Efficient Perspective-Accurate Silhouette Computation

G. Barequet C.A. Duncan M.T. Goodrich S. Kumar M. Pop

Silhouettes of geometric models (see Fig.1) are salient for many applications, including visualization [LE97] and shape analysis [BV95, Cro77]. The video describes a technique for efficient tracking of perspective-accurate silhouettes by reducing the problem to point location queries. Perspective-accurate silhouette computation is known to be difficult [KM96, KM98, KMGL96]. Note that silhouettes formed under perspective projection are significantly different from those under parallel projection and are more difficult to compute.

Given a polyhedral model with well-defined normals, the polygons (facets) whose normals point away from the viewpoint are called back-facing. Similarly, normals of front-facing polygons point toward the viewpoint. An edge shared by a back-facing polygon and a front-facing polygon is said to be on the silhouette of the model. The silhouette of a geometric model is among its most significant visual features. It may be used to describe the shape of a model with only very few details of its geometry. It can also be used for computing the visible portions of a model as it forms the boundary between the visible front-facing and the hidden back-facing polygons. Silhouettes are also useful in shadow generation [Cro77], model simplification [LE97], collision detection [BV95], image registration [LSB95], and several other applications.

The video presents an incremental algorithm for computing silhouettes of polyhedral models and updating them as the viewpoint changes. Most applications of interest exhibit coherence in silhouettes between successive computations as viewpoints change slowly. Our algorithm exploits this coherence to obtain fast updates of the silhouettes. Assume that we know the silhouette of a model from a given view point. An edge ceases to be (resp., becomes) a silhouette edge if and only if it is shared by one front-facing and one back-facing polygon (resp., two polygons of the same facing), and exactly one of its adjacent polygons changes orientation. Thus, when the viewpoint is changed we are interested in finding such orientation-change events. We reduce this problem to a point-location query in a planar-map data structure.

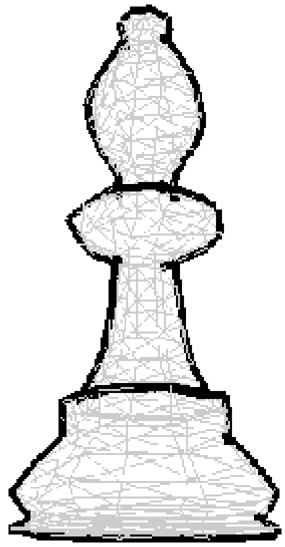

Figure 1: Wireframe bishop with its silhouette showing

The input model is assumed to be a file which describes a valid polyhedral model. The data are usually stored in the OFF file format (recognized by Geomview and other systems). Our system first parses the input file and builds a doubly-connected edge list data structure, as well as some additional data structures used for computing the silhouettes, as we describe below.

As defined above, given a viewpoint v, an edge e shared by the facets f_1 and f_2 is a silhouette edge if f_1 is facing v and f_2 is facing away from v (or vice versa). Finding all the silhouette edges by checking all the edges of the model can be prohibitively slow. Our algorithm for computing the silhouette edges is based on dualizing the polygons, or rather, their supporting planes, and on solving the dual problem.

We employ the following notion of duality described in [PS85]. Let f be a facet whose supporting plane has the equation $Ax+By+Cz+D=0$. The dual of f is the canonic point $p=(A/D,B/D,C/D)$. The primal edge shared by the adjacent facets f_1 and f_2 is mapped into the dual edge that connects between the the points dual of f_1 and f_2. It is important to note that the dual edge corresponds to the dual of the path (rotation in space) taken by one of the two primal facets until its supporting plane identifies with that of the other primal facet. If in order to align the two facets we need to cross the origin of coordinates, then the dual edge connects the two points by passing through infinity. This situation arises when the origin is on different sides of the two facets, i.e., the D components of the supporting planes have different signs. Likewise, the dual of a point $p = (a, b, c)$ is the plane $ax + by + cz + 1 = 0$. In particular, the dual of the viewpoint $v = (v_1, v_2, v_3)$ is the plane $v_1x + v_2y + v_3z + 1 = 0$.

The crucial observation is that a primal edge is a silhouette edge if and only if its dual edge intersects with the plane dual of the viewpoint. Thus, the problem of computing the silhouettes in the primary space reduces to the following problem in the dual space: preprocess a collection of spatial segments (possibly going through infinity) and store them in a data structure, so that given a query plane, we may quickly report all the segments intersected by the query plane. We solve this problem by using a variant of the BAR-tree (balanced aspect ratio tree) data structure [DGK99].

In order to use coherency of silhouette computing (solving a sequence of problems with nearby viewpoints), we store the duals of the facets (points in the dual space) in a data-structure that allows spatial double-wedge queries. The double wedge is bounded by the planes representing the duals of two consecutive viewpoints. The query thus seeks the *changes* that occur between two consecutive viewpoints. The reported points are those which moved from one side to the other of the "view-plane" (the dual of the viewpoint), corresponding to primal facets that were front facing and became back facing, or vice versa. For each reported facet we check the status of the edges on its boundary (whether it indeed became a silhouette edge), and update the list of silhouette edges if necessary. This procedure is guaranteed to report all changes in the silhouette since an edge can change its status only when one of its adjacent facets changes its status, thus being reported by our algorithm.

The video illustrates the silhouette computation process, visually demonstrates the significance of silhouettes and describes one application of silhouette finder, namely fast rendering. Some applications require interactive rendering of polygonal models but are unable to allocate sufficient computing resources to enable interactive rendering. 'Fast-rendering', on the other hand, displays only the edges on the silhouette of the model, and thus can provide interactivity in many cases without significantly reducing the feel of the model. Also note that the time required in silhouette computation is substantially lower than that required for rendering the full-scale model.

We have plugged our silhouette tracker into a variety of application programs including fast rendering, surface simplification and image registration and obtained significant performance improvement. We are currently in the process of refining these applications to further improve the performance of these systems. We also plan to build other applications like collision detection and shadow computation on top of the silhouette finder.

References

[BV95] W. Bouma and G. Vanecek. Velocity-based collision detection. In A. Paeth, editor, *Graphics Gems V*, pages 380–385, Academic Press, 1995.

[Cro77] F. C. Crow. Shadow algorithms for computer graphics. *ACM Computer Graphics*, 11(3):242–248, 1977. (SIGGRAPH Proceedings).

[DGK99] C.A. Duncan and M.T. Goodrich and S. Kobourov. Balanced Aspect Ratio Trees: Combining the Advantages of k-d Trees and Octrees. In *10th ACM-SIAM Symposium on Discrete Algorithms (SODA)* 1999.

[KM96] S. Kumar and D. Manocha. Hierarchical visibility culling for spline models. In *Proc. Graphics Interface*, pages 142–150, Toronto, Canada, 1996.

[KM98] S. Krishnan and D. Manocha. Efficient visibility computation. In *International Conference on Vision, Graphics and Image Processing*, New Delhi, India, Dec 1998.

[KMGL96] S. Kumar, D. Manocha, B. Garrett, and M. Lin. Hierarchical back-face culling. In *7th Eurographics Workshop on Rendering*, pages 231–240, Porto, Portugal, 1996.

[LE97] D. Luebke and C. Erikson. View-dependent simplification of arbitrary polygonal environments. In *Proc. ACM SIGGRAPH*, pages 199–208, 1997.

[LSB95] S. Lavallée and R. Szeliski and L. Brunie. Anatomy-based registration of three-dimensional medical images, range images, X-ray projections, and three-dimensional models using octree-splines. In *Computer-Integrated Surgery*, pages 115–143, MIT Press 1995.

[PS85] F.P. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag, New York, 1985.