Efficient View Point Selection for Silhouettes of Convex Polyhedra*

Therese Biedl[1], Masud Hasan[1**], and Alejandro López-Ortiz[1]

School of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. {biedl,m2hasan,alopez-o}@uwaterloo.ca.

Abstract. The silhouette of polyhedra is an important primitive in application areas such as machine vision and computer graphics. In this paper, we study how to select view points of convex polyhedra such that the silhouette satisfies certain properties. Specifically, we give algorithms to find all projections of a convex polyhedron such that a given set of edges, faces and/or vertices appear on the silhouette.
We present an algorithm to solve this problem in $O(k^2)$ time for k edges. For orthogonal projections, we give an improved algorithm that is fully adaptive in the number l of connected components formed by the edges, and has a time complexity of $O(k \log k + kl)$. We then generalize this algorithm to edges and/or faces appearing on the silhouette.

1 Introduction

Polyhedra are solids in 3-dimensional space. When looking at a polyhedron from a view point, the eye or camera computes a 2-dimensional projection of the polyhedron which may be an orthogonal or a perspective projection, depending on whether the view point is at infinity or not. In particular, some features of the polyhedron, such as vertices, edges or faces, are visible, while others are hidden in this projection. Especially noticeable are those features that reside on the *shadow boundary* of the projection, i.e., those that are just barely visible. Closely related is the concept of the *silhouette*, which are those edges for which exactly one incident face is visible; the two concepts describe the same set for convex polyhedra.

Silhouettes are useful in various settings, especially in the area of **machine vision**. For 3D gauging systems, in order to gauge a given part, video cameras are used to acquire silhouettes of the part along with the locations of the lighting element. These silhouettes help identify key elements of the part [16]. For assembling purposes, silhouettes are used to compute the boundary and the orientation of the mechanical parts to be picked up by a robot for assembly [15]. Silhouettes are also used for quality control [24], object recognition [17, 23], illuminating critical features [18] and others.

* Research supported by NSERC.

** On leave from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

For **image recognition**, researchers consider the topological graph representation of projections of polyhedra, which are called a *characteristic view* [19]. A similar topological graph is also considered for the silhouette [10]. Silhouettes from the projection of an object can be matched against stored pre-computed characteristic views hence aiding in recognition of the object (see [19] and the references therein). This has the advantage that the views from two nearby view points likely result in the same characteristic view, which makes the system robust under small positional errors. The characteristic views of a polyhedron are better known as *aspect graphs*. See [22] for a detailed survey on aspect graphs. **Reconstruction from images** concerns the problem of approximately reconstructing a 3D object from one or more images [13, 14, 25]. Instead of full images of an object, often only silhouettes are used in a process called *volume intersection* [13, 14]. In **computer graphics**, silhouette edges represent discontinuities in the visibility of an object, and are one of the strongest visual cues of the shape of an object [12]. When rendering the object, it often suffices to render only the edges on the silhouette, which can result in substantial speedup [20].

The computation of silhouettes has been studied extensively both in the computational geometry and in the computer graphics community. Pop et al. [20] gave an algorithm for perspective projections that maintains the silhouette of polyhedra during arbitrary changes of the view point. They use duality theory (see also Section 3.2) to develop a practical and efficient heuristic to maintain the corresponding visibility properties. Efrat et al. [10] presented several combinatorial bounds on the silhouette structure of a collection of convex polyhedra when the view point moves along a straight line or along an algebraic curve. They compute the silhouette map which is the arrangement of the silhouettes of all objects with their hidden parts removed. Their combinatorial complexity is the bound on the number of combinatorial changes in the silhouette map during the motion of the view point. For orthogonal projections only, Benichoe and Elber [4] give output-sensitive algorithms to find silhouettes from polyhedral scenes for a given view point. By mapping all projection directions onto the surface of a sphere and then mapping the sphere onto the surface of a cube, they reduce this problem to a segment intersection problem in 2D. Using known techniques from [1, 8], they find the silhouette in time linear in the size of the output.

Our results. In general the field has concentrated in computing the silhouette efficiently, or reconstruction and/or recognition of a polyhedron from a given set of silhouettes. In contrast to this, in this paper we do not consider the view point as given or fixed, instead we ask the question how to choose it suitably. Thus we consider the problem of given a polyhedron and given some desired property of the silhouette, how easy is it to find one or all projections that have the property?

This question is motivated by numerous applications of the silhouettes mentioned before. Two applications specifically benefit from the ability to bring certain features such as edges or faces to the silhouette. In **quality control** of a manufacturing process such as casting, we can check for flaws such as air pockets by examining whether each edge is a smooth and continuous line. This can be done efficiently if edges appear on the silhouette using video cameras to acquire the silhouette of the part. In **visualization**, crucial features should be forced to the silhouette to make them easily detectable. Also, if features are to be labeled it is advantageous to move them to the silhouette, since the outside area allows for space to place labels.

In this paper we consider how to select a projection of a polyhedron such that the silhouette satisfies certain properties. A straightforward approach to doing so is to compute all possible projections (depending on the conditions imposed on the silhouette, there is usually only a finite number of connected regions of view points that have these properties). Brunet et al. considered the case of computing the viewpoints from which the projections of a given polygonal chain projects a convex shadow [6]. They apply this special case to compute efficiently the occlusion properties in a 3-D scene.

This paper addresses the following question: Given a set of edges $\mathcal{E}$, a set of faces $\mathcal{F}$ and a set of vertices $\mathcal{V}$ of a convex polyhedron, how can we efficiently find all projections such that all elements in $\mathcal{E}$, $\mathcal{F}$, and/or $\mathcal{V}$ are on the silhouette under a perspective or orthogonal projection?

The straightforward algorithm for this problem would have a runtime of $O(k^3)$ for k edges (see Section 2 for a more detailed discussion). We show in Section 3 that the time can be improved to $O(k^2)$ by using geometric duality and transversal theory. In Section 4, we develop an adaptive algorithm in terms of the number of connected paths l formed by the edges, with time complexity $O(k \log k + kl)$. We conclude in Section 5 with open problems.

2 Preliminaries

A *convex polyhedron* is the intersection of finitely many half-spaces. We will not consider non-convex polyhedra in this paper and hence occasionally drop "convex" from now on. A *face/edge/vertex* of the convex polyhedron is the maximal connected set of points which belong to exactly one/exactly two/at least three planes that support these half-spaces. Every polyhedron with n vertices has $\theta(n)$ edges and $\theta(n)$ faces. Throughout this paper, we assume that the given polyhedron is fully-dimensional and that the origin o is inside the polyhedron. A *perspective projection* is defined relative to a point p, whereas an *orthogonal projection* is defined relative to a point at infinity, i.e., a direction d_p. A face f of a convex polyhedron with supporting plane π is *visible* with respect to a view point p (possibly at infinity) if the normal vector of f has a positive inner product with the vector from the origin to p (which is the direction vector d_p for the orthogonal projection). The *projection* of a polyhedron with respect to a view point p is the set of all those faces visible from p. An edge or vertex is in the projection if and only if at least one incident face is in the projection.

The *silhouette* of a projection from view point p is the set of all those edges for which one incident face is in the projection and the other face is not. In particular, note that we do not consider an edge to be on the silhouette if its projection is the degenerate case of a single point. The *shadow boundary* is the set of edges of the silhouette that are incident to the unbounded region. For a convex polyhedron, the notion of silhouette and shadow boundary is identical.

In the following, we will study how to find all view points for which a given set of edges is in the silhouette. Note that the set of such view points is in general not connected. We let a *viewing region* be a maximal connected region for which all points in it are view points with the desired property. Note that since projections in which edges project to points are considered degenerate and hence not to be proper view points, the viewing regions are always open sets.

First we study when exactly is an edge on the silhouette. Assume edge e is incident to faces f_1 and f_2, which are defined by half-spaces h_1 and h_2 with supporting planes π_1 and π_2. For $i = 1, 2$, face f_i is visible from view point p if and only if p is not in half-space h_i (recall that the origin is inside the polyhedron and hence belongs to all half-spaces of all faces). Edge e is on the silhouette if and only if exactly one of its incident faces is visible from p, so p must be in exactly one of the half-spaces h_1 and h_2. Thus, p belongs to $(h_1 \cap \overline{h_2}) \cup (h_2 \cap \overline{h_1})$ (or more precisely, to the maximal open set contained within this set), which is a double-wedge formed by planes π_1 and π_2.

3 Perspective projections

In this section, we study how to find efficiently all view points of a convex polyhedron such that a given set of edges, vertices and/or faces is on the silhouette under perspective projections. Our results rely heavily on duality theory and transversal theory, which we review in Section 3.2.

3.1 Computing perspective projections from plane arrangements

A straightforward approach to find all view points from which a set $\mathcal{E}$ of edges is on the silhouette is to compute the intersection of all the double-wedges associated with the edges in $\mathcal{E}$. In general the intersection of k double-wedges may have as many as $\theta(k^3)$ connected components under perspective projections. Therefore, finding all projections with certain properties can be done in $O(k^3 T)$ time, where T is the time to check whether a projection from a given view region has the desired property. In what follows we use duality theory to improve on this.

3.2 Geometric duality and transversal theory

Given a point $p = (a, b, c)$, the dual of the point is a plane $dual(p) = \{(x, y, z) : ax + by + cz = 1\}$. For a plane $\pi = \{(x, y, z) : ax + by + cz = d\}$, the dual is a point $dual(\pi) = (a/d, b/d, c/d)$. If π passes through the origin, then $dual(\pi)$ is at infinity. Recall that a point p lies in plane π if and only if $dual(\pi)$ lies in $dual(p)$. One can also easily prove the following observation:

Lemma 1. *Let π be a plane that does not contain the origin o, and let p be a point that is not the origin o. Then π intersects the line segment $[o, p]$ if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$.*

For an edge e, the dual is defined as the line segment $[dual(\pi_1), dual(\pi_2)]$, where π_1 and π_2 are the planes supporting the two incident faces of e. Pop et al. [20] made the following crucial observation.

Lemma 2. [20] *An edge e of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects $dual(e)$.*

A *geometric transversal* is an affine subspace of $\mathbb{R}^d$, such as a point, line, plane, or hyper-plane, that intersects every member of a family of convex sets. The set of [point, line, plane, or hyper-plane] transversals of a family forms a topological space in $\mathbb{R}^d$. Geometric transversal theory concerns the complexity and efficient computation of this topological space, especially in 2D and 3D. We will use the following result:

Theorem 1. ([9], see also [11], Theorem 5.6) *Let $\mathcal{A}$ be a family of n compact convex polytopes in 3D with a total of n vertices. All plane transversals of $\mathcal{A}$ can be found in $O(n^2\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function. If $\mathcal{A}$ consists of n line segments, then all plane traversals can be found in $O(n^2)$ time.*

We combine duality with transversal theory. Interestingly enough, the theorem above in turn uses dual geometric space (thus returning to the primal space) and analyzes double-wedges; it would thus be possible to express our algorithm directly in terms of double-wedges by tracing the results from transversal theory. We will not do this here for brevity and clarity's sake.

3.3 View point selection algorithm

Lemma 2 characterizes when an edge is on the silhouette of a projection. Combining this with transversal theory gives an algorithm to find all projections with a given set of k edges in $O(k^2)$ time. We apply the same approach to obtain an algorithm for given sets of vertices and faces. We first need to clarify what it means for a vertex or a face to be on the silhouette. This is relatively straightforward for a vertex, which is on the silhouette if and only if two incident edges are on the silhouette. The notion of a face being on the silhouette is not entirely obvious, since the silhouette by nature consists of line segments, so the entire face cannot be on it. However, for the purpose of displaying the face "near" the silhouette, the following definition seems appropriate: A face f is considered to be on the silhouette from view point p if and only if f is visible from p and at least one edge of f is on the silhouette.

As in Lemma 2, we characterize when a vertex or a face is on the silhouette. Assume that v is a vertex, and let $f_1, \ldots, f_l$ be the faces, in circular order, adjacent to the vertex v. In dual space, the dual of v is a plane and the duals of the planes supporting $f_1, \ldots, f_l$ are points in $dual(v)$. So the dual points of the planes of $f_1, \ldots, f_l$ are co-planar and thus form a polygon, which we call the *dual polygon* associated with vertex v. Observe that this dual polygon is convex as we assume that the origin is inside the polyhedron.

Lemma 3. *A vertex v is on the silhouette from view point p if and only if its associated dual polygon is intersected by dual(p).*

Proof. The polygon associated with v consists exactly of the union of the dual of the edges incident to v. If $dual(p)$ intersects this polygon, then it intersects exactly two edges incident to v. These two edges are then on the silhouette, and in consequence, v is also on the silhouette. $\square$

For a face f, let $f_1, f_2, ..., f_l$ be the faces adjacent to f and let $\pi, \pi_1, \ldots, \pi_l$ be the planes that support $f, f_1, \ldots, f_l$. Define the *dual polyhedron* of face f to be the convex hull of $dual(\pi), dual(\pi_1), \ldots, dual(\pi_l)$ (see Figure 1).

Lemma 4. *A face f with supporting plane π of a convex polyhedron is on the silhouette from view point p if and only if $dual(p)$ intersects both the line segment $[o, dual(\pi)]$ and the dual polyhedron associated with f.*

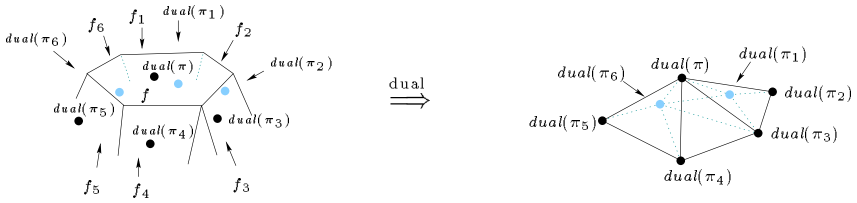

Fig. 1. The dual polyhedron associated with a face.

Proof. Let P' be the dual polyhedron of f. This polyhedron has one vertex v_f for f, which is incident to all other vertices. Now, if $dual(p)$ intersects P', then it separates the vertices of P' into two groups, and in particular intersects some of the edges incident to v_f, since all vertices are incident to v_f. Therefore, some of the edges of f are on the silhouette if and only if $dual(p)$ intersects P'.

For f itself to be on the silhouette, we additionally need that f is visible (i.e., not occluded) from view point p. This holds if and only if the plane through f separates the origin o (which is inside the polyhedron) from p. By Lemma 1, this holds if and only if $dual(p)$ intersects the line segment $[o, dual(\pi)]$. $\square$

Using the above lemmas, in combination with transversal theory, we can now compute all projections that have a given set of features on the silhouette.

Theorem 2. *Let P be a convex polyhedron with n vertices, and let $\mathcal{E}, \mathcal{V}$, and $\mathcal{F}$ be sets of edges, vertices and faces, respectively. All view points from which all edges in $\mathcal{E}$, all vertices in $\mathcal{V}$, and all faces in $\mathcal{F}$ are on the silhouette under perspective projections can be found in $O(n^2\alpha(n))$ time. The time reduces to $O(|\mathcal{E}|^2)$ if only edges are specified.*

Proof. Compute the dual of the edges in $\mathcal{E}$, the dual polygons associated with vertices in $\mathcal{V}$ and the dual polyhedra associated with faces in $\mathcal{F}$. Also, for each face $f \in \mathcal{F}$, compute the line segment $[o, dual(\pi)]$, where π is the plane supporting

f. Now find all plane transversals that intersect these convex objects. By the above lemmas this gives exactly the perspective projections for which the features are on the silhouette.

Every edge creates one line segment to be transversed; by Theorem 1 we can find the projections for a set of edges in $O(|\mathcal{E}|^2)$ time. For a vertex v, the associated polygon has *degree*(v) many vertices, and for a face f, the associated polyhedron has *degree*$(f) + 1$ many vertices. For a set of vertices and faces, the total size of the associated polygons and polyhedra can be at most the number of edges and faces in P, which is $O(n)$. Hence, by Theorem 1 we can find all projections for a set of edges, vertices and faces in $O(n^2\alpha(n))$ time. □

4 Orthogonal projections

In this section, we show how to compute efficiently all orthogonal projections such that a given set of edges is on the silhouette. This can be done with the same approach as in Theorem 2 (i.e., using duality theory and transversal theory.) However, a much simpler approach also works. Since the location of the origin is irrelevant in an orthogonal projection, we can identify directly the wedge for each edge and translate it such that all wedges intersect in one point. Then the hyperplane arrangement defined by the k wedges has only $O(k^2)$ cells, and we can thus find all projections in $O(k^2)$ time.

We improve on this further to give an algorithm that is adaptive in the number of paths formed by the set of edges. We study the case of one path in Section 4.1 and the case of many paths in Section 4.2. Finally, the ability to search all $O(k^2)$ cells of the arrangement allows us more flexibility in choosing projections; we study in Section 4.3 how to choose projections such that certain edges are *not* on the silhouette.

4.1 One connected path

If a set of edges forms a path in the polyhedron, then it is easier to compute viewing regions for which they are on the silhouette, mostly because (as we will show) there are only two viewing regions. This is not the case for arbitrary edges. For example Figure 2 shows a polyhedron where we have at least four viewing regions with edges e_1, e_2, e_3 on the silhouette. Two views from two different viewing regions are shown in (a) and (c); the other two viewing regions are origin-symmetric to the ones illustrated here. We now show that there are only two regions from which the path can be realized[1].

Theorem 3. *Given a path of k edges on a convex polyhedron P, there are only two viewing regions from which all k edges of the path are in the silhouette of P, and we can find them in $O(k \log k)$ time.*

Proof. Let the path consist of edges $e_1, \ldots, e_k$. For $1 \leq i \leq k$, let f_i be the face to the left of edge e_i, where "to the left" is taken with respect to walking along the

[1] This theorem is implicitly assumed without proof in [6].

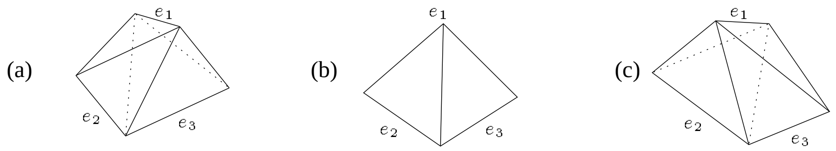

Fig. 2. The polyhedron has four viewing regions for edges e_1, e_2, e_3. (a) One incident face of e_1 is visible. (b) A rotation such that e_1 reduces to a point; this view point is on the boundary between two viewing regions. (c) The other incident face of e_1 is visible.

path from e_1 to e_k. Let f_i' be the other face incident to e_i. The crucial observation is that if the path is on the silhouette, then we either see all of $f_1, \ldots, f_k$ or all of $f_1', \ldots, f_k'$. To prove this, let v be the common vertex of e_1 and e_2. The clockwise order of faces around v is then f_1, (possibly) some other faces, f_2, f_2', (possibly) some other faces, f_1' (see Figure 3). Since the visible incident faces of v are

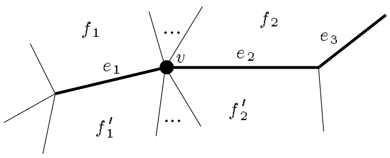

Fig. 3. The order of faces around a vertex.

connected, and for each of $\{f_1, f_1'\}$ and $\{f_2, f_2'\}$ exactly one is invisible, there are only two possibilities: either all of $f_1, \ldots, f_2$ are visible (and the others are invisible), or all of $f_2', \ldots, f_1'$ are visible (and the others are invisible). So either $\{f_1, f_2\}$ are visible or $\{f_1', f_2'\}$ are visible, but it is not possible (for example) that f_1 and f_2' are visible. Assume f_1 and f_2 are visible. Repeating the argument for e_2 and e_3, this shows that f_3 must also be visible, and so on, so $f_1, \ldots, f_k$ are all visible (and $f_1', \ldots, f_k'$ are invisible). Alternatively, if f_1' and f_2' are visible, then all of $f_1', \ldots, f_k'$ are visible (and $f_1, \ldots, f_k$ are invisible).

Recall that a face f is visible if and only if the view point is not in the half-space that defined the face. Thus, the view points from which $f_1, \ldots, f_k$ are all visible and $f_1', \ldots, f_k'$ are all invisible are defined as the intersection of $2k$ half-spaces. This defines one viewing region. A second viewing region is the one from which $f_1', \ldots, f_k'$ are all visible and $f_1, \ldots, f_k$ are invisible. This region is again the intersection of half-spaces —the opposite half-spaces as those for the first viewing region. The half-spaces can be found in $O(k)$ time, and their intersection can be computed in $O(k \log k)$ time (see e.g. [21]). There are no other viewing regions. Thus, all viewing regions can be computed in $O(k \log k)$ time. $\square$

Note that the above theorem applies to both orthogonal and perspective projections, and it is hence possible to find all viewpoints for which a path P with k edges is on the silhouette in $O(k \log k)$ time for both orthogonal and perspective projections. In contrast, the results in the next subsections apply only to orthogonal projections.

4.2 Multiple paths

In this section, we show how to use the above results to improve the time complexity in the case when k edges are not all disjoint. Assume that k edges are in l connected components which we refer to as $\mathcal{P}_1, \ldots, \mathcal{P}_l$; obviously $l \leq k$. From Theorem 3 we know how to compute all projections from which $\mathcal{P}_i$ is on the silhouette. We now show that for orthogonal projections, we can intersect these viewing regions in $O(k \log k + kl)$ time, which is an improvement over the $O(k^2)$ result of Section 3 if $l = o(k)$. Observe that for orthogonal projections, the set of view points from which e is on the silhouette is translation-invariant, since the view points are at infinity and correspond to directions. Hence, we are allowed to translate the double-wedge arbitrarily, and in particular we may assume that the intersection of the two planes contains the origin.

Theorem 4. *Given l disjoint paths of a convex polyhedron P, we can find all orthogonal projections of P such that all k edges of the paths are on the silhouette in $O(k \log k + kl)$ time.*

Proof. From Theorem 3 we know that for $1 \leq i \leq l$ there are exactly two viewing regions from which $\mathcal{P}_i$ is on the silhouette. Since we are considering orthogonal projections, these two viewing regions are two disjoint convex cones, say C_i^+ and C_i^-, and after a suitable translation we may assume that the apex of each cone is at the origin. Let $C_i = C_i^+ \cup C_i^-$ be the *double-cone* for path $\mathcal{P}_i$, and set $C^* = \bigcap C_i$; the desired viewing regions are then exactly the connected components of C^*.

In general, l double-cones may have $\theta(l^3)$ connected components in their intersection. For double-cones with origin at the apex this reduces to $O(l^2)$, but computing all connected components of C^* directly is still too slow. We therefore consider a projection of the double-cones onto a 2D surface (see [4, 5]). Consider a unit cube D centered at the origin. To compute C^*, it suffices to compute the intersection of C^* with each face of D. We explain how to do this in the following for one face f of D only.

For any i, where $1 \leq i \leq l$, both C_i^+ and C_i^- can intersect f, but these intersections are disjoint (see Figure 4). Set $B_i = C_i \cap f$; then B_i is a single convex polygon or the union of two convex polygons. We call each B_i a *cone polygon*. Let $B^* = \bigcap B_i$, then each connected component of B^* corresponds to one viewing region.

To compute B^*, we compute the arrangement $\mathcal{A}$ of the cone polygons $B_1, \ldots, B_l$, which has at most $2l$ convex polygons with a total of at most $2k$ edges. This can be done in $O(k \log k + I)$ time where I is the number of intersecting points [3, 7]. Within the same time bound, we can also compute the planar graph G defined by this arrangement (see Figure 5(a)). G has $O(k+I)$ vertices, edges and faces. Now we want to find all cells in the arrangement $\mathcal{A}$ that belong to all cone polygons. To do so, we compute a *modified directed dual graph* G' of G by computing the dual graph of G and replacing each edge by a directed 2-cycle (see Figure 5(b)). Note that here we use the term "dual" in the graph theoretical sense, which is distinct from the geometric duality used before.

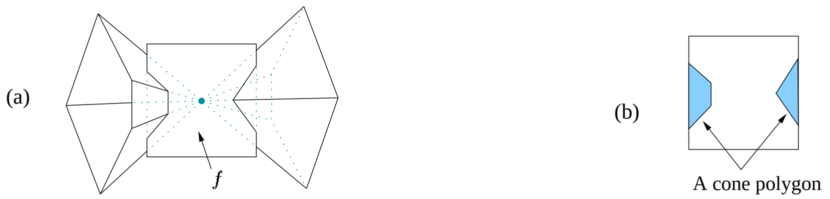

Fig. 4. (a) Intersection of a double-cone C_i with a face f. (b) The corresponding cone polygon B_i has two disjoint components.

Each vertex in G' is a cell in $\mathcal{A}$ and each directed edge e in G' corresponds to entering or leaving a polygon of $B_1, \ldots, B_l$. We store with each edge of G' whether traversing this edge means entering or leaving a cone polygon. Finding all cells for which we are inside all l cone polygons can then be done by traversing the graph G' in such a way that all vertices are visited (e.g. with a DFS-traversal) and maintaining a counter of the number of cone polygons that the currently visited vertex is in. Since G' has $O(k + I)$ vertices and edges, this can be done in $O(k+I)$ time. The time complexity of our algorithm hence is $O(k \log k + I)$. To find an upper bound on I, observe that there are $O(l)$ convex polygons of $O(k)$ edges total. Each edge can intersect each convex polygon at most twice, so $I \in O(kl)$ (and examples can be found where this is tight [2]). So the run-time of our algorithm is $O(k \log k + kl)$. $\square$

Fig. 5. (a) The arrangement $\mathcal{A}$; the desired viewing regions are shown shaded. (b) The corresponding modified dual graph. Some edges and vertices have been omitted for clarity's sake.

4.3 Hiding Edges

Another application of view point selection is industrial design, where we might wish to make certain features easily visible or prominent, while some other features (such as service trap doors or unsightly wiring) should be hidden. Thus we would like to force edges, vertices or faces *not* to be on the silhouette; we say that these features are *hidden from the silhouette*. Note that a face/vertex is hidden from the silhouette if and only if all its incident edges are hidden from the silhouette, so it suffices to explain how to hide edges.

Lemma 5. *The set of all view points from which a set of k edges is not visible under an orthogonal projection can be computed in $O(k^2)$ time.*

Proof. Each double-wedge from which an edge is visible also defines, by its complement, a double-wedge from which the edge is *not* visible. Since we are considering orthogonal projections, we can translate all double-wedges such that all hyperplanes that define them intersect the origin. Therefore, the hyperplane arrangement now has only $O(k^2)$ cells and can be computed in $O(k^2)$ time. By using a traversal technique similar to the one in Section 4.2, we can check whether there is any cell in which all edges are hidden in $O(k^2)$ time. □

Note that we can at the same time force some edges into the silhouette and hide some other edges from the silhouette. With the same proof as in Lemma 5, this can be done in $O(k^2)$ time. However, hiding edges unfortunately cannot easily be made adaptive in the number of paths that these edges form. The main obstacle is that Theorem 3 ("there are only two viewing regions") does not necessarily hold for hiding edges in one path.

5 Conclusions

In this paper, we studied the question of how to find all view points of a polyhedron that force a given set of edges onto the silhouette of a convex polyhedron, and gave an efficient algorithm to do so, as well as an adaptive algorithm in terms of the number of connected paths formed by the edges for orthogonal projections. A number of open questions remain:

- Transversal theory allowed us to force not only edges, but also vertices and faces, onto the silhouette but at a higher cost since the size of the objects to be transversed is proportional to the degree of the vertex or face involved. Is it possible to improve on this, say find all projections with k vertices or k faces on the silhouette in $O(k^2)$ or even $O(k)$ time?
- The general adaptive algorithm (Section 4.2) applies only to orthogonal projections. Does a similar result hold for the perspective projections?
- Hiding edges in a perspective projection can effectively be phrased as a transversal problem again, but Theorem 1 cannot be applied, since the resulting shapes are not convex and compact. So how can we efficiently find all view points from which a given set of edges is hidden?
- Given a family of edge sets $\mathcal{E}_1, \ldots, \mathcal{E}_s$, how quickly can we find all projections such that for each set $\mathcal{E}_i$, at least one (but not necessarily all) of the edges in $\mathcal{E}_i$ are on the silhouette?
- Rather than specifying the edges that must be on the silhouette, we might be interested in the number of such edges. So given a number k, how can we find all projections such that the silhouette has exactly k edges?

Last but not least, all our results were for convex polyhedra. What are efficient algorithms for the same problems for non-convex polyhedra?

References

[1] P. K. Agarwal and M. Sharir. Applications of a new space partitioning technique. *Disc. Comp. Geom.*, 9:11–38, 1993.

[2] B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Comp. Geom.*, 8:139–149, 1997.
[3] I. J. Balaben. An optimal algorithm for finding segments intersections. *ACM Symp. Comp. Geom.*, pages 211–219, 1995.
[4] F. Benichou and G. Elber. Output sensitive extraction of silhouettes from polygonal geometry. In *7th Pac. Conf. Comp. Grap. Appl.*, pages 60–69, 1999.
[5] P. Bose, F. Gomez, P. Ramos, and G. T. Toussaint. Drawing nice projections of objects in space. *J. Vis. Comm. Image Represent.*, 10:155–172, 1999.
[6] P. Brunet, I. Navazo, J. Rossignac, and C. Saona-Vazquez. Hoops: 3d curves as conservative occluders for cell-visibility. *Comp. Grap. Forum*, 20(3):431–442, 2001.
[7] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *J. ACM*, 39:1–54, 1992.
[8] B. Chazelle, M. Sharir, and E. Welzl. Quasi-optimal upper bounds for simplex range searching and new zone theorems. *Disc. Comp. Geom.*, 8:407–429, 1992.
[9] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Disc. Comp. Geom.*, 4:311–336, 1989.
[10] A. Efrat, L. Guibas, O. Hall-Holt, and L. Zhang. On incremental rendering of silhouette maps of a polyhedral scene. In *ACM-SIAM Symp. Disc. Alg.*, pages 910–917, 2000.
[11] J. E. Goodman, R. Pollack, and R. Wenger. Geometric transversals theory. In *New Trends in Discrete and Computational Geometry*. Springer-Verlag, 1993.
[12] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.
[13] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object? *Comp. Vis. Image Under.*, 67(1):81–87, 1997.
[14] A. Laurentini. Introducing a new problem: Shape-from-silhouette when the relative positions of the viewpoints is unknown. *IEEE Pat. Ana. Mach. Int.*, 25(11):1484–1493, 2003.
[15] Melles Griot Corporation. *Machine Vision Guide*, 2003. http://www.mellesgriot.com/products/machinevision/lif_3.htm.
[16] J. Miller. Low-cost in-process machine vision gauging system. Technical report, Dept. Elec. Comp. Eng., Univ. Michigan-Dearborn, April 1998. http://www.engin.umd.umich.edu/ceep/reports/96-97/jmiller.html.
[17] F. Mokhtarian. Silhouette-based occluded object recognition through curvature scale space. *J. Mach. Vis. Appl.*, 10(3):87–97, 1997.
[18] J. A. Muratore. Illumination for machine vision. http://www.pinnaclevision.co.uk/illum02.htm.
[19] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *Intl. J. Comp. Vis.*, 5(2):137–160, 1990.
[20] M. Pop, G. Barequet, C. A. Duncan, M. T. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation and applications. In *ACM Symp. Comp. Geom.*, pages 60–68, 2001.
[21] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1995.
[22] R. D. Schiffenbauer. *A Survey of Aspect Graph*. PhD thesis, Dept. Comp. Inf. Sci., Polytechnic Univ., New York, 2001.
[23] Siemens. *Outline inspection with SIMATIC VS 110*. Product literature. http://www.ad.siemens.de/dipdata/mk/pdf/e20001-a60-p285-x-7600.pdf.
[24] SIGHTech Vision Systems. *Eyebot Application, Inspecting Hard Disk Media*. Product literature. http://www.sightech.com/hard_disk_app_note.pdf.
[25] K. Sugihara. *Machine Interpretation of Line Drawing*. MIT Press, 1986.