Output Sensitive Extraction of Silhouettes from Polygonal Geometry

F. Benichou * G. Elber
Technion, Israel Institute of Technology,
Haifa 32000, Israel

December 28, 1998

Abstract

An algorithm to allow real time interactive extraction and display of silhouettes of complex polygonal object(s) is presented. An off-line pre-processing of all the edges of all polygons allows the efficient extraction of the silhouette edges, in real time. During the interactive session, the time complexity of extracting the silhouette edges is linear in the number of edges in the silhouette, and is typically in the order of $O(\sqrt{n})$, where n is the number of polygons in the scene. The time complexity of the pre-processing stage is linear in n.

Key Words: Silhouette, Real Time Display, Gaussian Sphere, Range Searching.

1 Introduction

Consider a polygonal object, $\mathcal{O}$, and some viewing direction, $\vec{V}$. In this work, we assume that $\mathcal{O}$ is a two-manifold. That is, every point on every edge, $E_i \in \mathcal{O}$, serves as a boundary point of at most two polygons, $\mathcal{P}_1(E_i)$ and $\mathcal{P}_2(E_i)$, denoted the *associated polygons* of E_i. A polygon is considered *locally visible* if it presents a non zero area toward the viewing direction, $\vec{V}$. Then,

Definition 1 *E_i is a silhouette edge if exactly one of its two associated polygons is locally visible from $\vec{V}$.*

Local visibility does not ensure that a polygon is indeed visible. Nevertheless, local invisibility guarantees the invisibility of the polygon.

The silhouette edges of a given geometry are highly intuitive and useful in depicting the shape of the object, reflecting on the limits or the horizons of the geometry. The silhouette edges provide significant cues to the objects' shape and serves well in tools that display and/or portray geometry. Furthermore, the silhouette edges of an object can aid in a whole variety of applications, such as sweeping [4] and collision detection.

*Ecole Nationale Supérieure d'Electronique, d'Electrotechnique, d'Informatique et d'Hydraulique de Toulouse (ENSEEIHT), France

The set of all silhouette edges of $\mathcal{O}$ as viewed from direction $\vec{V}$ will be denoted $\mathcal{S}(\mathcal{O}, \vec{V})$. While we are allowed to pre-process all the geometric data off-line, we would like to answer the following query in the most efficient way:

> **Query 1** *Given a viewing direction, $\vec{V}$, find the set of edges of a polygonal object $\mathcal{O}$, that are silhouette edges from $\vec{V}$, $\mathcal{S}(\mathcal{O}, \vec{V})$.*

Let $\mathcal{E}$ be the set of all edges of all the polygons of $\mathcal{O}$. While each edge typically shows up twice, in each of its two associated polygons, we count this edge only once throughout this work. Let the number of polygons in $\mathcal{O}$ be of $O(n)$ and denote by s the number of silhouette edges, $|\mathcal{S}(\mathcal{O}, \vec{V})|$. Assume the number of edges of a single polygon is bounded. Then, the size of $\mathcal{E}$, $|\mathcal{E}|$, is also $O(n)$. Here after, we assume $|\mathcal{E}| = n$.

Consider a polygonal approximation of a sphere of radius r that is tiled by small triangles of area d each. The area of the sphere is proportional to r^2 while the silhouette edges of the sphere are proportional to the length of a great circle on the sphere or proportional to r. Therefore, $n = O\left(\frac{r^2}{d}\right)$ whereas $s = O\left(\frac{r}{\sqrt{d}}\right)$. In other words, the number of polygons in a spherical object relates to the area of the object is a similar way to the relation between s and the diameter of the sphere.

One can clearly construct objects where the number of silhouette edges, s, will be proportional to the total number of edges, n. For example, consider a polygonal approximation of a cylinder, where $s \geq \frac{n}{3}$ holds for any view. In contrast, one can envision an object that from a certain view, s is arbitrarily small, when an arbitrarily freeform complex feature of the model generates no silhouette edge.

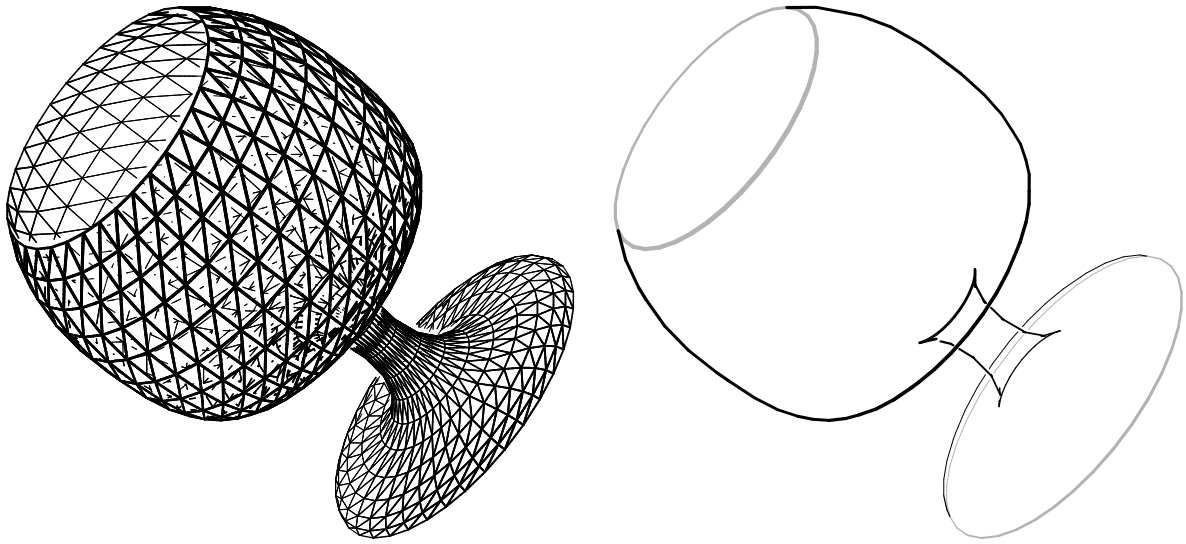
Nevertheless and on the average, we will expect that the number of silhouette edges will be proportional to $\sqrt{n}$. In this work, we are aiming at an *output sensitive extraction of silhouettes* and hence, we seek an algorithm to answer silhouette Query 1 with a time complexity of $O(\sqrt{n} + s)$.

1.1 The Detection of Silhouette Edges

Given edges $E_i \in \mathcal{E}$, $i = 1, .., n$ and a viewing direction, $\vec{V}$, we are interested in detecting all edges $E_i \in \mathcal{S}(\mathcal{O}, \vec{V})$. Consider again the two associated polygons of E_i, $\mathcal{P}_1(E_i)$ and $\mathcal{P}_2(E_i)$. Let the two normals of $\mathcal{P}_1(E_i)$ and $\mathcal{P}_2(E_i)$ be $N_1(E_i)$ and $N_2(E_i)$, that are also denoted as the *associated normals* of E_i. Then and following Definition 1, one can express the constraint for E_i to be a silhouette edge as,

$$E_i \in \mathcal{S}(\mathcal{O}, \vec{V}) \quad \Leftrightarrow \quad \left\langle N_1(E_i), \vec{V} \right\rangle \left\langle N_2(E_i), \vec{V} \right\rangle \leq 0. \tag{1}$$

Unlike Definition 1, Equation (1) includes the special case were one of the two associated polygon is invisible (back facing) while the other presents a zero area (with a normal that is orthogonal to



(a) (b)

Figure 1: In (a), a polygonal model of a wine glass is shown. In (b), the boundary edges are shown in gray while the silhouette are shown in black.

$\vec{V}$. Nevertheless, this singularity can clearly be tested via the examination of the individual signs of $\left\langle N_j(E_i), \vec{V} \right\rangle$, $j = 1, 2$ and will be ignored from now on.

Figure 1 (a) shows a polygonal model of a wine glass. In Figure 1 (b), the silhouette edges from the same viewing direction are displayed in black.

A naive algorithm to extract $\mathcal{S}(\mathcal{O}, \vec{V})$ can traverse all $O(n)$ polygons, $\mathcal{P}_j \in \mathcal{O}$, $j = 1, .., O(n)$ and extract the set $\mathcal{E}$. Then, for each edge $E_i \in \mathcal{E}$, $i = 1, .., n$, the two associated polygons of E_i are identified, and, following Equation (1), a test whether E_i is a silhouette edge or not can be conducted. See algorithm 1.

The time complexity of Algorithm 1 is clearly of order $O(n)$, again assuming the number of edges of every polygon is bounded by some constant. But even such an algorithm requires some pre-processing. First, the set $\mathcal{E}$ must be computed. Then, given $E_i \in \mathcal{E}$, the identification of $\mathcal{P}_1$ and $\mathcal{P}_2$ requires topological neighborhood information. Furthermore, while Algorithm 1 assumes a two-manifold object, each edge might be shared by more than two polygons, a singular condition that can be resolve by neither Algorithm 1 nor the output sensitive algorithm we are about to present as part of this work. In Section 1.2, we will introduce the problem as well as a possible simple remedy.

1.2 Regularized object

An edge can be shared by more than two polygons, in a two-manifold object. For example, consider Figure 2 (a). A single edge, E_i, is shared by polygon $\mathcal{P}_1$ on one side, and by polygons $\mathcal{P}_2$ and $\mathcal{P}_4$ on the

Algorithm 1

Input:
$\mathcal{O}$, A model to interactively display its silhouette;
$\vec{V}$, A viewing direction for the real time query;

Output:
$\mathcal{S}(\mathcal{O}, \vec{V})$, the silhouettes of $\mathcal{O}$ from viewing direction $\vec{V}$;

Algorithm:
NaiveSilhtExtrct($\mathcal{O}$, $\vec{V}$ **)**
begin
$\mathcal{S}(\mathcal{O}, \vec{V}) \Leftarrow \{\ \}$;
$\mathcal{E} \Leftarrow$ All n edges of all polygons, $\mathcal{P}_j \in \mathcal{O},\ \ j = 1, .., O(n)$;
For each edge E_i **in** $\mathcal{E}$ **do**
If $\left(\left\langle N_1(E_i), \vec{V}\right\rangle \left\langle N_2(E_i), \vec{V}\right\rangle \ \leq\ 0\right)$ **then**
$\mathcal{S}(\mathcal{O}, \vec{V}) \Leftarrow \mathcal{S}(\mathcal{O}, \vec{V}) \cup \{E_i\}$;
endfor;
return $\mathcal{S}(\mathcal{O}, \vec{V})$;
end.

other. E_i is called a *non-regular edge*. To complicate things, note the different unit normals of polygons $\mathcal{P}_2$ and $\mathcal{P}_4$, N_2 and N_4. From the given viewing direction, and according to Equation (1), only a *portion* of E_i is a silhouette edge, the portion that is shared by $\mathcal{P}_1$ and $\mathcal{P}_4$.

The solution lays in the regularity condition we are about to impose over the polygons of $\mathcal{O}$:

Definition 2 *A two-manifold object, $\mathcal{O}$, is called* regular *if no vertex in $\mathcal{O}$ lays on the interior of an edge of a neighboring polygon.*

In other word, for a two-manifold regular object, all edges are shared by at most two polygon.

In the example of Figure 2 (a), the remedy is found in splitting polygon $\mathcal{P}_1$ into two new polygons, $\mathcal{P}_1^a$ and $\mathcal{P}_1^b$, in such a way that obtains two different edges that exactly match the two respective edges of polygons $\mathcal{P}_2$ and $\mathcal{P}_4$. This entire process is called *regularization*.

1.3 The Detection of Boundary Edges

Geometric objects may be open. For example in Figure 1 (a), the top and the bottom of the surface of revolution of the wine glass is open.

All edges that have only one associated polygon are called *open edges*. The set of all *open edges* is denoted the *object's boundary* or $\mathcal{B}$, and is also called the set of boundary edges. Unlike $\mathcal{S}(\mathcal{O}, \vec{V})$, $\mathcal{B}$ is

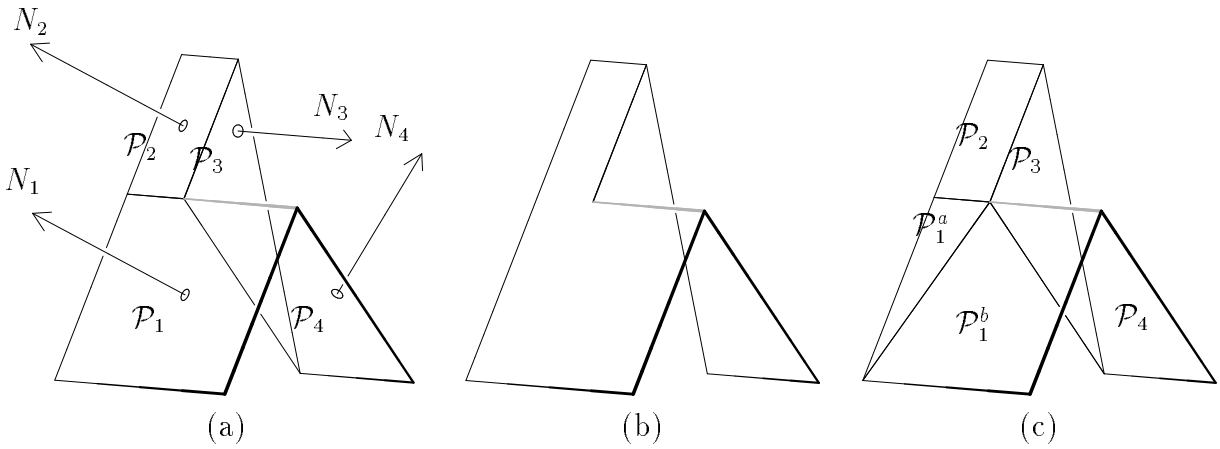


Figure 2: A non-regular object (a) prohibits the proper extraction of the silhouette output as is shown in (b). Indeed the corresponding single edge between $\mathcal{P}_1$ and $\mathcal{P}_2$ and $\mathcal{P}_4$, can't be split whereas the silhouette output needs it. The solution consists of splitting $\mathcal{P}_1$ to match exactly the neighboring edges from $\mathcal{P}_2$ and $\mathcal{P}_4$ as in (c).

view-independent and will always contribute to the display of the object. In other words, $\mathcal{B}$ solely depends on the geometry of the object, $\mathcal{B} = \mathcal{B}(\mathcal{O})$. Figure 1 (b) shows, in a gray line, the boundary set of the polygonal model of the wine glass shown in Figure 1 (a).

Efficient extraction of silhouettes, given the viewing direction, is potentially useful for many other applications, such as hidden line removal, collision detection, diameters of objects, and the construction of the swept volume of $\mathcal{O}$ along some rotational trajectory. Nevertheless and while the presented approach is clearly useful for all these applications, we will demonstrate the use of the proposed algorithm only in the application of real time display.

This paper is organized as follow. In Section 2, we present the different stages of the algorithm. In Section 3, some result are presented, while we conclude in Section 4.

2 Algorithm

In this section, we introduce the output sensitive extraction algorithm of silhouette edges. We assume that the input, two-manifold, polygonal object $\mathcal{O}$ has already been regularized. We begin the pre-processing stage, in Section 2.1, by mapping the normals of all the polygons of $\mathcal{O}$, along with all the edges of the neighboring polygons of $\mathcal{O}$, onto a data structure called the *Gaussian Sphere* [5], $\mathcal{G}_\mathcal{S}$, on S^2, the unit sphere. Then, in the second pre-processing stage, the data on the Gaussian Sphere is projected onto six different planes of a circumscribing cube. This second stage is described in Section 2.2. The third stage is the stage that satisfies the silhouette query, and is described in Section 2.3. This query request is the only stage

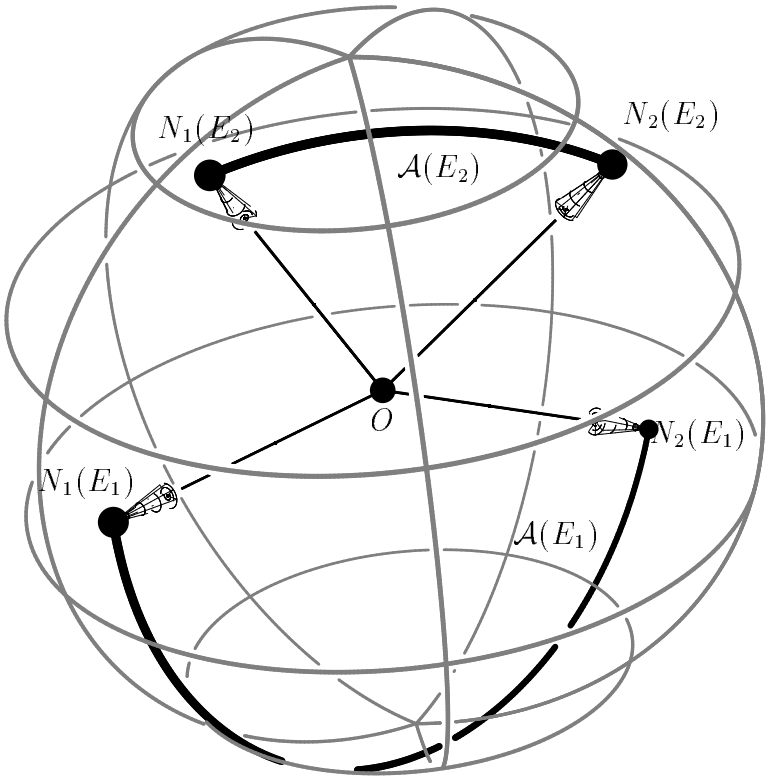


Figure 3: A Gaussian Sphere, $\mathcal{G}_{\mathcal{S}}$, containing two *associated arcs*, $\mathcal{A}(E_1)$ and $\mathcal{A}(E_2)$, of two different edges, E_1 and E_2.

that is conducted during the interactive session, in real time.

2.1 The Mapping onto the Gaussian Sphere

Given a viewing direction, the containment of edge E_i in $\mathcal{S}(\mathcal{O}, \vec{V})$ solely depends on its two associative unit normals, $N_1(E_i)$ and $N_2(E_i)$. Denote the origin of the Euclidean space by O, and let arc $\mathcal{A}(E_i) = \angle N_1(E_i) O N_2(E_i)$ be the *associated arc* of E_i on the Gaussian Sphere (see Figure 3). $\mathcal{A}(E_i)$ is a great arc and while there exist two great arcs between $N_1(E_i)$ and $N_2(E_i)$ on $\mathcal{G}_{\mathcal{S}}$, the shortest one is the only one considered.

Let $\mathcal{P}_{\vec{V}}$ be the plane that is orthogonal to $\vec{V}$ through the origin O. $\mathcal{P}_{\vec{V}}$ is denoted the *viewing plane*. Then,

Lemma 1

$$E_i \in \mathcal{S}(\mathcal{O}, \vec{V}) \quad \Leftrightarrow \quad \mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}} \neq \emptyset. \tag{2}$$

Proof: From Equation (1), E_i is a silhouette edge if and only if $\left\langle N_1(E_i), \vec{V} \right\rangle \left\langle N_2(E_i), \vec{V} \right\rangle \leq 0$.

Consider first the case of $\left\langle N_j(E_i), \vec{V} \right\rangle = 0$, for $j = 1$ or $j = 2$. Then, Equation (1) holds and hence E_i is a silhouette. But if $\left\langle N_j(E_i), \vec{V} \right\rangle = 0$, $N_j(E_i)$ must be orthogonal to $\vec{V}$, or $N_j(E_i) \subset \mathcal{P}_{\vec{V}}$. Therefore, $\mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}} \neq \emptyset$.

Now assume $\left\langle N_1(E_i), \vec{V} \right\rangle \left\langle N_2(E_i), \vec{V} \right\rangle < 0$. From the continuity of $\mathcal{A}(E_i)$, it is clear that there exist $N \in \mathcal{A}(E_i)$ such that $\left\langle N, \vec{V} \right\rangle = 0$. Then again, N must be in $\mathcal{P}_{\vec{V}}$. Therefore, $\mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}} \neq \emptyset$.

Finally assume $\left\langle N_1(E_i), \vec{V} \right\rangle \left\langle N_2(E_i), \vec{V} \right\rangle > 0$. Then, both $N_1(E_i)$ and $N_2(E_i)$ are on the same side of $\mathcal{P}_{\vec{V}}$. Clearly the shorted great geodesic arc on S^2 between $N_1(E_i)$ and $N_2(E_i)$ does not intersect $\mathcal{P}_{\vec{V}}$. This geodesic is exactly $\mathcal{A}(E_i)$. Therefore, $\left\langle N, \vec{V} \right\rangle \neq 0$, $\forall N \in \mathcal{A}(E_i)$. Hence, no vector in $\mathcal{A}(E_i)$ is contained in $\mathcal{P}_{\vec{V}}$, or $\mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}} = \emptyset$.

The other direction follows the exact same (yet reversed) line of reasoning, concluding the proof.

■

The silhouette extraction problem has therefore been reduced into searching the (associated) arcs, $\mathcal{A}(E_i)$, on the unit sphere of $\mathcal{G}_{\mathcal{S}}$, that are crossed by the viewing plane, $\mathcal{P}_{\vec{V}}$. An edge with two identical associated normals will never be displayed. Hence, and while mapping all the normals to the $\mathcal{G}_{\mathcal{S}}$, we filter out, during this pre-processing stage, edges with zero arc-length associated arcs.

2.2 Projecting the Gaussian Sphere onto a Circumscribing Cube

As in many cases, it is easier to deal with two dimensional geometry than with three dimensional one. The goal of this stage is to map the three dimensional query problem of arcs on the unit sphere into a two dimensional query problem of line segments on a two dimensional plane. Toward this end, we employ the central (or gnomonic) projection [6]. Every point $P \in \mathcal{A}(E_i)$ is projected by the central projection along a ray that emanated from the origin of the Euclidean space, O, which is also the center of $\mathcal{G}_{\mathcal{S}}$. Given some planar face $\mathcal{F}_i$, the central projection of point P on $\mathcal{F}_i$ is the intersection point of ray $\overline{OP}$ and $\mathcal{F}_i$, $\overline{OP} \cap \mathcal{F}_i$. Interestingly enough, great arcs such as $\mathcal{A}(E_i)$, are centrally projected into *line segments* in $\mathcal{F}_i$.

Here, the normal information on the Gaussian Sphere, $\mathcal{G}_{\mathcal{S}}$, is centrally projected onto a convex circumscribing polyhedra along with all its associated arcs. The convexity condition ensures that each point on the unit sphere is centrally projected to exactly one point on the polyhedra. The fact that the polyhedra circumscribes the sphere guarantees that the central mapping from the unit sphere to the polyhedra is indeed bijective.

We have selected the convex circumscribing polyhedra to be an axis-parallel cube due to the simplicity in the expected projections onto the six faces of the cube. Each arc $\mathcal{A}(E_i)$ on the Gaussian Sphere is

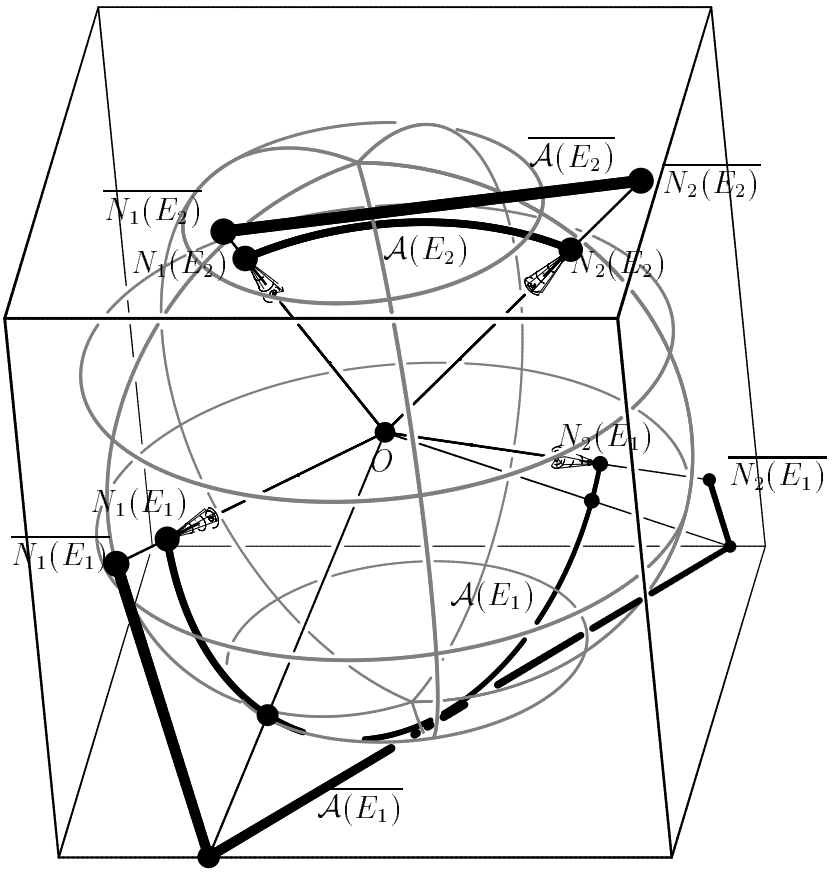


Figure 4: Projection onto a circumscribing cube of the two *associated arcs*, $\mathcal{A}(E_1)$ and $\mathcal{A}(E_2)$, from Figure 3.

centrally projected along with its two end points' normals onto the circumscribing cube. Recall that $\mathcal{A}(E_i)$ always spans less than 180 degrees. Then, the central projection might project $\mathcal{A}(E_i)$ on the cube as one to four line segments on one to four adjacent faces of the cube, respectively. See Figure 4.

Let $\overline{N_1(E_i)}$ and $\overline{N_2(E_i)}$ denote the central projections of $N_1(E_i)$ and $N_2(E_i)$ onto the cube. Again, if $\overline{N_1(E_i)}$ and $\overline{N_1(E_i)}$ belong to the same face of the cube, the projection of the arc is the simple line segment $\overline{\overline{N_1(E_i)}\ \overline{N_2(E_i)}}$. Otherwise, we obtain a list of up to four line segments on adjacent faces of the cube. The projection of $\mathcal{A}(E_i)$ onto the cube is denoted the *associated segment* of E_i, $\overline{\mathcal{A}(E_i)}$.

Denote by $\mathcal{F}_i$, $i = 1, .., 6$, the six faces of the circumscribing cube, and let $\mathcal{L} = \left\{\mathcal{L}_i(\vec{V})\right\}_{i=1}^{6} = \left\{\mathcal{P}_{\vec{V}} \cap \mathcal{F}_i\right\}_{i=1}^{6}$. In general, a non-empty intersection of a cube and an infinite plane is a polygon with three to six edges. Nevertheless, because the plane $\mathcal{P}_{\vec{V}}$ contains the origin, O, which is the center of the cube, the resulting intersection of the cube and $\mathcal{P}_{\vec{V}}$ might consist of either four or six line segments, $\mathcal{L}_i(\vec{V})$, forming a closed convex polygon. Then, the central projection allows us to perform a second mapping that further simplifies the complexity of answering silhouette Query 1:

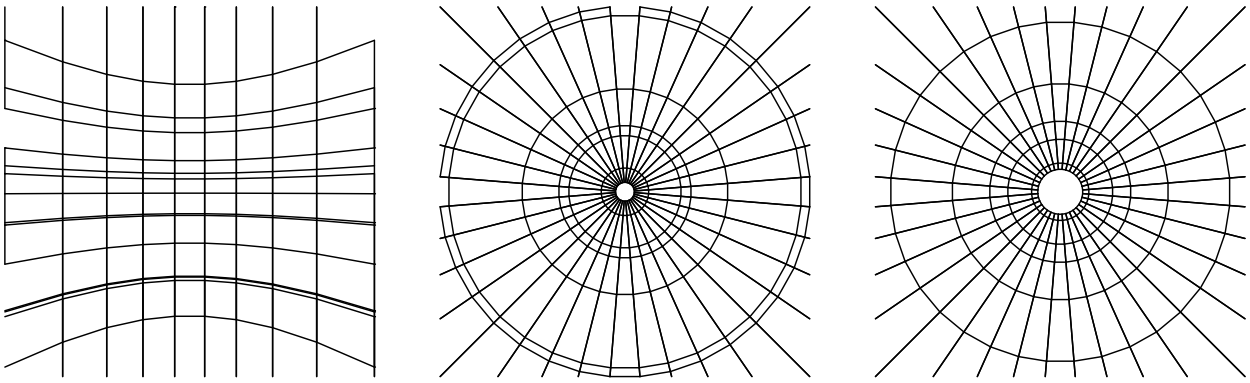


Figure 5: Three projection planes of the Gaussian Sphere data, $\mathcal{G}_{\mathcal{S}}$, onto a circumscribing cube. Data is from the glass object of Figure 1. Note that all drawn polylines are, in fact, formed out of numerous small connected line segments. Planes $X = -1$, $Y = \pm 1$ are similar to Plane $X = 1$. Also note the missing edges on the boundaries which are added at random, to either one of the two faces that share that edge.

Lemma 2

$$E_i \in \mathcal{S}(\mathcal{O}, \vec{V}) \;\Leftrightarrow\; \overline{\mathcal{A}(E_i)} \cap \mathcal{L} \neq \emptyset. \tag{3}$$

Proof: From Lemma 1, we have $E_i \in \mathcal{S}(\mathcal{O}, \vec{V}) \;\Leftrightarrow\; \mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}}$. Hence, we need to show that $\overline{\mathcal{A}(E_i)} \cap \mathcal{L} \neq \emptyset \;\Leftrightarrow\; \mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}}$. Now $\mathcal{A}(E_i) \cap \mathcal{P}_{\vec{V}} = \mathcal{A}(E_i) \cap (\mathcal{P}_{\vec{V}} \cap S^2)$. Furthermore, $\mathcal{L}$ is the central projection of $\mathcal{P}_{\vec{V}} \cap S^2$ and $\overline{\mathcal{A}(E_i)}$ is the central projection of $\mathcal{A}(E_i)$. Then, relation (3) is a direct consequence of the bijective property of the central projection of the unit sphere onto its circumscribing cube, preserving all intersections.

■

Figure 5 shows an example of the six $\mathcal{F}_i$ planes that are obtained by projection the $\mathcal{G}_{\mathcal{S}}$ data of the wine glass of Figure 1 onto six faces of a circumscribing cube. The symmetry of the surface of revolution of the glass yields the same projection for the four planes of $X = \pm 1$ and $Y = \pm 1$ and hence three of these four planes are omitted from the figure.

So far, completing the pre-processing stages, we have obtained a set of six planes that contain $O(n)$ line segments representing all the edges of the object(s). In the next section, Section 2.3, Lemma 2 will be employed to efficiently compute the silhouette edges of the polygonal model, from an arbitrary viewing direction, $\vec{V}$.

2.3 Querying the silhouette edges

The problem in hand can now be formulated as follows. Given a set of $O(n)$ line segments, $\left\{\overline{\mathcal{A}(E_i)}\right\}_{i=1}^{n}$, in the plane, one is seeking the line segments that are intersected by an infinite prescribed line, $\mathcal{L}_i(\vec{V})$. Solutions to this problem are known and we do not attempt to survey then here; we only recall the results that were obtained for this *stabbing or ray tracing query* as it is sometimes called. Chazelle [2] derived a lower bound of $\Omega(n/\sqrt{\Delta})$ for this problem where Δ is the storage used. Matoušek [7] gave an algorithm whose performance matches this lower bound. Yet, the implementation of his algorithm is based on a linear-space data structured which is complicated. In Argawal and Sharir [1], similar results are presented with a complex algorithm that based on a new hierarchical space-partitioning scheme defined by Chazell *et al.* [3]. Both methods move the initial problem into a half-plane range-searching problem, which consists of reporting points in one part of an hyper-plane.

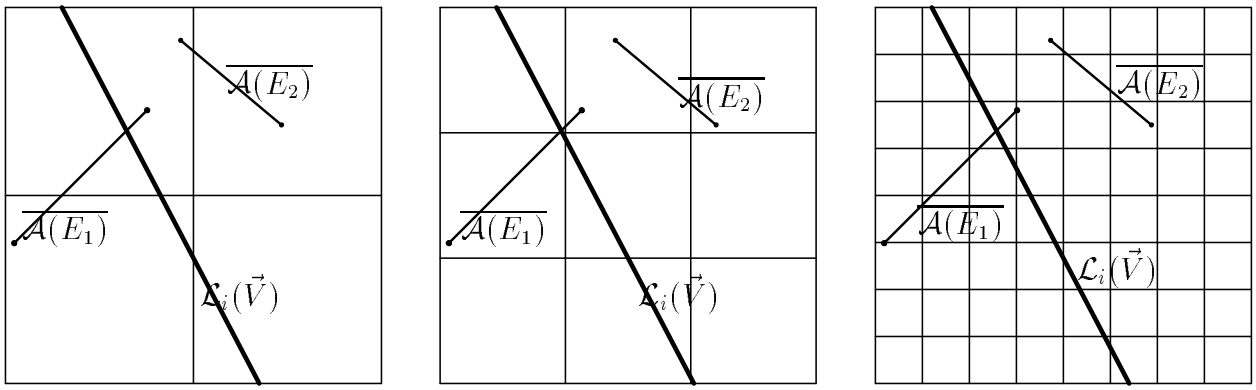
It is clear that if Δ is in the order of the number of line segments, n, the segments of interest can be reported in $O(\sqrt{n})$. Because s is also in the order of $O(\sqrt{n})$, we are able to report the silhouette edges in output sensitive manner that is linear in the output set and hence optimal.

As part of the implementation of this work, we have chosen a simple yet efficient alternative to the above optimal stabbing query algorithms. As the segments are located in six bounded planar square areas, we impose six grids that are denoted $\mathcal{G}_i,\ i = 1, .., 6$, on these six planar square faces. These $\mathcal{G}_i$ grids are set to follow some prescribed resolution. Each cell, $C \in \mathcal{G}_i$, stores references to the line segments from the set of projected arcs, $\overline{\mathcal{A}(E_i)}$, that intersects C. Then, and following Lemma 2, the query consists of visiting only the set of cells that are stabbed by line $\mathcal{L}_i(\vec{V})$ and reporting segments $\overline{\mathcal{A}(E_i)}$ that indeed intersect with $\mathcal{L}_i(\vec{V})$.

Figure 6 illustrates several grids of different resolutions containing two line segments and the infinite query line, $\mathcal{L}_i(\vec{V})$. In Figures 6 (a) and (b), $\overline{\mathcal{A}(E_1)}$ and $\overline{\mathcal{A}(E_2)}$ have at least one common cell with $\mathcal{L}_i(\vec{V})$. Hence, the query process will visit the two segments. Nonetheless, in Figure 6 (c), no cells are simultaneously crossed by $\overline{\mathcal{A}(E_2)}$ and $\mathcal{L}_i(\vec{V})$, so $\overline{\mathcal{A}(E_2)}$ is not considered.

A line segment, $\overline{\mathcal{A}(E_i)}$, might be contained in more than one cell. Therefore, and in order to prevent from testing $\overline{\mathcal{A}(E_i)}$ several times for intersections against $\mathcal{L}_i(\vec{V})$, we mark $\overline{\mathcal{A}(E_i)}$ as tested on its first intersection test, avoiding unnecessary further redundant computations. Obviously, the finer the grid, the less line segments one needs to test in each query, but more cells are to be visited and more memory is to be allocated.

Algorithm 2 summarizes the first two pre-processing stages that were described in Sections 2.1 and 2.2.



(a) Grid of 2 (b) Grid of Size 3 (c) Grid of Size 8

Figure 6: Two segments, $\overline{\mathcal{A}(E_i)}$, $i = 1, 2$ and a query line, $\mathcal{L}_i(\vec{V})$, processed in three grids of different resolutions. Note that in (c), $\overline{\mathcal{A}(E_2)}$ need not be tested against $\mathcal{L}_i(\vec{V})$, because $\overline{\mathcal{A}(E_2)}$ and $\mathcal{L}_i(\vec{V})$ share no common cells.

The time complexity of this pre-processing stage is clearly bounded by $O(ng^2)$ where g^2 is the number of cells in each one of the six grids, $\mathcal{G}_i$. Hence, for a grid with a fixed size, the time complexity of Algorithm 2 is $O(n)$.

Algorithm 3 summerizes the process of the real time stabbing query, using the simple grid based approach. Assume that the average edge covers m cells on grid $\mathcal{G}_i$ and select a grid resolution of size $\sqrt{n}$ by $\sqrt{n}$ for $\mathcal{G}_i$. Hence, the average density of a cell will be m edges per cell. A line cutting through $\mathcal{G}_i$ will visit $O(\sqrt{n})$ cells or $O(m\sqrt{n})$ edges will be examined for being silhouette edge, on the average. Unfortunately, the value of m is highly geometry dependent, a hindering factor which prevents us from establishing a clear upper bound for the grid based query algorithm. Moreover, it is clear that one can stage cases for each we will unnecessarily test $O(n)$ edges, using Algorithm 3, even when $s = O(\sqrt{n})$. Nevertheless, the average behavior of this algorithm is far better that the naive approach presented in Algorithm 1 as will be demonstrated in Section 3.

3 Results

We have applied the proposed algorithm to various polygonal objects and recorded the average times to answer the silhouette query in Table 1. In all the results that are shown, the pre-processing stage took a very short time, from a fraction of a second to few seconds. All computations were conducted on an $O2$ SGI system equipped with a 180 Mhz R5000.

The times presented for the computation of the silhouette query have been calculated from an average

Algorithm 2

Input:
$\mathcal{O}$, A model to interactively display its silhouette;

Output:
$\mathcal{G}_i,\ i = 1,..,6$, Six planar grids holding the line segments representing all the edges of $\mathcal{G}_\mathcal{S}$'s geometry projected onto the six planes of a cube;

Algorithm:
OtptSnstvSilhtExtrctPreprocess($\mathcal{O}$, $\vec{V}$ **)**
begin
  For i from 1 to 6 step 1 **do**
    $\mathcal{G}_i \Leftarrow \emptyset$;
    $\mathcal{F}_i \Leftarrow i$'th face of circumscribing cube;
  end;
  $\mathcal{E} \Leftarrow$ All edges of all polygons, $\mathcal{P}_j \in \mathcal{O},\ j = 1,..,O(n)$;
  For each edge E_i **in** $\mathcal{E}$ **do**
    $N_1(E_i),\ N_2(E_i) \Leftarrow$ Normals of the two associated polygons of E_i on $\mathcal{G}_\mathcal{S}$;
    $\mathcal{A}(E_i) \Leftarrow$ The associated arc of E_i on $\mathcal{G}_\mathcal{S}$;
    $\overline{N_1(E_i)},\ \overline{N_2(E_i)},\ \overline{\mathcal{A}(E_i)} \Leftarrow$ The central projection of $N_1(E_i),\ N_2(E_i),\ \mathcal{A}(E_i)$ on the circumscribing cube;
    For i from 1 to 6 step 1 **do**
      If $\left(\overline{\mathcal{A}(E_i)} \cap \mathcal{F}_i \neq \emptyset\right)$ **then**
        For each cell $C \in \mathcal{G}_i$ such that $\left(\overline{\mathcal{A}(E_i)} \cap C\right) \neq \emptyset$ **do**
          Update C to contain a reference to $\overline{\mathcal{A}(E_i)}$ and E_i;
        endfor;
    endfor;
  endfor;
  return $\{\mathcal{G}_i\}_{i=1}^{6}$;
end.

of one thousand different views. Note that even though we are not using an optimal algorithm for resolving the stabbing query, we have obtained a significant reduction in the computation time of the silhouette query by factors that range from ten times and up to almost one hundred times less than the straight forward or naive method of Algorithm 1.

4 Conclusion

We have presented an optimal algorithm to interactively extract the silhouette edges of polygonal model(s), that is linear in the output silhouette set.

Algorithm 3

Input:

$\mathcal{O}$, A model to interactively display its silhouette;
$\vec{V}$, A viewing direction for the real time query;
$\{\mathcal{G}_i\}_{i=1}^{6}$, The result of the preprocessing stage (Algorithm 2);

Output:

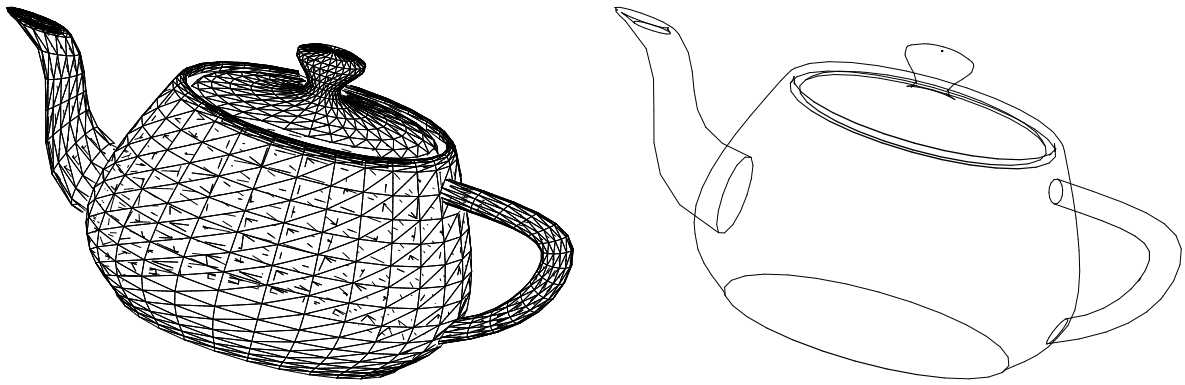
$\mathcal{S}(\mathcal{O}, \vec{V})$, The silhouettes of $\mathcal{O}$ from viewing direction $\vec{V}$;

Algorithm:

OtptSnstvSilhtExtrctQuery($\mathcal{O}$, $\vec{V}$, $\{\mathcal{G}_i\}_{i=1}^{6}$ **)**
begin
$\mathcal{S}(\mathcal{O}, \vec{V}) \Leftarrow \emptyset$;
$\mathcal{P}_{\vec{V}} \Leftarrow$ Plane orthogonal to $\vec{V}$ through the origin;
$\mathcal{L} \Leftarrow$ four/six line segments of the intersection of $\mathcal{P}_{\vec{V}} \cap \mathcal{F}_i, \ i = 1, .., 6$;
For each non empty $\mathcal{L}_i(\vec{V})$ in $\mathcal{L}$ **do**
For every cell $C \in \mathcal{G}_i$ that interest $\mathcal{L}_i(\vec{V})$ **do**
For each untested projected associated arc $\overline{\mathcal{A}(E_i)} \in C$ **do**
Mark E_i as tested;
If ($\mathcal{L}_i(\vec{V}) \cap \overline{\mathcal{A}(E_i)} \neq \emptyset$) **then**
$\mathcal{S}(\mathcal{O}, \vec{V}) \Leftarrow \mathcal{S}(\mathcal{O}, \vec{V}) \bigcup \{E_i\}$;
endfor;
endfor;
endfor;
return $\mathcal{S}(\mathcal{O}, \vec{V})$;
end.

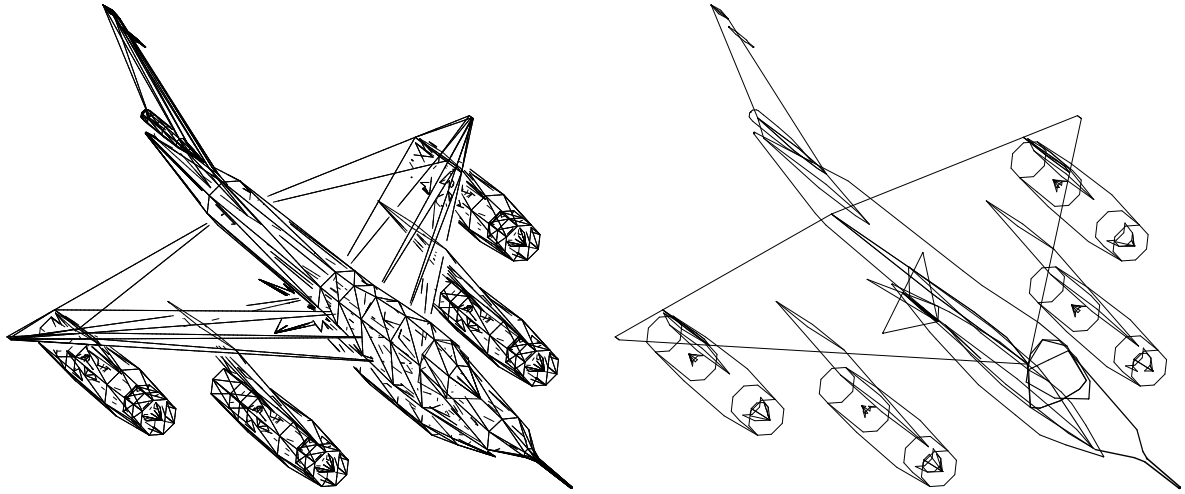
Objects	# Triangles	Average # Silhouette Edges	Average Silhouette Query Time			
			Straight forward Method (Alg. 1)	Grid Resolution		
				2	10	50
Glass Figure 1	75,048	715	0.768 Secs [225,144]	0.060 Secs [32,866]	0.018 Secs [7,845]	0.009 Secs [2,173]
Teapot Figure 7	85,176	1,258	0.887 Secs [255,528]	0.085 Secs [45,010]	0.027 Secs [11,896]	0.013 Secs [3,598]
Plane B58 Figure 8	78,844	2,851	0.828 Secs [236,532]	0.076 Secs [37,040]	0.040 Secs [15,280]	0.030 Secs [6,141]
Dinner Figure 9	59 000	3,390	0.631 Secs [177,000]	0.076 Secs [34,890]	0.037 Secs [11,347]	0.027 Secs [5,075]

Table 1: Average times to answer the silhouette query and the number of segments visited according to the object and the method used. Computed on $O2$ SGI system equipped with a 180 Mhz R5000. In square brackets, the number of edges that are actually visited is listed.



(a) (b)

Figure 7: The Utah Teapot. In (a), the polygonal model and in (b), the silhouette and the boundary are shown.

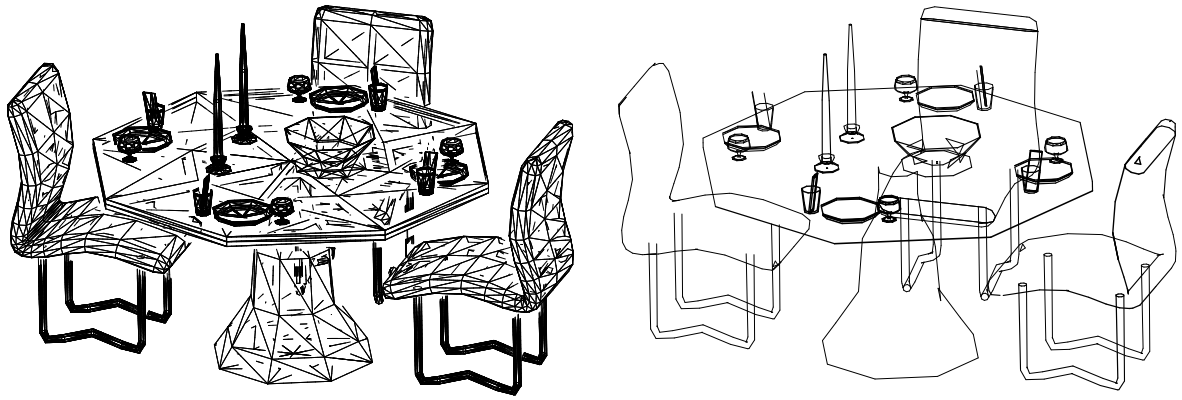


(a) (b)

Figure 8: A B58 plane. In (a), the polygonal model and in (b), the silhouette and boundary.

Several improvements can augment the presented method. First, this algorithm is limited by the real memory capacity of the exploited system. While the memory consumption of the presented algorithm is bounded by $O(n)$ and hence is in the same order of the magnitude of the input, large models can take full advantage of the real memory of contemporary workstations. Because large models is where the presented algorithm shines, even a decrease by a constant factor of the memory use is significant, and effort should be invested to minimize this memory overhead.

The use of this efficient silhouette extraction algorithm should be examined in various other applications



(a) (b)

Figure 9: A dinner scene. In (a), the polygonal model and in (b), the silhouette and boundary.

from hidden line removal, through general swept volume of object $\mathcal{O}$ [4], to collision detection.

While in this work, we have selected a circumscribing cube for simplicity, one should expect somewhat better behavior if a tetrahedra will be used, minimizing the number of faces that need to be processed. While each face is now somewhat more complex, the over whole cost should be evaluated, seeking the best convex circumscribing polyhedra. Furthermore, because objects have different silhouette complexity from different views, it is also worth while to ask the question of the best circumscribing convex polyhedra, for some special objects.

Because of the way the arcs are centrally projected onto the plane of the circumscribing polyhedra, one could also have chosen to subdivide the planes in *a radial way*. This subdivision scheme should improve our results since the associated segments are expected to be more densely populated in the centers of the six planes, due to the behaviour of the central projection, while making the traversal of cells for a given $\mathcal{L}_i(\vec{V})$ more complex. This density should be computed to follow the Riemannian metric of the central projection. In addition, the influence of the resolution of the grid on the query time, considering both the shape of the model and the number of silhouette edges, is yet to be explored as well, for optimality.

Throughout this discussion, the assumtion was made that the viewing is orthographic. The extension of the proposed scheme to perspective views is clearly desired and should be sought.

Finally, we have not exploited the fact that the stabbing line is *continuously* moving. This coherency can clearly be used in finding the new set of cells and/or edges that needs to be visited in the grid, building the solution of the next silhouette query based on the solution of the previous query.

References

[1] P. K. Argawal and M. Sharir. Applications of a New Space-Partitioning Technique. Discrete Computational Geometry, 9:11-38, 1993.

[2] B. Chazelle. Lower bounds on the complexity of polytope range searching. Journal of the American Mathematical Society, 2(4):637-666, 1989.

[3] B. Chazelle, M. Sharir and E. Wezl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. Algorithmica, 8:407-429, 1992.

[4] S. Coquillart. A control Point Based Sweeping Technique. IEEE Computer Graphics and Applications, Vol. 7, No. 11, pp 36-44, November 1987.

[5] M. P. DoCarmo. Differential Geometry of Curves and Surfaces. Prentice-Hall 1976.

[6] S. Iyanaga, Y. Kawada and K. O. May. Encyclopedic Dictionary of Mathematics. The MIT Press Cambridge, Massachusetts, and London, England.

[7] J. Matoušek. Range Searching with Efficient Hierarchical Cuttings. Discrete Computational Geometry, 10:157-182, 1993.