Efficient View Point Selection for Silhouettes of Convex Polyhedra ⋆,⋆⋆

Therese C. Biedl [a], Masud Hasan [b,1,2], Alejandro López-Ortiz [a]

[a] *Cheriton School of Computer Science*

University of Waterloo, Waterloo, Ontario N2L 3G1, Canada

[b] *Department of Computer Science and Engineering,*

Bangladesh University of Engineering and Technology

Dhaka-1000, Bangladesh

Abstract

Silhouettes of polyhedra are 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.

Preprint submitted to Elsevier Science *9 December 2010*

Key words: Convex polyhedra, duality, geometric transversal, projections, silhouette, views

---

⋆ Research supported by NSERC.

⋆⋆An earlier version appeared in *Proc. 29th International Symposium on Mathematical Foundation of Computer Science (MFCS'04)*, volume 3153 of *Lecture Notes in Computer Science*, pp. 735-747, Prague, Czech Republic, August 2004.

Email addresses: biedl@uwaterloo.ca (Therese C. Biedl), masudhasan@cse.buet.ac.bd (Masud Hasan), alopez-o@uwaterloo.ca (Alejandro López-Ortiz).

[1] The research was done while the author was a PhD student at the University of Waterloo.

[2] Corresponding author. Phone: +880-155-663-4106, Fax: +880-2-966-5612

1 Introduction

Polyhedra are solids in 3D space. When looking at a polyhedron from a view point, the eye or camera computes a 2D 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 the edges for which exactly one incident face is visible; the two concepts describe the same set for convex polyhedra.

The problem considered in this paper has potential application in machine vision, image recognition, reconstruction from images, and computer graphics, as describe below. Additionally, the generality of the primitive operations proposed here make it suitable for novel applications, such as security, see also [5].

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 [22]. For assembling purposes, silhouettes are used to compute the boundary and the

orientation of the mechanical parts to be picked up by robot for assembly [21]. Silhouettes are also used for quality control [28,29], object recognition [28], illuminating critical features, and others.

For **image recognition**, researchers consider the topological graph representation of projections of polyhedra, which are called *characteristic views*, or simply *views* [23]. Similar topological representations are also considered for the silhouette [14,19,27]. Silhouettes computed from the projection of an object can be matched against stored pre-computed views, hence aiding in recognition of the object (see [23,30] for similar argument). This method has the advantage that the views from two nearby view points likely result in the same view, which makes the system robust under small positional errors. The characteristic views of a polyhedron are better known as *aspect graphs*. See [26] for a detail survey on aspect graphs.

Reconstruction from images concerns the problem of approximately reconstructing a 3D object from one or more images [7,8,20,31]. Instead of full images of an object, often only silhouettes are used in a process called *volume intersection* [7,8,20].

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 [18]. When rendering the object, it often suffices to render only the silhouette edges, which can result in substantial speedup [24]. This is because

for a polyhedron the number of silhouette edges is usually much smaller than the total number of edges [17].

1.1 Results on silhouette computation

The computation of silhouettes has been studied extensively both in the computational geometry community and in the computer graphics community. Pop et al. [24] gave an algorithm for perspective projections that efficiently maintains the silhouette of polyhedra during arbitrary changes of the view point. They use geometric duality (see also Section 3.2) to develop a practical and efficient heuristic to maintain the corresponding visibility properties.

Efrat et al. [14] 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. From there they find the silhouette in time linear to the size of the output.

1.2 View point selection

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 given or fixed, instead 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 on 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.

There are some studies on how to compute "nice" projections of polygonal objects [6,15] and 3D graph drawings [11], where criteria for "nice" include crossings among the edges, monotonicity of polygonal chains, and coincidences among edges and vertices, and the resulting nice projections are in terms of set of view points or regions.

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 view points from which the projections of a given polygonal chain projects a convex shadow [9]. They apply this special case to compute efficiently the occlusion properties in a 3D scene.

More specifically, 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 perspective or orthogonal projections? By "all projections" we too mean all possible view points or regions from which the projection has the desired property. An example in Figure 1 shows four views corresponding to four viewing regions of a polyhedron that would be generated by our algorithm. The edges e_1, e_2 and e_3 are on the silhouette. Observe that the views in (a) and (b) are opposite to the views in (c) and (d) respectively.

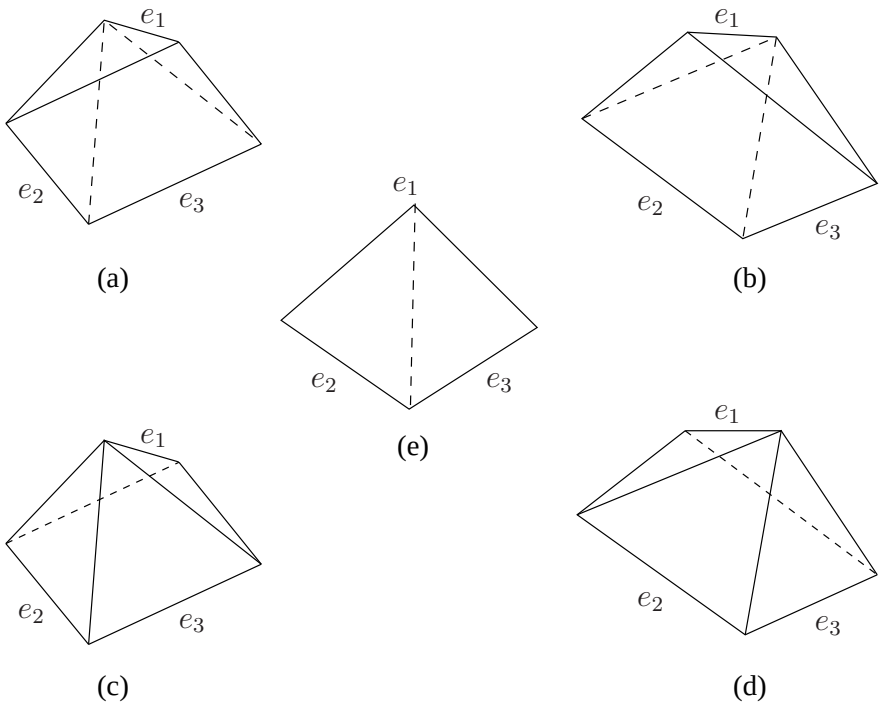


Fig. 1. The polyhedron has four views when the edges e_1, e_2 and e_3 are on the silhouette. (a,b,c,d) The four views. (e) The degenerate projection common to all four viewing regions.

The projection in (e) is such that e_1 reduces to a point; the view point for this degenerate projection is on the common boundary of all four resulting viewing regions.

Recall that a straightforward approach would be to compute all regions where projections are different, which amounts for our problem to finding cells in an arrangement of hyperplanes. One approach in the literature to do this is the reverse search technique by Avis and Fukuda [27]. However, this technique is not directly applicable to our problem. The reverse search algorithm expects that at least one object among all that are to be enumerated is given. For our

problem, finding an initial cell among the resulting ones would not be easy and would take as much time as to compute the entire solution in the case when only edges are given and they form a single path (see Section 4.1 for detail for this case of our problem). Moreover, in general the time complexity of reverse search is $O(d \cdot a \cdot o)$, where d is the maximum degree of "adjacency" among the objects, a is the time required for finding an adjacent object of a given one, and o is the output size (see [2] for detail). In an implementation of reverse search for our problem for k edges, the value of d and o would be k and k^2, respectively, and would thus result in an inefficient solution.

On the other hand a straightforward algorithm would have a runtime of $O(k^3)$ for k edges (see Section 3 for a more detailed discussion). We show in Section 3 that the time complexity can be improved to $O(k^2)$ by using geometric duality and transversal theory. In Section 4, we study the special case of edges in l connected paths, for which we develop an adaptive algorithm with time complexity $O(k \log k + kl)$ for orthogonal projections. 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 from the origin to a point at infinity. The *projection* of a polyhedron with respect to a view point p (possibly at infinity) is defined as the set of all those faces visible from p.

The *silhouette* of a projection from view point p is the set of edges for which one incident face is visible and the other one 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 on 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.

We precisely define 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 $(h_1 \cap \overline{h_2}) \cup (h_2 \cap \overline{h_1})$ 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 geometric duality and transversal theory, which we review in Subsection 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, and they can be computed (by computing the *arrangements* of $2k$ hyperplanes) in $O(k^3)$ time[13]. Therefore, finding all projections with certain properties can be done in $O(k^3T)$ time, where T is the time to check whether a projection from a given viewing region has the desired property. In what follows we use geometric duality 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 $\mathit{dual}(p) = \{(x, y, z) : ax + by + cz = 1\}$. For a plane $\pi = \{(x, y, z) : ax + by + cz = d\}$, the dual is a point $\mathit{dual}(\pi) = (a/d, b/d, c/d)$. If π passes through the origin, then $\mathit{dual}(\pi)$ is at infinity. Recall that a point p lies in plane π if and only if $\mathit{dual}(\pi)$ lies in $\mathit{dual}(p)$. We will make use of a simple 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 $\mathit{dual}(p)$ intersects the line segment $[o, \mathit{dual}(\pi)]$.*

PROOF. Let $\pi = \{(x, y, z) : ax + by + cz = 1\}$ and $p = (p_x, p_y, p_z)$. Plane π intersects $[o, p]$ if and only if $ap_x + bp_y + cp_z \geq 1$, which in turn holds if and only if the plane $\{(x, y, z) : p_x x + p_y y + p_z z = 1\}$ has o on one side and (a, b, c) on the other side, thus $\mathit{dual}(p)$ intersects $[o, \mathit{dual}(\pi)]$. □

For an edge e, the dual is defined in terms of the dual of the facets defining e. More formally, let f_1 and f_2 be the facets defining e, and let π_1 and π_2 be the planes supporting f_1 and f_2 respectively. Then we define the dual of the edge e as the line segment $[dual(\pi_1), dual(\pi_2)]$. This definition of the dual of an edge depends upon the assumption that the origin is inside of the polyhedron and was actually given by Pop et al. [24], where they also define the dual of an edge when the origin is not inside the polyhedron. Observe that both e and its dual are line segments and not infinite lines. Also note that if f_1 or f_2 changes while e remains the same, the dual of e would also change. Pop et al. [24] prove the following crucial lemma.

Lemma 2 [24, Theorem 2] *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* to a family of convex set in d-dimension is an affine subspace, such as a point, line, plane, or hyper-plane, that intersects every member of the family. Geometric transversal theory, more familiarly in two and three dimensions, concerns the complexity and efficient computation of the space of all [point, line, plane, or hyper-plane] transversals. We will in particular deal with plane transversals of a family of line segments, convex polygons, and convex polyhedra in 3D. The complexity of such plane transversals is the space of all planes that intersect every member of that family. For such a family of total n vertices, the complexity of all plane transversals is $O(n^2\alpha(n))$, where $\alpha(n)$ is the inverse Ackerman function. The computation of

these plane transversals is to compute their entire space and present them in suitable data structures [16], which can be done optimally as mentioned in the following theorem. See [12,16] for detailed discussions on geometric transversal theory.

Theorem 3 ([12], see also [16], Theorem 5.6) *Let $\mathcal{C}$ be a family of line segments, convex polygons, or convex polyhedra in 3D with a total of n vertices. All plane transversals of $\mathcal{C}$ can be found in $O(n^2\alpha(n))$ time, where $\alpha(n)$ is the inverse Ackerman function. If $\mathcal{C}$ 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, Theorem 3 in turn uses dual geometric space (thus returning to the primal space) and analyzes double-wedges. In dual space, the family of objects maps to an arrangement of hyperplanes in 3D, and a plane transversal maps to a point between the lower and upper envelopes of the arrangement. Thus, the problem of computing all plane transversals turns to finding all such points in dual space [16]. 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.

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[3] 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 $dual(v)$ is a plane and the duals of the planes supporting $f_1, \ldots, f_l$ are points in $dual(v)$. Two points in $dual(v)$ are connected by an edge which is the dual of the common edge of corresponding

[3] Instead of "at least one edge", "at least one vertex" could also be considered in this definition and the following related characterization and result could be achieved similarly.

two faces in primal space. Thus the dual of v is a plane and the duals of the adjacent edges and faces of v form a polygon in that plane. We call this polygon the *dual polygon* associated with vertex v.

Lemma 4 *The dual polygon O associated with vertex v is convex.*

PROOF. Consider any plane π in dual space. We will prove that if π intersects O, then it intersects exactly two edges of O. π corresponds to a view point p in primal space. (We consider that $\pi \neq dual(v)$, because the view point in primal space corresponding to $dual(v)$ is v, which is not a valid view point.) If v is not on the silhouette from p, then none of its adjacent edges are on the silhouette from p. If v is on the silhouette from p, then exactly two of its adjacent edges, say e_1 and e_2, are on the silhouette from p. By Lemma 2, the $dual(p)$, which is π, intersects, among the duals of the adjacent edges of v, exactly $dual(e_1)$ and $dual(e_2)$.

Corollary 5 *A vertex v is on the silhouette from view point p if and only if its associated dual polygon is intersected by dual(p).*

For a face f, proceed as follows: 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 $f, f_1, \ldots, f_l$, respectively. Define the *dual polyhedron* of face f to be the convex hull of $dual(\pi), dual(\pi_1), \ldots, dual(\pi_l)$.

Lemma 6 *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*.*

PROOF. Let P' be the dual polyhedron associated with f. $dual(p)$ intersects P' means it separates at least one vertex, say $dual(\pi_i)$, of P' from $dual(\pi)$. By Lemma 2 the edge between f and f_i is on the silhouette. 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 $[o, dual(\pi)]$. □

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 7 *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 edges 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 3 we can find the projections for a set of edges in $O(|\mathcal{E}|^2)$ time. For a vertex v, the associated polygon has $\mathit{degree}(v)$ many vertices, and for a face f, the associated polyhedron has $\mathit{degree}(f)+1$ 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 3 we can find all projections for a set of edges, vertices and faces in $O(n^2\alpha(n))$ time. □

Transversal theory allowed us to force not only edges, but also vertices and faces, on 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. Can the time complexities be improved in general?

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 3 (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 [23] and they can be found in $O(k^2)$ time [13,23].

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 features (i.e. edges, vertices, faces) 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. Figure 1 is such an example where we have four viewing regions for three edges (e_1, e_2, e_3) that are on the silhouette but do not form a single path.

We now show that there are only two regions from which the path can be realized [4].

[4] This theorem is implicitly assumed without proof in [9].

Theorem 8 *Given a path of k edges on a convex polyhedron P, there are only two viewing regions from which all k edges are on the silhouette of P, and we can find them in $O(k \log k)$ time.*

PROOF. Let the path consists 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 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 see either 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 ordering of faces around v is then f_1, (possibly) some other faces, f_2, f_2', (possibly) some other faces, and f_1' (see Figure 2). Assume that $e_1, \ldots, e_k$ are on the silhouette. Consider e_1 and e_2. Exactly one of f_1 and f_1' and one of f_2 and f_2' are visible, while the other two are invisible. Without loss of generality, assume that f_1 is visible. We shall prove that f_2' can not be visible. Observe that it is possible to have $f_1 = f_2$ or $f_1' = f_2'$, but not both. We shall also use the fact that for a convex polyhedron, a vertex on the silhouette has exactly two of its adjacent edges on the silhouette. By way of contradiction assume that f_2' is visible too. Then f_2 is invisible. Since f_1 is visible, it can not happen that $f_1 = f_2$. Now, among the faces around v in counter-clockwise order from f_2 to f_1 (including f_2 and f_1), there are both visible and invisible faces. Let f' and f'' be two such visible and invisible faces that are also adjacent. Let e' be the common edge of f' and f''. By definition, e' is on the silhouette. So, v has three adjacent edges e_1, e_2 and e' on the

silhouette, which is a contradiction. Therefore, f_1, f_2 are visible and f_1', f_2' are invisible. Repeating this for e_2 and e_3, we get that f_2 and f_3 are visible and f_2' and f_3' are invisible. We continue this for the next $k-2$ pairs of consecutive edges and get that $f_1, \ldots, f_k$ 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 viewing region is again the intersection of half-spaces (in fact, 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. [25]). There are no other viewing regions. Thus, all viewing regions can be computed in $O(k \log k)$ time. □

Note that the two viewing regions computed by the above theorem contains both finite and infinite view points and it is hence possible to find all view-

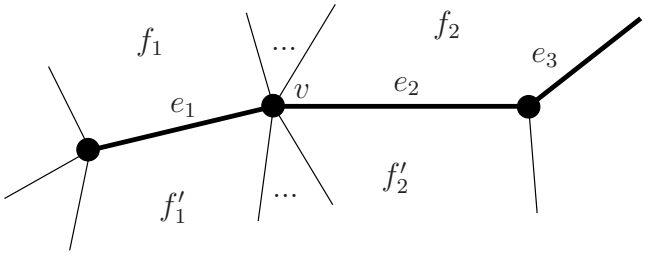


Fig. 2. The order of faces around a vertex.

points 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 8 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 is significantly smaller than k. As mentioned earlier, 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 9 *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. We assume that for $1 \leq i \leq l$, $\mathcal{P}_i$ has k_i edges. From Theorem 8, we know that for each i there are exactly two viewing regions from which $\mathcal{P}_i$ is on the silhouette and they can be computed in $O(k_i \log k_i)$ time. 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 they have origin as the common apex. Over all disjoint paths, the $2l$ such convex cones can be found in $\sum O(k_i \log k_i) \leq \sum O(k_i \log k) = O(k \log k)$ time.

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 (similar as in [4,6]). 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 3). 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.

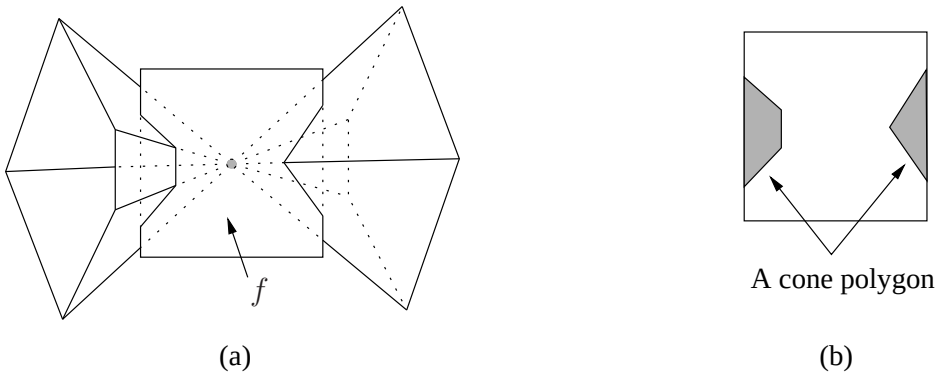
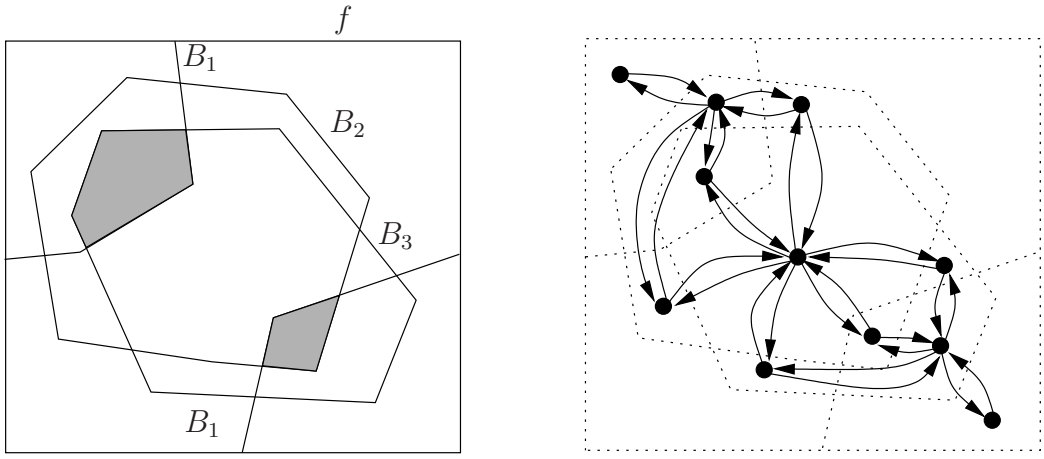


Fig. 3. (a) Intersection of a double-cone C_i with a face f. (b) The corresponding cone polygon B_i has two disjoint components.

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 intersection points [3,10]. Within the same time bound, we can also compute the planar graph G defined by $\mathcal{A}$ (see Figure 4(a)). G has $O(k + I)$ vertices, edges and faces. Now we want to find all cells in $\mathcal{A}$ 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 4(b)). Note that here we use the term "dual" in the graph-theoretic sense, which is distinct from the geometric duality used before.

Each vertex in G' is a cell in $\mathcal{A}$ and each directed edge e in G' corresponds to entering or leaving a cone 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) (b)

Fig. 4. (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.

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 [1]). So the run-time of our algorithm is $O(k \log k + kl)$. $\square$

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 vertex is hidden from the silhouette if and only if all its incident edges are hidden from the silhouette. So for hiding a vertex from the silhouette it suffices to explain how to hide edges.

Lemma 10 *The set of all view points from which a set of k edges is not on the silhouette under orthogonal projections can be computed in $O(k^2)$ time.*

PROOF. Each double-wedge from which an edge is on the silhouette also defines, by its complement, a double-wedge from which the edge is *not* on the silhouette. 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 from the silhouette in $O(k^2)$ time. □

Corollary 11 *The set of all view points from which a set of vertices is not*

on the silhouette under orthogonal projections can be computed in $O(n^2)$ time.

PROOF. All k vertices are not on the silhouette if and only if all of their adjacent edges are hidden from the silhouette, and there can be $O(n)$ such adjacent edges. □

Unlike a vertex, it is not true that a face is hidden from the silhouette if and only if all of its incident edges are hidden from the silhouette. Rather, a face is hidden from the silhouette if and only if it is invisible or it is visible and all of its incident edges are hidden from the silhouette. But this difference in the definitions is not very difficult to handle (see the following lemma).

Lemma 12 *The set of all view points from which a set of faces is not on the silhouette under orthogonal projections can be computed in $O(n^2)$ time.*

PROOF. For each face f in the given set we have two cones from which f is not on the silhouette. One cone is defined by the whole half-space from which f is invisible (i.e. the half-space that defines f). The other cone corresponds to the view directions from which f is visible and all of its adjacent edges are not on the silhouette. This cone is computed as follows. For each adjacent edge e of f there is exactly one wedge from which f is visible and e is not on the silhouette. After translating all such wedges to the origin, their intersection gives the desired cone for f.

The time required to compute the second cone for f (as discussed above) is no more than $O(d^2)$, where d is the number of edges of f. Since each edge counts twice over all faces of the polyhedron and since $\sum {d_i}^2 \leq (\sum d_i)^2$ for $d_i \geq 1$, the total time for computing the pairs of cones for all faces is $O(n^2)$. After having all these pairs of cones, using a technique similar to one in Section 4.2, we can compute their intersection and then find the cells in which all the faces are not on the silhouette in $O(k \log k + kl)$ time where l is the given number of faces and k is twice the number of edges in those faces. Since k is $O(n)$, our total time remains $O(n^2)$. $\Box$

Note that we can at the same time force some edges, vertices and faces on the silhouette and hide some other edges vertices and faces from the silhouette. With similar proofs as in the above lemmas and corollary, this can be done in $O(n^2)$ time.

Theorem 13 *Let P be a convex polyhedron with n vertices. Let $\mathcal{E}, \mathcal{E}'$ be sets of edges, $\mathcal{V}, \mathcal{V}'$ be sets of vertices, and $\mathcal{F}, \mathcal{F}'$ be sets of faces of P. All view points from which all elements of $\mathcal{E}, \mathcal{V}, \mathcal{F}$ are on the silhouette while all elements of $\mathcal{E}', \mathcal{V}', \mathcal{F}'$ are hidden under orthogonal projections can be found in $O(n^2)$ time. The time reduces to $O((|\mathcal{E}| + |\mathcal{E}'|)^2)$ if only edges are specified.*

Before we conclude this hiding section, we note that hiding edges unfortunately cannot easily be made adaptive in the number of paths that these edges form. The main obstacle is that Theorem 8 ("there are only two viewing regions")

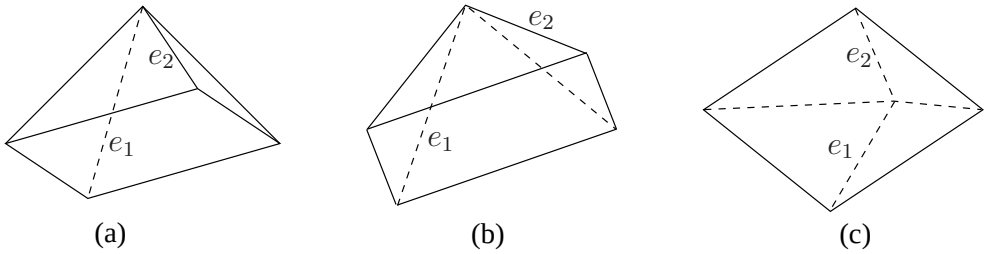


Fig. 5. The polyhedron has more than two viewing regions with the path consisting of edges e_1 and e_2 not on the silhouette. (a) e_1 is invisible and e_2 is visible. (b) The path is partially not on the silhouette because of e_2 on the silhouette. (c) Both e_1 and e_2 are invisible.

does not necessarily hold for hiding edges in one path. For example, consider the polyhedron in Figure 5 which is similar to that in Figure 1. The path consisting of e_1 and e_2 is not on the silhouette from at least four viewing regions. 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.

5 Conclusions

In this paper, we studied the question of how to find all view points of a polyhedron that satisfy certain properties. We focused on how to force a given set of edges on the silhouette of a convex polyhedron, and gave an efficient algorithm to do so, as well as an adaptive algorithm for orthogonal projections if the edges are in one or very few connected paths. The most immediate open problem is whether such an adaptive algorithm exists for perspective

projections? Also, can the time complexities be improved further?

Our study was motivated by the more general problem of finding all projections that have some desired property. Many questions deserve studying here:

- Hiding edges in a perspective projection can effectively be phrased as a transversal problem again, but Theorem 3 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?
- Recall that a vertex or face is considered to be on the silhouette if one of its incident edges is on the silhouette. This naturally raises the following question: 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 computing projections of non-convex polyhedra such that the silhouette satisfies given properties?

References

[1] B. Aronov and M. Sharir. The common exterior of convex polygons in the plane. *Computational Geometry: Theory and Applications*, 8(3):139–149, 1997.

[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete and Applied Mathematics*, 6(1–3):21–46, 1996.

[3] I. J. Balaben. An optimal algorithm for finding segments intersections. In *11th ACM Symposium on Computational Geometry*, pages 211–219, Vancouver, British Columbia, Canada, June 1995.

[4] F. Benichou and G. Elber. Output sensitive extraction of silhouettes from polygonal geometry. In *7th Pacific Conference on Computer Graphics and Applications*, pages 60–69, Seoul, Korea, October 1999.

[5] T. C. Biedl, M. Hasan, and A. López-Ortiz. Projections of polyhedra in optical physical security. Manuscript in progress.

[6] P. Bose, F. Gomez, P. Ramos, and G. T. Toussaint. Drawing nice projections of objects in space. *Journal of Visual Communication and Image Representation*, 10(2):155–172, 1999.

[7] A. Bottino, L. Jaulin, and A. Laurentini. Reconstructing 3D objects from silhouettes with unknown viewpoints: The case of planar orthographic views. In *8th Iberoamerican Congress on Pattern Recognition*, pages 153–162, Havana, Cuba, November 2003.

[8] A. Bottino and A. Laurentini. Introducing a new problem: Shape-from-

silhouette when the relative positions of the viewpoints is unknown. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(11):1484–1493, 2003.

[9] P. Brunet, I. Navazo, J. Rossignac, and C. Saona-Vazquez. Hoops: 3D curves as conservative occluders for cell-visibility. *Computer Graphics Forum*, 20(3):431–442, 2001.

[10] B. Chazelle and H. Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of ACM*, 39(1):1–54, 1992.

[11] P. Eades, M. E. Houle, and R. Webber. Finding the best viewpoints for three-dimensional graph drawings. In *5th International Symposium on Graph Drawing*, pages 87–98, Rome, Italy, September 1998.

[12] H. Edelsbrunner, L. Guibas, and M. Sharir. The upper envelope of piecewise linear functions: Algorithms and applications. *Discrete and Computational Geometry*, 4:311–336, 1989.

[13] H. Edelsbrunner, J. O'Rourke, and S. Seidel. Constructing arrangements of line and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

[14] A. Efrat, L. Guibas, O. Hall-Holt, and L. Zhang. On incremental rendering of silhouette maps of a polyhedral scene. In *11th ACM-SIAM Symposium on Discrete Algorithms*, pages 910–917, San Francisco, California, January 2000.

[15] F. Gomez, F. Hurtado, T. Sellares, and G. Toussaint. Nice perspetive projections. *Journal of Visual Communication and Image Representation*,

12(4):387–400, 2001.

[16] J. E. Goodman, R. Pollack, and R. Wenger. Geometric transversals theory. In J. Pach, editor, *New Trends in Discrete and Computational Geometry*. Springer-Verlag, 1993.

[17] L. Kettner and E. Welzl. Contour edge analysis for polyhedron projections. In W. Strasser, R. Klein, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 379–394. Springer, 1997.

[18] J. J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13(3):321–330, 1984.

[19] M. R. Korn and C. R. Dyer. 3D multiview object representations for model-based object recognition. *Pattern Recognition*, 20(1):91–103, 1987.

[20] A. Laurentini. How many 2D silhouettes does it take to reconstruct a 3D object? *Computer Vision and Image Understanding*, 67(1):81–87, 1997.

[21] Melles Griot Corporation. *Machine Vision Guide*, 2003. `http://www.mellesgriot.com/products/machinevision/toc.htm`.

[22] J. W. Miller, V. Shridhar, E. Wicke, and C. Griffin. Very low-cost in-process gauging system. In B. G. Batchelor, J. W. Miller, and S. S. Solomon, editors, *Machine Vision Systems for Inspection and Metrology VII*, SPIE, pages 14–19, October 1998.

[23] H. Plantinga and C. R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.

[24] M. Pop, G. Barequet, C. A. Duncan, M. T. Goodrich, W. Huang, and S. Kumar. Efficient perspective-accurate silhouette computation and applications. In *17th ACM Symposium on Computational Geometry*, pages 60–68, Massachusetts, June 2001.

[25] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, New York, 1985.

[26] R. D. Schiffenbauer. *A Survey of Aspect Graph*. PhD thesis, Department of Computer and Information Science, Polytechnic University, New York, 2001.

[27] W. B. Seales and C. R. Dyer. Viewpoint from occluding contour. *Computer Vision, Graphics and Image Processing: Image Understanding*, 55:198–211, 1992.

[28] Siemens.

Outline inspection with SIMATIC VS 110. Product literature. `http://www.siemens.be/eit/microautomation/downloads/SIMATIC_VS110_EN.pdf`.

[29] SIGHTech Vision Systems. *Eyebot Application, Inspecting Hard Disk Media*. Product literature. `http://www.sightech.com/hard_disk_app_note.pdf`.

[30] M. R. Stevens and J. R. Beveridge. Interleaving 3D model feature prediction and matching to support multi-sensor object recognition. *Image Understanding Workshop, APRA96*, pages 699–706, August 1996.

[31] K. Sugihara. *Machine Interpretation of Line Drawing*. The MIT Press Series in Artificial Intelligence. MIT Press, Cambridge, Massachusetts, 1986.