Computational Geometry 44 (2011) 399–408

Contents lists available at ScienceDirect

Computational Geometry: Theory and Applications

www.elsevier.com/locate/comgeo

Efficient view point selection for silhouettes of convex polyhedra [☆,☆☆]

Therese C. Biedl [a], Masud Hasan [b,*,1], 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

ARTICLE INFO

Article history:
Received 15 September 2007
Accepted 8 April 2011
Available online 14 April 2011
Communicated by G. Toussaint

Keywords:
Convex polyhedra
Geometric transversal
Projections
Silhouette
Views

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.

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

[☆] Research supported by NSERC.
[☆☆] An earlier version appeared in: Proc. 29th International Symposium on Mathematical Foundation of Computer Science (MFCS'04), Lecture Notes in Computer Science, vol. 3153, Prague, Czech Republic, August 2004, pp. 735–747.
* Corresponding author. Tel.: +880 155 663 4106; fax: +880 2 966 5612.
E-mail addresses: biedl@uwaterloo.ca (T.C. Biedl), masudhasan@cse.buet.ac.bd (M. Hasan), alopez-o@uwaterloo.ca (A. López-Ortiz).
[1] The research was done while the author was a PhD student at the University of Waterloo.

0925-7721/$ – see front matter
doi:10.1016/j.comgeo.2011.04.001

[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, Benichou 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 Fig. 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. 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 [2]. 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).