An Accurate Method for Voxelizing Polygon Meshes

Jian Huang[1], Roni Yagel[1,2], Vassily Filippov[1], and Yair Kurzion[1]

[1]Department of Computer and Information Science, The Ohio State University, Columbus OH
[2]BioMediCom Ltd., Jerusalem, Israel

Abstract

The process of generating discrete surfaces in a volumetric representation, termed voxelization, is confronted with topological considerations as well as accuracy and efficiency requirements. We introduce a new method for voxelizing planar objects which, unlike existing methods, provides topological conformity through geometric measures. We extend our approach to provide, for the first time, an accurate and coherent method for voxelizing polygon meshes. This method eliminates common voxelization artifacts at edges and vertices. We prove the method's topological attributes and report performance of our implementation. Finally, we demonstrate that this approach forms a basis for a new set of voxelization algorithms by voxelizing an example cubic object.

1. INTRODUCTION

Polygons are fundamental primitives for 3-D surface graphics because they approximate arbitrary surfaces as a mesh of polygonal patches. Polygons also serve as the basis for a wide variety of additional capabilities, such as clipping planes and cutting planes.

In volumetric graphics, one has to voxelize polygon meshes for graphical purposes [4][11] and for intermixing polygonal objects with sampled data [9]. Planar objects are also used for other operations, such as extracting an oblique cut-plane. Some block operations (bitblt) are performed on a sub-volume region bounded by multiple clipping planes. The identification of the voxels belonging to a desired region also involves voxelization of the planar boundaries of the region.

A voxelization algorithm for any geometric object should meet several criteria. It must be efficient and accurate and it must generate discrete surfaces that are thick enough so as not to allow rays to penetrate (the *separability* requirement). On the other hand, discrete surfaces should contain only those voxels indispensable for separability (the *minimality* requirement).

Early work in scan-converting planar objects was only concerned with filling in 2D polygons or 2D projections of 3D polygons. A 2D polygon is defined as a closed sequence of line segments (edges). Scan-line algorithms for converting a polygon are widely known (e.g., [10][3]). Voxelization algorithms for planar polygons have been proposed in [1][2][5][6][7]. These algorithms extend the widely known 2D scan-line algorithm, where at each scan-line the third dimension is also interpolated. By selecting a correct orientation, the algorithm employs numerical considerations to guarantee that no gaps appear in the third dimension.

Existing algorithms do not provide a mechanism to guarantee *both* separability and minimality especially for polygon meshes where the edge and vertex regions pose some difficulties. In addition, these techniques do not extend naturally to other types of objects. We present a method that uses a distance criterion to voxelize. This criterion is a function of the normal vector to the surface and provides numerical accuracy as well as topological correctness for polygons *and* polymeshes. We show that this technique can be extended to other types of surfaces.

We introduce discrete spaces in Section 2 and Section 3. In Section 4 we explain our method for voxelizing lines and planes and prove its correctness. In Section 5, we present our extension to polymeshes and report on our implementation in Section 6. In Section 7 we show that our method can be extended to handle other types of objects. We discuss our results and conclude in Section 8.

2. 3D DISCRETE SPACES

A 3D discrete space Z^3 is a set of integer grid points in a 3D Euclidean space denoted by Σ. A 3D grid point is a zero dimensional object defined by its Cartesian coordinate *(x,y,z)*. The *Voronoi neighborhood of grid point p* is the set of all points in the Euclidean space that are closer to *p* than to any other grid point. The Voronoi neighborhood of a 3D grid point is a unit cube around it, known also as a *voxel*.

The aggregate of all voxels is a tessellation of 3D Euclidean space. A voxel's value is mapped into the set *{0,1}*: voxels assigned the value "1" are called "black" or "non-empty" voxels, while those assigned the value "0" are called "white" or "empty" voxels.

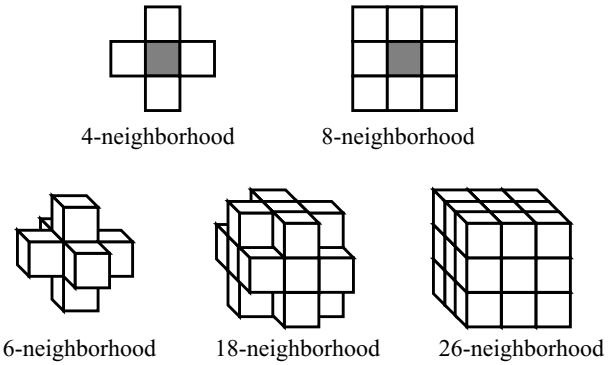


Figure 1. The set of 2D pixels that are N-adjacent to the dark pixel (top) where $N \in \{4, 8\}$. The set of 3D voxels that are N-adjacent to the voxel at the center (bottom) where $N \in \{6, 18, 26\}$.

Two voxels are *26-adjacent* if they share a vertex or an edge or a face (see Figure 1). Every voxel has 26 such neighbors. Eight of

the 26 neighbors share a vertex (corner) with the center voxel, twelve of the 26 neighbors share an edge with the center voxel, and six of the 26 neighbors share a face with the center voxel. Accordingly, the face-sharing voxels are defined as *6-adjacent*, and the voxels that are both edge-sharing and face-sharing are defined as *18-adjacent* (see Figure 1).

The prefix '*N*' is used to denote the adjacency relation, where $N \in \{6, 18, 26\}$. An *N-path* is a sequence of black voxels such that consecutive pairs are N-adjacent. Two black voxels are said to be *N-connected in* Σ if there exists a connecting N-path consisting only of black voxels in *W*. It is easy to show that "connectedness in Σ" is an equivalence relation. The equivalence classes under this relation are the *connected components*. A *(closed) N-curve* is an N-path *P* that either contains a single voxel or each voxel in *P* has exactly two N-adjacent voxels also in *P*. An o*pen N-curve* is an N-curve with two exceptions called *endpoints*, each of which has only one N-adjacent voxel in *P*.

In continuous space, it is impossible to pass from the region enclosed by a curve to the region outside the curve without crossing the curve itself. In discrete space, however, the opposite is possible. Figure 2 shows a 2D discrete 8-connected curve penetrating through another 8-curve without meeting it. In order to avoid this discrepancy between continuous and discrete space, the convention is to define opposite types of connectivity for white and black sets [8]. Opposite types in 2D space are 4 and 8, while in 3D space 6 is "opposite" to 26 and 18. In 3D, however, the situation is much more complex because the connectivity of a surface does not fully characterize its topology. The failure of the connectivity to properly characterize a surface is the main motivation behind the notion of *separability.*

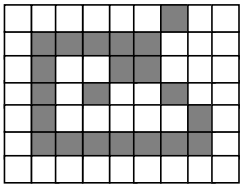


Figure 2. An 8-connected curve penetrating through another 8-curve.

3. TOPOLOGICAL AND GEOMETRICAL CONSIDERATIONS

Let *S* be a C^0 continuous surface in R^3, we denote by $\tilde{S}$, the discrete representation of *S*. $\tilde{S}$ is a set of black ("1") voxels generated by some arbitrary computation. There are three major requirements that $\tilde{S}$ should fulfill. First, we would like to preserve the analogy between continuous and discrete space by assuring that $\tilde{S}$ is not penetrable since *S* is C^0 continuous. This requirement is called *separability* and it is formally defined and characterized in this section, based on [2]. In addition, $\tilde{S}$ should be the most *accurate* discrete representation of *S* according to some appropriate error metric. Finally, the voxelization should be *minimal*, that is, it should not contain voxels that, if removed, make no difference in terms of separability. Next, we describe each one of these requirements, namely separability, minimality, and accuracy, in more detail.

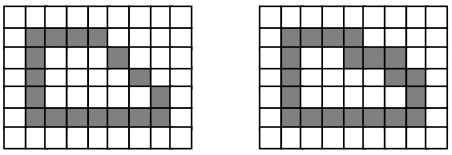


Figure 3. A 4-separating curve (left) and an 8-separating curve (right).

Let *A, B* and *C* be three disjoint sets of voxels. *A* is said to *N-separate B* and *C* if any *N*-path Π between a voxel in *B* and a voxel in *C* meets *A* (i.e, $\Pi \cap A \neq \varnothing$). (See Figure 3).

Let $S \in R^3$ be a C^0 continuous surface such that $R^3 - S$ has exactly two connected components, *I* and *O*. Let $\tilde{S}$ be a voxelization of *S*. If $\Sigma - \tilde{S}$ have exactly two *N*-connected components $\tilde{I}$ and $\tilde{O}$, then $\tilde{S}$ is said to be *N-separating with respect to S*, or in short, *N-separating*[1]. This property is topological and does not reflect the accuracy for the quantization of $\tilde{S}$ to the continuous surface *S*. In practice, to avoid dealing with the special cases for surface boundaries, we regard *N*-separability as a local surface property, for example, in a $3 \times 3 \times 3$ neighborhood.

A voxel belonging to an *N*-separating surface is called an *N-simple voxel* if deleting it will not affect the surface separability. An N-separating surface is called *N-minimal* if it does not contain any *N*-simple voxels, as shown in Figure 4.

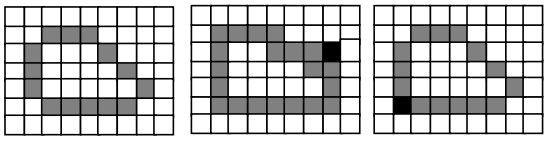


Figure 4. Examples of a 4-minimal curve (left), 8-simple point (center), and a 4-simple point (right).

Denote by S^d the set of all voxels that meet *S*. That is, if we define the extent of the unit voxels as having closed boundaries, each voxel that covers a portion of *S* is in S^d. Denote by I^d and O^d the set of voxels that are fully contained in *I* and *O*, respectively. One can show that S^d N-separates I^d and O^d, for any N. However, S^d is not always N-minimal, for any N. That is, although S^d is 'close' to the continuous surface, it contains too many voxels. By removing all *N*-simple voxels from S^d we can generate a surface $\tilde{S} \subseteq S^d$ that is both *N*-separating and *N*-minimal.

1.This simple definition assumes that the resolution of the discretization is enough to have discrete points representing both sub spaces I and O. A more robust definition will include treatment of low resolution discretization where, for example, a circle is discretized into one black point.

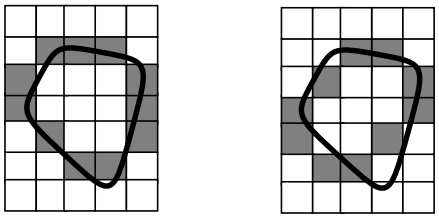


Figure 5. Two 8-separating curves with different accuracies; the pixelization in the left better approximates the continuous curve than the one on the right (notice the length of the curve that is 'covered' by each pixelization).

Finally, in addition to separability and minimality, $\tilde{S}$ is required to closely approximate *S*, for example, although the two forms of pixelization of the same curve shown in Figure 5 are both 8-separating, the left one better approximates the curve. The mechanisms needed to assess the accuracy of $\tilde{S}$ are not discussed in this paper.

4. VOXELIZATION OF LINES AND PLANES

Given a 3D polygon, an integer *N*, where $N \in \{6, 26\}$, and a uniform grid in 3D space, one would like to construct a discrete N-separating and *N*-minimal representation of the polygon. Our work is based on the observation that separability is a topological manifestation of thickness, which we use to control separability.

Given a plane *S*, construct two parallel planes such that *S* lies between these planes and is parallel to both of them, as shown in Figure 6.

Given the equation of plane *S*:

$$A_p X + B_p Y + C_p Z + D_p = 0 \quad (1)$$

the equations for the planes *G* and *H* will have the following form:

$$A_p X + B_p Y + C_p Z + D_p \pm t = 0 \quad (2)$$

We assume that the normal of the polygon is normalized.

$$\sqrt{A_p^{\,2} + B_p^{\,2} + C_p^{\,2}} = 1 \quad (3)$$

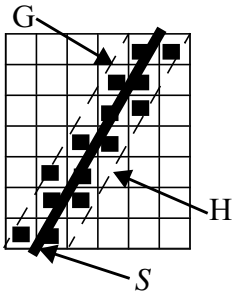


Figure 6. The plane S (solid line) and the two planes parallel to S at distance t, G and H (dashed lines). All voxels between G and H are included in the discrete representation of S.

A voxel centered at grid point (x, y, z) will be incorporated into $\tilde{S}$ if the point (x, y, z) lies between the planes *G* and *H*.

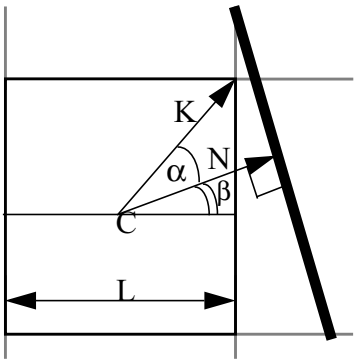
$$-t \le A_p x + B_p y + C_p z + D_p \le t \quad (4)$$

4.1 PIXELIZING A 2D LINE

In the 2D case, we must deal with straight lines instead of planes, the conditions in the problem will require 4- or 8-separability and minimality. Furthermore, we will have *lines G* and *H* instead of planes *G* and *H*, and for a 2D point (x, y) Equation (4) will take the form:

$$-t \le A_p x + B_p y + D_p \le t \quad (5)$$

Let's derive the distances between lines *G* and *H* corresponding to 8-separating or 4-separating lines. In the equations below, *L* is the size of the grid square (pixel), as shown in Figure 7. *C* is the pixel's center and *N* is the normal to the line through the point *C*.



$K = (L/2)\sqrt{2}$

$t_8 = K\cos\alpha = (L/2)\sqrt{2}\cos\alpha$

$t_4 = (L/2)\cos\beta$

Figure 7. The 2D case: L is the pixel's width, C denotes its center, and K is the pixel's diagonal. N is the normal to the line through the pixel's center. α is the angle between N and K.

For generating a 4-separating line we use $t = K\cos\alpha = (L/2)\sqrt{2}\cos\alpha$, denoted by t_8. For generating 8-separating lines we define $t_4 = (L/2)\cos\beta$.

Theorem 1.

The set of pixels $\tilde{S} = \{(x, y) | -t_8 \le A_p x + B_p y + D_p \le t_8\}$ is an 8-separating and 8-minimal representation of the line *S* defined by A_p, B_p and D_p.

Proof:

Separability:

We prove by contradiction. For a line to have an 8-hole, there must be a pixel *P* that is not included in $\tilde{S}$ but is intersected by line. *P* must also have two 8-neighbors, *Q* and *R*, which are not included in $\tilde{S}$ and are on opposite sides of line *S*, as shown in Figure 11.

We denote the center of P by *C*. In our case, points *V* and *W* are on

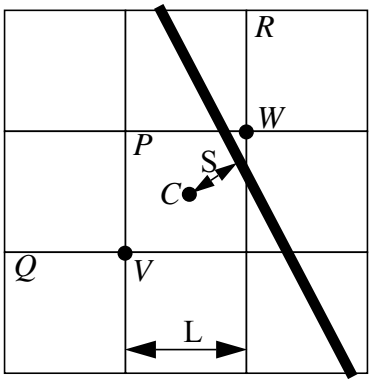


Figure 8. Proof of the 2D case for 8-separability and 8-minimality.

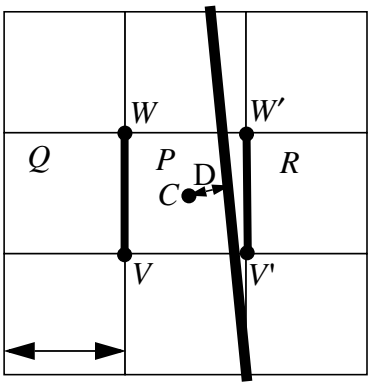


Figure 9. Proof of the 2D case for 4-separability and 4-minimality.

opposite sides of line *S* and *C* is on the same side of the line as point V. The distance between *C* and point *W* is $(L/2)\sqrt{2}$. Since the line only intersects *P*, but not *Q* or *R*, the distance between the center of the pixel and the line is $D \le (L/2)\sqrt{2}\cos\alpha$, where α is the angle between the normal to the line from *C* and the diagonal VW. If $D > (L/2)\sqrt{2}\cos\alpha$, then this line cannot intersect pixel P. On the other hand, $t_8 = (L/2)\sqrt{2}\cos\alpha$, so $S \le t_8$. Therefore, pixel *P* should have been included in the representation of the line, contradicting our assumption. Therefore, we conclude that $\tilde{S}$ is 8-separating.

Minimality:

Suppose there is a pixel in the representation of the line that can be removed without introducing an 8-hole. For this pixel, the condition $D \le t_8$ must hold, therefore the distance from *C* to the line is $D \le (L/2)\sqrt{2}\cos\alpha$. It follows that there is at least one vertex of this pixel on each of the two sides of the line. Removing this pixel from the representation of the line will introduce an 8-hole, and therefore affects the separability. Thus our representation of the line is 8-minimal.

Let's now use the value of t_4 in place of *t* in equation

$-t \le A_p X + B_p Y + D_p \le t$.

Theorem 2.

The set of pixels $\tilde{S} = \{(x, y) | -t_4 \le A_p x + B_p y + D_p \le t_4\}$ is a 4-separating and 4-minimal representation of the line *S* defined by A_p, B_p and D_p.

Proof:

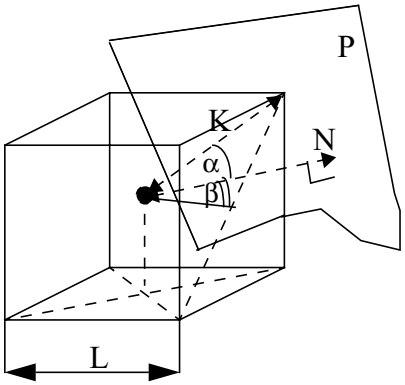
Separability:

We prove by contradiction. For a line to have an 4-hole, there must be a pixel *P* that is not included in $\tilde{S}$ but is intersected by this line. *P* must also have two 4-neighbors *Q* and *R* which are not included in $\tilde{S}$ and are on opposite sides of line *S*. *C* is closer to the line than the centers of *Q* and *R*.

Without loss of generality, let's consider the case in Figure 9. In this case, edges *VW* and *V'W'* are on opposite sides of the line. The distance between *C* and the edge *V'W'* is *(L/2)*. Since *C* is closer to the line than the center of either Q or R, the distance between *C* and the line is $D \le (L/2)\cos\beta$, where β is the angle between *VV'* and the normal to the line. $t_4 = (L/2)\cos\beta$, so $D \le t_4$. In this case, pixel *P* should have been included in the representation of the line, contradicting our assumption. Therefore, we conclude that $\tilde{S}$ is 4-separating.

Minimality:

4-minimality is similarly proven as Theorem 1.



$K = (L/2)\sqrt{3}$

$t_{26} = K\cos\alpha = (L/2)\sqrt{3}\cos\alpha$

$t_6 = (L/2)\cos\beta$

Figure 10. The 3D case (extended from Figure 7): L is the voxel's width, and K is the voxel's diagonal. N is the normal to the plane through the voxel's center, C . α is the angle between N and K. β is the angle between N and the normal to the face of the voxel.

4.2 VOXELIZING A 3D PLANE

In 3D, the decision as whether to include a given voxel in the discrete representation $\tilde{S}$ of the polygon *S* is handled in a similar way as in the 2D case. If the center of a voxel is between planes G and H, this voxel is included in $\tilde{S}$. In 3D, we are concerned with build-

ing either 6- or 26-separating representations of planar objects.

We use the same reasoning as in the 2D case to come up with the distance between planes G and H that gives $\tilde{S}$ the desired separability. For generating a 26-separating surface we use $t = K\cos\alpha = (L/2)\sqrt{3}\cos\alpha$, which we denote by t_{26}. For generating 6-separating line we define $t_6 = (L/2)\cos\beta$.

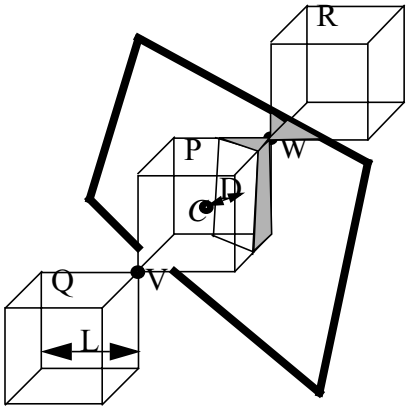


Figure 11. Proof of the 3D case for 26-separability and 26-minimality.

Theorem 3.
The set of voxels:

$\tilde{S} = \{(x, y, z) | -t_{26} \le A_p x + B_p y + C_p z + D_p \le t_{26}\}$ is a 26-separating and 26-minimal representation of the plane S defined by A_p, B_p, C_p and D_p.

Proof:

Separability:

We prove by contradiction. For a plane to have a 26-hole, there must be a voxel that is in $\tilde{S}$ but has at least one 26 neighbor on each of the two opposite sides of the plane, and which are not included in $\tilde{S}$.

Let P be such a voxel, and Q and R be its neighbors, which are not included in $\tilde{S}$, as shown in Figure 11. Vertices V and W (of voxel P) are on different sides of the plane. The distance between C and point V is $(L/2)\sqrt{3}$. Since the plane intersects P, but not Q or R, the distance between C and the plane is $D \le (L/2)\sqrt{3}\cos\alpha$, where α is the angle between VW and the normal to the plane. $t_{26} = (L/2)\sqrt{3}\cos\alpha$, so $D \le t_{26}$. In this case, voxel P should have been included in the representation of the plane. Contradiction. So, our initial assumption that there exists such a voxel, P, and the plane has a 26-hole is false. Therefore, the plane is 26-separating.

Minimality:

Suppose there is a voxel in the representation of the plane which can be removed without introducing a 26-hole. For this voxel, condition $D \le t_{26}$ must hold, so the distance from C_p to the plane is $D \le (L/2)\sqrt{3}\cos\alpha$. There are at least two vertices of this voxel lying on the two opposite sides of the plane. In this case, removing this voxel from the representation of the plane will introduce a 26-hole. Thus, our representation of the plane is 26-minimal.

Let us now use the value of t_6 in place of t in equation (3).

The following theorem is proven in a similar manner:

Theorem 4.
The set of voxels:

$\tilde{S} = \{(x, y, z) | -t_{26} \le A_p x + B_p y + C_p z + D_p \le t_6\}$ is a 6-separating and 6-minimal representation of the plane S defined by A_p, B_p, C_p and D_p.

5. VOXELIZING POLYGON MESHES

In Section 4, we dealt with infinite planes only. In practice, however, what is needed are finite planes, i.e. polygons and polygon meshes. Thus, we have to further consider border situations, i.e. edges and vertices.

The voxelization of polygon meshes is a long standing problem. First, we must choose the same set of voxels for voxelizing the common edge between two neighboring polygons. In addition, the final outcome has to maintain the separability and minimality attributes. Existing voxelization algorithms fail to provide these features.

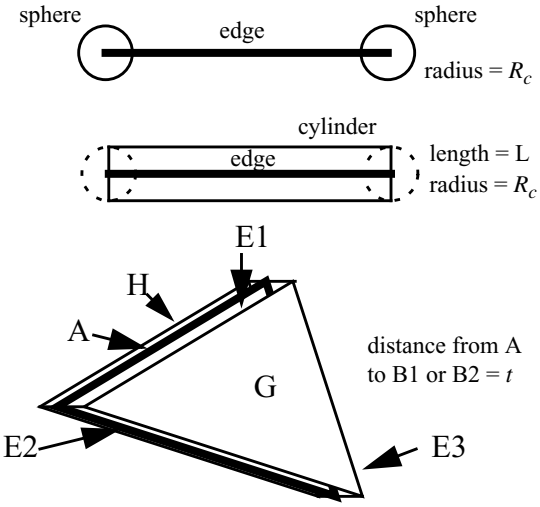


Figure 12. The set of voxels comprising a polygon. (1) (top) the voxels lying inside the spheres that surround the vertices (2) (center) those that lie inside the cylinder that surrounds the vertices, and (3) (bottom) those that lie between the planes that bound the plane.

We divide the discrete representation $\tilde{S}$ of polygon S into three

overlapping sets of voxels, $\tilde{S}_v$, $\tilde{S}_e$, and $\tilde{S}_b$. After having generated these three sets, we combine them into the final representation of the polygon $\tilde{S}$.

Let t denote the desired connectivity distance, either t_6 or t_{26}, as specified above. Define R_c to be $L/2$ for 6-separability, and $(\sqrt{3}/2)L$ for 26-separability.

Assume the polygon S has n edges and n vertices. We consider three sets of voxels:

1. The set of voxels $\tilde{S}_v$ for the representation of S's vertices. For each vertex of S, we define a bounding *sphere* of radius R_c as shown in Figure 12 (top). The center of the bounding sphere is at the corresponding vertex. All the voxels whose voxel centers fall inside any one of the n bounding spheres belong to $\tilde{S}_v$.
2. The set of voxels $\tilde{S}_e$ for the representation of S's edges. For each edge of S, we define a regular bounding *cylinder* of radius R_c and length L, where L is the length of the edge, as shown in Figure 12 (center). All the voxels whose voxel centers fall inside any one of the n bounding cylinders belong to $\tilde{S}_e$.
3. The set of voxels $\tilde{S}_b$ for the representation of the body of S. For the body of S, we define a set of bounding *planes*. First, we define two bounding planes, G and H, as defined in Section 4. Second, for each edge of the polygon, we define a plane, E_i $(i=1,...,n)$. E_i passes through edge i and is perpendicular to S, G, H and the set of E_i's form a closed region in 3D space, as shown in Figure 12 (bottom). All the voxels with their voxel centers falling inside this closed region form $\tilde{S}_b$.

The final discrete representation $\tilde{S}$ of the polygon S is a union of $\tilde{S}_v$, $\tilde{S}_e$ and $\tilde{S}_b$.

We claim that combining the $\tilde{S}$ representations, for all polygons S in a polymesh, forms an N-separating surface. The proof of this claim is an extension of the proofs of Theorem 3 and Theorem 4, given in Section 4.

The reason we define both t and R_c here is to ensure separability. If we only define t, and use t instead of R_c for the first two cases in Figure 12, then we cannot guarantee that the edge is voxelized correctly. Let us look at an example in 2D, assuming we are voxelizing 4-separating polyline, as shown in Figure 13:

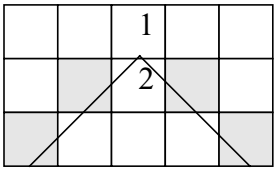


Figure 13. Voxelized lines which meet in a way that leaves a hole at the intersection point unless the value of R_c is used.

Assume both lines have a slope of 45 degrees, and the distance of both lines to the center of the shaded squares is $\frac{\sqrt{2}}{4}L - \varepsilon$. Therefore, the intersection point of the two lines must fall in square 1. The distance from the center of square 1 to the intersection point is $\frac{L}{2} - \sqrt{2}\varepsilon$. It is possible that this distance is larger than the value of t_4 for both lines. Then, both square 1 and 2 are not included in the discrete model, and we have a hole. Using 'R_c' instead of 't' shall solve that problem.

6. RESULTS

We implemented our algorithm and voxelized four polygon mesh models of mechanical parts: *bowl, brevi, connector* and *pump*. All four models were voxelized into a $512 \times 512 \times 512$ resolution volume. Performance results are quoted for a SGI workstation, with 195MHz R10000, processor and 640 MB memory. Table 1 describes the experiment setup. Table 2 and Table 2 provide results of voxelizing the objects using 6-separability and 26-separability, respectively.

Table 1: Experiment Setup

Part Name	# of Facets	Avg Tri Size	Avg T_Size in VS	Physical Size (x, y, z)
Connector	242	0.4	1371.6	406, 118, 171
Brevi	1812	3.7	102.4	200, 183, 503
Bowl	328	21.0	199.4	512, 250, 490
Pump	3030	7.7	72.9	506, 368, 435

Table 2: Experiment Results for 6-separability

Part Name	# Voxels (1000s)	Total Time (sec)	# Voxel per ms	$\tilde{S}_v$	$\tilde{S}_e$ in (1000s)	$\tilde{S}_b$ in (1000s)
Contr	220	4.7	46.4	63	16	204
Brevi	466	12.7	36.7	524	75	391
Bowl	839	33.3	25.1	112	46	794
Pump	1483	67.8	21.8	983	163	1320

Table 3: Experiment Results for 26-separability

Part Name	# Voxels (1000s)	Total Time (sec)	#Voxel per ms	$\tilde{S}_v$	$\tilde{S}_e$ in (1000s)	$\tilde{S}_b$ in (1000s)
Contr	270	6.5	41.1	309	44	226
Brevi	636	18.2	35.0	2290	187	447
Bowl	1115	44.7	24.9	407	119	995
Pump	2019	95.4	21.2	3729	418	1597

In Table 1, the column 'Avg T_Size in VS', stands for the average triangle size in terms of voxel space unit area, which is computed as $\frac{\text{average triangle size}}{\text{voxel size}^2}$. In Tables 2 and 3, the column '*Number of Voxels*' shows the total number of voxels in the discrete representation of the model. The '*Total Time*' column shows the time used in the process of voxelizing the polygon mesh. The column '*#Voxel per ms*' shows the average number of voxels produced by the algorithm per millisecond. The last three columns show the number of voxels in each of the three groups $\tilde{S}_v$, $\tilde{S}_e$ and $\tilde{S}_b$ described in Section 5.

The time consumed by this voxelization algorithm depends on many factors, including implementation dependencies. We use a linked list to store the volume model. When voxelizing different model at a constant resolution, the complexity of the polymesh plays an important part in determining the execution time. The more complicated the data set, the more voxels in the final discrete volume model, thus, the more linked list elements to be initialized. This linked list data structure also explains why with the increasing 'Number of Voxels', we get decreasing '#Voxel per ms'(VPM). The VPM reflects the relative efficiency in the voxelization process. It takes on values in a rather small range, as shown in the tables above. Obviously, an increasing number of voxels requires longer linked lists and thus a higher maintenance cost on these lists.

The average size of triangles in the polymesh, in terms of unit area in voxel space, also plays an important role. In our current implementation, the procedure to extract necessary triangle information takes a fixed amount of time, thus, it is more efficient to voxelize larger polygons in terms of cost per voxel. 'Connector' has a significantly larger average polygon size in terms of voxel space unit area, while 'Pump' has the smallest such size. This partly explains the violent contrast between 'VPM' values.

We notice that 26-separating voxelization produces many more Vertex Voxels $\tilde{S}_v$ and Edge Voxels $\tilde{S}_e$ than 6-separating voxelization, while the difference in the number of Body Voxels $\tilde{S}_b$ is smaller. The reason is that in our implementation, we first try to classify voxels as Vertex Voxels, then as Edge Voxels and finally as Body Voxels. Another factor that affects the final voxel count in the discrete models is the alignment of the polygon-based models to the discrete grid.

Note that our timing results are based on a non optimized scan conversion implementation. An efficient incremental algorithm needs to be devised to achieve even faster voxelization.

We used a splatting-based renderer to create our images [12]. Visually, the images of 6- and 26- separating models are virtually indistinguishable, as shown in Figure 14. Since 6-separating voxelization takes less time (see Table 2) for display purposes, we choose 6-separating objects for our algorithm. The images of the 6-separating models of *Bowl, Brevi*, and *Connector* are shown in Figure 15.

7. EXTENDING THE METHOD TO OTHER OBJECT TYPES

So far, we have discussed how to voxelize polygon meshes. Traditionally, objects are first converted to polygon meshes (as done in surface graphics) and are then voxelized. We observe that our voxelization method includes voxels in the discrete representation $\tilde{S}$ of the surface *S,* if the voxel's distance to the surface is smaller than some value *t*. Obviously, if one can scan a subset of the voxel space in the vicinity of the surface *S*, efficiently measure the distance to surface *S* for each voxel, then one could generate a 6- or 26-separating representation of *S*. Furthermore, the ability to directly voxelize geometric objects without having to first convert them to polygon meshes, is most desired for efficiency and accuracy reasons.

We have implemented this approach (in a non optimized manner) for the cases of cubic objects such as spheres, cylinders and the function $x^2 + ky + kz = 0$, and show the results in Figure 15. For spheres and cylinders, which have a regular shape, measuring the distance from each voxel to the surface is straightforward. However, a more general way to measure such distances is to use the Lagrangian Method, as follows.

The problem of finding the distance between a point and a surface could be more clearly defined as: given a random point in 3D space (x_0, y_0, z_0), one wants find the point on a surface defined by $f(x, y, z) = 0$ minimizing:

$$p(x, y, z) = (x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 \quad .$$

This problem can be readily solved by the Lagrangian Method. We note that solutions for this problem by the Lagrangian Method may require solving high order polynomial equations which don't have analytical solutions. For surfaces in the form $x^2 + ky + kz = 0$ there are analytical solutions. We claim that our method works even when there is no analytical solution, because there are many existing numerical methods to solve high order polynomial equations with a high enough accuracy.

The scene shown as Figure 15 (left) is composed of a sphere with radius 61.4, a cylinder with radius 49.7 and height 158.2, voxelized at $256 \times 256 \times 256$ for 6-separability. The scene shown as Figure 15 (right) is the function $x^2 + ky + kz = 0$, with $k = 10.0$, voxelized at $256 \times 256 \times 256$ for 6-separability. We chose these parameter values arbitrarily.

8. CONCLUSION

We presented an accurate algorithm for the voxelization of polygon meshes which, for the first time, provides a comprehensive treatment of separability and guarantees minimality and accuracy. This method eliminates common voxelization artifacts at edges and vertices. We proved the topological attributes of our method and reported on the performance of our implementation. The algorithms we have presented need to be further optimized and accelerated. In addition, the topological correctness needs to be proven in the case of non-planar objects.

We demonstrated that this approach can be further extended to voxelize certain cubic objects. The approach could also be extended to support non-binary (i.e., antialiased) objects where the separability requirement is replaced by a 'uniform thickness' and the voxel value is a function of its distance to the surface. There-

fore, we believe that our approach will form the basis for a new breed of voxelization algorithms that rely on geometric distance to provide desirable topological and geometrical characteristics.

Acknowledgments

This paper has been proof read by Naeem Shareef of our group. The project has been partially supported by Ford Motor Company.

References

[1] Cohen, D. and Kaufman, A., "Scan-Conversion Algorithms for Linear and Quadratic Objects", in *Volume Visualization*, A. Kaufman, (ed.), IEEE Computer Society Press, Los Alamitos, CA, 1990, 280-301.

[2] Cohen Or, D., Kaufman, A., "Fundamentals of Surface Voxelization", Graphical Models and Image Processing, 57, 6 (November 1995), 453-461.

[3] Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F., C*omputer Graphics Principles and Practice*, Addison-Wesley, Reading, MA, 1990.

[4] Greene, N., "Voxel Space Automata: Modeling with Stochastic Growth Processes in Voxel Space", *Computer Graphics,* 23, 3 (July 1989), 175-184.

[5] Kaufman, A. and Shimony, E., "3D Scan-Conversion Algorithms for Voxel-Based Graphics", *Proceedings of the 1986 Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October 1986, 45-75.

[6] Kaufman, A., "An Algorithm for 3D Scan-Conversion of Polygons", *Proceedings of EUROGRAPHICS'87 Conference*, Amsterdam, The Netherlands, August 1987, 197-208.

[7] Kaufman, A., "Efficient Algorithms for 3D Scan- Converting Polygons", *Computers & Graphics*, 12, 2 (1988), 213-219.

[8] Kong, T. Y. and Rosenfeld, A., "Digital Topology: Introduction and Survey", *Computer Vision, Graphics and Image Processing*, 48, 3 (December 1989), 357-393.

[9] Levoy, M., "A Hybrid Ray Tracer for Rendering Polygon and Volume Data", *IEEE Computer Graphics & Applications*, 10, 3 (March 1990), 33-40.

[10] Newman, W. M. and Sproull, R. F., *Principles of Interactive Computer Graphics*, (2nd ed.) McGraw-Hill, New York, 1979.

[11] Yagel, R., Cohen, D. and Kaufman, A., "Discrete Ray Tracing", *IEEE Computer Graphics & Applications*, 12, 5 (September 1992), 19-28.

[12] R. Yagel, D.S. Ebert, J. Scott, and Y. Kurzion "Grouping Volume Renderers for Enhanced Visualization in Computational Fluid Dynamics," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 1, No. 2, July 1995, pp. 117-132.

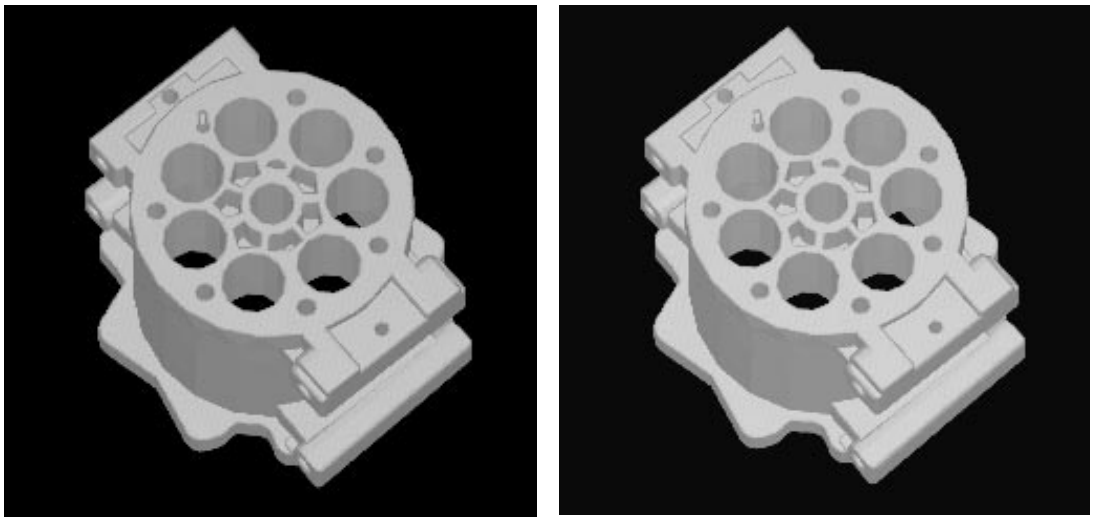


Figure 14. The ‘Pump’ data set voxelized in 26-separability (left) and 6-separability. They are almost identical visually, but are different topologically.

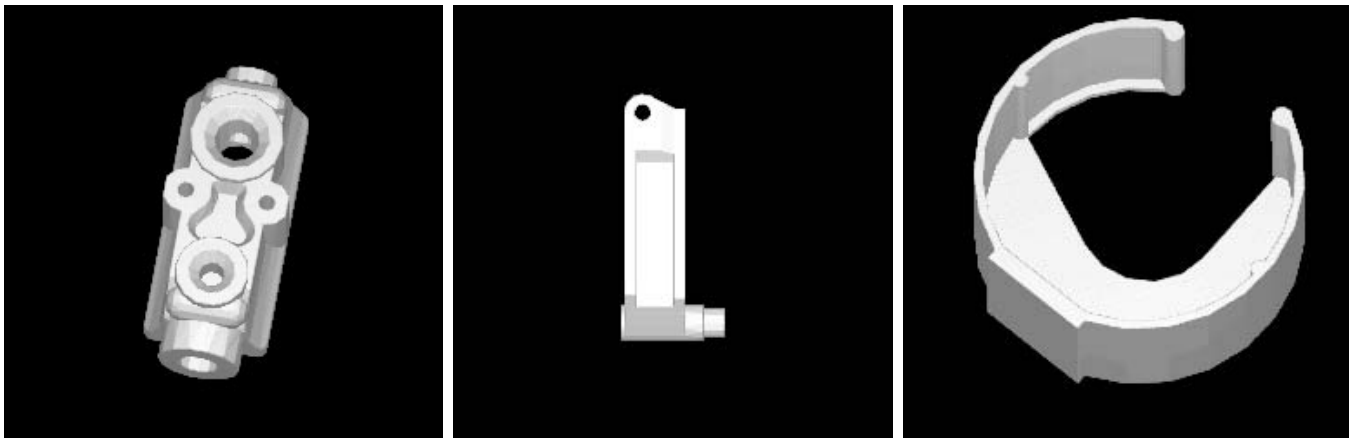


Figure 15. 6-separating voxelization of Bowl (left), Brevi (right), and Connector (middle).

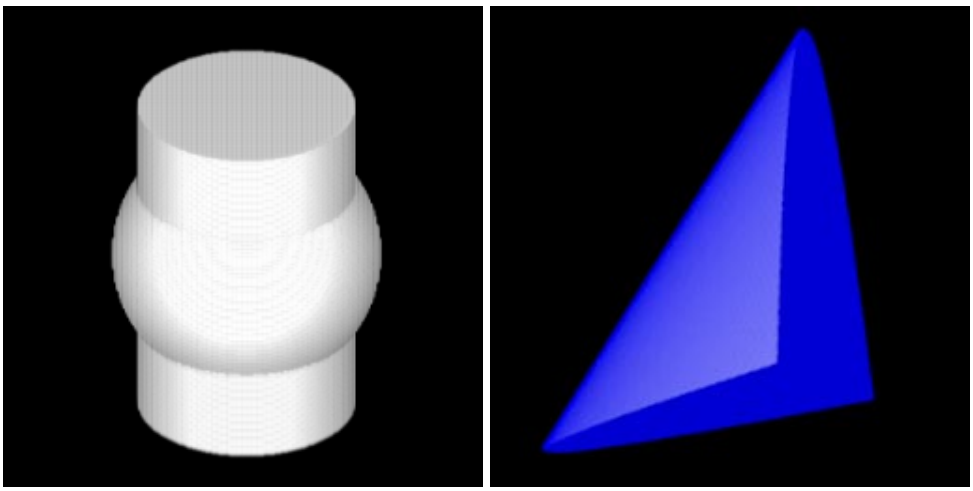


Figure 16. 6-separating voxelization of some cubic surfaces.