Intersecting two polyhedra one of which is convex

by

Kurt Mehlhorn and Klaus Simon

Fachbereich 10, Informatik, Universität des Saarlandes, 6600, Saarbrücken, West Germany.

Abstract: Let P and Q be polyhedra one of which is convex. Let n and m be the number of edges of P and Q respectively and let s be the number of edges of the intersection $P \cap Q$. We show how to compute $P \cap Q$ in time $O((n + m + s)\log(n + m + s))$. Previously only algorithms with running time $O(nm)$ were known.

1. Introduction:

The treatment of solids and surfaces in three-dimensional space is one of the fundamental topics in computer graphics and computer-aided design. One of the goals of computational geometry (cf. Mehlhorn, Vol 3 for a textbook treatment) is to provide efficient algorithms for the basic operations (e.g. intersection, visibility, hidden line elimination) on these objects. In this paper we show how to compute the intersection of two polyhedra, one of which is convex, efficiently. More precisely, we show:

Theorem 1. Let P be a convex polyhedron with n vertices, let Q be an arbitrary polyhedron with m edges, and let $P \cap Q$ be a polyhedron with s edges. Given P and Q one can compute $P \cap Q$ in time $O((n + m + s) \log(n + m + s))$.

Previously only solutions with running time $O(nm)$ were known. The new algorithm is better provided that $s \leq n/\log(n + m)$. It remains open whether a subquadratic algorithm for intersecting two arbitrary polyhedra exists.

There are at least three $O((n+m)\log(n+m))$ solutions in the literature (Muller-Preparata, Dobkin-Kirkpatrick, Hertel-Mäntylä-Mehlhorn-Nievergelt) for the intersection of two convex polyhedra. The solution by Muller and Preparata is based on duality and seems to be intimately tied to convexity. However, the results of the other two papers can be combined (and slightly extended) to yield the new result presented here.

We borrow from [HMMN] the idea of reducing the intersection problem to the support problem which we now define. Let $\bar{P}$ and $\bar{Q}$ be the surfaces of P and Q respectively. Then $\bar{P} \cap \bar{Q}$ is a straight-line graph embedded into three-dimensional space. The *support problem* is to compute at least one point of every connected component of $\bar{P} \cap \bar{Q}$. Our proof of the main theorem is based on the following two lemmas.

Lemma 1. The support problem for P and Q can be solved in time $O((n + m)\log(n + m))$.

Lemma 2. Given a solution for the support problem for P and Q one can compute the intersection of P and Q in time $O((n + m + s)\log(n + m + s))$.

A proof of lemma 2 can be found in section 3. The basic idea is to compute first $\bar{P} \cap \bar{Q}$ and then $P \cap Q$ by a graph traversal algorithm. This is similar to [HMMN]. The proof of lemma 1 can be found in section 2. Our solution for the support problem is based on the hierarchical representation of convex polyhedra introduced by Dobkin-Kirkpatrick and later used in Edelsbrunner-Maurer. This representation allows for "binary search" on convex objects and thus yields fast algorithms for the intersection of a convex polyhedron with lines and planes.

Throughout this paper we assume for simplicity that P and Q are in general position, i.e. that no edge of P intersects an edge of Q. The general case will be treated in the full paper. If P and Q are in general position then every connected component C of $\bar{P} \cap \bar{Q}$ is a polygon whose vertices are given by edge-face intersections and whose edges are given by face-face intersections.

2. Data structures and a solution to the support problem

A polyhedron consists of vertices, edges and faces. A vertex is a point in $\mathbf{R}^3$, an edge is a line segment connecting two vertices and a face is a subset of some plane whose boundary is a polygon; cf. Figure 1 for an example.

We assume that polyhedra are represented by the quad-edge data structure of Guibas-Stolfi, see also Edelsbrunner-Maurer. In this data structure each undirected edge e is stored as a pair of directed edges e and **sym** e. Also each face is essentially stored as a circular list of its boundary edges. More precisely, each directed edge e stores its origin **org** e and pointers to neighboring edges $e' =$ **onext** e and $e'' =$ **dnext** e where e' and e'' are edges on the boundary of the face to the left of e (as seen from outside the polyhedral body) and **org** e'' = **org sym** e and **org** e' = **org** e, cf. Figure 2. Note that the quad-edge structure allows the clockwise and counterclockwise traversal of the edges incident to any given vertex.

For convex polyhedra we use in addition the hierarchical representation introduced by Dobkin-Kirkpatrick. A sequence $P_0, P_1, \ldots, P_k$ of convex polyhedra is a hierarchical representation of P if

a) P_0 is a tetrahedron and $P_k = P$
b) $V_i = V_{i+1} - I_{i+1}$ where V_i is the vertex set of P_i and I_{i+1} is an independent set of vertices of P_{i+1}, i.e. no two vertices of I_{i+1} are connected by an edge of V_{i+1}.

Fact (Kirkpatrick, Dobkin-Kirkpatrick)

a) There are constants c_1 and c_2 such that I_{i+1} exists with $|I_{i+1}| \geq |V_{i+1}|/c_1$ and $\deg(v) \leq c_2$ for all vertices $v \in I_{i+1}$ where $\deg(v)$ is the degree of vertex v. Moreover, I_{i+1} can be determined in time $O(|V_{i+1}|)$.
b) A hierarchical representation with $k = O(\log n)$ can be constructed in time $O(n)$.

We stipulate that each P_i in the hierarchical representation is stored as a quad-edge structure and that P_i and P_{i+1} are tied together as follows. Let e be an edge of P_i. If e is also an edge of P_{i+1} then e (as an edge of P_i) points to e (as an edge of P_{i+1}). If e is not an edge of P_{i+1} then there is a unique vertex $v \in I_{i+1}$ such that the interior of the triangle spanned by e and v does not intersect P_i, cf. Figure 3. In this case we let e point to edge g of P_{i+1} with **org'g** $= v$ and **org sym** g $=$ **org** e. Dobkin-Kirkpatrick and later Edelsbrunner-Maurer have shown how to use the hierarchical representation for detecting intersections and for performing extremal queries on convex polyhedra. The following theorem can be proved by their methods.

Theorem 2 Let P be a convex polyhedron given by its hierarchical representation.
a) If l is a line then $l \cap \bar{P}$ can be computed in time $O(\log n)$.
b) If h is a plane then it can be decided in time $O(\log n)$ whether $h \cap \bar{P}$ is empty. If $h \cap \bar{P} \neq \emptyset$ then one point $p \in h \cap \bar{P}$ can be computed in the same time bound.

We will next show how Theorem 2 can be used to solve the support problem.

Consider the following algorithm. It computes a set S which is a solution to the support problem.

```
(1)  $S := \emptyset$
(2)  for all edges $e$ of $Q$
(3)  do $S := S \cup (e \cap \bar{P})$ od;
(4)  for all faces $f$ of $Q$
(5)  do let $h$ be the plane supporting $f$
(6)      if $h \cap \bar{P} \neq \emptyset$
(7)      then let $p$ be the intersection of $h$ with an edge of $\bar{P}$;
(8)           if $p \in f$ then $S := S \cup \{p\}$ fi
(9)      fi
(10) od
```

Theorem 3 The algorithm above solves the support problem in time $O(m \log n)$

Proof: We argue correctness first. Let C be a connected compound of $\bar{P} \cap \bar{Q}$. We distinguish two cases, cf. Figure 4.

Case 1 C contains a point on an edge e of Q. Then a point of C is clearly determined in line 3.

Case 2 C does not contain a point on an edge e of Q, i.e. C is **completely contained** in a face f of Q. Let h be the plane supporting f and let p be any intersection of h with an edge of P. Then p is a point of C.

This shows that at least one point of every connected component C is added to S. Conversely, it is trivial to see that only points of $\bar{P} \cap \bar{Q}$ are added to S. This proves correctness.

The running time is also easily estimated. For each edge or face of Q we need time $O(\log n)$ for computing intersections in lines 3 or 6 respectively for a total cost of $O(m \log n)$. Also for each f of Q we need to test for at most one point p whether $p \in f$. This can clearly be done in time proportional to the number of vertices of f. Thus the total cost of line 8 is $O(m)$. This completes the proof of theorem 3. ∎

3. Computing the intersection

Let S be a solution to the support problem for P and Q. Each point $p \in S$ is given as an edge-face intersection. We will now describe how to compute $P \cap Q$ from S. We compute $\bar{P} \cap \bar{Q}$ first and then compute $P \cap Q$ in a second stage. We assume throughout this section that all faces of P and Q are triangulated. A triangulation can be computed in time $O((n+m)\log m)$ by a number of triangulation algorithms, e.g. Hertel-Mehlhorn.

Let C be a connected component of $\bar{P} \cap \bar{Q}$ and let $p \in S$ be a vertex of C. It is easy to trace C starting in p in time proportional to the number of vertices of C as follows. Let p be the intersection of edge e and face f and let f_1 and f_2 be the two faces incident to e. Then $f \cap f_1$ and $f \cap f_2$ define edges of C and also the vertices adjacent to $p, \ldots$ (cf. Figure 5).

In this way we can trace all components of $\bar{P} \cap \bar{Q}$ in time $O(s)$ except for one small detail. We have to avoid to trace the same component twice. We therefore insert all vertices encountered into a dictionary and use this dicitionary to make sure that no component is traced twice. Thus the total cost of tracing the components of $\bar{P} \cap \bar{Q}$ is $O(s \log(n+m))$.

All this point we have computed $\bar{P} \cap \bar{Q}$. It remains to compute those edges of $P \cap Q$ which are part of edges of P and Q. Again this can be done by a simple graph traversal algorithm. The details are as follows. We first sort for every edge e of P (and Q) the vertices of $\bar{P} \cap \bar{Q}$ which lie on e. This takes time $O(s \log(n+m))$ and splits every edge into intervals which alternately belong to $P \cap Q$ and do not. We subdivide the edges as given by these intervals and throw away all parts outside $P \cap Q$. A simple graph traversal will then yield $P \cap Q$.

This proves lemma 2 and theorem 1 in the case that $\bar{P} \cap \bar{Q} \neq \emptyset$. So let us finally assume that $\bar{P} \cap \bar{Q} = \emptyset$. Then either $P \cap Q = \emptyset$ or $P \supseteq Q$ or $Q \supseteq P$. We can distinguish these three cases by testing a single vertex of P for containment in Q and a single vertex of Q for containment in P. These tests can certainly be done in time O(n+m) and therefore $P \cap Q$ can be computed in the desired time bound.

4. Conclusion

We presented a simple and efficient algorithm for computing the intersection of a convex and an arbitrary polyhedron. The algorithm combines ideas developed previously for the intersection of two convex polyhedra; in particular, it uses the reduction to the support problem [HMMN] and the hierarchical representation of convex polyhedra [DK].

5. References

[1] H. Edelsbrunner, H.A. Maurer: "Finding Extreme Points in Three Dimensions and Solving the Post-Office Problem in the Plane", Information Processing Letters, to appear.

[2] St. Hertel, M. Mäntylä, K. Mehlhorn, J. Nievergelt: "Space Sweep Solves Intersection of Two Convex Polyhedra Elegantly", Acta Informatica, 21, 501-519, 1984.

[3] St. Hertel, K. Mehlhorn: "Fast Triangulation of Simple Polygons", FCT 83, LNCS Vol 158, 207-218.

[4] D.P. Dobkin, D.G. Kirkpatrick: "Fast Detection of Polyhedral Intersections", 9th ICALP, LNCS 140, 154-165, 1982.

[5] D. Kirkpatrick: "Optimal Search in Planar Subdivisions", SICOMP 12, 28-35, 1983.

[6] K. Mehlhorn: "Data Structures and Efficient Algorithms", Vol 3: "Multi-dimensional Data Structures and Computational Geometry", Springer Verlag, EATCS Monographs in Computer Science, 1984.

[7] D.E. Muller, F.P. Preparata: "Finding the Intersection of Two Convex Polyhedra", TCS 7, 217-236, 1978.

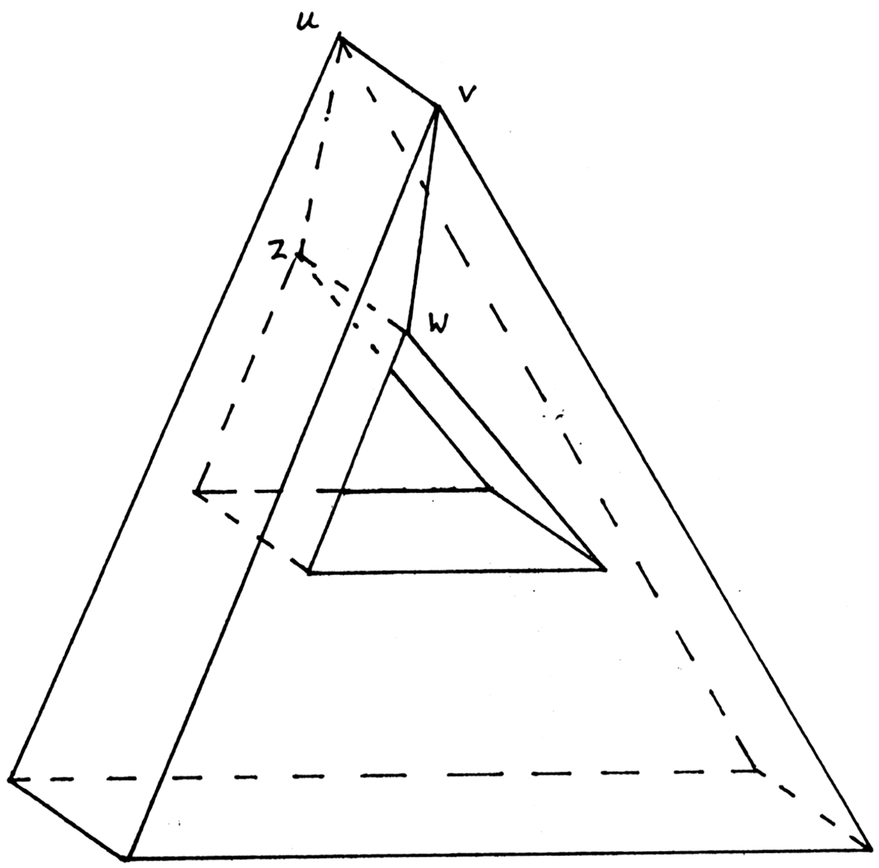

Figure 1: A triangular cylinder with a triangular cylindrical hole. The edges (v,w) and (u,z) have to be included because the boundary of every face is required to be a polygon.

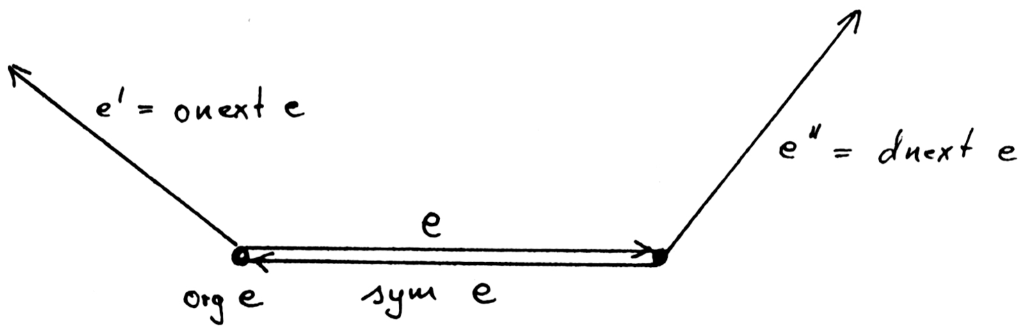

Figure 2:

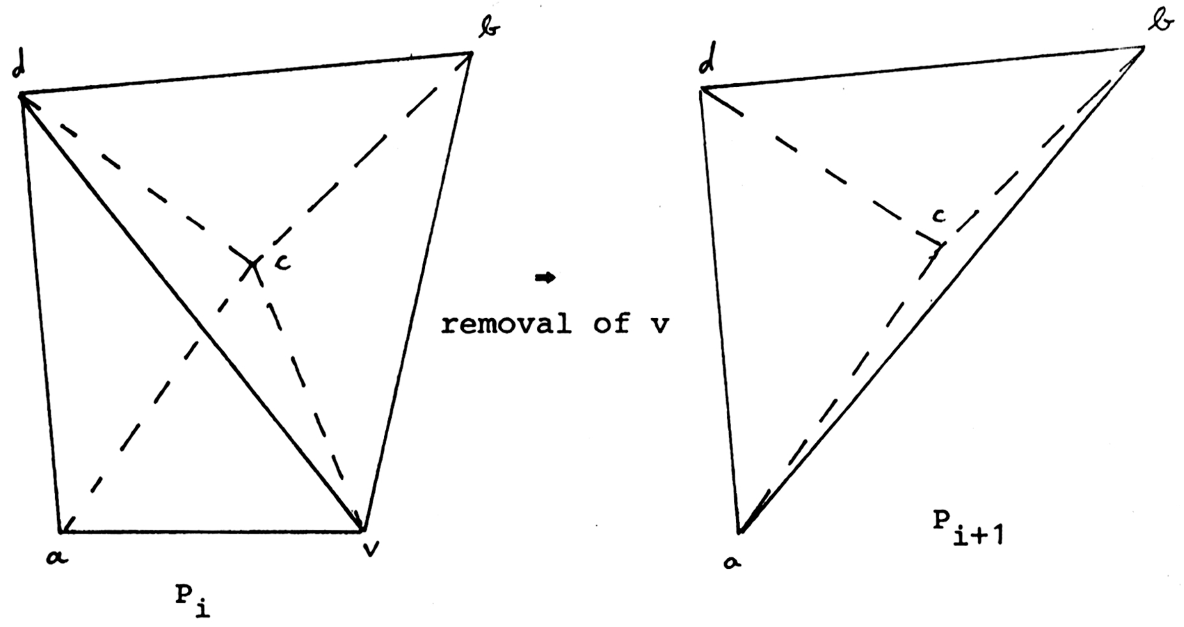

Figure 3: P_i is obtained from P_{i+1} by removal of v, edge (a,b) of P_i points to edge (v,a) of P_{i+1}.

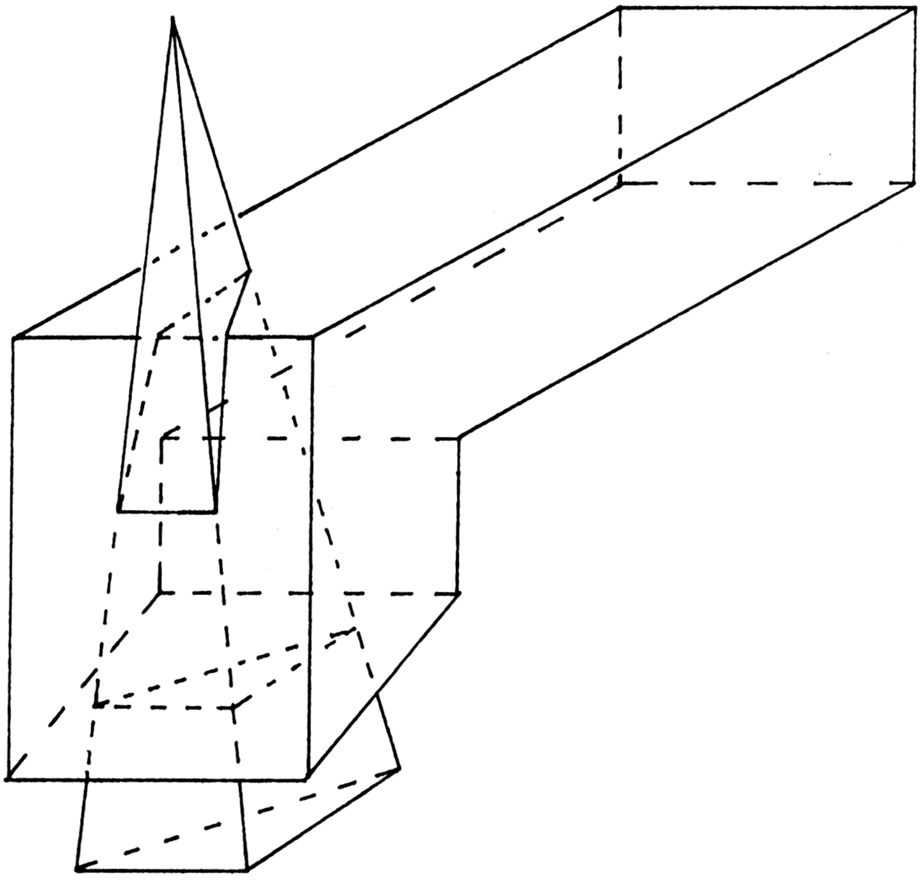

Figure 4: P is a tetrahedron, Q is L-shaped. $\overline{P} \cap \overline{Q}$ consists of two components one of which is completely contained in a face of Q.

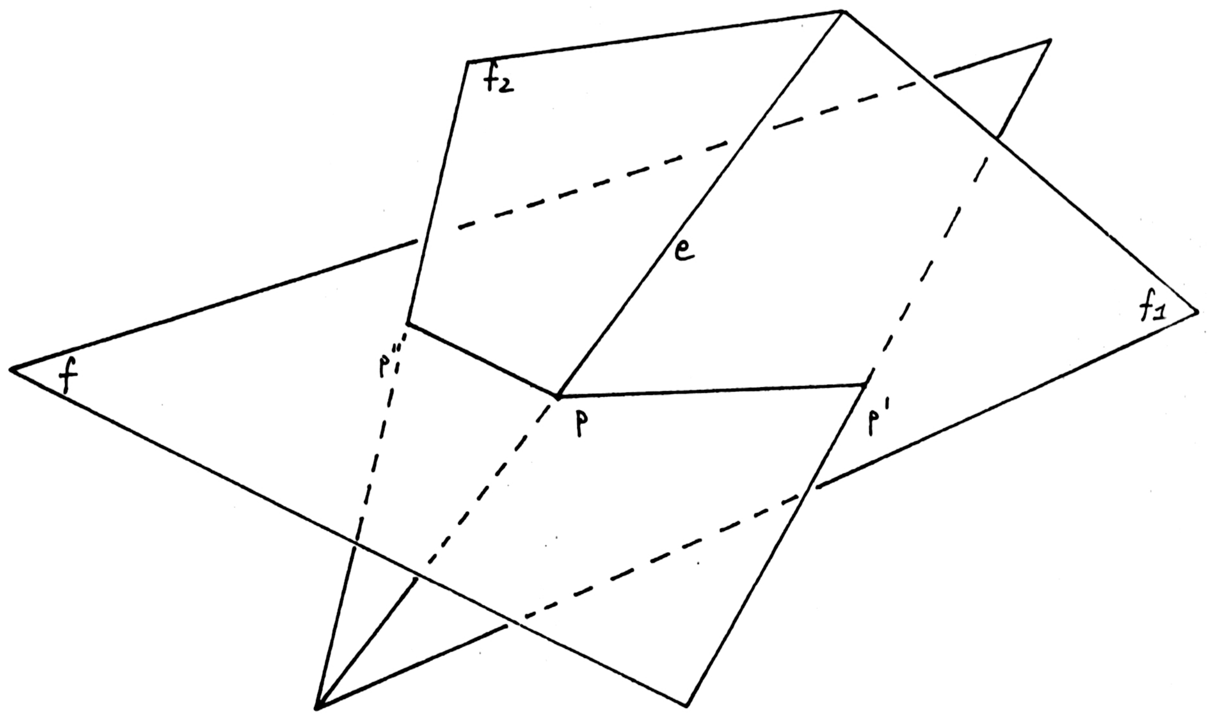

Figure 5: