Hidden Surface Removal for Accurate Painting-Area Calculation on CAD Models

Lucas Figueiredo, Paulo Ivson, Waldemar Celes
Computer Science Department and
Tecgraf Institute
Pontifical Catholic University of Rio de Janeiro - PUC-Rio
Rio de Janeiro, Brasil
Email: lucascf, psantos, celes@tecgraf.puc-rio.br

***Abstract*—3D CAD models are widely used to improve management of large-scale engineering projects. Examples include Building Information Modeling (BIM) and Oil & Gas industrial plants. Maintaining these facilities is a critical task that often involves anti-corrosive painting of equipment and metallic structures. Existing CAD software estimates the painting area including hidden surfaces that are not actually painted in the field. To improve these computations, we propose an approach based on Adaptively-Sampled Distance Fields (ADFs) exploiting the relationship between object areas and Constructive Solid Geometry (CSG) operations. Tests with synthetic models demonstrate that our technique achieves an accuracy of 99%. In real-world 3D CAD models, we were able to reduce the estimated area by 38% when compared to the naïve calculations. These result in significant cost savings in material provision and workforce required for maintaining these facilities.**

I. Introduction

Computer-Aided Design (CAD) systems are widely used to design, construct, and operate large-scale engineering projects. In the context of civil construction, Build Information Modeling (BIM) [1], [2] is a widely known methodology that employs data-rich 3D CAD models for diverse analysis throughout the life-cycle of a facility. The Oil & Gas industry also takes advantage of these virtual models to improve the quality of design and management of its enterprises [3]–[5].

Facility maintenance is a complex and decisive task during the operation phase of these projects. In particular, offshore oil platforms are subject to strong inclement weather which demands the constant renewal of anti-corrosive coating. Any errors in material and work estimation significantly increases maintenance costs. One of the main efforts to reduce these is to diminish the amount of people working on the platform [6]. Therefore, 3D CAD systems can improve the maintenance planning by efficiently estimating the amount of required materials while avoiding boarding time and expenses.

Wagner et al. [6] and Cho et al. [7] developed systems that utilize 3D CAD models to assist the surface area estimation of structures and equipment. Both used a similar approach to achieve this goal, in which painting zones must first be defined, as visualized in Figure 1, and then the objects inside these zones have their surface area computed and summed-up, obtaining a rough estimation of the overall paint requirements.

In fact, CAD models typically includes surfaces which are not painted in real life. These surfaces are produced by the contact or interpenetration between among objects, as illustrated in Figure 2. In this paper, we reference these surfaces as *hidden surfaces*. To estimate more accurately the amount of painting material required, these hidden surfaces should not be taken into account. However, naïve painting area calculations include such hidden surfaces.

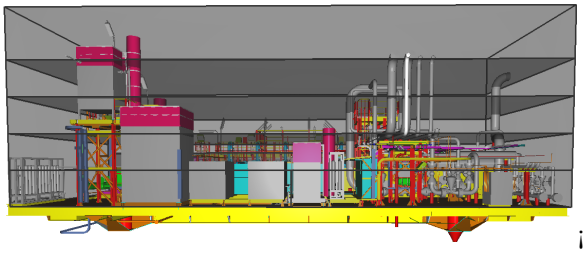


Fig. 1. Example of zones delimited by volumes on the platform.

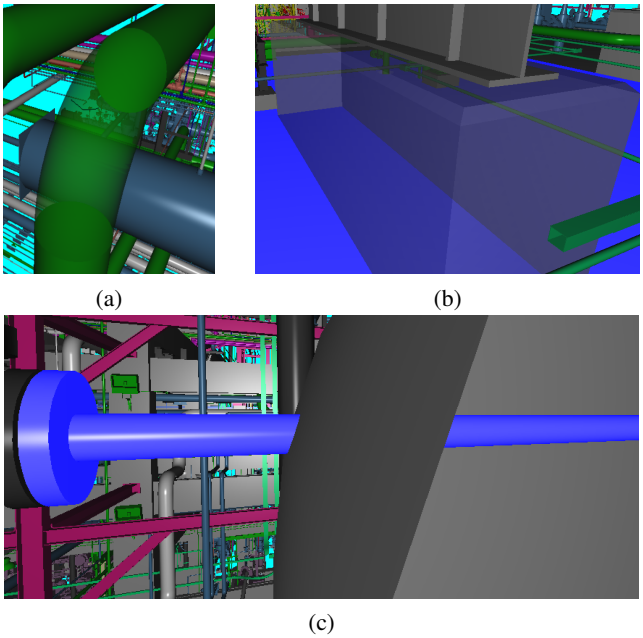


Fig. 2. Examples of hidden surfaces. In (a), cylinders and circular torus caps on contact, in (b), a gray box and blue floor on contact and in (c), a blue cylinder and gray object on interpenetration.

The Boundary Representation (B-Rep) of 3D objects could be used to eliminate hidden surfaces from the paint area computation, since it enables determining which portion of each triangle belongs to the geometries' union boundary. However, this task can be computationally expensive, especially for massive 3D CAD models of oil platforms. In addition, identifying contact can be complicated due to potential numerical inaccuracies while computing close adjacent surfaces.

Distance fields, as opposed to B-Rep, can more adequately identify contact and interpenetration among 3D geometries. This structure efficiently computes the smallest distance between any point and a geometric surface [8]. Consequently, distance fields can be employed for solid modeling [9] through Constructive Solid Geometry (CSG) operations [10].

In this paper, we present an approach, based on adaptively sampled distance fields (ADFs) and constructive solid geometry (CSG), capable of computing the painting area from 3D CAD models, without taking into account their hidden surfaces. Using the developed technique, we were able to obtain a minimum accuracy of 99.16% in synthetic models and a reduction of up to 38% in the total painting surface area of real 3D CAD models, when compared to naïve approaches. These results contribute to significant material provision and workforce required for maintenance.

The remainder of this paper is as follows: Section 2 makes an overview about adaptively sampled distance fields, highlighting its triangles surface reconstruction and CSG application. In Section 3, we present the proposed approach for painting area estimation with ADFs. We show the obtained results in Section 4, demonstrating the effectiveness and accuracy of our solution. Finally, we conclude this paper in Section 5.

II. Adaptively-Sampled Distance Fields

A discrete distance field is a structure that represents an object of interest by storing the minimum distance from points in a grid to the object surface. These distances can be stored with or without sign, distinguishing the object's interior from the exterior. Although it is very common to implement discrete distance fields using regular grids, to capture details on the object geometry a high resolution grid is necessary, consuming an excessive amount of memory [8].

Aiming to produce a better representation that consumes less memory, Frisken et al. [11] introduced the concept of Adaptively-Sampled Distance Fields (ADF). During its construction, cells that contain the represented surface are subdivided until some predefined maximum depth, similar to the three-color octree [12]. However, on the three-color octree, these border cells are always subdivided, while the ADF subdivides only when the error of the reconstructed distance inside the cell is above a given threshold. Thus, ADFs consume fewer memory than regular grids and three-color octrees, and still retains sufficient geometry detail information. The resulting structure needs fewer cells to represent complex geometries and produces deeper subdivisions only where there is enough detail to be preserved.

Adaptively-Sampled Distance Fields can be constructed using a top-down or bottom-up approach [11]. In the first, the distance is computed for the root cell's vertices; then, the cells are recursively subdivided until the reconstruction error becomes smaller than the tolerance threshold. In the second approach, the distance field is computed using a maximum resolution regular grid; then, groups of eight cells are merged if none of these cells have children and the error of reconstructed distances on the merged cell is below the tolerance threshold.

According to Figueiredo et al. [13], an ADF can be classified as "boundary" or "global", depending on which types of cells are subdivided. For the boundary ADF, only the cells that contain the boundary of the object are considered for subdivision. For the global ADF, cells that are in the interior or exterior of the object can also be subdivided. Hence, boundary ADFs only guarantee accurate distance computation in border cells, while global ADFs guarantee accuracy for any point in its domain.

A. Constructive Solid Geometry

Constructive Solid Geometry (CSG) operations [14] can be combined with ADFs to model complex shapes. Perry and Frisken [15], for example, developed a system based on ADFs for sculpting digital characters. The authors implemented a sculpting tool that applied CSG operations between the object and the tool fields in order to reshape the geometry surface.

Using the positive-inside/negative-outside sign convention established in [11], the authors also formalized the CSG operations using minimums and maximums. From these, the union, difference and intersection operations can be obtained by equations 1, 2 and 3, respectively.

$$dist(A \cup B) = max(dist(A), dist(B)) \tag{1}$$

$$dist(A - B) = min(dist(A), -dist(B)) \tag{2}$$

$$dist(A \cap B) = min(dist(A), dist(B)) \tag{3}$$

B. ADF Triangulation

Perry and Frisken [15] proposed an algorithm, based on *SurfaceNets* [16], to extract from ADFs isosurfaces represented by a topologically-consistent mesh with high quality triangles (close to equilateral) and without cracks. The surface is reconstructed from distances sampled in the adaptive grid. The enhanced *SurfaceNets* algorithm has three basic steps:

1) Assign one vertex to each boundary leaf cell, positioned at the cell center.
2) Triangulate vertices of three neighboring cells that share a common edge containing a zero crossing. In order to avoid redundant triangles, only 6 of the 12 possibles edges are considered, as shown in Figure 3.
3) After producing all triangles, move vertices from cell centers towards the surface. Then, to improve the quality of triangles, vertices are moved towards the average of its neighbors' positions.

Triangulation based on adaptive grids may produce two types of cracks, as shown in Figure 4. The first, at the top of

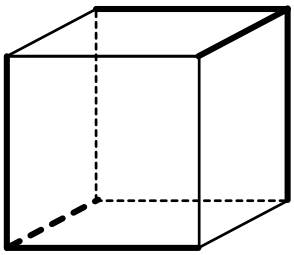


Fig. 3. ADF cell with edges considered on triangulation highlighted: up-right, up-front, front-right, bottom-left, bottom-back and back-left.

the figure, occurs when the cells have different sizes and when the vertices are generated on the faces or edges [17]–[19]. In this case, the interpolated position produced by neighboring cells may differ. As the proposed algorithm generates vertices only at cell centers, this type of crack does not happen.

The second type of crack, at the bottom of Figure 4, occurs when neighboring cells have different numbers of edge crossings. Perry and Frisken [15] used a pre-processing step to avoid this crack, in which the number of zero-crossings for each face of each boundary cell is compared to the total number of zero-crossings in faces shared between the boundary and the face-adjacent neighboring cells. When the zero-crossings differ, the cell is subdivided using the distance information of its face-adjacent neighbors, until they match.

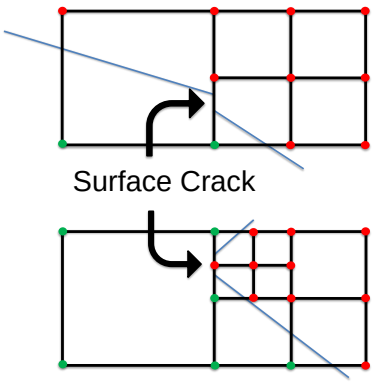


Fig. 4. Types of cracks produced while triangulating adaptive grids. The top crack does not occur using the enhanced *SurfaceNets*. The bottom crack is prevented with a pre-processing step.

III. Painting Area Calculation Using ADFs

3D CAD models have a variety of elements represented as triangle meshes and parametric objects. Hidden surfaces, which are not painted, appear in two flavors. There are hidden surfaces due to simplified modeling operations, such as not removing the caps of piping sections, as illustrated in Figure 2(a). Others are correctly modeled, but hidden by contact after assembling the objects in the plant, as in Figure 2(b).

In order to estimate the painting area not taking into account these hidden surfaces, we propose a two-step routine. The first step consists in computing the ADFs of each mesh. It is worth pointing out that objects already represented by implicit surfaces do not require an ADF construction, since their implicit equations already compute the smallest signed distance to any given point. The second step of our routine aims to estimate the individual painting area for all CAD elements.

A. Computing the ADFs

Occasionally, the meshes found in real models are not watertight, as seen in Figure 5, which is a prerequisite for the signed ADF generation. To ensure that we are able to generate an ADF for all CAD meshes, we first submit them to a pre-processing stage. During this step, holes found on the meshes are covered employing a simple fan triangulation strategy, assuming they are convex. In fact, no non-convex hole was found in our tests.

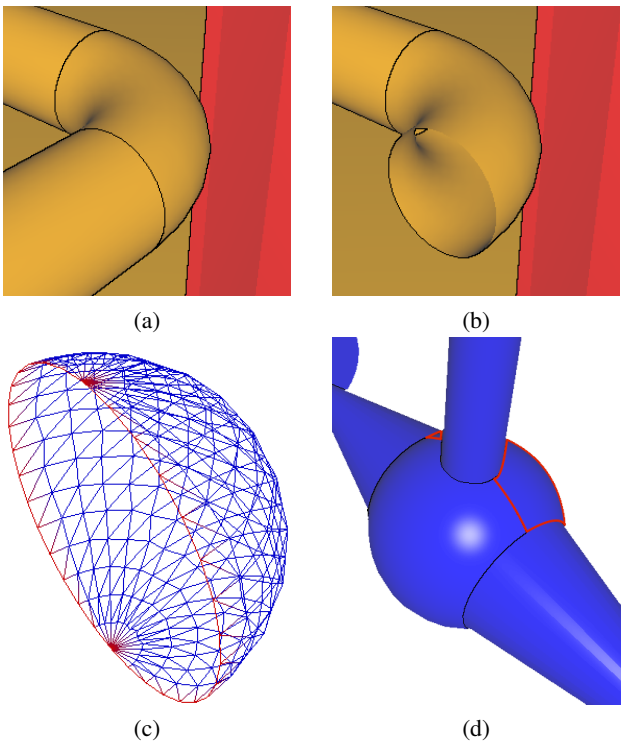


Fig. 5. Non watertight real examples. (a) and (b) shows a circular torus without cap. In (c) we show the opening, in red, of the highlighted semi-sphere in (d).

After the pre-processing stage, inspired by the tiled generation used in [15], we use a hybrid approach to construct an ADF for the mesh. We apply the bottom-up method to obtain an ADF with half of the maximum allowed depth, avoiding an initial high-resolution grid and its associated disadvantages, and proceed to the maximum depth with the top-down approach. We follow the strategy of a signed global ADF, since it provides accurate distance calculation at any point of its domain and, consequently, accurate area estimation.

During ADF construction, the criterion established to decide if a cell needs subsequent division is the reconstruction error

at the cell center and each of its faces' centers. This criterion differs from the one applied in [15], which also uses the reconstruction error in the edges' centers. By disregarding the error at the edge centers we greatly improve the time spent to compute the ADF without significant influence on the later area estimation.

Aiming to accelerate the ADF computation, we take advantage of a kd-tree structure [20] to obtain faster signed distances from points to the represented mesh. The sign of the distance can be determined by the dot product between the normal of the closest triangle and the direction from the point to the triangle. However, when the point is closer to a vertex or edge, this can lead to errors, since the normal at a vertex or edge is not well defined. To address this issue, we used the pseudo-normal concept developed by Aans and Brentzen [21], [22].

B. Painting Area Estimation

An alternative method to estimate the total painting area would be to construct an union ADF using all objects of interest in the scene. However, to obtain such ADF for a significant amount of objects would demand an excessive memory consumption. Furthermore, we want to have access to each object area, since different painting material can be employed to different objects.

We then estimate the painting area of an object relative to the scene, relying on a relation between the object surface areas and boolean operations. Without the loss of generality, consider a scene composed by three objects, illustrated in Figure 6. The area of object A is composed by its painting and its hidden surface areas, so $A_o = A_p + A_{ho}$, where A_o is the total object area, A_p is the painting area and A_{ho} is the object hidden surface area, demonstrated in Figure 7.

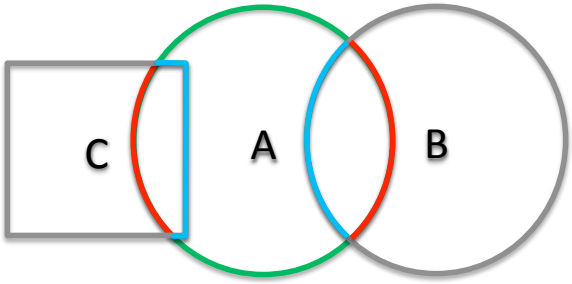


Fig. 6. Scene example in which we aim to compute the painting area of object 'A'.

We observed that the surface area of the difference between A and the scene, A_d, illustrated in Figure 8, is the combination of A painting area, A_p, and the scene's hidden surface area, A_{hs}. Thus the difference area is given by $A_d = A_p + A_{hs}$. Finally, the intersection surface area, A_i, illustrated in Figure 9, is the sum of the hidden surfaces from both A and the scene, that is $A_i = A_{ho} + A_{hs}$.

Thus, the painting area of A can be estimated as:

$$\begin{aligned} A_o + A_d - A_i &= (A_p + A_{ho}) + (A_p + A_{hs}) - (A_{ho} + A_{hs}) \\ &= 2(A_p) \end{aligned}$$

Thefore,

$$A_p = \frac{A_o + A_d - A_i}{2} \tag{4}$$

To compute the difference and intersection surface areas, first we obtain the difference and intersection ADFs. We construct the difference ADF applying the CSG difference operation between A and the scene with N objects, that is $dist(A - Scene) = min(dist(A), -dist(i), -dist(i+1), ..., -dist(N))$. The intersection ADF is built as the union of the intersection between A and each object of the scene, so $(A \cap Scene) = (A \cap i) \cup (A \cap (i+1)) \cup ... \cup (A \cap N)$.

Constructing the difference and intersection ADFs in a simplistic way, testing the object of interest with all other objects in the scene, can lead to a high computational cost. To reduce this cost, we take advantage of Bounding Volume Hierarchy (BVH) [23], a hierarchical tree often used in collision detection that provides the scene objects that may have interference with an object of interest.

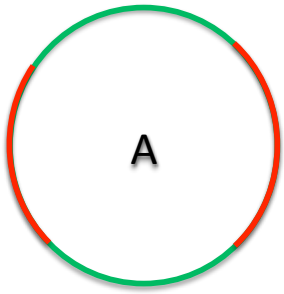


Fig. 7. Object 'A' with green as painting area and red as hidden surface area.

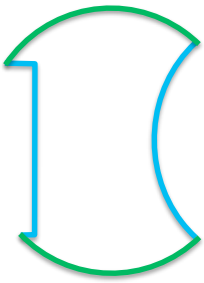


Fig. 8. $A - Scene$ with blue as scene hidden surface area and green as object painting area.

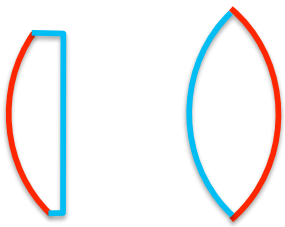


Fig. 9. $A \cap Scene$ with blue as scene hidden surface and red as object hidden surface area.

After computing the difference and intersection ADFs, we modify the enhanced *SurfaceNets* algorithm to only compute the surface area of the reconstructed surface instead storing all produced triangles, thus avoiding an excessive memory consumption. To achieve this, we consolidate the second and third steps of the original reconstruction algorithm into a single step, in which, for each triangle generated in the second step, we move its vertices towards the surface, compute its area and discard it. The areas are then accumulated, obtaining the area of the represented surface. This modification worsens the quality of the triangles, since the vertices are not moved towards the average of its neighbors' positions. However, generating triangles close to equilateral is not a decisive factor for area computation.

C. Handling Contact

The approach described so far does not handle contact, since there is no real intersection between contacting objects. We introduce the concept of *access tolerance* to address this shortcoming. Access tolerance is a constant that represents an envelope on the objects, illustrated in Figure 10, indicating a minimum distance for physical access so that the painting can be performed. When applying this tolerance the intersection operation between two objects is given by:

$$dist(A \cap B) = min(dist(A) + tol, dist(B) + tol) \quad (5)$$

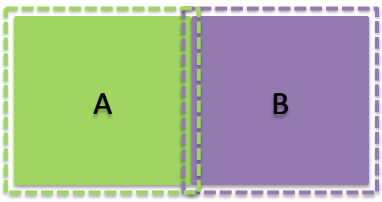


Fig. 10. Scene with access tolerance (dashed) applied to construct the intersection ADF between 'A' and 'B'.

However, as we apply the access tolerance concept to calculate only the intersection area, an error is inserted on the area estimation. The hidden surface areas of both scene and object get slightly bigger, observed in Figure 11. Thus, the intersection area computation introduces an error:

$$A_i = A_{ho} + A_{hs} + error, \quad (6)$$

and the area estimation:

$$A_o + A_d - A_i = 2(A_p) - error$$

Therefore,

$$\frac{A_o + A_d - A_i}{2} = A_p - \frac{error}{2} \quad (7)$$

We attenuate this error while moving the triangles vertices towards the surface during the modified enhanced *SurfaceNets*. Our approach discounts the access tolerance by increasing the step taken in the gradient direction. This process can be visualized in Figure 12. It is worth pointing out that the described approach will always estimate a slightly smaller area, due to the error inserted by the access tolerance.

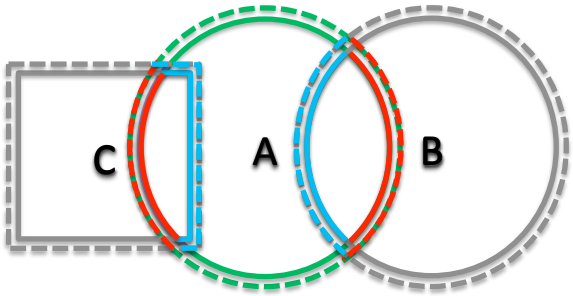


Fig. 11. Access tolerance applied in Figure 6 scene with the intersection area growth highlighted.

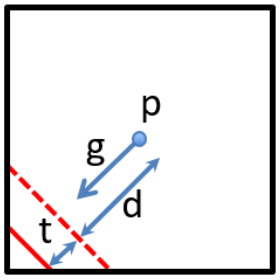


Fig. 12. Vertex p moving towards the surface in the gradient direction g with a step that discounts the access tolerance t.

IV. Results and Discussion

To evaluate our methodology, we ran tests with synthetic controlled models, as seen in Figure 13. We designed them to represent scenes that simulate both contact and interpenetration of hidden surfaces. Setting the maximum ADF depth at 8, reconstruction error threshold at 10^{-4} and access tolerance at 5×10^{-3}, we were able to estimate the total painting area with an accuracy of 99.16% for the sphere scene, 99.4% for the box scene and 99.45% for the cylinder scene.

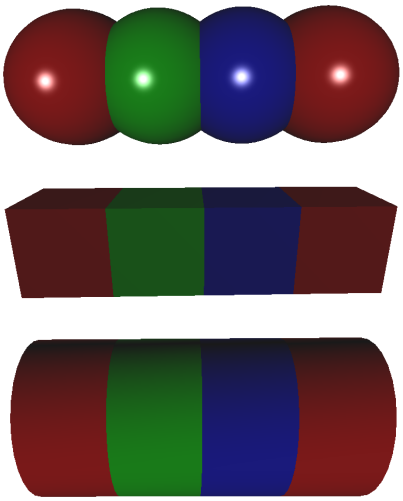


Fig. 13. Three synthetic models: spheres with interpenetration, boxes with contact and cylinders also with contact.

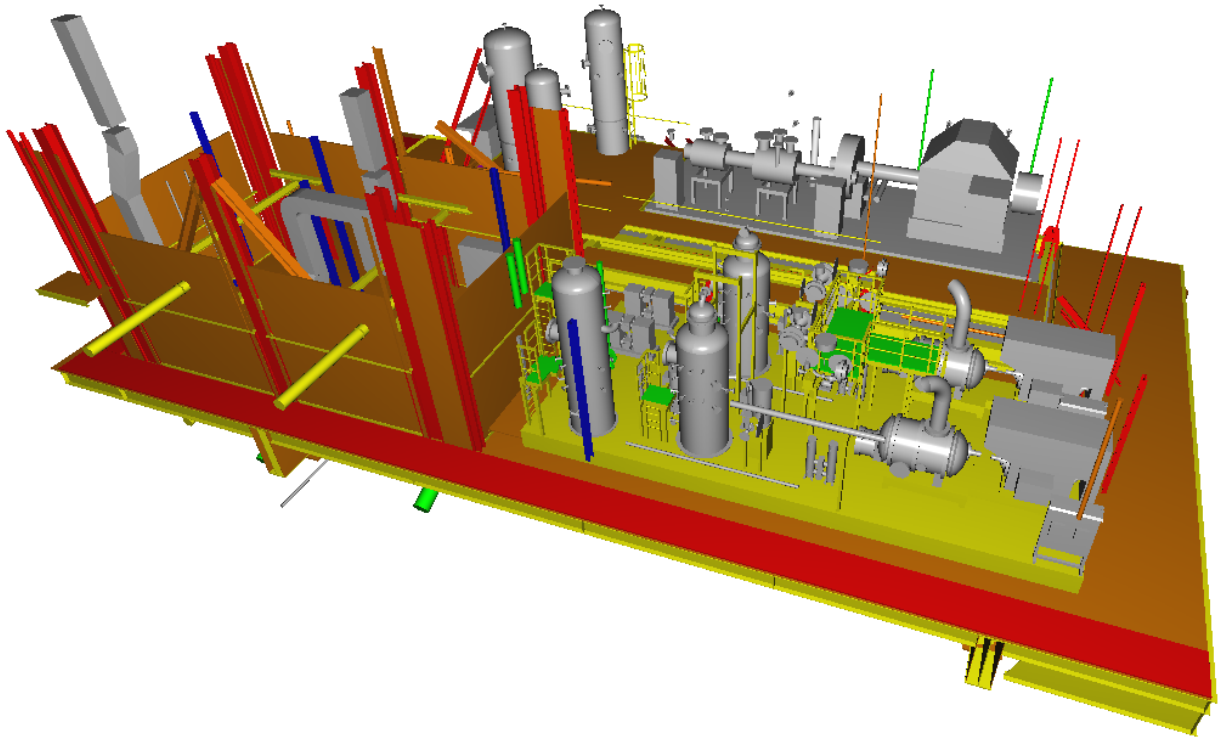
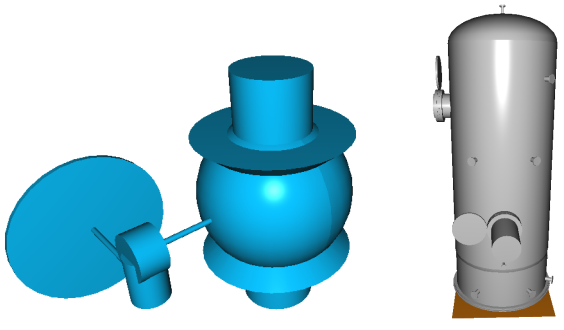


Fig. 14. Model 5, a real delimited zone.

We then tested our painting area calculation method on 5 real models:

1) Valve with 12 geometric objects (Figure 15(a)).
2) Vessel with 73 geometric objects (Figure 15(b)).
3) Piping with 725 geometric objects.
4) Real delimited zone with 1,582 geometric objects.
5) Real delimited zone with 4,450 geometric objects (Figure 14).



(a) Model 1 (b) Model 2

Fig. 15. Another real models examples.

We then compare the painting area estimation with the simplistic approach that computes the entire surface area, as used in [6]. Analyzing the results shown in Table I, we observe a maximum reduction of 38.0% and minimum of 7.9%. We also notice that a smaller percentage reduction can represent a significant amount of area in large models, such as in model 5.

TABLE I
COMPARISON BETWEEN OUR APPROACH AND THE TOTAL SURFACE AREA (TSA), SHOWING THE PERCENTAGE AND ABSOLUTE REDUCTION.

Model	**TSA**[m^2]	**Ours**[m^2]	**Red.**[m^2]	**Red.**[%]
1	2.387	1.579	0.808	33.8
2	190.376	117.954	72.422	38.0
3	29.228	19.303	9.925	33.9
4	396.675	365.245	31.430	7.9
5	7,132.650	6,551.490	581.16	8.1

To perform the painting area estimations we used a computer with 32GB of RAM and an Intel(R) Core(TM) i7-4810MQ processor. The memory consumption and processing times were measured to analyze how our approach scales for different scenes. We noticed that memory consumption is directly influenced by the number of meshes, as observed in Table II, since we compute all their ADFs before proceeding to the area calculation step. It is important to highlight that the ratio between the number of meshes and the number of implicit

surfaces is highly dependent of modeling process and can vary widely from model to model. We also noticed that the time spent depends on the total number of objects, regardless of being meshes or implicit surfaces. This makes sense since all objects must have their difference and intersection computed to estimate the overall painting area.

TABLE II
NUMBER OF IMPLICIT SURFACES AND MESHES, MEMORY CONSUMPTION AND PERFORMANCE OBTAINED DURING THE PAINTING AREA ESTIMATION ON REAL MODELS.

Model	Imp. Surf.	Meshes	Memory[MB]	Time[sec.]
1	10	2	69.62	18.12
2	35	38	689.87	119.50
3	427	298	1,986.56	577.42
4	435	1,147	11,448.32	1,656.38
5	844	3,606	24,094.72	5,655.53

Naturally, complex scenes demands more memory and time to be processed. Nevertheless, when reducing the maximum ADF depth to 7, the time spent for model 5 decreased to 1,371.27 seconds, 75.7% smaller, and consumed 14,069.76 MB, 41.6% less memory. At the same time, the estimated area was 6,578.580m^2, 0.4% greater. Therefore, ADFs with smaller depths use less resources but also produce relatively accurate results. The user must then decide if the lower precision is acceptable for a specific application.

V. CONCLUSION

This paper presented a novel approach for accurate estimation of painting area using 3D CAD models. Our technique successfully eliminates the hidden surfaces that appear during the modeling process to improve the precision of the area calculation. To achieve this, we used ADFs combined with CSG operations, accelerated with kd-tree and BVH structures. The proposed algorithm achives high performance by exploiting the relationship between an object's area, its difference and intersection with the remainder of the scene.

To handle contact between geometries, we introduced the concept of *access tolerance*: a minimum distance for physical access so that painting can be performed. When applying the access tolerance for intersection computation, an error is inserted. However, we attenuate this error by discounting the tolerance while reconstructing the triangle meshes.

The results demonstrated the effectiveness of our approach by estimating the painting area with at least 99.16% accuracy in synthetic models and a reduction of 38.0% or 581.16m^2 in real-world 3D CAD models. This reduction directly translates into significant cost savings in material provision and workforce for facility maintenance.

It is worth pointing out that our approach can be used for other goals such as validating the 3D CAD model during early design stages, prior to construction. To do so, we could use the access tolerance concept to make sure that all surfaces that need painting maintenance can be physically reached once the facility is built. Another possible application of our technique is to estimate the amount of material required to construct a model using a 3D printer.

The high memory consumption when the model contains several meshes deserves attention. As future work, we suggest that the ADF meshes should be constructed on demand, during processing, and discarded when it is guaranteed that the structures will not be used anymore. Also as future work, we suggest the investigation of an adaptive maximum ADF depth, in which the refinement ceases when the difference between the N and $N-1$ levels are below some tolerance, in terms of surface areas.

ACKNOWLEDGMENT

The present work was supported by the CNPq, Conselho Nacional de Desenvolvimento Cientfico e Tecnolgico - Brasil. We also would like to thank Petrobras for providing the 3D CAD models and supporting scientific research in its partnership with the Tecgraf Institute at PUC-Rio.

REFERENCES

[1] C. Eastman, C. Eastman, P. Teicholz, and R. Sacks, *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors*, 2011.
[2] B. Hardin and D. McCool, *BIM and construction management: proven tools, methods, and workflows*, 2015.
[3] Process Industries STEP Consortium, *STEP In The Process Industries: Process Plant Engineering Activity Model*, 1994.
[4] W. Gielingh, "An assessment of the current state of product data technologies," *Computer-Aided Design*, vol. 40, no. 7, pp. 750–759, 2008.
[5] B. C. Kim, Y. Jeon, S. Park, H. Teijgeler, D. Leal, and D. Mun, "Toward standardized exchange of plant 3d cad models using iso 15926," *Computer-Aided Design*, vol. 83, pp. 80–95, 2017.
[6] G. Wagner, R. Delerue, A. Raposo, E. Corseuil, and I. Santos, "Calculating paint area in engineering virtual models," in *XI Simposium on Virtual and Augmented Reality–SVR 2009*. Porto Alegre, Brasil, 2009, pp. 161–167.
[7] D.-Y. Cho, S. Swan, D. Kim, J.-H. Cha, W.-S. Ruy, H.-S. Choi, and T.-S. Kim, "Development of paint area estimation software for ship compartments and structures," *International Journal of Naval Architecture and Ocean Engineering*, vol. 8, no. 2, pp. 198–208, 2016.
[8] M. W. Jones, J. A. Baerentzen, and M. Sramek, "3D distance fields: A survey of techniques and applications," *IEEE Transactions on visualization and Computer Graphics*, vol. 12, no. 4, pp. 581–599, 2006.
[9] S. F. Frisken and R. N. Perry, "Designing with distance fields," in *ACM SIGGRAPH 2006 Courses*. ACM, 2006, pp. 60–66.
[10] S. Ghali, "Constructive solid geometry," *Introduction to Geometric Computing*, pp. 277–283, 2008.
[11] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively sampled distance fields: A general representation of shape for computer graphics," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 249–254.
[12] H. Samet, *The design and analysis of spatial data structures*. Addison-Wesley Reading, MA, 1990, vol. 199.
[13] L. H. de Figueiredo, L. Velho, and J. B. De Oliveira, "Revisiting adaptively sampled distance fields," in *Computer Graphics and Image Processing, 2001 Proceedings of XIV Brazilian Symposium on*. IEEE, 2001, p. 377.
[14] C. M. Hoffmann, "Geometric and solid modeling," 1989.
[15] R. N. Perry and S. F. Frisken, "Kizamu: A system for sculpting digital characters," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001, pp. 47–56.
[16] S. Gibson, "Using distance maps for accurate surface reconstruction in sampled volumes," in *1998 Symposium on Volume Visualization*, 1998, pp. 23–30.
[17] J. Bloomenthal, "Polygonization of implicit surfaces," *Computer Aided Geometric Design*, vol. 5, no. 4, pp. 341–355, 1988.

[18] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM siggraph computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169.
[19] D. B. Karron, J. Cox, and B. Mishra, "New findings from the SpiderWeb algorithm: toward a digital morse theory," in *Visualization in Biomedical Computing 1994*. International Society for Optics and Photonics, 1994, pp. 643–657.
[20] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[21] H. Aanæs and J. A. Bærentzen, "Pseudo–Normals for Signed Distance Computation," in *Vision, modeling, and visualization 2003, Munich, Germany*, 2003.
[22] J. A. Bærentzen and H. Aanaes, "Signed distance computation using the angle weighted pseudonormal," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 3, pp. 243–253, 2005.
[23] C. Ericson, *Real-time collision detection*. CRC Press, 2004.