square distance. To robustly produce the best vertex, we use the following equation to solve the least square optimization problem [17]:

$$x = \bar{x} + A^{-1}(b - A\bar{x}) \quad (2)$$

where the seed point $\bar{x}$ is the center of the cell. If the matrix A is singular or near singular, the minimization can't be solved by the Gaussian elimination method. In these cases, we adopt the singular value decomposition method (SVD) [25].

Using the center as the seed point often causes the minimized point to be outside the cell [28], as shown in Figure 2a. In the SVD method, although part of the solution space is inside the cell, the computed solution is not guaranteed to be in the cell. Actually it only gives a solution that has minimized the sum of the square distances to the planes and is closest to the seed point. Schaefer et al. [28] have suggested to replace the center point with the mass point (the average of the intersection points on the edges). In our system, since the intersection points are unavailable, we use the corner point which is the closest to the surface. The distance of a corner point to the surface is efficiently estimated using Frisken and Perry's method [8]. In the minimization process, we first use the cell center. If the representative vertex is outside the cell, we re-compute the representative vertex using the closest corner point.

Figure 2: Minimizing one QEF. (a) Using a cell center as the seed. (b) Using a corner point as the seed.

4.3 Refinement Criteria

To define the refinement rules, the results of point classification and quadric error minimization are used. For each non-empty cell, the criteria are defined as:

1. The minimized quadric error should be less than a user-specified error threshold.
2. The representative vertex should be within the cell.
3. The in/out signs of the corner points indicate only one surface component.
4. There should be no cell edge which contains two or more intersection points.

The first two rules give constraints on vertex positions. The QEF for a cell is constructed by taking account of all the triangles intersected or contained by this cell. This sampling approach is more accurate than just considering the intersection points on the cell edges. The third criterion tells us whether the surface components inside this cell can be represented by one vertex or not. This prevents topology problems even when a large error threshold is used. From the viewpoint of surface extraction, the isosurface inside a cell might be extracted into several disconnected components. The disconnected components may cause topology errors if collapsed into one vertex. In the discretized MC algorithm [19], a LUT of surface configurations is given for binary volume datasets — the same data as the classification signs used in our sampling method. There are totally 16 unique configurations. Among them, 8 configurations contain only one surface component. The others have two or more disconnected components (cf. Figure 3 in [19]). We store one extra field in the discretized MC LUT to indicate the surface component count.

The last criterion is used to ensure that there is at most one intersection point on each edge. This criterion is checked by computing the intersection between each cell edge and the surface component passing through this cell. We use the following optimization: only an edge which shows no sign change should perform this test. Furthermore, edge coherency is exploited between the neighboring cells. The child cells will also inherit the results from their parent cells on the shared edges. As the result, less than 30% cell edges are checked in Figure 8, for example.

5 CSG OPERATIONS

The CSG operations are evaluated between two adaptively sampled octrees. For each pair of cells used in a CSG operation, they should be the same size. The CSG operations are applied on the two parts of the PIC representation separately. Each part is handled differently.

5.1 CSG on Corner Points

The purpose of performing CSG operations on the corner points is to maintain the vertex connectivity. For the classification signs, we use Boolean algebra as CSG operators. The inside sign is assigned as 1, while the outside sign is 0. These operations are similar to the CSG operations performed on binary volumes. The results are stored and are also used for determining the count of disconnected surface components in cells, as in Section 4.3. Since the PIC representation can only represent one surface component, a cell has to be subdivided if more than one surface component exist.

5.2 Vertex CSG

There are three basic leaf octree cell types: black, white, and gray. CSG operations between black and white cells are trivial. CSG operations between a gray cell and a black or white cell can also be easily computed. For example, to compute the difference of a black cell and a gray cell, we simply copy the vertex position and the inverted signs of the gray cell to the black cell. For CSG operations between gray cells, the vertex CSG algorithm is applied. It uses a combination of recursive subdivision and minimization of quadric error functions associated with the representative vertices.

The vertex CSG algorithm contains a preprocessing step and a vertex evaluation step. In the preprocessing step, for each PIC in the overlapping region, a triangle fan passing through it is constructed.

The triangle fan is centered at the representative vertex. The union of the triangle fans of all the overlapping cells in an octree forms a partial contour of that octree. The partial contours for the two octrees are extracted using the algorithm described in Section 5.2.1.

The vertex evaluation step is executed for every pair of octree cells of the same size. If two cells are not the same size, the larger one must be subdivided. Our recursive CSG algorithm is described as follows:

1. If one of the two cells is a black or white cell, CSG operations between them can be evaluated easily.
2. For two gray cells, if both of them are non-leaf cells, a recursive process is called to process their child cells. Otherwise, if one of them is a leaf cell, subdivide this leaf cell first and process the child cells recursively.
3. If both of the cells are gray cells, we test the count of representative points in the resulting cell. If more vertices are required, the cells are further subdivided. Figure 3 illustrates the gray cell CSG in detail.

```

procedure GraycellCSG(GrayCell A, GrayCell B)
  QEF  $Q_{sum} = QEF_A + QEF_B$ ;
  minimize  $Q_{sum}$ ;
  if residue of  $Q_{sum} < \text{threshold}$ 
    if no more than one intersection point on each cell edge
      return;
    else
      goto label;
    if B can be culled by A
      return;
    else
      goto label;
  label:
    subdivide A and B;
    perform the CSG operation on non-gray child cells;
    for each gray child cell C of A, gray child cell D of B
      GraycellCSG(C, D);

```

Figure 3: CSG algorithm for gray cells.

There are three key procedures in the vertex CSG algorithm: extracting the partial contours and creating quadric functions for vertices, generating new representative vertices, and maintaining the PIC definition without any violation. We describe these procedures in detail in the remainder part of this section.

5.2.1 Extracting Partial Contours

The quadric error functions are not stored in the PIC representation. To retrieve these functions, the underlying surfaces have to be extracted before each CSG operation. Then, the QEFs are constructed from the triangles in the surfaces. Besides, a partial contour provides the base mesh for the cells to be subdivided. The adaptive sampling algorithm in Section 4 is also used for forced subdivision.

Essentially, for each CSG operation, we need to compute one partial contour for each object. To determine the overlapping region, a timestamp technique is used. The system maintains a global timestamp. Each octree cell stores a local timestamp. Before each CSG operation, the global timestamp is increased by one. The overlapping cells are assigned the global timestamp by a recursive tagging procedure, while other cells are untouched. The benefit of using a timestamp is that there is no need to traverse the whole octree to

find the overlapping region. There is also no need to reset all the timestamps for the next CSG operation. The performance of interactive applications, such as sculpting, is therefore improved.

In order to reduce the computation load of contouring, we use a truncated SurfaceNets algorithm. The SurfaceNets algorithm is truncated in two cases: (1) only the overlapping region needs to generate the partial contours; and (2) since the CSG result between a black or white cell and another cell involves no QEFs, these black or white cells can be skipped. The partial contouring algorithm uses a recursive octree traversal rule similar to the full contouring algorithm [23]. The difference is that it checks the timestamp field stored in each cell. Since a new timestamp is generated for each CSG operation, it is unnecessary to construct the triangle fan when a cell's timestamp is older than the current value and none of its 18-neighborhood cells contains the current timestamp. The partial contour extraction procedure only takes time proportional to the count of affected cells. Unnecessary computations are saved, such as contouring the cutting region of object A in Figure 4. Using the above optimization technique, the preprocessing of partial contouring is achieved on-the-fly.

Figure 4: In the CSG operation of objects A and B, the partial contours are extracted on the overlapping region.

In accordance with the optimization, the following two rules are applied to update the timestamp field in each cell:

- None of the black cells and white cells should be assigned the global timestamp.
- If a gray cell's counterpart (in the other octree) is a black or white cells, there is no need to assign this cell and its descendant the global timestamp.

5.2.2 Creating Representative Vertices

One significant difference between our vertex CSG algorithm and directed distance based CSG algorithms is that we minimize the sum of the quadrics collected from the cells directly. In the directed distance representation, the min/max operations are applied on distance values. The underlying assumption is that one cell contains only one feature. This assumption may not be true for the adaptively sampled cells. Figure 5 gives examples where the min/max operators generate wrong results, where two or more features arise after a CSG operation. Although the PIC representation can't represent more than one feature, the CSG algorithm detects the violating cases and uses cell subdivision to avoid them.

There is no need to compute the exact count of features generated by a CSG operation. In fact, we just need to detect whether the feature count exceeds one or not. Similar to the sampling stage, we can minimize a QEF Q_{sum} :

$$Q_{sum} = Q_A + Q_B \quad (3)$$

where Q_A and Q_B are the QEFs associated with the two cells A and B. The seed used in the minimization is the average of their

Figure 5: For large cells tagged with directed distances, the min/max based CSG operations may fail to generate the correct result. (upper row) Union, difference and intersection between two objects. The correct results are shown in dash lines; (lower row) The incorrect results.

vertices. If the minimization error is less than a user-specified error threshold, a representative vertex is created and a representation safety test is performed, which is described in Section 5.2.3. Otherwise, there may be more than one feature. However, a triangle culling process is still needed to test whether the two triangles sets intersect each other in the cell space. If intersection is confirmed, we have to subdivide the cells. Otherwise, there is no need for subdivision. Figure 6 gives the explanation, where $A-B$ is evaluated. Suppose both of the minimization errors are greater than the error threshold, the result on the right remains A . On the left, three feature points are created and a recursive cell subdivision has to be performed to distinguish them.

Figure 6: In the CSG difference operation $A-B$, the result of (left) has three feature points (shown in blue), while the result of (right) remains to be object A .

The triangle culling procedure used in our algorithm is simple, due to the existence of only one surface component in each cell. The procedure is similar to the algorithm used in the extended octree [4]. Since the partial contours have been extracted, the triangle fan for each vertex is used for culling. Compared to the algorithm in [4], the culling procedure is largely simplified since our goal is just to cull one set of triangles. We only need to detect the intersections between the surfaces. Early termination is applied when the first intersection is detected. There is no need to compute intersection lines and maintain connectivity information.

5.2.3 Handling Double Intersections

To maintain the closure of the PIC representation after a CSG operation, the intersection count on an edge can't exceed one. However,

even when the QEF residue is zero, there is still a possibility that one edge has two intersection points. Figure 7 shows such a scenario, where operation $A-B$ is computed to generate a sharp angle. In Figure 7, although the minimized quadric error is zero, two intersection points are created on one edge. This case occurs frequently and is found in the sculpting stage of Figure 10c. To solve this problem, such a cell should be recursively subdivided until the two intersection points are separated.

Figure 7: When evaluating $A-B$, two intersection points are generated on a cell edge, although the minimized error is zero.

Performing the intersection tests is time-consuming. Fortunately, it is unnecessary to perform the detection for all the edges. Table 1 gives the sign configurations for a pair of edges that may contain two intersection points after a CSG operation. These configurations mean that there is possibility of double intersections only when there are sign changes in both edges but no sign changes on the resulting edge. For every edge of both cells, the test is performed only when a table item is satisfied. The lookup table saves about 85% of the tests on cell edges in Figure 10. The test is achieved by computing the directed distances on the edges followed by 1D CSG operations.

Table 1: Sign configurations for the two end points of a cell edge to perform the double-intersection test.

	Object A edge		Object B edge	
Operations	end 1	end 2	end 1	end 2
Union	0	1	1	0
Union	1	0	0	1
Intersection	0	1	1	0
Intersection	1	0	0	1
Difference	0	1	0	1
Difference	1	0	1	0

6 IMPLEMENTATION AND RESULTS

Our experiments have been conducted on a uni-processor 3.0GHz Intel Pentium IV PC running Windows XP, with 1024M RAM and NVIDIA GeForce 6800GT graphics board. In all the examples, the physical size of the octree is set to 1.0. In the PIC structure, we store the vertex coordinates (12 bytes), the signs (1 byte), a timestamp (2 bytes) and some other tagging information (3 bytes). Totally 18 bytes are used for storing the PIC representation, plus the octree overhead. Compared to the data structures requiring hundreds of bytes per cell [13, 30], our method provides a neat solution. It also shows efficiency in sampling objects with large flat regions and sharp features relative to general point sampling.

Figures 10 shows the adaptiveness of our representation. A slab, initially represented by 8 PICs, is progressively refined using three CSG operations. For the carving on the front-left corner of the slab, the subdivision level is low (Figure 10b). For the carving on the

Figure 8: The fandisk dataset is sampled into the PIC representation, where a level-12 octree is used and $\delta^2 = 10^{-8}$. (a) The original model. (b) Surface reconstructed from the PIC representation.

Table 2: Adaptive sampling time (in sec.) for four models, where the $\delta^2 = 10^{-8}$ and $\text{octree}_{\text{max_level}} = 12$.

Model	Figure	Triangle #	PIC #	Time
fandisk	8	13K	14K	0.65
cylinder	11a	2K	8K	0.24
gazebo	11b	68K	238K	7.91
dolphin	11c	4K	66K	2.82

front-right corner, much deeper subdivisions are performed (Figure 10b). Thus, the object subdivides itself according to the modeling details required. Figures 8 and 11 show several other objects represented in PIC. These models are faithfully reconstructed by our adaptive sampling algorithm. For example, the reconstructed fandisk dataset of Figure 8b is visually identical to the original mesh of Figure 8a. Even the small details are preserved well. The facet property of the polygon mesh is also preserved well in the gazebo (Figure 11b) and the dolphin models (Figure 11c). The gazebo model is an isosurface extracted from a density volume using iso-value 120.5. The dolphin dataset is poorly sampled using a regular grid of 256^3 in Figure 11e, since the mouth, the eyes and the fins are tiny or thin, sharp features and thus require a very high sampling resolution. Meanwhile, the major part of the dolphin is smooth and can be sampled in low resolution. In Figure 11f, this model is properly sampled by our method.

The timing of our sampling algorithm is listed in Table 2. During sampling of these datasets, we set the maximum octree depth = 12 and the QEF error tolerance $\delta^2 = 10^{-8}$. This QEF error is equivalent to the order of 10^{-4} in distance. We find that sampling efficiency is determined by several factors, such as the count of triangles, the smoothness of the mesh, and the relative position of the mesh and the grid. Among these factors, the relative position is probably the most important since a cell may be recursively subdivided many levels if two or more intersection points are found on an edge. This becomes particularly dominant when the error threshold is small. Figure 9 shows this scenario, where three octrees are generated by sampling the fandisk dataset in error thresholds 10^{-4} , 10^{-6} , and 10^{-8} . In Figure 9, the octree of $\delta^2 = 10^{-8}$ has the largest amount of high level gray cells. Although a large error threshold, such as 10^{-4} , is specified, there is still a considerable amount of cells subdivided to the finest level. This is because of the double intersections on the edges. Since recursive subdivision is a very expensive operation, it has a major impact on the sampling efficiency.

The timing of our CSG algorithm is presented in Table 3, where a CAD tool and a dolphin are used. The setting is the same as in Table 2. The CAD tool in Figure 11g is created by three consec-

Figure 9: Statistics on the count of PICs. The octrees are created for the fandisk dataset using three QEF error thresholds: 10^{-8} , 10^{-6} , and 10^{-4} .

utive CSG operations between a cube and three cylinders. These objects are pre-sampled in the PIC representation. The operation sequence is union, difference and difference. The dolphin model in Figure 11i is created in the same way. The results of Table 3 indicate that we have achieved nearly interactive CSG evaluations. The PIC sampling time is not included and can affect the total CSG timing dramatically, if either one is too complex. However, usually we treat it as the pre-processing cost. The resulting models after CSG operations have two advantages. (1) They are adaptively sampled and optimized for rendering. Since polygon rendering is accelerated by graphics hardware, such as ATI Radeon X800 and NVIDIA Quadro 3400, the display cost is dominated by the polygonization time and can be reduced by incremental surface construction. Our system has a polygonization speed of 8 million triangles per second. (2) The resulting PICs are always sorted. In other point CSG algorithms [1, 22], the resulting points need to be re-organized. Sometimes, the re-organization is more expensive than the CSG operations themselves [1]. The advantages of our algorithm make it suitable for interactive CSG visualization.

7 CONCLUSIONS AND FUTURE WORK

We have proposed PIC to represent solid objects adaptively. Our method strikes the mid-ground between the distance field representation and point-based modeling. The surface in a cell is represented by two parts: a representative vertex and the in/out classification values of the corner points. The PIC representation has many advantages, such as compact storage, feature-preservation, and adaptive sampling. It is also easy to construct the surface. The conversion of polygonal mesh into the PIC representation is efficient, adaptive, and feature-preserving. The CSG algorithm for PIC objects is unique since cell subdivision and quadric error minimization are combined in the evaluation. We have demonstrated the application of our method for various models.

There are limitations to our method. For example, since the vertex position is explicitly stored and the connectivity needs to be maintained, our representation is unable to perform offsetting efficiently. There is also the possibility of too many levels of subdivision when surface components are too close to separate. In such cases, the PIC representation becomes less effective. Some objects may contain rich information of textures and materials. Our sampling algorithm hasn't consider these yet.

We plan to extend the PIC representation in our future work. To

Table 3: CSG time (in ms.) for the CAD tool model (Figure 11g) and the dolphin model (Figure 11i).

Operation	CAD tool				Dolphin			
	Clay partial contour	Tool partial contour	CSG	Total	Clay partial contour	Tool partial contour	CSG	Total
Union	1	4	41	46	11	5	241	257
Difference	7	2	50	59	7	12	203	222
Difference	11	2	1622	1635	6	3	133	142

avoid deep subdivisions, we are investigating a new encoding approach of using multiple representative vertices in a cell, where each vertex contains its own classification values. We are also considering sampling objects with textures and materials using the extended quadric error metrics [12]. Finally, since the underlying polygonal mesh is only C^0 continuous, we plan to investigate other underlying surfaces, such as the MPU implicits [20], and error metrics other than QEM, to represent curved surfaces.

ACKNOWLEDGMENTS

This work is supported in part by an NSF grant CCR-0306438 and the Army High Performance Computing Research Center (AHPERC) under the auspices of the Department of the Army, Army Research Laboratory (ARL) under Cooperative Agreement number DAAD19-01-2-0014. The content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. The dolphin dataset is courtesy of Flash Fire Designs Company (<http://www.flashfire.com/free3dm.htm>).

REFERENCES

- [1] B. Adams and P. Dutré. Interactive boolean operations on surfel-bounded solids. In *SIGGRAPH Proceedings*, pages 651–656, July 2003.
- [2] M. Alexa, J. Behr, D. Cohen, S. Fleishman, D. Levin, and Claudio T. Silva. Point set surfaces. In *IEEE Visualization*, pages 21–28, October 2001.
- [3] D. Breen, S. Mauch, and R. Whitaker. 3D scan conversion of CSG models into distance volumes. In *ACM/IEEE Symposium on Volume Visualization*, pages 7–14, October 1998.
- [4] P. Brunet and I. Navazo. Solid representation and operation using extended octrees. *ACM Trans. on Graphics*, 9(2):170–197, 1990.
- [5] C. Butcher. Interactive CSG. In *SIGGRAPH Conference Abstracts and Applications*, page 251, 1999.
- [6] I. Carlbom. An algorithm for geometric set operations using cellular subdivision techniques. *IEEE Computer Graphics and Applications*, 7(5):44–55, 1987.
- [7] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH Proceedings*, pages 67–76, August 2001.
- [8] S. F. Frisken and R. N. Perry. Efficient estimation of 3D euclidean distance fields from 2D range images. In *IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics*, pages 81–88, October 2002.
- [9] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields. In *SIGGRAPH Proceedings*, pages 249–254, July 2000.
- [10] M. Garland. *Quadric-Based Polygonal Surface Simplification*. PhD thesis, Carnegie Mellon University, May 1999.
- [11] S. Gibson. Using distance maps for smooth surface representation in sampled volumes. In *IEEE Visualization*, pages 23–30, October 1998.
- [12] H. Hoppe. New quadric metric for simplifying meshes with appearance attributes. In *IEEE Visualization*, pages 59–66, October 1999.
- [13] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of Hermite data. In *SIGGRAPH Proceedings*, pages 339–346, July 2002.
- [14] L. P. Kobbelt, M. Botsch, U. Schwanecke, and H. Seidel. Feature-sensitive surface extraction from volume data. In *SIGGRAPH Proceedings*, pages 57–66, August 2001.
- [15] H. Lee, M. Desbrun, and P. Schröder. Progressive encoding of complex isosurfaces. *ACM Transactions on Graphics*, 22(3):471–476, 2003.
- [16] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3D scanning of large statues. In *SIGGRAPH Proceedings*, pages 131–144, July 2000.
- [17] P. Lindstrom. Out-of-core simplification of large polygonal models. In *SIGGRAPH Proceedings*, pages 259–262, July 2000.
- [18] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH Proceedings*, pages 163–169, July 1987.
- [19] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In *IEEE Visualization*, pages 281–287, October 1994.
- [20] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H. Seidel. Multi-level partition of unity implicits. In *SIGGRAPH Proceedings*, pages 463–470, July 2003.
- [21] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled surfaces. In *IEEE Visualization*, pages 163–170, October 2002.
- [22] M. Pauly, R. Keiser, L. P. Kobbelt, and M. Gross. Shape modeling with point-sampled geometry. In *SIGGRAPH Proceedings*, pages 641–650, July 2003.
- [23] R. N. Perry and S. F. Frisken. Kizamu: A system for sculpting digital characters. In *SIGGRAPH Proceedings*, pages 47–56, August 2001.
- [24] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *SIGGRAPH Proceedings*, pages 335–342, July 2000.
- [25] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C++*. Cambridge University Press, Cambridge, UK, 2001.
- [26] H. Qu, A. Kaufman, R. Shao, and A. Kumar. A framework for sample-based rendering with o-buffers. In *IEEE Visualization*, pages 441–448, October 2003.
- [27] S. Rusinkiewicz and M. Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *SIGGRAPH Proceedings*, pages 343–352, July 2000.
- [28] S. Schaefer and J. Warren. Dual contouring: The secret sauce. Technical Report 02-408, Department of Computer Science, Rice University, 2002.
- [29] M. Sránek and A. Kaufman. Alias-free voxelization of geometric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):251–266, 1999.
- [30] G. Varadhan, S. Krishnan, Y. J. Kim, and D. Manocha. Feature-sensitive subdivision and isosurface reconstruction. In *IEEE Visualization*, pages 99–106, October 2003.

Figure 10: A PIC object created from several CSG operations. (a) A slab model. (b) The space partitioning for the model showing the adaptiveness of our method. The red dots are the representative vertices in each cell. (c) A zoom-in view of the sharp edges.

Figure 11: Some PIC models and CSG results. (a) A cylinder. (b) A gazebo converted from the isosurface of a density volume. (c) A dolphin model. (d) The mouth of the dolphin, rendered from the original polygon mesh. (e) The same area, sampled in 256^3 uniform grid. (f) The same area, sampled by our adaptive sampling algorithm. (g) A CAD tool, created via three CSG operations. (h) The same object with octree subdivision (green lines) and representative vertices shown (red points). (i) The resulting dolphin model after several CSG operations.